Security and Complexity Analysis of LUT-based Obfuscation: From Blueprint to Reality

Gaurav Kolhe*, Hadi Mardani Kamali†, Miklesh Naicker‡, Tyler David Sheaves‡,
Hamid Mahmoodi‡, Sai Manoj P D†, Houman Homayoun* Setareh Rafatirad†, Avesta Sasan†
* Department of Electrical and Computer Engineering, University of California, Davis, CA, USA.
† Department of Electrical and Computer Engineering, George Mason University, VA, USA.
‡ School of Engineering, San Francisco State University, San Francisco, CA, USA.
*{gskolhe, hhomayoun}@ucdavis.edu †{hmardani, spudukot, srafatir, asasan}@gmu.edu
‡{tsheaves, miklesh1, mahmoodi}@sfsu.edu

Abstract—**Recent obfuscation schemes have leveraged reconfigurable logics to alleviate various hardware security threats. However, existing reconfigurable logic-based obfuscation schemes focus on specific design factors such as gate replacement strategy or an optimization metric such as SAT-hardness. Despite meeting the focused metrics such as security, the obfuscation also incurs overheads, which are not well analyzed in the existing works.**

In this work, we provide a comprehensive analysis on reconfigurable logic obfuscation schemes i.e., LUT-based obfuscation by investigating 3-key design factors such as (1) LUT size, (2) number of LUTs, and (3) replacement strategy as they have a considerable impact on design criteria, i.e., Power-Performance-Area (PPA) and Security (PPA/S). Our results show that among the studied parameters the size of LUT has the most prominent impact on improving the resiliency of LUT-based obfuscation against the SAT and removal attacks. However, using large size LUTs incur significant PPA overheads, making such solutions unfeasible and unpractical.

To address this challenge, this work proposes a pragmatic solution based on a customized LUT, where the security provided by each LUT is superior to that of traditional LUT-based obfuscation. The proposed solution primarily benefits from LUT-based obfuscation reinforced with additional logic/routing obfuscation that is implemented using small 2-input LUTs. We evaluate the hardware security and overhead of the proposed customized LUT-based obfuscation on various benchmarks to prove that the customized LUT-based obfuscation breaks the PPA trade-offs while exhibiting robustness against the SAT and removal attacks. The customized LUT-based obfuscation comes with 8× reduced area and 2× reduced power on an average compared to state-of-the-art LUT-based obfuscation without compromising security.

I. Introduction

To minimize the increasing fabrication costs and design complexity of Integrated Circuits (ICs), many IC design companies are becoming fabless [1]. Employing third-party Intellectual Property modules (3PIP) benefits the Intellectual property (IP) holder by reducing the time-to-market while cutting down the design flow efforts. Despite the economic benefits, this trend poses significant challenges to hardware security in numerous forms [2]–[5].

To thwart the prevalent security threats, many hardware design-for-trust techniques have been introduced such as split manufacturing [6], [7], IC camouflaging [8], and logic locking [9]. Among multiple aforementioned techniques, logic locking can thwart the majority of the attacks at various phases in the IC Production chain [10]. This is because logic locking requires the correct keys to unlock the true functionality of the design. Additionally, as a part of the post-manufacturing process, the activation of IC (i.e., providing correct keys) will be accomplished in a trusted regime to hide the functionality from the untrusted foundry and other attacks.

Although logic locking schemes enhance the security of the IP, the advent of Boolean satisfiability (SAT) based attack [11] [12], as an "oracle-guided" threat model, shows that by applying a few stimuli to the design and analyzing the output, the key value and functionality of an IC could be extracted in the order of a few minutes.

To overcome the security threats posed by the SAT attacks [13], numerous SAT resilient logic locking techniques [14]–[17] have been proposed. However, many of these techniques are vulnerable to other RE threats such as removal and Satisfiability Modulo Theory (SMT) attacks [13], [18], [19]. Among the proposed logic locking schemes, some recent obfuscation methods [20]–[28] focus on using configurable barriers, such as Look-Up-Tables (LUTs), as a key-programmable logic to achieve resiliency against state-of-the-art attacks.

In this work, we first study the existing reconfigurable logic obfuscation schemes and discuss their strengths and weaknesses against the SAT attacks and the corresponding overheads. We find that, most of the previously proposed obfuscation techniques remain vulnerable to the SAT attack when executed for a sufficient amount of time [11]. Among the previously proposed reconfigurable logic obfuscation schemes, we investigate LUT-based obfuscation and observe that in the LUT-based obfuscation, utilizing large-size LUTs guarantees the security against state-of-the-art SAT attacks, but at the cost of significant area-power overheads. To make the LUT obfuscation a practically viable solution, we need to radically reduce the design overheads without compromising on the security aspect. Unfortunately, these two goals are contradictory, as reducing the size of LUT helps in mitigating the design overheads, but, significantly compromises the security. To address these challenges, we propose a pragmatic customized LUT design that not only benefits from the configurable barriers for obfuscation but also mitigates the incurred area and power overheads without sacrificing the security. The customized LUT replaces the *fairly large LUT with a blend of two LUTs*, which is arranged in a way that 2-input LUTs are preceded by a large LUT, thus, small LUTs provide routing obfuscation, while the large LUT primarily provides logic obfuscation. This combination of these two obfuscation techniques assists in elevating security provided by logic obfuscation while concurrently reducing the design overheads.

The proposed customized LUT-based obfuscation is rigorously tested for various metrics such as power, performance, area, and particularly security on different benchmarks such as AES, which are representative in terms of size and complexity of real-world designs. The main contributions of this work are outlined as follows:

- **Study existing reconfigurable obfuscation techniques**: We study the existing reconfigurable obfuscation techniques and discuss their limitations for practical implementation. LUT-based obfuscation is further investigated using three parameters: (1) LUT size, (2) number of LUTs, and (3) replacement strategy, as they have a considerable impact on design criteria.
- **Increasing the size of LUTs (scale-up) as the most important factor for SAT Resiliency**: To address the design security using LUT-based obfuscation we show that size of the LUT is the crucial element in guaranteeing security against state-of-the-art SAT attacks but comes at the cost of hefty overheads, which makes the LUT-based obfuscation an idealistic method for obfuscation.
- **Proposed Pragmatic Customized LUT**: As a practical solution to implement LUT-based obfuscation, we propose a customized LUT that breaks the design overhead and security trade-offs, whereas recent works are optimized for either security or power, performance, and area.

II. Logic Obfuscation and SAT Attack

A. Logic Obfuscation

In logic obfuscation, 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 de-layering by introducing ambiguity. The strength of traditional logic obfuscation is based on the location of the inserted/replaced gates [9], [29], [30]. Traditional attacks, such as justification/sensitization [31] try to RE the design using heuristic techniques and Automatic Test Generation Pattern (ATPG) based approaches to extract the keys.

B. SAT Attack

To retrieve the keys, SAT attack [11] iteratively eliminates the incorrect keys based on specific input patterns, called *Distinguished Input Patterns (DIPs). DIP* is an input that produces two different outputs for two different keys. By iteratively finding the *DIPs*, SAT attack can eliminate all incorrect keys within a range of few seconds to minutes, even for large circuits. At each iteration and after finding a *DIP*, a new constraint is added to the satisfiability problem posed on the SAT solver, until it can no longer find a satisfying assignment. At this point, any key that satisfies all previously found *DIPs*, is considered as the correct key for the obfuscated circuit.

C. Post-SAT Obfuscation and Challenges

All logic locking and camouflaging algorithms after SAT attack try to increase the number of iterations needed for finding the correct key in the SAT attack. Most of them concentrate on decreasing the size of wrong key elimination in each iteration to maximize the number of iterations and consequently exponentially increase the SAT attack execution time. SARLock [14], Anti-SAT [15], and SFLL-HD^0 [32] are the state-of-the-art approaches with a common goal of increasing the number of iterations exponentially. Also, some obfuscation schemes are based on adding cycles in the circuit, especially stateful/oscillating cycles [33], [34]. When the SAT attack encounters these cycles, two states can happen: (1) The SAT solver returns wrong key or (2) The SAT solver will be stuck in an infinite loop. In fact, the SAT solver cannot extract the correct key in cyclic-based approaches with stateful/oscillating cycles. However, CycSAT, SMT-Solver, and SRCLock [19], [34], [35] show that by considering some conditions and having some pre-processing, cyclic-based schemes can be decrypted. This motivates the research community to build a comprehensive obfuscation scheme robust against different hardware threats.

III. Investigation on LUT-based Obfuscation

A plethora of work have focused on utilizing reconfigurable logic such as LUT-based obfuscation as a defense against reverse engineering (RE) attacks [20]–[28]. In LUT-based obfuscation, the design is partially mapped to LUTs, for example, if a 2-input AND gate has to be obfuscated using LUT of size 2, one can set the configuration bits of the LUT to ‘0001’ as per the truth table of the AND gate. The partial mapping of the circuit results in a design implementation that is a hybrid of custom (ASIC) and programmable (FPGA) styles. Besides, reconfigurable bits can be stored in non-volatile memory (NVM) such as magnetic tunnel junction (MTJ). These, stored bits are highly susceptible to be lost during RE de-layering process. With the incorrectly configured LUT block, the design will remain unintelligible, which will refrain the attacker from understanding the functionality of the design. Figure 1 shows the LUT-based obfuscation.

From the SAT attack perspective, to model the LUT-based obfuscation, each LUT is substituted with a (2+)-level MUX. The work in [11] has shown that many of the reconfigurable schemes such as [20]–[23], [30] are vulnerable to the SAT attacks. To thwart the SAT attacks recent works such as [27] focuses SAT resiliency, however with no investigations on PPA

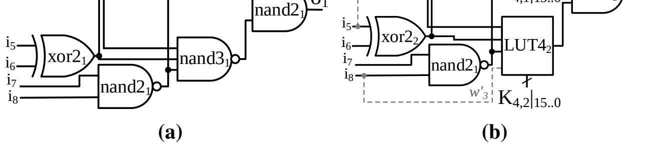

Fig. 1: Using Enlarged LUTs for obfuscation, (a) A Sample Circuit, (b) Obfuscation with LUTs.

overheads, while the work in [26] fails to acknowledge the resiliency of their solution against the SAT attack. One can model the reconfigurable blocks proposed in previous works using LUT of size 2 for the SAT attack modeling, as LUT of size 2 can render all the functions realized by reconfigurable blocks proposed in [24]–[27].

Figure 2 shows the resiliency of the LUT obfuscation by leveraging LUT of size 2 against the SAT attack. Various benchmarks are obfuscated with from 5% up to 35% using LUT of size 2. *It can be observed that irrespective of the benchmark, the SAT attack can reverse engineer the design in less than 5 days.* It needs to be noted that obfuscating 35% of the circuit isn't a viable solution, as it adds an enormous amount of overhead. Therefore, traditional LUT-based obfuscation techniques wouldn't cater to providing resiliency.

When using the LUT with size n to replace a gate, each n-input LUT can provide all 2^{2^n} possible functions, which increases the key length along with the search space to find the correct key configuration for LUT. However, increasing the size of large LUTs impose large area and performance overheads. Therefore, the trade-off to address is (1) using large LUT size while obfuscating less number of gates *or* (2) using small LUT size while obfuscating a large number of gates to enhance the attack resiliency. Therefore, in this work, we perform a large design-for-security space exploration for LUT-based obfuscation using three key factors namely (1) LUT size, (2) number of LUTs, and (3) replacement strategy.

A. *Replacement Strategies*

The strength of obfuscation is also dependent on the location of the gate(s) that are obfuscated i.e., replacement strategy. We discuss a few replacement policies below.

*1) **Random Selection (RND)**:* As the baseline replacement strategy, we implement random selection/replacement, as opposed to the *independent selection* in [21]. As the name implies, the gates are selected randomly.

*2) **Low Output Corruptibility (LC)**:* As mentioned previously, the SAT attack works on the concept of Conflict-Driven Clause Learning (CDCL). In *CDCL*, the SAT attack searches for the conflict clauses to make the cut-off for learning clauses. Finding the conflict clauses depends on the comparison between two different outputs for two different keys. Consequently, the higher the *hamming distance* of two outputs, finding the *DIP* is less complex. If obfuscation strategy influences different outputs by applying each input, the probability of *hamming distance* > 1 will be increased drastically, which leads to faster de-obfuscation. Therefore, the higher the corruptibility, the easier de-obfuscation by the SAT attack is. Therfore, the optimal solution is to minimize the *hamming distance* to 1.

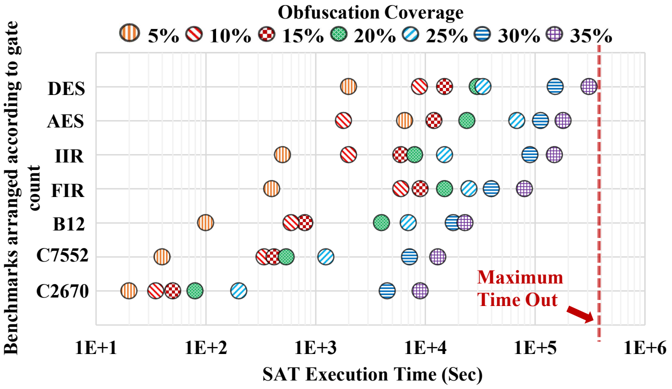

Fig. 2: SAT Attack Execution Time on various benchmarks obfuscated using LUT size 2, while obfuscation coverage is varied from 5% to 35%.

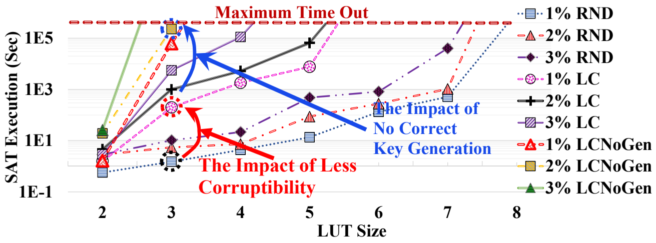

Fig. 3: SAT Attack Execution Time for Different (1) Replacement Strategy, (2) LUT Size (scale up), and (3) Number of LUTs (scale out) in ISCAS-85 c7552.

To have the lowest output corruptibility, one needs to employ a custom Breadth-First Search (BFS) on the logic design based to find the set of logic gates that have the lowest corruption metric.

*3) **Avoiding Unintentionally Correct Key Generation (LC_NoGen)**:* Due to the capability of LUTs for implementing all 2^{2^n} possibilities for a LUT of size n, connecting LUTs directly may generate some additional correct keys. This scenario provides additional options for the SAT solver to find the key, which results in diminishing execution time of the SAT solver. Also, increasing the number of LUTs which are directly connected significantly increases the number of correct possibilities in design. As a simple example, connecting two inverters directly has the same functionality as connecting two buffers back to back. Hence, since previous work uses a large number of gates to be replaced with LUTs, the probability of choosing directly connected gates are extremely high.

Figure 3 illustrates the overall impact of the three key design factors on execution time. As it can be seen, even for random insertion strategy, for LUT size larger than 8, obfuscating ∼1% of each circuit is sufficient to provide SAT resiliency. Further, it is evident that *LC_NoGen*, which has both conditions considered in replacement strategy, remarkably increases the SAT execution time, which shows its effectiveness on SAT resiliency. However, increasing the size of LUTs significantly increases the hardness of obfuscation regardless of the replacement strategy and the number of LUTs obfuscated.

B. *LUT Size versus Number of LUTs*

One of the straightforward approaches for LUT-based obfuscation is to either increase the number or the size of LUTs to enhance the security against the SAT attack(s). For example, instead of using a LUTn for the n-input gate, a LUTn+ (i.e., LUT$n+1$, LUT$n+2$,...) is used. When LUTs are replaced by MUXs for SAT modeling leads to a $\log_2(n)$-level MUX-based structure. Thus, by increasing the size of the LUTs, the SAT attack replaces them with more deeper MUX trees, and consequently, the de-obfuscation time gets exponentially

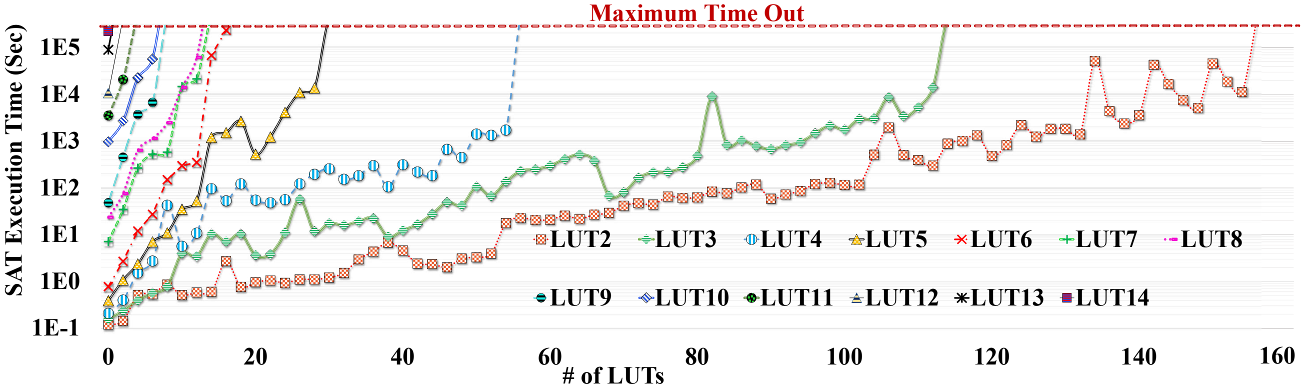

Fig. 4: SAT Attack Execution Time on Synthesized ISCAS-85 c7552 with Different Number of LUTs and Different LUT Sizes.

longer to exploit the value of keys for LUTs, making large LUT resilient against the SAT attack.

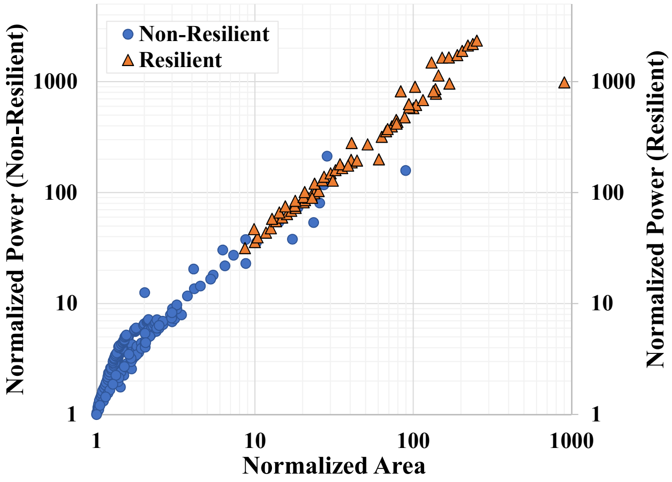

Fig. 5: Normalized area and power overhead of LUT-based obfuscation. Instances that result in the SAT execution time out are marked as SAT resilient configuration.

Additionally, the SAT attack work based on the cconcept of Conflict-Driven Clause Learning (CDCL) for finding *DIPs* [14]. Due to the symmetric structure of the MUX tree model, there is no shortleaf in the equivalent logic to find the cut-off (conflicts) in the logic tree. Due to the virtue of this, enlarging the size of LUTs (scale-up) increases the depth of this symmetric tree which makes it harder against the SAT solver to find conflicts.

Figure 1 shows an example of using large-sized LUTs in the design. As it can be seen, two 2-input NOR gate ($nor2_1$) and 3-input NAND gates ($nand3_1$) are substituted by LUT_4. Using larger LUTs provide some extra input 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 the intermediate wires.

Figure 4 demonstrates the SAT execution time with more details on ISCAS-85 C7552 for different size of LUTs, and different number of LUTs using the replacement strategy LC_NoGen. This experiment shows that using larger sizes of LUT provides higher SAT resiliency than obfuscating a higher number of gates with smaller sizes of LUTs. For instance, only replacing a single gate with a LUT_{13} is sufficient to make the design perfectly resilient against the SAT attack.

Although our evaluations from Figure 4 show that LUT scale-up is the most straightforward approach for LUT-based obfuscation for yielding high resiliency against state-of-the-art attacks, Figure 5 shows that precipitous increase in the number and the size of LUT can render inefficient solutions. Every possible combination from Figure 4 is synthesized using Synopsys Design Compiler to obtain area and power overhead which are plotted in Figure 5. In this example, we consider the design to be resilient to the SAT attack if the execution results in time out. It can be observed that securing design for more than 5 days using LUT obfuscation, incurs at least 10x area and power overhead, making LUT-obfuscation an idealistic solution for hardware security.

Therefore, to make LUT obfuscation as a realistic solution one needs to 1) radically reduce the design overhead, and 2) do not compromise the security against various attacks. Unfortunately, the results show that these two goals are contradictory to the discussed LUT-based obfuscation. To reduce the design overhead, one can reduce the size of LUT, however, this will significantly compromise the security. To address the design overhead, in the following section, we propose a customized LUT design.

IV. Proposed Customized LUT

To reduce the design overhead, we need to break the trade-off between the security offered by the LUT and the design overhead. In the proposed solution, we leverage a 2-input LUT (LUT_2) which is preceded by a large LUT as shown in Figure 6, where the LUT_2 can be used for routing obfuscation or logical obfuscation, while the other larger LUT is primarily used for logical obfuscation. If security grows at a higher rate than the incurred overhead, then replacing few gates with large LUTs can realize the proposed solution a viable means of hardware security.

When a large LUT is used for obfuscation, besides creating larger key search space, it also creates the SAT-hard instance.

When a netlist is converted to the bench format for the SAT attack simulation, the LUTs that are used for obfuscation can be represented using the MUXes as discussed in Section III. In the SAT solver, the clause to variable ratio is one of the important metrics for qualitatively evaluating the complexity of the problem. For example, a SAT-instance to be proven as SAT-hard, the clause to variable ratio needs to be around 4.2 [17]. However, when the clauses are short, there are many satisfying assignments and the clauses are said to be under-constrained. On the other hand, long clauses are over-constrained (contradictions can often be easily found), facilitating determining a large number of *DIPs*. The SAT solver uses the DPLL (Davis Putnam Logemann Loveland) method for backtracking, and with an increasing ratio of clauses to the variable, the computational cost of DPLL calls decreases monotonically. When LUT is represented using the MUX in the bench format, it has 4 clauses. By leveraging the LUTs for obfuscation, it assists in averaging clause/variable ratio in the Conjunctive Normal Form (CNF) to 4. Furthermore, symmetric problems pose a great difficulty to the SAT solver in finding a solution [17]. To further aggravate the security offered by traditional LUT, we add extra LUTs at the select line of large LUT. The scenario can be visualized as the addition of MUX-tree by LUT_2 to the existing MUX-tree of Large LUT. The resulting customized block benefits from both LUT and routing obfuscation. As the LUT is a fully reconfigurable logical block, by addition of a 2-input LUT increases the number of ways large LUT can be configured, which significantly increases the key search space. For the customized LUT to function as intended, the LUT_2 needs to be configured correctly, followed by the larger LUT, which adds additional constraints in finding the keys. All these factors altogether help in elevating the security offered by the customized LUT.

We leverage the proposed algorithm *LC_NoGen* for gate replacement which adds an extra layer of resiliency to the customized LUT solution. From figure 6, we can see the process of obfuscation using customized LUT. From previous experiment, for C7552 design to be SAT resilient gates are identified and replaced using LUT_8 in traditional LUT obfuscation, whereas, in the customized LUT, the gate can be mapped to the combination of LUT_2 and LUT_n such that $n < 8$ where n is the size of LUT. Therefore we compare proposed LUT obfuscation against traditional LUT obfuscation with size 8.

In proposed LUT obfuscation we map the gate which is selected for obfuscation to the large LUT. This large LUT is accompanied by extra 2-input LUT which are placed in front of the large LUT. Previously if the single gate was mapped to LUT_8, using customized LUT we can map the gate to LUT_4, and 4-inputs of LUT_4 can be driven using 4 LUT_2. This customized LUT will require 8 inputs among which few can be driven using dummy inputs. Using LUT_4 and 4 LUT_2 for obfuscation, we can have $4 \times 2^{2^2} \times 2^{2^4}$ configurations.

Replacing a single gate with such a bigger LUT leaves a place for further research of mapping additional gates to the LUT such that PPA overhead can be mitigated. However, we still show that even mapping a single gate to such large LUT makes the customized LUT obfuscation a practical solution in the hardware security domain. In this example, we show that the large LUT_8 is replaced by LUT_4 with 4 2-input LUT, however, we are not limited to this single configuration. We can use the configuration such as LUT_6 with 6 2-input LUT.

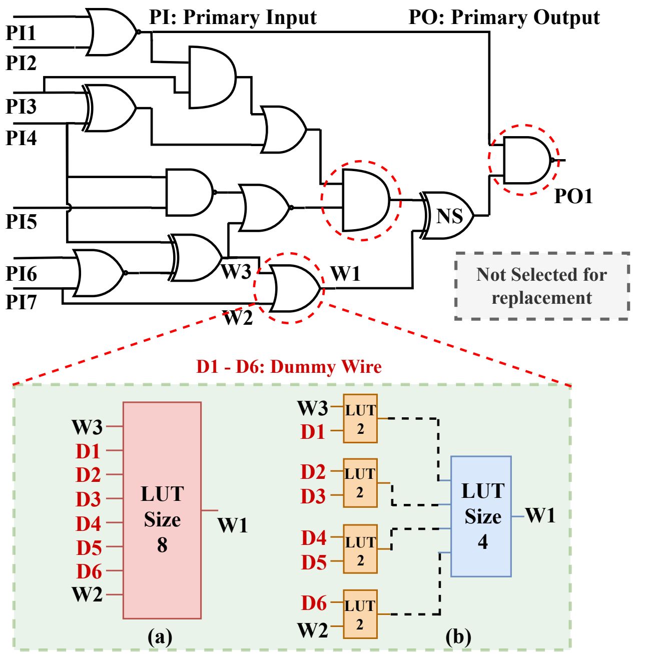

Fig. 6: Gate replacement using (a) traditional LUT of size 8, (b) customized LUT with LUT size 4 and 4: 2-input LUT, Marked gates are selected for obfuscation using Algorithm. LC_NoGen

A. ASIC Iterative Security-driven Design Flow

As NVM based spin-transfer torque (STT) MTJ-based LUTs have shown higher PPA efficiency [21], we consider STT-based LUT instead of SRAM based LUT for the purpose of obfuscation in this work, because 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 the CMOS fabrication process [21]. Additionally,it provides on-die re-configurability which enables to achieve high performance. 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. The proposed logic obfuscation needs gate level netlist. The result of the obfuscation is a new netlist in which identified gates for logic obfuscation are replaced with NV-LUTs. Since the NV-LUTs are firm macros that contain RTL code of a MUX, the netlist with NV-LUT inserted needs to be re-synthesized and optimized, given their impact on timing constraint. In order to obtain the best PPA results as well as SAT resiliency, we introduce an iterative-based design flow.

Figure 7 shows the proposed iterative security-driven ASIC design flow optimization. The logic library of the NV-latch is used in the re-synthesizing step and the multiplexer of the LUTs and the rest of the netlist is further re-optimized to meet

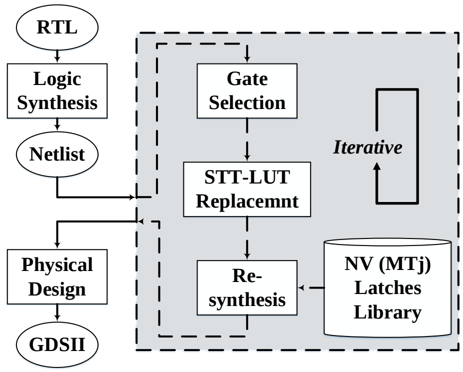

Fig. 7: Iterative-based Security-driven Design Flow with Maximizing PPA optimization with the best security solution.

the original timing. This involves executing the gate selection algorithm such that the obfuscated LUT is not the part of critical path. Therefore, 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[1]. Also, the iterative flow involves altering the size of the customized LUT and the number of gates to obfuscate.

As discussed earlier, leveraging large LUT size, adds an enormous amount of PPA overheads, however, its resiliency is far superior to that of the traditional LUT obfuscation. Since the security per PPA overhead of customized LUT rendered by leveraging LUTn is greater than security rendered by traditional LUT obfuscation with LUTn, we can afford to replace less number of gates with Large LUT sizes in customized LUT-based solution to create resiliency with permissible overheads. After meeting the timing and area/power constraints and security requirements, the physical design would be manufactured. The iterative flow largely benefits from Design-space-exploration and adds additional overhead, but guarantees optimal design configuration. While for our studied circuit benchmarks, less than 4 levels of iteration were found to be sufficient to meet security and design parameter constraints, for large circuits more iterations might be required, which leaves further scope for the research.

V. Experimental Evaluation

A. Experimental Setup

In order to demonstrate the efficacy of the proposed LUT-based obfuscation, 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. The jobs can be run 5 days without interruption, which is the reason time-out states have been set to 5 days. We employed benchmarks from different sources for evaluation, illustrated in Table I. The selected benchmarks comprises of the circuit that are representative of the real world IPs. For security evaluation, we employed recently proposed SAT attack i.e. SMT-attack [19], which is super-set of the SAT attack. It uses one or more theory solvers in addition to its internal SAT solver. For this reason, it is capable of modeling far more complex behaviors and could formulate much stronger attacks.

TABLE I: Benchmarks used for Experimentation.

Circuit	c2670	c7552	b12	AES	FIR	IIR	DES
#Gates	430	1296	2780	9511	14971	17054	25450
Source	ISCAS'85	ISCAS'85	ISCAS'89	OpenCores	CEP	CEP	OpenCores

We comprehensively measure and explore the SMT solver execution time by sweeping the size of large LUT in customized-LUT from 4 to 7 to demonstrate the impact of customized LUT for security design.

According to the security-driven PPA optimization, the benchmarks are first synthesized in Synopsys generic 28nm technology using Synopsys' Design Compiler. After the first synthesis and determining different solutions based on replacement strategy, an iterative-based re-synthesis finds the optimal solution that satisfies the timing and security constraints. Using the *ASIC Iterative Security-driven Design Flow*, the delay overhead can be eliminated for both traditional LUT-based obfuscation and proposed solution and hence only area and power overhead are discussed in fine granularity.

B. Security Analysis against the SAT-based Attack

Figure 8(a) shows the Design space exploration for benchmark C7552 by leveraging proposed obfuscation[2]. The most significant observation is that the resiliency provided by traditional obfuscation by using 14 LUTs with size 8 can be achieved using small LUT size and number. Replacing 2 gates with 7-LUT$_2$ preceded by LUT$_7$ and replacing 6 gates with 6-LUT$_2$ preceded by LUT$_6$ renders the obfuscation resiliency to create a time-out scenario. Hence, we can conclude that the resiliency provided by customized LUT with size 6 and 7 is greater than the traditional LUT with size 8. When using LUT size 4 and 5 for the proposed obfuscation, we need to obfuscate more gates to achieve time-out states. Hence, for proposed custom LUT-based obfuscation scaling up LUTs (increasing the size of LUTs used for obfuscation) is beneficial in rendering higher resiliency. On the other hand, customized LUT obfuscation using LUT 4 and 5 nevertheless, provides higher security per PPA overhead than leveraging obfuscation using traditional LUT of size 8. It is worth noting that both instances of obfuscation render SMT-attack timeout, thus resulting in the same level of resiliency, however, Figure 8(b) shows that obfuscating 14 gates with customized LUT using size 4 adds 3.58x overhead compared to 14.76x overhead added by traditional LUT. One may argue that the SAT results are bounded by 5 day execution time, however, it is evident that, customized LUT can yield higher security compared to traditional LUT at the cost of lower PPA overheads.

Also from Figure 8(b), it can be noted that for all the cases of customized LUT obfuscation, the normalized power and area overhead is smaller than that of the traditional obfuscation using LUT$_8$. We can also observe that leveraging LUT$_7$ for customized LUT occurs less PPA overhead than other LUT sizes. **Using a small number of large LUTs render maximum resiliency in case of proposed LUT solution**, as

[1] gray part in Figure 7 is iterative to obtain the best result.

[2] LUTm + n:LUT$_2$ represents Customized LUT where n LUTs of size 2 is preceded by LUT$_m$, where LUT$_m$ represents LUT of size m.

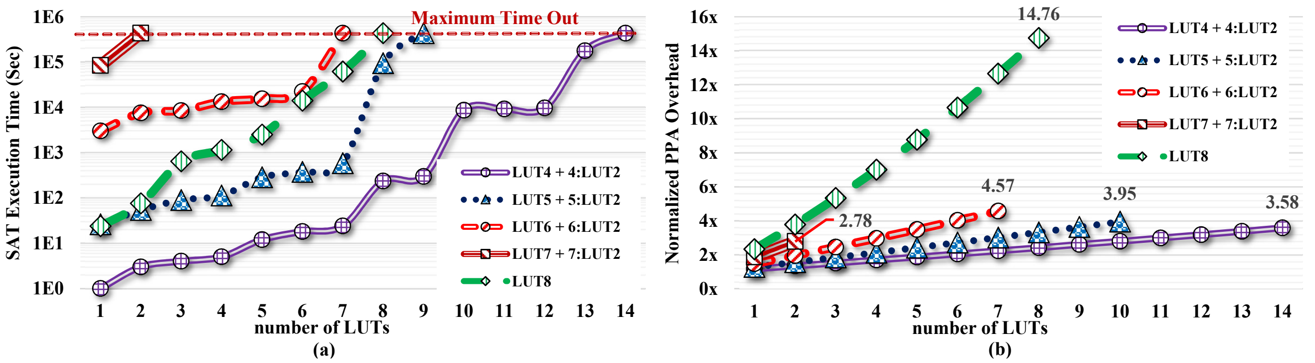

Fig. 8: Comparison of traditional LUT-based obfuscation with LUT_8 against customized LUT-based obfuscation. The size of LUT in customized LUT is varied from 4 to 7. Figure (a) shows the SAT execution time whereas Figure (b) shows the normalized PPA overhead, both as the function of LUT size and number of gates replaced. The aim of the experiment is to have a obfuscation configuration that maximizes SAT resiliency with minimal overheads.

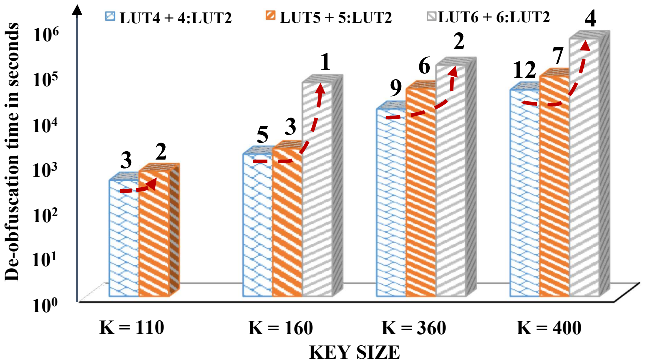

Fig. 9: SMT solver de-obfuscation time as a function of the size and the number of LUTs being used in customized LUT obfuscation. Number of LUTs to be replaced are constrained by the key size and PPA overhead. The Number of gates replaced is denoted by the number above the bar.

the SAT resiliency provided by each individual customized LUT is superior compared to traditional LUT.

Figure 9 also demonstrates a similar effect. The number of key bits is the number of fuses that are required to store the configuration bits of the LUT. The increased number of fuses results in increased overheads. Hence, by keeping the constraint on the key size imposes the constraint on the overhead. By adding the constraint on the key size, the size and number of LUTs that can be used for obfuscation is also constrained. We obfuscate ’AES’ benchmark while keeping the key lengths restricted to 110, 160, 360 and 400 bits respectively. The number of gates that were replaced is shown on the top of the bar. In each run of the SMT-solver, it can be observed that using less number of large LUT sizes for obfuscation, provided better resiliency than using small LUT sizes in large measures for obfuscation. Only by using 4 LUTs of size 6, we encounter the time-out, which is more than leveraging 12 LUTs of size 4 for obfuscation. This is due to the virtue of the stronger resiliency created by utilizing the large LUT sizes. While the key sizes are equal, the overhead added in each case are roughly equal, however, the security provided by large LUT size is evident. Therefore, it can be concluded that security grows at a higher rate than the added overhead, or the additional security comes with lower PPA overheads and the trade-off between security and design overhead is mitigated in the proposed LUT.

Figure 10 shows the Power and Area overhead for the different benchmarks using proposed LUT of size 7. The overheads are compared against the traditional obfuscation using LUT size 8, as we observe that from Figure 3 using LUT_8 for any replacement strategy rendered time-out instances of SMT-execution. The sufficient number of gates are encrypted such as to meet the time-out condition. It can be observed that the 8x and 2x average reductions in area and power overheads on average compared to traditional LUT-based obfuscation. The overheads are inclusive of all the wiring/routing overhead. One may argue that for circuits such as C7552 the incurred area overheads that are still high i.e. (2x), however the circuit size of the C7552 is small (only 1290 gates), therefore the overhead added by the LUTs is significant. However, with large and more practical circuits like AES, DES which are representative of real-world IPs, demonstrates that the overhead added by the proposed LUT-based obfuscation is small and justifiable ($\sim$1x) which renders this technique a practical solution for rendering SAT resilient designs. It is also important to note that the obfuscated circuit is normally a small part of a larger millions or billions gates design.

C. Resiliency to Other Attacks

The proposed customized LUT-based obfuscation is also resilient to the removal attacks. One cannot remove the LUT or MUX as removing them can strip the functionality of the circuit. The layout of the LUT is visually similar and nothing can be inferred by visual inspection. Though the Electron Microscopy (EM) can be applied for read-out data during run-time, the technology is currently not mature enough to reverse engineer the switching elements [24].

Acknoledgement

This work was funded by Defense Advanced Research Projects Agency (DARPA-AFRL, #FA8650-18-1-7819).

VI. Conclusions

In this work, we studied the logic locking by means of using reconfigurable logic, and performed comprehensive analysis of LUT-based obfuscation. Our experimental results show that the size of LUT is the most influential and straightforward factor in SAT resiliency, however, it introduces significant and unacceptable PPA overheads. To mitigate the

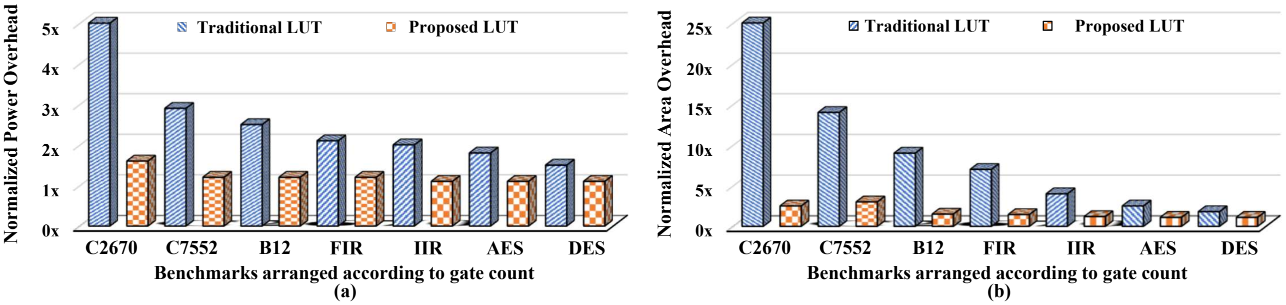

Fig. 10: Comparison of (a)Power, (b) Area overhead in Traditional LUT obfuscation with LUT size 8 and Proposed LUT Obfuscation with LUT size 7. The number of gates is chosen such that each of the obfuscated instance results in SMT execution time-out. The added overhead in case of Proposed LUT is less as each of the gate replaced using proposed LUT renders higher resiliency compared to that of the traditional LUT thus requiring few gates to be obfuscated resulting in lower PPA overheads.

overhead occurred due to the large LUT sizes, we introduce a customized LUT that breaks the trade-off between security and the design overheads. The introduced obfuscation using proposed custom LUT reduces the design overhead to an acceptable range without compromising the SAT resiliency. The obtained PPA results indicate that, for today's state-of-the-art attacks, our proposed obfuscation solution, that uses the novel and customized LUT introduced in this work, is an effective solution for meeting the security and resiliency requirements while keeping the PPA overhead of the design in check. We have illustrated that our solution resists both SMT-based and removal attacks. To further mitigate the overhead, further research can be focused on mapping more gates to the LUT and enhancing the dummy wire selection strategies.

References

[1] U.S.D. Of Defense. (2005) Defense Science Board Task Force on High Performance Microchip Supply. [Online]. Available: www.acq.osd.mil/dsb/reports/2000s/ADA435563.pdf,

[2] R. Karri *et al.*, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[3] M. Rostami *et al.*, "A Primer on Hardware Security: Models, Methods, and Metrics," *Proc. of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[4] Sai Manoj P D *et al.*, "Adversarial attack on microarchitectural events based malware detectors," in *Design Automation Conf.*, 2019.

[5] Brasser Ferdinand *et al.*, "Advances and throwbacks in hardware-assisted security: Special session," in *Int. Conf. on Compilers, Architecture and Synthesis for Embedded Systems*, 2018.

[6] J. Rajendran *et. al*, "Is Split Manufacturing Secure?" in *Design, Automation Test in Europe Conf. Exhibition (DATE)*, 2013, pp. 1259–1264.

[7] K. Z. Azar *et al.*, "COMA: Communication and Obfuscation Management Architecture," in *Int'l Symp. on Research in Attacks, Intrusions and Defenses (RAID)*, 2019, pp. 1–13.

[8] R. P. Cocchi *et al.*, "Circuit Camouflage Integration for Hardware IP Protection," in *Design Automation Conf. (DAC)*, 2014, pp. 1–5.

[9] J. A. Roy *et al.*, "Ending Piracy of Integrated Circuits," *Computer*, vol. 43, no. 10, pp. 30–38, 2010.

[10] M. Yasin *et al.*, "On Improving the Security of Logic Locking," vol. 35, no. 9, pp. 1411–1424, 2016.

[11] P. Subramanyan *et al.*, "Evaluating the Security of Logic Encryption Algorithms," in *Int'l Symp. on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.

[12] S. Roshanisefat and *et al.*, "Benchmarking the capabilities and limitations of SAT solvers in defeating obfuscation schemes," in *IEEE Int. Symp. on On-Line Testing And Robust System Design (IOLTS)*, 2018.

[13] K. Z. Azar *et al.*, "Threats on logic locking: A decade later," in *Great Lakes Symp. on VLSI (GLSVLSI)*, 2019, pp. 471–476.

[14] M. Yasin *et al.*, "SARLock: SAT Attack Resistant :Logic Locking," in *HInt'l Symp. on Hardware Oriented Security and Trust (HOST)*, 2016.

[15] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT Attack on Logic Locking," in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, 2019, pp. 199–207.

[16] M. Li *et al.*, "Provably Secure Camouflaging Strategy for IC Protection," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2018.

[17] H. M. Kamali *et al.*, "Full-Lock: Hard Distributions of SAT Instances for Obfuscating Circuits using Fully Configurable Logic and Routing Blocks," in *Proc. of Design Automation Conf. (DAC)*, 2019, p. 89.

[18] M. Yasin *et al.*, "Security analysis of Anti-SAT," in *Asia and South Pacific Design Automation Conf. (ASP-DAC)*, 2017, pp. 342–347.

[19] K. Z. Azar *et al.*, "SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond the SAT Attacks," *IACR Trans. on Cryptographic Hardware and Embedded Systems*, no. 1, pp. 97–122, 2019.

[20] A. Baumgarten *et al.*, "Preventing IC Piracy Using Reconfigurable Logic Barriers," *Design Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[21] T. Winograd *et al.*, "Hybrid STT-CMOS Designs for Rreverse-Engineering Prevention," in *Design Automation Conf. (DAC)*, 2016.

[22] B. Liu and B. Wang, "Embedded Reconfigurable Logic for ASIC Design Obfuscation against Supply Chain Attacks," in *Design, Automation Test in Europe Conf. Exhibition (DATE)*, 2014, pp. 1–6.

[23] A. Attaran *et al.*, "Static Design of Spin Transfer Torques Magnetic Look Up Tables for ASIC Designs," in *Great Lakes Symp. on VLSI (GLSVLSI)*, 2018, pp. 507–510.

[24] S. Patnaik *et al.*, "Advancing Hardware Security using Polymorphic and Stochastic Spin-Hall Effect Devices," in *Design, Automation Test in Europe Conf. Exhibition (DATE)*, 2018, pp. 97–102.

[25] A. Rezaei *et al.*, "Cyclic Locking and Memristor-based Obfuscation against CycSAT and Inside Foundry Attacks," in *Design, Automation Test in Europe Conf. Exhibition (DATE)*, 2018, pp. 85–90.

[26] J. Yang *et al.*, "Exploiting Spin-Orbit Torque Devices As Reconfigurable Logic for Circuit Obfuscation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 1, pp. 57–69, 2019.

[27] H. M. Kamali *et al.*, "LUT-Lock: A Novel LUT-Based Logic Obfuscation for FPGA-Bitstream and ASIC-Hardware Protection," in *IEEE Computer Society Annual Symp. on VLSI (ISVLSI)*, 2018, pp. 405–410.

[28] G. Kolhe *et al.*, "On custom lut-based obfuscation," in *Great Lakes Symp. on VLSI (GLSVLSI)*, 2019, pp. 477–482.

[29] J. Rajendran *et al.*, "Security Analysis of Logic Obfuscation," in *Design Automation Conf. (DAC)*, 2012, pp. 83–89.

[30] J. Rajendran *et al.*, "Fault Analysis-Based Logic Encryption," *IEEE Trans. on Computers*, vol. 64, no. 2, pp. 410–424, 2015.

[31] J. Rajendran *et al.*, "Security Analysis of Integrated Circuit Camouflaging," in *Proc. of ACM SIGSAC Conf. on Computer & Communications Security (CCS)*, 2013, pp. 709–720.

[32] M.Yasin *et al.*, "Provably-secure logic locking: From theory to practice," in *2017 ACM SIGSAC Conf. on Computer & Communications Security (CCS)*, 2017, pp. 1601–1618.

[33] K. Shamsi, M. Li, T. Meade, and et.al., "Cyclic obfuscation for creating sat-unresolvable circuits," in *Great Lakes Symposium on VLSI 2017*. USA: ACM, 2017, pp. 173–178. [Online]. Available: http://doi.acm.org/10.1145/3060403.3060458

[34] S. Roshanisefat *et al.*, "SRCLock: SAT-Resistant Cyclic Logic Locking for Protecting the Hardware," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2018, pp. 153–158.

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