On Custom LUT-based Obfuscation

Gaurav Kolhe, Sai Manoj P D, Setareh Rafatirad, Avesta Sasan, Houman Homayoun
George Mason University, Fairfax, VA, USA
{gkolhe,spudukot,srafatir,asasan,hhomayou}@gmu.edu

Hamid Mahmoodi
LogiMem Corporation
San Francisco, CA, USA
mahmoodi@logimem.com

ABSTRACT

Logic obfuscation yields hardware security against various threats, such as Intellectual Property (IP) piracy and reverse engineering. Evolving Boolean satisfiability (SAT) attacks have challenged the hardware security assurance rendered by various obfuscation methods. Recent works have centered on using re-configurable components such as Look-Up-Tables (LUTs) to enhance resiliency against SAT attacks. Resiliency against SAT attack is guaranteed when the size of LUT (number of inputs) is large. However, this incurs significant power, area and performance overheads. To address this challenge, this work proposes logic encryption based on customized LUT to make this practical. We propose two variants of the customized LUT based obfuscation: LUT+MUX based obfuscation, securing the design through routing obfuscation by MUX (multiplexer) and logic obfuscation of LUTs; and LUT+LUT based obfuscation, benefiting from LUT based obfuscation reinforced with additional logic/routing obfuscation. We evaluate the hardware security and overheads of the proposed two variants of customized LUT-based obfuscation on various benchmarks. Proposed customized LUT-based obfuscation breaks the security, power, and area trade-offs. The proposed solution is shown to be robust against SAT-attacks and power analysis-based side-channel attacks with 8× reduced area and 3× reduced power on an average compared to state-of-the-art LUT-based obfuscation.

KEYWORDS

Reverse Engineering, hardware obfuscation, logic encryption, STT-LUT, LUT obfuscation, SAT-attack.

ACM Reference Format:
Gaurav Kolhe, Sai Manoj P D, Setareh Rafatirad, Avesta Sasan, Houman Homayoun and Hamid Mahmoodi. 2019. On Custom LUT-based Obfuscation. In *Great Lakes Symposium on VLSI 2019 (GLSVLSI '19), May 9–11, 2019, Tysons Corner, VA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3299874.3319496

1 INTRODUCTION

To minimize the increasing fabrication costs and design complexity of Integrated Circuits (ICs), IC companies are becoming fabless [10] and have started employing third party Intellectual property modules (IPs) to meet the reduced time-to-market constraints and minimize the design flow efforts. Despite economic benefits, this trend poses significant challenges to hardware security assurance in various forms such as reverse engineering (RE) [4, 7, 15, 16].

In order to thwart these security threats, many hardware design-for-trust techniques have been introduced; split manufacturing [12], IC camouflaging [5], and logic locking [26] to name a few. Among multiple aforementioned techniques, logic locking can thwart the majority of the attacks at any phase in the IC Production chain [26]. This is because the logic locking requires keys to unlock the correct functionality of the design. In addition, as a part of the post-manufacturing process, the activation of the IC (providing correct keys) will be accomplished in a trusted regime after fabrication to hide the functionality from the untrusted foundry, and attackers. Having key-programmable gates allow the designer or user to control the functionality using these key inputs.

Although introducing logic locking schemes have enhanced the security of the design, through advent of advanced Boolean satisfiability (SAT) based attack [2, 19] also known as "oracle-guided" threat model shows that by applying a few stimuli to the design and monitoring the output, the key value and functionality of an IC could be extracted in the order of a few minutes.

Numerous logic locking techniques [22, 23] have been proposed to improve the resiliency against SAT-attack. However, many of these techniques are vulnerable to other RE practises such as removal and SPS (Signal Probability Skew) attacks [24, 25]. Among the proposed logic locking schemes, some recent obfuscation methods focus on using configurable barriers, such as Look-Up-Tables (LUTs), as a key-programmable logic to achieve resiliency against attacks like SAT [1, 8]. Despite the achieved resiliency, the incurred area and power overheads with such LUT-based obfuscation acclimatizes the feasibility of implementation. Even employing emerging Spin-Transfer-Torque (STT) based LUTs that provides much higher density, performance, and near zero leakage power compared to CMOS-based LUTs, does not address the area and power overhead problem [20].

In this work, first, we study the previously proposed LUT-based obfuscation schemes [1, 3, 8, 20], and discuss the challenges to make them resilient against SAT-attack. To make LUT-based obfuscation a practical solution, we need to 1) radically reduce the design overhead, and 2) do not compromise the security benefits. Unfortunately, these two goals are contradictory with general LUT-based obfuscation. To address the design overhead, we propose a customized LUT design that not only benefits from configurable barriers for obfuscation but also mitigates the incurred area and power overheads of previous works. Two different variants of the proposed customized LUT is presented in this work for security: LUT+MUX based obfuscation, and LUT+LUT based obfuscation that benefits from re-configurability of the LUT along with the routing obfuscation supplemented by additional MUX/LUT. This

GLSVLSI '19, May 9–11, 2019, Tysons Corner, VA, USA

ACM ISBN 978-1-4503-6252-8/19/05…$15.00
https://doi.org/10.1145/3299874.3319496

blend of two obfuscation techniques assist in elevating security provided by logic obfuscation while concurrently creating SAT-hard instances.

The proposed customized LUT-based obfuscation is rigorously tested for various metrics such as power, performance, area. The contributions of this work are outlined as follows:

- To address the design overhead of LUT-based obfuscation, we propose a customized LUT-based obfuscation, where LUT is the crucial element in providing security, as replacing a gate with LUT realizes 2^{2^n} logical function where n is the LUT size.
- Two variants of customized LUT are proposed: LUT+MUX and LUT+LUT. LUT+MUX enhances the security through the confluence of routing obfuscation (by MUX) and logic obfuscation (by LUT). Similarly, LUT+LUT fortifies the logic obfuscation by adding an extra layer of routing/logic obfuscation.
- The proposed customized LUT breaks the design parameters and security trade-offs and is optimized for power, performance, area, and security simultaneously, whereas state-of-the-art works are optimized for either security or power, performance, and area.

2 BACKGROUND AND RELATED WORK

2.1 Logic Obfuscation

Logic obfuscation referred as logic locking [26] is a hardware security solution that facilitates to hide the IP using key-programmable logic gates to combat against RE and other hardware threats. The activation of the obfuscated IP must be accomplished in a trusted regime before releasing the product into the market. During the activation phase, the correct key is applied to these key-programmable gates to recover the correct functionality of the IC/IP. In addition, the correct key will be stored in a tamper-proof memory.

2.2 SAT-Attack and SAT-resilient Obfuscation

Although logic locking schemes try to minimize the probability of determining the correct key by the attacker, introducing SAT-attack shows that these schemes can be broken as fast as in a few minutes [19]. To perform a SAT-attack, attacker requires an access to the functional IC along with the extracted obfuscated netlist. By searching for the Distinguishing Input Patterns (DIP) which is applied as the input to the obfuscated circuit, produces different outputs (Y_i) when different key values are applied. This DIP is used to distinguish between the set of correct and incorrect keys. The number of DIPs discovered during the SAT-based attack is equal to the number of iterations needed to unlock the obfuscated design.

Different SAT-hard schemes such as [22, 23] are proposed to curb the SAT-attack. The motive of these defenses is to increase the number of DIPs required to prune the correct key. These type of defenses are mostly point function, where the number of iterations needed can be increased up to 2^{n-1}. Hence, the queries that SAT-attack must make and apply grows exponentially as the size of the key (n) increases. To overcome such obfuscation variants, SAT-attack such as AppSAT and Double-DIP [17, 18] are proposed. Furthermore, new obfuscation schemes that focus on non-Boolean behavior of circuits [21], that are not convertible to a SAT circuit are proposed for SAT resilience. Some of such defenses include adding cycles into the design [14]. Adding cycles into the design may cause SAT-attack to get stuck in the infinite loop, however, advanced SAT-based attacks such as cycSAT [27], SMT-attack [2] can extract the correct key despite employing such defenses.

Recently proposed obfuscation schemes relying on the concept of polymorphism [11, 13] facilitate to configure the Boolean function at run-time. However, polymorphic Memristor-CMOS gate based implementations are limited to only 2 logical functions, while giant spin-Hall effect (GSHE) is reconfigurable to 16 Boolean functions. While converting the obfuscated circuit to SAT representation using [11, 13], one can replace obfuscated gates using MUX which lets us choose between different functions implemented by GSHE and polymorphic Memristor-CMOS devices. Our simulations have shown that the above obfuscation schemes [11, 13] on C7552 circuit with as much as 50% obfuscation coverage, the reverse engineering (RE) time using SAT solver is found to be less than 5 days.

2.3 Obfuscation Using Look-Up-Tables

Look-up-tables (LUTs) built using STT can be used as a means of obfuscation. When a gate is replaced with a LUT, only the IP holder knows the function implemented by the LUT. As the content of the LUT is stored in tamper-proof memory, unauthorized users do not have access to the configuration of the LUT and cannot RE the IP. With the increase in LUT size, the functionality of the obfuscated logic can become complex, leading to increased robustness against SAT and other attacks. In this type of obfuscation, the partial part of the circuit is mapped to the LUT.

There has been some prevalent research using LUTs for logic obfuscation. The work in [3] proposes obfuscation and discusses multiple replacement strategies to secure the design, but falls short in providing the resiliency against SAT-based attacks. Three different LUT replacement policies are discussed in [20], which suggests that using STT based LUT could reduce the Power, Performance, and Area (PPA) overhead. Despite depicting low overheads, similar to the aforementioned works [3], the work in [20] also does not guarantee resiliency against state-of-the-art SAT-attack. The primitive in [8] has shown robustness against SAT-attack but does not evaluate the PPA overheads, which are equally critical in hardware design.

One of the most straightforward approaches in LUT-based obfuscation is to increase the number of LUTs. However, this does not guarantee the best security promises. It is observed that using larger LUTs enhance the security by an exponential margin because, the SAT-attack replaces the LUTs with deeper MUX trees, and the time to find the correct key grows exponentially. An example of obfuscation is demonstrated in Figure 1. The gates to be replaced are identified and are further replaced by LUT of size 8. However, this adds large power and area overheads.

Another solution to circumvent utilizing large LUTs is the choice of replacement policies. However, replacement policies alone do not solve the area and power overhead challenges, as these policies cannot reduce the size of LUT beyond a certain limit. For example, we observe that for a circuit like C7552, security can be achieved against SAT-based attack by replacing 1% of the gates in the random fashion with LUT size 12. However, when we use better replacement policy such as Algorithm 1, even with a LUT size of 8, a similar level of resiliency against SAT-attack can be achieved. Nevertheless,

the PPA incurred by the relatively lower LUT size 8 is still very large and unacceptable. We observe that for circuit C7552, when 1% of the gates are obfuscated with LUT of size 8 using improved replacement strategy, the power and area overhead are 7× and 38× higher compared to original C7552 un-obfuscated circuit.

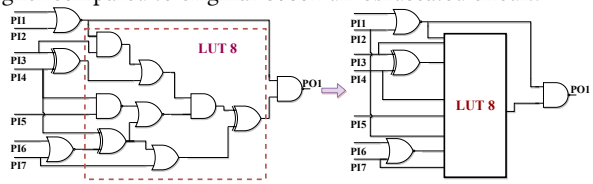

Figure 1: LUT Obfuscation using LUT of Size 8.

3 PROPOSED CUSTOMIZED LOGIC OBFUSCATION

In this section, we present the proposed customized LUT which offers secure obfuscation along with a significant reduction in the area and power overheads. We form the rationale for using STT technology for proposed LUT obfuscation followed by the replacement policies. This section further delves into the implementation of customized LUT Variant using STT-LUT along with additional MUX and LUT. We propose LUT+MUX based obfuscation for CMOS compatible devices benefiting using the fewer non-volatile (NV) elements while LUT+LUT can be used for emerging technology.

3.1 STT-LUT based Obfuscation

For obfuscation, we rely on STT-LUTs because of the following reasons. The employed design of STT-LUT is similar to that in [20].

- The configuration bits are stored in the Magnetic Tunnel Junction (MTJ), which are inserted between metal layers. The stored bits are highly susceptible and will be lost during RE, thus making it nearly impossible for the attacker to retrieve the configuration bits and determine the functionality of the LUT.
- STT based LUT provides higher integration density than SRAMs along with high endurance and retention time.
- MTJs hav near-zero leakage, and have soft error resilience.
- STT-LUT is highly integrative in CMOS fabrication process. The work in [20] has shown the feasibility to create the full-custom design and optimize one-bit MTJ latch cell using the Synopsys generic 28nm process for fabrication purposes.

Compared to static CMOS design, the timing overhead can be mitigated when employing the LUT-based obfuscation using STT-LUTs. As the ASIC standard cells are static in nature, the STT-LUT has implemented in a way that the path from LUT inputs to the LUT output is a MUX. This allows the designers to implement such type of STT-LUTs to be written in synthesizable RTL code for Electronic Design Automation (EDA) and optimization.

3.2 Gate Replacement Policy

In this work, we have used a replacement policy that is discussed in Algorithm 1 to reduce the size of the LUT needed for obfuscation. However other replacement policies can also be applied [8]. We start the Design Space Exploration (DSE) by obfuscating 1% of the gates with varying LUT size (from 2 to 15) to produce desired attack resiliency. The gate replacement algorithm used to find the gates for replacement is formulated in the Algorithm 1.

Algorithm 1 Gate Replacement Policy

```
Remove the Sequential Element                          ▷ PO: Primary Outputs
Convert the Netlist to Graph Representation
for each (PO in Output_cone) do
    for each (gate in PO) do
        tag_key(PO)
        gate_obfuscation_gate_count += 1
Sort (Output Cone, gate_obfuscation_count, descending)
find Combination of PO where (gate_obfuscation_count > target_number)
while (obfuscated_gate_count < targated_number) do
    for each gate in Combination_Output_Cone do
        if parent_gate not replaced by LUT then
            Replace_LUT (gate, LUT_size)
    obfuscated_gate_count += 1
s = SAT_Attack (obfuscated_netlist)
PPA = DC_Compiler (obfuscated_netlist)
Normalize (S, PPA)
Calculate (S/PPA)
```

In this gate replacement policy, we choose gates such that the output corruptibility is low. When the output corruptibility is high i.e., the probability of having the hamming distance between the outputs of locked and activated IC greater than 1, it enables the SAT-attack to discover conflict clauses easily, which in turn enables SAT solver to eliminate the incorrect keys. On the other hand, when the output corruptibility is low, SAT solver requires more iterations to learn the underlying clauses for assigning a satisfying assignment. This increases the complexity by a significant margin. Along with this replacement policy, the gates are replaced such that LUTs are never connected to each other directly in back-to-back manner. Avoiding connecting the LUTs in this manner ensures only one single valid key for the circuit [8]. The number of output cones being selected for obfuscation depends on the number of gates that are being targeted for the obfuscation. To summarize, this algorithm finds the combination of output cone such that maximum number of gates are selected for obfuscation while less number of output cones are affected. If the resiliency against SAT-attack [19] can't be achieved with the selected number of gates even at higher LUT sizes, the obfuscated gate count are further increased. The gates to be replaced are again obtained using Algorithm 1. The selected gates are then replaced with, different sizes of the LUTs.

By iteratively sweeping the LUT size and gate count, we obtain the power, area and delay overheads along with the security metric. By calculating the S/PPA metric (Security per unit Power, Performance, and Area overhead) for each run, we find the optimal configuration that achieves the maximum robustness against SAT and other attacks with lower overheads. A large value of S/PPA indicates higher security assurance with lower PPA overheads. For further mitigating the overheads, We propose two variants of the customized LUT-based obfuscation..

3.3 Customized LUT Variant 1: LUT with MUXes

In, a SAT solver the clause to variable ratio (CVR) is one of the important metrics for qualitatively evaluating the security. For instance, to be proven as SAT-hard, the CVR needs to be around 4.2

[9]. 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. Hence, one needs to determine a trade-off between the clause and variables.

In this work, we have chosen MUX to be integrated with LUTs for enhanced security. The rationale can be explained as follows: MUX has 4 clauses thereby using the MUX for obfuscation assists in averaging clause/variable ratio in the Conjunctive Normal Form (CNF) to 4. Furthermore, symmetric problems pose a great difficulty to SAT solver in finding a solution [6]. By effectively using the LUTs with MUX, one can formulate symmetry, to increase the SAT-hardness. In this work, we limit the size of the MUX to 2 for ensuring minimum PPA overheads. By employing the LUT + MUX configuration, the design benefits from both LUT and Routing obfuscation. As the LUT is a fully reconfigurable logical block, adding MUX tree increases the number of ways LUT can be configured, which significantly increase the key search space. Further, in SAT, the LUTs are represented using MUX, and hence combining MUX with LUT of size n, increases the depth of the MUX tree, thereby creating even harder SAT-instances.

Once the optimal LUT size and gates for replacement are determined, we replace the LUT using the proposed customized LUT +MUX to address the power and area overhead challenges. For example, it is observed that by using LUT size 8, the design can be made resilient to the SAT-attack however, leveraging LUT size 8 for obfuscation, the design suffers from the hefty design overheads. We address the PPA overhead issue by replacing the LUT size of 8 with smaller LUT of size 4 with an additional 4-2:1 MUX tree at the input of LUT size 4. Figure 2 demonstrate the gate replacement using LUT+MUX based obfuscation. While trading the size of the LUT, the size of the MUX tree has to be determined. We therefore iteratively reduce the LUT size while increasing the number of MUX to find the configuration which has a higher S/PPA.

Figure 2 shows the obfuscation using proposed LUT + MUX design. Gates are selected for replacement using Algorithm 1 and the gates are replaced by LUT Size of 4. In addition, 2:1 MUXes are added to configure 4-input LUTs in $2^4 \times 2^{2^4}$ ways. By utilizing this type of configuration, one can ensure high resiliency against SAT-attack while using a LUT size of 4 and lower PPA overheads. The additional dummy wires that create the confusion can be routed anywhere from the circuit while making sure we do not introduce cycles.

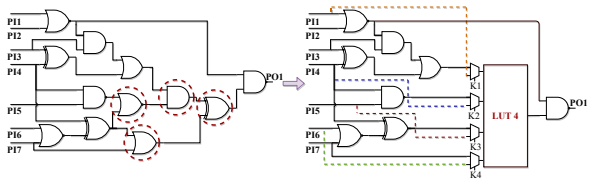

Figure 2: Customized LUT obfuscation using 4 input LUT and 4-2:1 MUXes.

Perhaps one can use switch-box instead of MUXes for creating routing obfuscation. However, attacker can eliminate a few permutations formed by m×n switch-box, thus reducing the search space. For instance, by creating a 4×4 switch-box at the input of the LUT, the switch-box can be configured in 16 different ways, however, the attacker can eliminate this switch-box and directly feed these 4 wires to the LUT and still identify the true functionality.

3.4 Customized LUT Variant 2: LUT + LUT

Another variant of proposed customized LUT for obfuscation is LUT cascaded with LUT. In this arrangement, to increase the complexity and security promises, we propose to replace the gates that reside at the input of the LUT with a smaller LUTs, as shown in Figure 3 (a). The main motive behind this is to increase the possible ways in which the larger LUT can be configured. The newly added smaller LUTs can take the form of a MUX or some arbitrary logical function. In order to translate LUT+MUX strategy to LUT+LUT, one needs to replace each of the 2:1 MUXes with a LUT of size 2. Note that it is stated that back-to-back implementation of the LUTs increases the number of correct combination [8], therefore we refrain from replacing two back-to-back gates with LUT+LUT or even LUT+MUX.

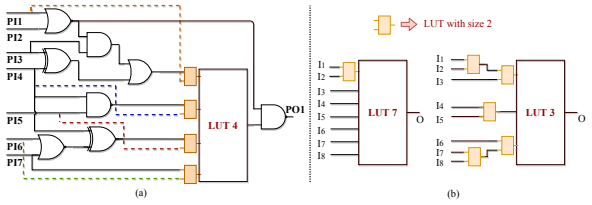

Figure 3: (a) Customized LUT Obfuscation using LUT Size 4 and LUT Size 2. (b) Different LUT+LUT configuration

Since, the LUT can take 2^{2^n} possible forms, and by adding the extra m-LUT, the total number of possible combinations becomes $m \times 2^{2^2} \times 2^{2^n}$ with the size of smaller LUT fixed to 2. This is chosen to keep the overhead margin as low as possible. The Figure 3 (b) shows the different configuration of the LUT+LUT that can replace the LUT Size 8 with the proposed method.

4 EXPERIMENTAL EVALUATION

4.1 Experimental Setup

To demonstrate and test the capabilities of the proposed customized LUTs, we are using a cluster computing environment which consists of 50 nodes, totaling 1060 cores and over 3TB of memory. We allocated a maximum of 128GB to each run of the SAT solvers. It is assumed that an attacker (fab or end-user) has no access to tamper-proof memory, however, all other flip-flops in the design can be accessed using the scan-chain that are implemented for the design-for-test. Uninterruptedly the experiments are run with a time-out of 10 days, which is higher than the time-out used in recent SAT-hard works. As the SAT solvers does not have the support for the LUTs, we construct the LUTs using the 2:1 MUXes [8].

The benchmarks that are employed for validating the proposed customized LUT-based obfuscation and its details are presented in Table 1 along with the gate count. The evaluated benchmarks comprise of 2 combinational and 5 sequential circuits.

As discussed, the proposed customized-LUT employs STT-LUTs for obfuscation to alleviate the overheads. To obfuscate with STT-LUTs, we define the STT based LUT as a Verilog module, and identified gates in the netlist are replaced by these modules. The logic library of NV latch is used to re-synthesize the design putting do-not-touch on the STT-LUT gate, to prevent them from being replaced during the subsequent rounds of synthesis.

DSE is used to find the best S/PPA ratio by varying LUT Size and gate count to obfuscate. For instance, as shown in Table 2, the LUT size 8 achieves the highest S/PPA for C7552, hence LUT size 8 is chosen as an optimal size for obfuscating C7552 circuit. For the purpose of experimentation, the constraint on the obfuscation is set such that, the design should remains resilient against the SAT-attack for 10 consecutive days. Using LUT size 8, we can achieve the resiliency for 10 days, and this is the reason, we refrain from using the LUT size 9. Further to reduce hefty PPA overheads, we have replaced the 8 input LUT with the proposed LUT+MUX techniques, so that it matches the overall input size of 8 with LUT of size 4 and 4 2:1 MUXes. In case of the LUT+LUT technique, LUT size 8 is replaced with LUT size 3 with 5 2-input LUTs as shown in Figure 3(b). The same LUT configuration is followed in all experiments.

Table 1: Benchmark Summary

Benchmark	Synthesized Gate Count	Selected Gate Count	Source
C2670	430	12	ISCAS'85
C7552	1296	15	ISCAS'85
B12	2780	28	ISCAS'89
AES	9511	50	OpenCores
FIR	14971	50	CEP
IIR	17054	50	CEP
DES	25450	50	OpenCores

4.2 Security Analysis Against SAT-Attack

The execution time of the SAT solver with traditional LUT-based obfuscation for C7552 benchmark is presented in Table 2. For traditional obfuscation, the LUT size is varied. The final LUT size is fixed to 8, as it achieves the best S/PPA metric. Using the proposed obfuscation technique, we can replace LUT 8 with smaller LUT size. It can be seen from Table 3, that LUT+MUX with LUT size of 4 and LUT+LUT using LUT size 3 makes it impossible to decode the key in the provided time-out of 10 days, but with traditional LUT obfuscation it can be done in the range of few seconds to minutes. Due to leveraging lower sizes of LUT for obfuscation, the proposed LUT-based obfuscation incurs 8× lower area with 3× lower power compared to traditional obfuscation. The ∞ denotes time-out i.e., SAT solver was unable to find the keys within 10 days. Customized obfuscation using LUT+LUT relies on lower LUT sizes, thus further minimizing the PPA overheads. Figure 3 also shows deobfuscation time for various configuration of Size of LUT + number of (2 input MUX/LUT).

Table 2: Impact of LUT Size on S/PPA metric

LUT Size	8	7	6	5	4	3	2
S/PPA	34.57	0.002	0.004	0.007	0.0017	0.02445	0

There have been several methods proposed for logic obfuscation in the recent year. The most recent and prominent one, which is closesr to the proposed obfuscation is GSHE [11]. Therefore, in this paper we are comparing our proposed custom LUT-based obfuscation with two other methods: 1) standard LUT-based and 2) GSHE. Table 4 demonstrates the empirical results obtained using the proposed LUT-based obfuscation (with two variants) and the traditional LUT-based obfuscation using LUT Size of 8. Table 4 also shows the de-obfuscation time using the method proposed in [11] while obfuscating 10% of the design. As [11] did not report the timing and power, we only compare the SAT execution time.

Table 3: SAT-attack execution time using different LUT sizes on C7552 circuit

Traditional LUT-Obfuscation	LUT Size used for Obfuscation	8	7	6	5	4	3	2
	SAT Execution time for LUT Obf.	∞	612.3	363.2	224.8	121.7	87.3167	0.598
Proposed Custom LUT-Based Obfuscation	SAT Execution time for LUT+MUX Obf.	-	∞	∞	∞	∞	14549	7.85
	SAT Execution time for LUT+LUT Obf.	-	∞	∞	∞	∞	∞	2642.8
	Customized LUT Configuration	-	7+1	6+2	5+3	4+4	3+5	2+6

From Table 4, one can observe nearly 8× and 3× reduction in area and power overheads on average compared to traditional LUT-based obfuscation. It is also evident that LUT+LUT further mitigates the PPA overhead while utilizing lower LUT Sizes. In terms of resilience to the SAT-attack, one can observe that the proposed customized LUT-based solution secures the IC against SAT-attack with the time-out of 10 days (represented as ∞), whereas with the traditional LUT-based obfuscation, the IC can be reverse engineered in the range of seconds to a few minutes. The work in [11] despite having 10% obfuscation fails to demonstrate the resiliency against the SAT-solver. With the proposed LUT+MUX and LUT+LUT based obfuscation, even small circuits such as "C2670" can be hardened against SAT-attack. As the circuit size is small (only 430 gates), the overhead added by LUTs is significant. However, with larger and more practical circuits like AES, DES it can be observed that the overhead added by the proposed technique is small and justifiable 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 large millions or billions gates design. Therefore the extra 1∼3× overhead on a small 10K∼1M gates design results in a very small overall design overhead. While both variants of proposed customized LUT achieve similar SAT-hardness, LUT+LUT yields lower overhead while LUT+MUX requires less number of expensive NV memory elements. To secure the design for longer time-span, the designer can utilize the DSE to find the optimized LUT size and number of gates to be obfuscated to achieve the resiliency. Further the bigger LUT size can be replaced with the proposed methods to relieve the hefty overheads.

4.3 Side Channel Attack Vulnerabilities and Prevention

The secret key in this technique is the configuration bits that are stored in the non-volatile latches (NV-latches). The NV-latches are idle during run-time and their power consumption is leakage only and is independent of their states due to their differential design. Hence, the latches themselves do not directly generate any side channel signature such as the power or temperature map of the chip. However, the power of the MUX of the customized LUT and its fan-out cone depends on the LUT content, which could be a source of vulnerability for side-channel attack. If the attacker is given an option to reprogram the LUTs, an attacker may attempt to configure the LUTs with different keys and compare the power of the chip at each configuration with the power of a reference chip that has the

Table 4: The impact of Different methodology on de-obfuscation time (in seconds) along with power and area overhead for different benchmarks

Benchmarks	De-Obfuscation time using SAT-attack				Impact on Area Overhead*			Impact on Power Overhead*		
	GSHE Obf. Coverage = 10%	LUT Obf.	Proposed Obf		LUT Obf.	Proposed Obf		LUT Obf.	Proposed Obf	
			LUT + MUX	LUT + LUT		LUT + MUX	LUT + LUT		LUT + MUX	LUT + LUT
C2670	19.8	79.8	∞	∞	95.06x	7.51x	6.80x	15.14x	2.28x	1.96x
C7552	29.6	121.7	∞	∞	38.63x	3.65x	3.21x	6.39x	1.48x	1.14x
B12	31.8	150.7	∞	∞	27.56x	2.81x	2.04x	4.48x	1.41x	1.09x
FIR	745.5	28342.6	∞	∞	9.42x	1.63x	2.06x	2.62x	1.31x	1.19x
IIR	654.6	95838.4	∞	∞	8.21x	1.57x	1.36x	2.32x	1.22x	1.18x
AES	950.6	98637.5	∞	∞	11.63x	1.89x	1.59x	3.45x	1.76x	1.65x
DES	1438.7	154429	∞	∞	7.50x	1.51x	1.29x	1.90x	1.10x	1.05x

For LUT obfuscation, the size of the LUT used is 8, while proposed method (1) LUT + MUX use LUT Size 4 along with 4-MUX and (2) LUT + LUT uses LUT Size 3 with 5-LUT of Size 2 for replacing LUT Size 8. The number of gates obfuscated is reported in Table 1. However, for GSHE obfuscation 10% of gates are encrypted in random fashion. The SAT-deobfuscation time is given in seconds. * Figures represent the area and power overheads in comparison to the implementation of the circuit/design with no obfuscation.

correct configuration already programmed in. The proposed customized LUT-based solution increases the number of permutations to 2^{2^n}, where n is the size of LUT. As the side-channel attacks are compute-intensive and require precise measurements, one can argue that by achieving SAT-attack resilience, sufficient side-channel attack resilience can also be achieved. Nonetheless, there are solutions possible for integrating additional security mechanisms in place to make the aforementioned side-channel attack more difficult. The simplest option is to apply encryption to the configuration bits so that the bits given to the on-chip programmer get decrypted before being written into the LUTs. The encryption/decryption key will be generated by an on-chip PUF and is unique to each chip so it cannot be guessed or attacked. Such solutions will be integrated with the proposed customized-LUT as a part of future research.

4.4 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 switching elements [11].

5 CONCLUSIONS

In this work, we have introduced obfuscation using Customized LUT to mitigate the design overhead while not compromising the security. Further, to enhance the resilience against SAT-attack and make it a practical solution, we propose to employ LUTs in conjunction with MUX that facilitates routing based obfuscation or using an extra layer of LUT for increased depth of MUX for SAT-attack, therefore, making it resilient to power analysis-based side-channel attacks and EM-based reverse engineering attacks. Our experimental results show that with the proposed customized LUT-based obfuscation, higher robustness can be achieved against SAT-attack. Furthermore, nearly 3× power and 8× area overheads can be reduced on an average, compared to the state-of-the-art defenses. LUT+MUX offer competitive resiliency compared to LUT+LUT based obfuscation while using fewer NV-elements, while LUT+LUT technique benefits the IP holder from lower design overhead perspective.

REFERENCES

[1] A. Attaran, Tyler D. Sheaves, and P. Kumar et.al. Mugula. 2018. Static Design of Spin Transfer Torques Magnetic Look Up Tables for ASIC Designs. In *GLSVLSI*.
[2] Kimia Azar, Hadi Kamali, Houman Homayoun, and Avesta Sasan. 2019. SMT Attack: Next Generation Attack on Obfuscated Circuits with Capabilities and Performance Beyond the SAT Attacks. In *Conf. on CHES*.
[3] Alex Baumgarten, Akhilesh Tyagi, and J Zambreno. 2010. Preventing IC piracy using reconfigurable logic barriers. *IEEE Design & Test of Computers* (2010).
[4] Ferdinand Brasser and et. al. 2018. Hardware-Assisted Security: Understanding Security Vulnerabilities and Emerging Attacks for Better Defenses. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*.
[5] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang. 2014. Circuit camouflage integration for hardware IP protection. In *ACM/IEEE Design Automation Conf.*
[6] J. Crawford, M. Ginsberg, and et.al. Luks. 1996. Symmetry-breaking Predicates for Search Problems. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*.
[7] Ramesh Karri, Jeyavijayan Rajendran, Kurt Rosenfeld, and Mohammad Tehranipoor. 2010. Trustworthy hardware: Identifying and classifying hardware trojans. *Trans. Computer* (2010).
[8] H. Mardani, K. Zamiri, and et.al. Gaj, K. 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)*.
[9] Eugene N., Alex D., and Yoav S. et.al. [n. d.]. Understanding Random SAT: Beyond the Clauses-to-Variables Ratio. In *In Proc. of CP-04*.
[10] Department of Defense. 2005. Defense science board task force on high performance microchip supply. (2005). https://www.acq.osd.mil/dsb/reports/2000s/ADA435563.pdf
[11] S. Patnaik, N. Rangarajan, and J. Knechtel et.al. 2018. Advancing hardware security using polymorphic and stochastic spin-hall effect devices. In *Design, Automation Test in Europe Conference Exhibition (DATE)*.
[12] J. Rajendran, O. Sinanoglu, and R. Karri. 2013. Is split manufacturing secure?. In *Design, Automation Test in Europe Conference Exhibition (DATE)*.
[13] A. Rezaei, Y. Shen, S. Kong, and J. Gu et.al. 2018. Cyclic locking and memristor-based obfuscation against CycSAT and inside foundry attacks. In *Design, Automation Test in Europe Conference Exhibition (DATE)*.
[14] S. Roshanisefat, H. Mardani, and A. Sasan. 2018. SRCLock: SAT-Resistant Cyclic Logic Locking for Protecting the Hardware. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*.
[15] Masoud Rostami, Farinaz Koushanfar, and Ramesh Karri. 2014. A primer on hardware security: Models, methods, and metrics. *Proc. IEEE* (2014).
[16] Hossein Sayadi, Nisarg Patel, and Sai Manoj et.al. 2018. Ensemble Learning for Effective Run-time Hardware-based Malware Detection: A Comprehensive Analysis and Classification. In *Design Automation Conference*.
[17] K. Shamsi, M. Li, and T. Meade et.al. 2017. AppSAT: Approximately deobfuscating integrated circuits. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*.
[18] Y. Shen and H. Zhou. 2017. Double DIP: Re-Evaluating Security of Logic Encryption Algorithms. In *Proceedings of the on Great Lakes Symposium on VLSI*.
[19] P. Subramanyan, S. Ray, and S. Malik. 2015. Evaluating the security of logic encryption algorithms. In *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*.
[20] T. Winograd, H. Salmani, and H. et.al. Mahmoodi. 2016. Hybrid STT-CMOS designs for reverse-engineering prevention. In *Proceedings of the 53rd Annual Design Automation Conference*.
[21] Y. Xie and A. Srivastava. 2017. Delay locking: Security enhancement of logic locking against IC counterfeiting and overproduction. In *ACM/EDAC/IEEE Design Automation Conference (DAC)*.
[22] Y. Xie and A. Srivastava. 2018. Anti-SAT: Mitigating SAT Attack on Logic Locking. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* (2018).
[23] M. Yasin, B. Mazumdar, and J. J. V. Rajendran et.al. 2016. SARLock: SAT attack resistant logic locking. In *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*.
[24] M. Yasin, B. Mazumdar, and O. Sinanoglu et.al. 2017. Security analysis of Anti-SAT. In *Asia and South Pacific Design Automation Conference (ASP-DAC)*.
[25] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran. 2018. Removal Attacks on Logic Locking and Camouflaging Techniques. *IEEE Transactions on Emerging Topics in Computing* (2018).
[26] M. Yasin, J. J. Rajendran, and et.al. O. Sinanoglu. 2016. On Improving the Security of Logic Locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2016).
[27] H. Zhou, R. Jiang, and S. Kong. 2017. CycSAT: SAT-based attack on cyclic logic encryptions. In *IEEE/ACM International Conference on Computer-Aided Design*.