Breaking the Design and Security Trade-off of Look-up-table–based Obfuscation

GAURAV KOLHE and TYLER DAVID SHEAVES, University of California, Davis, USA
SAI MANOJ P. D., George Mason University, USA
HAMID MAHMOODI, San Francisco State University, USA
SETAREH RAFATIRAD, AVESTA SASAN, and HOUMAN HOMAYOUN, University of California, Davis, USA

Logic locking and Integrated Circuit (IC) camouflaging are the most prevalent protection schemes that can thwart most hardware security threats. However, the state-of-the-art attacks, including Boolean Satisfiability (SAT) and approximation-based attacks, question the efficacy of the existing defense schemes. Recent obfuscation schemes have employed reconfigurable logic to secure designs against various hardware security threats. However, they have focused on specific design elements such as SAT hardness. Despite meeting the focused criterion such as security, obfuscation incurs additional overheads, which are not evaluated in the present works. This work provides an extensive analysis of Look-up-table (LUT)–based obfuscation by exploring several factors such as LUT technology, size, number of LUTs, and replacement strategy as they have a substantial influence on Power-Performance-Area (PPA) and Security (PPA/S) of the design. We show that using large LUT makes LUT-based obfuscation resilient to hardware security threats. However, it also results in enormous design overheads beyond practical limits.

To make the reconfigurable logic obfuscation efficient in terms of design overheads, this work proposes a novel LUT architecture where the security provided by the proposed primitive is superior to that of the traditional LUT-based obfuscation. Moreover, we leverage the security-driven design flow, which uses off-the-shelf industrial EDA tools to mitigate the design overheads further while being non-disruptive to the current industrial physical design flow. We empirically evaluate the security of the LUTs against state-of-the-art obfuscation techniques in terms of design overheads and SAT-attack resiliency. Our findings show that the proposed primitive significantly reduces both area and power by a factor of 8× and 2×, respectively, without compromising security.

CCS Concepts: • **Security and privacy → Hardware reverse engineering**; *Hardware attacks and countermeasures*; Side-channel analysis and countermeasures;

Additional Key Words and Phrases: Hardware security, look up table (LUT), obfuscation, optimized reconfigurable obfuscation, SAT-hard LUT blocks, gate replacement algorithm, security-driven design flow

This work is an extension of our previous work [14].

This work was funded by Defense Advanced Research Projects Agency (DARPA-AFRL, Grant No. FA8650-18-1-7819) and NSF CHEST IUCRC Industrial Support.

Authors' addresses: G. Kolhe, T. D. Sheaves, S. Rafatirad, A. Sasan, and H. Homayoun, Department of Electrical and Computer Engineering, University of California, One Shields Avenue, Davis, CA, USA, 95616; emails: {gskolhe, tsheaves, srafatirad, asasan, hhomayoun}@ucdavis.edu; S. Manoj P. D., Department of Electrical and Computer Engineering, George Mason University, 4400 University Dr, Fairfax, VA, USA, 22030; email: spudukot@gmu.edu; H. Mahmoodi, School of Engineering, San Francisco State University, 1600 Holloway Avenue, HH 808A, San Francisco, CA, USA, 94132; email: mahmoodi@sfsu.edu.

1084-4309/2022/06-ART56
https://doi.org/10.1145/3510421

ACM Reference format:
Gaurav Kolhe, Tyler David Sheaves, Sai Manoj P. D., Hamid Mahmoodi, Setareh Rafatirad, Avesta Sasan, and Houman Homayoun. 2022. Breaking the Design and Security Trade-off of Look-up-table–based Obfuscation. *ACM Trans. Des. Autom. Electron. Syst.* 27, 6, Article 56 (June 2022), 29 pages.
https://doi.org/10.1145/3510421

1 INTRODUCTION

Massive integration of billions of transistors on a single **integrated circuit (IC)** has improved the performance of the IC at the cost of added design complexity. The increasing costs of manufacturing such intricate designs and the trend of incorporating heterogeneous IC components from various vendors for IP development have given rise to the era of "fabless manufacturing." Despite the cost-effectiveness of this trend, exposing designs to third-party vendors and fabrication facilities may expose the hardware designs to an array of hardware security threats. While the threat of malware, which is a software entity, can be mitigated with software methods [36–39], we need active countermeasures to thwart hardware attacks. Obfuscating the design makes it hard to understand the underlying design and thus helps curb some of the hardware attacks. The IP supply chain provides many opportunities for adversaries to employ state-of-the-art techniques to reverse engineer or compromise a customer's design, such as IC **reverse engineering (RE)** and hardware Trojans [13, 32, 46].

To combat this pervasive threat, hardware **design-for-trust (DFTr)** mechanisms, such as watermarking, IC metering, IC camouflaging, split manufacturing, camouflaging, and logic locking [1, 7, 10, 11, 33], have shown good resilience to many of the existing potential hardware-level reverse engineerings and tampering techniques [52]. Increased interest within the logic locking and IC camouflaging research community has persuaded Mentor Graphics, a major CAD tool provider, to release TrustChain, which is a CAD framework that supports logic locking and camouflaging as a means of curbing various hardware security threats [41].

Numerous obfuscation techniques, namely, logic locking and camouflaging, have been proposed to thwart existing hardware security threats. Some of these techniques make use of reconfigurable logic, a method based on hardware reconfiguration and/or transformation, such as **Look Up Tables (LUTs)** [5, 18, 45]. However, some recently introduced attacks have exploited vulnerabilities in various available logic obfuscation schemes. **Boolean satisfiability (SAT)**-based attacks are among the most effective de-obfuscation/de-camouflaging techniques. These attacks can reverse engineer a target design even when state-of-the-art logic locking and camouflaging protection mechanisms [19, 43] are used. Recent works in the domain of reconfigurable obfuscation make use of diverse approaches like increasing the number of reconfigurable blocks [45] or using different replacement strategies [20] to reinforce the security against the SAT-attack.

In this work, we demonstrate that security can be compromised despite considering all substantial and compelling factors for reconfigurable logic obfuscation. Furthermore, most of the existing work lacks the discussion on the impact of the reconfigurable obfuscation on the design in terms of **Power, Performance, Area (PPA)** overhead. Prior works focus on either security [20] or the overhead impact [48] of the proposed primitive and lack the comprehensive study of the vital elements that contributes to the overall success of the obfuscation technique. Moreover, the obfuscation strategies have been crafted based on heuristics, which may not apply to all designs and thus requires customization of IP-design flows.

Compared to the prior works, we comprehensively explore the design space of LUT-based obfuscation (a variant of reconfigurable obfuscation) to showcase its unparalleled resiliency and efficiency against state-of-the-art attacks. Using the design space exploration process, we study four factors that impact the design overhead and security of LUT-based obfuscation. These four factors

consist of (1) technology of the LUT used for logic obfuscation, (2) the size of LUT (LUT scale-up), (3) the number of cells replaced by LUTs (LUT count or scale-out), and (4) the replacement strategy. Further, based on the thorough experimental evaluation (i.e., an investigation of all possible combinations for these factors), we provide a LUT-based obfuscation, resilient against the SAT attack. Our investigations substantiate that the LUT size (LUT scale-up) plays a crucial role in enhancing the resiliency of the obfuscation scheme, even in the presence of a weak replacement policy such as random replacement. This reduces the IP designer's efforts as they do not have to abide by the obfuscation strategies developed based on heuristics. Finally, we gauge the LUT-based obfuscation's impact on the power-performance (delay)-area (PPA) overheads. Considering the large overheads imposed due to the large size of LUT, large-scale obfuscation using LUTs is idealistic and practically impossible to be used for obfuscation. There has not been any comprehensive analysis of the trade-off between resiliency and overhead across many different LUT topographies and scales, making LUT-based obfuscation an open challenge.

To overcome the challenge mentioned above by the LUT-based obfuscation, the incurred overhead by the LUT-based obfuscation must be reduced radically while offering robust resiliency against various hardware security attacks. However, both of these goals are conflicting for LUT-based obfuscation, and given the impossible nature of satisfying both goals at once, the IP owner usually achieves one objective at the expense of the other. For example, reducing the LUT size mitigates the hefty overheads but at the cost of reduced SAT-attack resiliency. However, in the proposed primitive of the LUT-obfuscation, the obfuscation leverages from the LUT's enhanced reconfigurability while achieving both goals of reducing the design overheads and increasing attack resiliency.

The proposed novel LUT obfuscation is also rigorously tested for various metrics including power, performance, area, and security by utilizing different benchmarks such as GPS, which is a representative example of the size and complexity of real-world designs. The current work is the extension of our previous work [14] and the current work expands the contribution of our [14] as follows.

(i) Investigate and evaluate current state-of-the-art re-configurable obfuscation techniques: We perform an extensive case study of the state-of-the-art reconfigurable obfuscation techniques. Among various reconfigurable obfuscation, we primarily focus on LUT-based obfuscation and evaluate it on four primary parameters: (1) LUT technology, (2) LUT size, (3) number of LUTs, and (4) replacement strategy as they significantly influence design performance and metrics. Additionally, we study each parameter's impact on hardware security and design overheads by performing extensive design space exploration.

(ii) We elaborate on the Proposed STT-LUT with Scan Chain Programming, and the Full implementation details of the custom layout of **Magnetic Tunnel Junction (MTJ)** latch in standard cell format along with the area breakdown between different blocks that are used for the obfuscation purposes. We have also added the results on the PPA of an STT-LUT for 28 nm technology.

(iii) We discuss the replacement strategies and provide theoretical analysis along with the complexity analysis of the proposed replacement strategies. Moreover, the discussion on how replacement strategy can be leveraged to increase the **Clause-driven Conflict Learning (CDCL)** runtime is discussed.

(iv) Additional details on testing the IP after obfuscation and the validation results are presented. Moreover, we have added a detailed comparison of the proposed LUT obfuscation against other state-of-the-art proposed obfuscation techniques such as Anti-SAT SFLL-HD^0 and InterLock obfuscation.

(v) Finally, to show the feasibility of the proposed novel LUT solution, we present a case study for the GPS module, which has around 170K gates.

Among the above contributions, we have provided minuscule details of the iterative design flow used to implement the proposed LUT-based obfuscation technique using off-the-shelf EDA tools to optimize PPA and security. The proposed flow is non-disruptive to the standard **Complementary Metal-Oxide Semiconductor (CMOS) Application Specific Integrated Circuit (ASIC)** design flow, which is vital for the accelerated design and manufacturing time. The resulting obfuscation flow can be generalized and scaled to various designs. Last, we discuss the performance of the LUT-based obfuscation against various attacks and obfuscation strategies.

2 LOGIC OBFUSCATION AND SAT ATTACK: BACKGROUND

2.1 Logic Obfuscation

Logic obfuscation aims to conceal the functionality of the design by inserting additional logic gates. These gates can be key-programmable XOR/XNOR gates or MUXes for interconnection. The ambiguity is created in the circuit due to the newly added gates. The strength of the logic obfuscation also depends on the location of the gate insertion [25, 27, 33]. Traditional attacks such as justification/sensitization [26] try to reverse engineer the design using heuristic techniques; however, with more advanced logic obfuscation, the attacker has been able to extract the keys using **Automatic Test Generation Pattern (ATPG)** [21, 43].

2.2 SAT Attack

Boolean **Satisfiability attack (SAT attack)** [43] is used to determine the correct key of the obfuscated or logic locked circuit. The SAT attack is an "oracle-guided attack," where the threat model assumes that an attacker has access to the activated IC along with the locked gate-level netlist, which can be retrieved through invasive reverse engineering [43] or through the GDSII of the design. The input to the SAT solver is a boolean formula in **Conjunctive Normal Form (CNF)** obtained from the transformation of the obfuscated netlist. *Distinguishing Inputs* also known as *DIPs*, helps the SAT-attack in eliminating the incorrect keys iteratively.

DIPs can be found using a miter circuit, which is used to check the equivalence of two hardware designs. The miter circuit is built using two copies of the locked netlist obtained after invasive reverse engineering efforts, with their outputs XORed together. The input is common to both instances of the locked netlists, which are part of the miter circuit. However, each of the locked netlists is programmed with a different key. The DIP is found when the two circuits' output differs, making the output of XOR gate or miter circuit "1." These two different outputs indicate that one of the keys is wrong, and the input that helped us distinguish between the keys is termed as *distinguishing input*. The found DIP is applied to the oracle circuit to identify the correct output. The SAT-attack in every iteration tries to find a new DIP until no new DIP exists.

2.3 Post-SAT Obfuscation and Challenges

After the advent of the SAT attack, the advanced logic locking and camouflaging algorithms have largely focused on increasing SAT-attack iterations required for successfully unlocking the logic locked circuit [43]. Another way of increasing the SAT attack time is to decrease the number of wrong keys getting eliminated in each iteration, such that the SAT attack needs more iterations and, consequently, more execution time. Various defense primitive such as SARLock [49], Anti-SAT [47], CamoPerturb [50], and **SFLL-HD^0 (Stripped Functionality Logic Locking)** [53] abides by the tenet of using a logic function that increases the number of SAT-iteration, resulting in longer SAT-execution times. Few obfuscation schemes are based on the addition of the cyclic loop in the

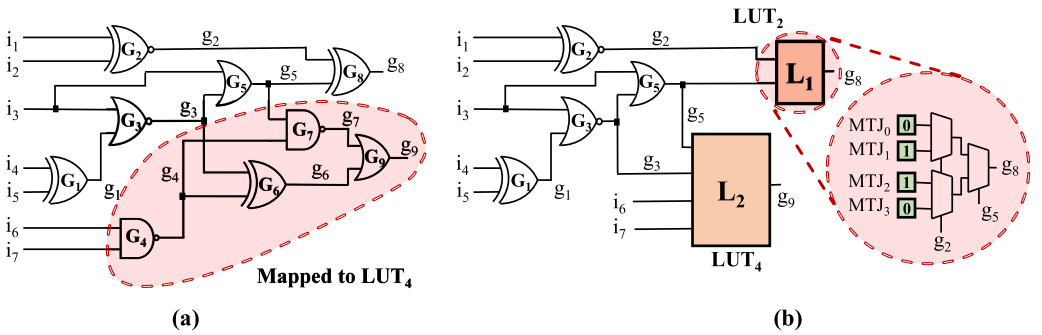

Fig. 1. (a) Sample circuit used for obfuscation with gates G_4, G_6, G_7, G_9 selected for obfuscation. (b) Obfuscation with LUT with representation of the LUT for SAT-attack simulation.

circuit, especially stateful/oscillating cycles [31, 35]. When the SAT attack encounters these cycles, the SAT solver either returns the wrong key, or else the SAT solver gets stuck in an infinite loop. And thus, it is unable to find the correct key required to restore the functionality of the obfuscated IP.

Although there have been advancements in logic locking techniques [9], researchers have been able to find the vulnerabilities in many of the most prominent obfuscation schemes. Attacks such as CycSAT, SMT-Solver [4, 6, 8, 31] show that by considering just a few conditions and pre-processing steps, designs obfuscated with cyclic obfuscation can be reverse-engineered. The state-of-the-art attack SFLL-HD [53] has also been successfully defeated using FALL-attack [40], which does not require an oracle IC to find the correct key making the attack even more feasible. Apart from these attacks, the mentioned SAT-resilient schemes are vulnerable to other types of attacks, such as **Signal Probability Skew (SPS)** attacks [51], and approximate-SAT attacks [34]. These vulnerabilities in the obfuscation techniques have impelled us to propose and build a more comprehensive and robust obfuscation scheme that can resist many hardware security threats.

3 INVESTIGATION ON LUT-BASED OBFUSCATION

The recent work in the reconfigurable logic obfuscation uses the reconfigurable devices to protect the IP against various hardware security attacks [2, 5, 18, 20, 24, 30, 45, 48]. We primarily consider the LUT-based obfuscation as it can represent all of the possible reconfigurable-based obfuscation scenarios given its ability to reconfigure. In obfuscation using LUT, the gates are selected from the design and are mapped to the LUTs. For example, to obfuscate a 2-input AND gate with LUT, one can replace the AND gate in the IP with the LUT whose configuration bits are set to "0001." Obfuscation using LUT thus results in a netlist as a hybrid mixture of ASIC and programmable FPGA styles. In the LUT-based obfuscation, the keys that denote the logical function of the LUT can be stored in a tamper-proof non-volatile memory. Without prior knowledge of the content of the non-volatile memory contents, the attacker does not have access to the IP's intended functionality and thus refrains the attacker from reverse engineering the IP. Figure 1 shows the LUT-based obfuscation, where part of the circuit is mapped to the LUT.

While storing the LUT configuration bits in tamperproof memory thwarts the attacker from understanding the content and functionality of the LUT, the attacker can still use SAT-attack to restore the content of the LUT. The attack on the LUT-based obfuscation using SAT-attack is described here. The SAT attack simulation does not offer the ability to model the LUT directly. Hence, we model the LUT-based obfuscation using the MUXes where each LUT is replaced with a (2+)-level MUX. The work in Reference [43] can be leveraged to attack the existing reconfigurable

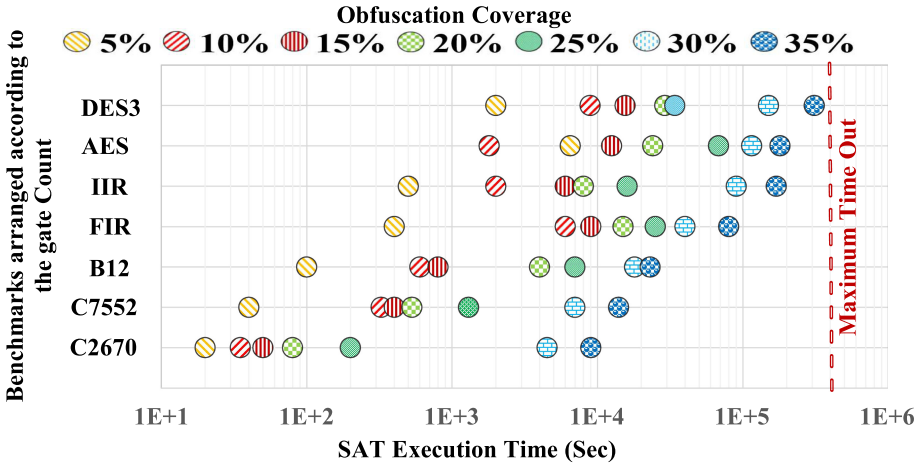

Fig. 2. De-obfuscation time of the various benchmarks obfuscated with LUT of size 2. Each benchmark is obfuscated with varying obfuscation ranging from 5% to 35% [14].

schemes such as References [2, 5, 18, 28, 45]. Figure 1(b) shows the representation of the LUT of size 2 for SAT simulation. The LUT of size 2 is built using 2:1 MUX. Since the attacker is interested in finding the LUT's configuration bits' content, representing the functionality imparted by the LUT, the MTJ's are treated as the key inputs. The SAT-solver tries to find the value of these key inputs during the de-obfuscation process. After replacing all the LUT with their equivalent representation using MUXes, one can perform the SAT attack.

Figure 2 shows the resiliency of obfuscation using a reconfigurable block against the SAT attack. The LUT of size 2 is chosen, because one can model the existing reconfigurable blocks proposed in previous work [20, 24, 30, 48] using LUT of size 2 for SAT-attack modeling. Consider Figure 1, Gate G_8 can be replaced with an emerging device proposed in Reference [24], which can implement at most 16 different combinations. For SAT simulation, we can replace it with LUT of size 2. The gate selection for obfuscation is made using random replacement policy [33]. Further, the obfuscation percentage is varied from 5% up to 35%. The obfuscation percentage of 5% corresponds to 5% of total gates being replaced with LUT of size 2. Each data point on the graph is the averaged run of 10 different SAT-solver execution. The benchmarks in Figure 2 are arranged in ascending order on the Y-axis, while the SAT de-obfuscation time is shown on the X-axis. The X-axis has a logarithmic scale for better representation.

From the result, it can be concluded that all the designs can be reverse engineered using the SAT-attack within 5 days (432×10^5 s).

Obfuscating 35% of the circuit to render the security against the SAT attack is not a viable solution from the manufacturing perspective, as it adds an enormous amount of overhead and manufacturing costs. Therefore, traditional reconfigurable obfuscation does not cater to providing resiliency. Modern works in the reconfigurable obfuscation domain that are crafted to thwart SAT attacks, such as Reference [20] lacks investigations of PPA overheads. In contrast, the work in Reference [48] acknowledges the overheads but fails to acknowledge their solution's resiliency against the well-known and most prevalent SAT attack. It is intuitive to use the large LUTs for obfuscation, because the number of possible functions rendered by a LUT is enormous. Therefore, to render better resiliency, one should employ reconfigurable blocks that can employ more functions. A Look-up table of input size n can provide 2^{2^n} logical functions, thus increasing the search space for the SAT-solver exponentially.

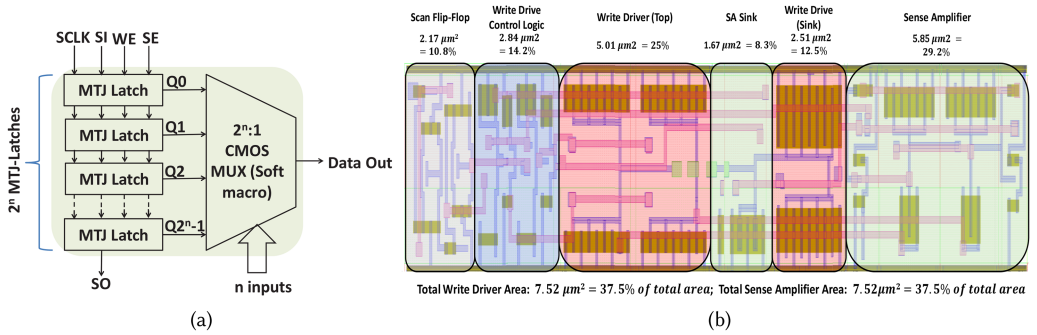

Fig. 3. (a) Proposed STT-LUT with scan chain programming. (b) Full custom layout of MTJ latch in standard cell format and area breakdown between different blocks.

While leveraging the larger sizes of the Look-up table seems the best idea, the implementation of LUT for obfuscation results in increased design overheads. Therefore, the trade-off to address is (1) whether to use a few numbers of large LUTs *or* (2) use more numbers of small LUT size to defend the SAT attack. To answer this question, we perform an extensive design-for-security space exploration for LUT-based obfuscation using four critical factors, namely, (1) LUT technology, (2) LUT size, (3) number of LUTs, and (4) replacement strategy to find the impact of each on SAT-resiliency.

3.1 Implementation of MTJ-based STT-LUT

In this section, we first discuss the design implementation of the MTJ-based LUT used in our work, followed by the impact of the SRAM- and MTJ-based LUT on the design overheads

3.1.1 Design and Integration of Spin Transfer Torque (STT)-based LUT. As STT-based LUTs have shown higher PPA efficiency [45], we consider STT-based LUT design and obfuscation in this work. STT technology can provide incredible features like (1) higher integration density than Static RAMs (SRAM), (2) high endurance and retention time, (3) near-zero leakage, and (4) soft error resilience [14–16]. Additionally, STT-LUTs have shown the ability to be highly integrative in the CMOS fabrication process [45]. Thus, STT-LUT, due to its virtue of on-die reconfigurability, enables us to achieve high performance and security against various hardware RE threats.

Additionally, for STT-based LUTs, reconfigurable bits are stored in a **magnetic tunnel junction (MTJ)** inserted between metal layers. The delayering process involved in the reverse engineering of IPs to retrieve the netlist results in the destruction of the structure of the MTJ device and, thus, will result in loss of data stored in the MTJ. The MTJ, in this manner, serves as the tamper-proof memory to store the configuration of the LUT. 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 limits 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 residing between the LUT inputs and the output is not suitable for the obfuscation [3]. In contrast, we propose an STT-LUT with a design concept in which the path from the LUT inputs to the LUT output is a MUX, as shown in Figure 3(a). The MUX of the LUT is a 2^n to 1 (2^n : 1) CMOS MUX implemented in static style, which can be written as a synthesizable RTL code for automatic implementation and optimization by the logic synthesizer tool in the process of design compilation.

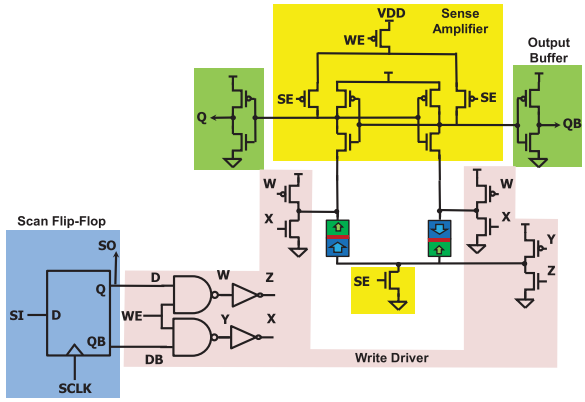

Fig. 4. MTJ latch with scan chain programming.

Each configuration bit is stored by a MTJ latch with scan chain programmability, as shown in Figure 4. The MTJ latch uses a pair of differentially programmed MTJs for non-volatile storage, a pre-charge sense amplifier for sensing the state of the MTJs, and three write driver schemes for parallel writing MTJs simultaneously, with each MTJ receiving full voltage swing, offering more write current. The **Sense Enable (SE)** signal must be low during the write operation, and the **Write Enable (WE)** signal must be low during the sensing operation. To avoid conflict of state between the pre-charge state of the sense amplifier (when SE=0) and the state of the write driver outputs in the write mode, the pre-charge path to V_{DD} is disconnected via the PMOS driven by the WE signal. The MTJ latch uses a dynamic latched sense amplifier that needs to be fired (SE low to high pulse) once on every power-up to convert the resistive state of the MTJs into the volatile voltage states at the outputs (Q and QB). In this configuration, the MTJs are read-only once, and for the remaining time in the active mode, the LUT read power and delay are determined by the static MUX. Moreover, by not reading from the MTJs repetitively in the active mode as in the dynamic STT-LUT styles, the stress is removed from the MTJs, enhancing their lifetime. Another critical thing to note here is that the scan chain used to program the LUT is separate from the one used in the design for testing and thus does not require critical MUX to switch from functional mode to test mode. Apart from programming the MTJ's, the separate scan-chain also allows us to thwart the Scan and Shift attack, which is discussed in Section 5.4.

The MTJ latch is designed in a full-custom manner and needs to be optimized for sensing reliability and area. The custom-designed MTJ latch is delivered as a standard cell for integration into the ASIC design flow. The full-custom design and optimization of the one-bit MTJ latch cell are performed in the Synopsys generic 28 nm process. The write drivers tend to require large transistors to produce sufficient current needed for MTJ write. The write transistors need to be optimized so that the write operation can succeed under process variations. We have performed a statistical transistor sizing optimization on the write driver for achieving near zero (less than 0.1%) write failure rate under process variation. After the write driver sizing optimization, the read path (i.e., the sense amplifier) transistor sizes are statistically optimized for achieving less than 0.1% sensing failure rate at the smallest possible area. Moreover, a minimum-sized scan flip-flop is inserted in front of the MTJ latch to store the data written to the MTJ latch. These scan flip-flops will form a scan chain for loading the configuration bits to the MTJ latches in a design. Figure 3(b) shows the full-custom layout of the one-bit MTJ latch designed in the format of a standard cell layout (fixed height). The write drivers occupy most of the layout area (37.5%), since the MTJ write current is still reasonably large. Notice that the MTJ devices are stacked on top of this layout between

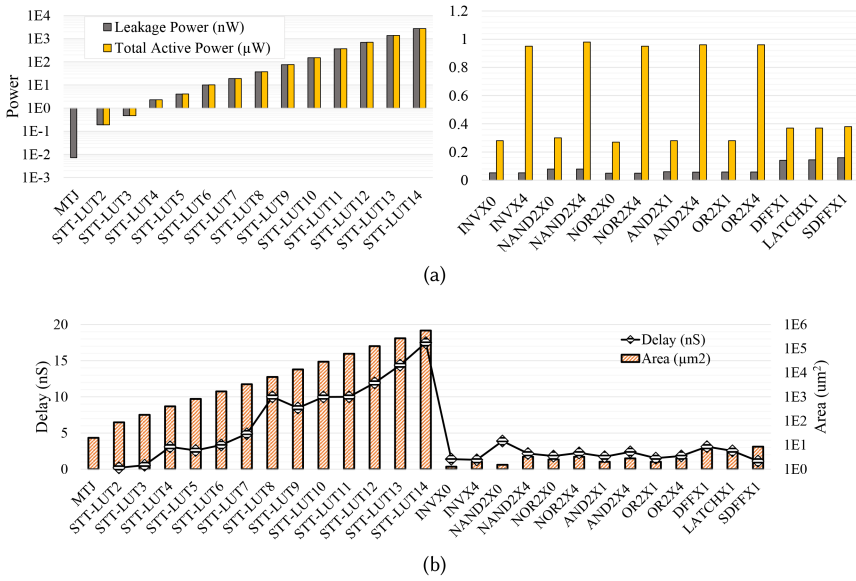

Fig. 5. Comparison of (a) power, (b) delay, and area of STT-LUT and standard cells in 28 nm.

two metal layers (assuming M3 and M4) and do not occupy the 2D area. M3 pins are placed for connection to the MTJ layers.

3.1.2 STT-based LUT Versus CMOS-based LUT. Figure 5 shows the comparison of the area of the MTJ latch and STT-LUTs, along with the areas of other standard cells in 28 nm. 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 than basic CMOS logic gates and 20× smaller than SRAM-based D-FF.

The delay and active mode power of the STT-LUT are determined by the multiplexer part of the LUT, which is optimizable by the logic synthesizer. Figure 5 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 the large MTJ latch area, LUTs are noticeably larger than the standard cells, and their area increases exponentially with fan-in. While it is evident that the scale at which the LUT's overheads increases is exponential, we can leverage LUTs up to a specific size where the increase in overhead is linear. The LUTs of sizes 8 and 9 have power in hundreds of uW, which is way larger than the standard cells, but we must also keep in mind that LUTs will replace the collection of gates. Going beyond LUT of size 8 and above makes it hard to find a collection of gates that can be effectively mapped to the LUT. Moreover, overheads at such high input LUT are hard to offset as they increase exponentially. This study helps us in establishing a practical bound until which we can scale up the LUTs.

The traditional methods have used SRAM-based Lookup tables for obfuscation, which are volatile and require programming during startup. Loading large keys for SRAM-LUTs comes with the added cost of time, power, and area overheads. One needs to also account for the overhead of programming and storage circuitry. With STT-LUT, using non-volatile MTJs, programming is required once in the trusted regime. Since the programming of the key will be done once, write circuitry can be shared between MTJs, reducing both area and power overhead of the design. Moreover, with SRAM-based LUT, there is a key movement from tamper-proof memory to the

SRAM cells for programming SRAM-based LUT on device startup, which can raise more security concerns.

Compared to the SRAM-based LUT, the STT-MTJ have only one downside in our case, i.e., the MTJ-based STT-LUT technology has not matured or commercialized. However, given that shortly, when the MTJ-based STT-LUTs emerge, it will benefit the LUT-based obfuscation in many ways. Thus, in our quest to realize LUTs as a feasible means of obfuscation, in this study, we proceed with STT-LUTs.

3.2 Replacement Strategies

The location of the gate(s) selected for the obfuscation is one of the critical factors that define the strength of the obfuscation. The replacement strategy finds the gate for obfuscation based on heuristics. As replacement strategies affect the SAT-resiliency and the PPA overheads, we study the effect of replacement strategy in this work. There are several conditions that an effective replacement strategy needs to meet to provide resiliency against the SAT attacks. The 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 the random placement strategy [33]. To better evaluate the impact of each condition, we compare three different strategies in this work as follows.

3.2.1 Random Selection (RND). In random selection algorithm, the gates are selected for obfuscation in the random fashion. We use this method as a baseline for comparison as opposed to the *independent selection* in Reference [45].

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

For the IP to exhibit the low output corruptibility, we employ a Breadth-first-search on the graph, which consists of graphs as a node and edges being the connection between the logical gates. While traversing from **logic cone outputs (LCO)** toward the inputs, a dictionary with all the gates and their corruptibility will be created, which allows us to pick the combination of the gates with the lowest output corruptibility. After traversing, based on the number of gates targeted for obfuscation, we do the multi-objective optimization, where we try to maximize the number of gates selected for obfuscation while minimizing the output corruptibility.

ALGORITHM 1: Avoiding Unintentionally Correct Key Generation (LC_NoGen)

```
1: for each LCO in Logic_cones do                          ▷ LCO: Logic Cone Output
2:     gate_list = BFS(LCO);                               ▷ Get all gates in the logic cone
3:     for each gate in gate_list do
4:         gate.listLCOs = find_affected_LCOs(gate)
           Find all the Logic Cone Output gates, which are affected by the current gate.
5: for each (gate in circuit) do
6:     for each (LCO in gate.listLCOs) do
7:         tag_key(LCO)
8:         if isExist(tag_key(LCO)) then
9:             dictionary.add(gate)
10:        else
11:            dictionary.add(gate)
12:            dictionary.addtag((tag_key(LCO))
13: CriticalPath = PrimeTime(Get Critical Path)
    Get List of gates that are on Critical Path using Synopsys PrimeTime.
14: for each (tag in dictionary) do
15:     for each (gate in tag) do
16:         if isExist(Parent(gate) in tag)
17:         or isExist(gate in CriticalPath) then
18:             dictionary[tag].delete(gate)
                Remove gates that are adjacent to each other to avoid back-to-back LUT replacement.
                Remove gates on critical path list
19: tag_key = Optimize ()
    find tag_key which have maximum gate coverage with lowest Output Corruption
20: for each (gate in tag_key) do
21:     Replace_LUT(gates, target_no)
        Replace gates with LUT.
```

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

By considering the above case of avoiding the obfuscation of two gates directly connected, we propose a *LC_NoGen* replacement strategy. The pseudocode for the *LC_NoGen* replacement strategy is illustrated in Algorithm 1. The algorithm involves traversing the graph followed by the dictionary creation. The dictionary is filtered to eliminate the back-to-back gates and gates that contribute to the critical path, and we use an optimizer that maximizes gate coverage while reducing the output corruption. The complexity of traversing the graph with V vertices and E edges is given as $O(V + E)$. With fewer modifications, the dictionary creation and filtering can be done

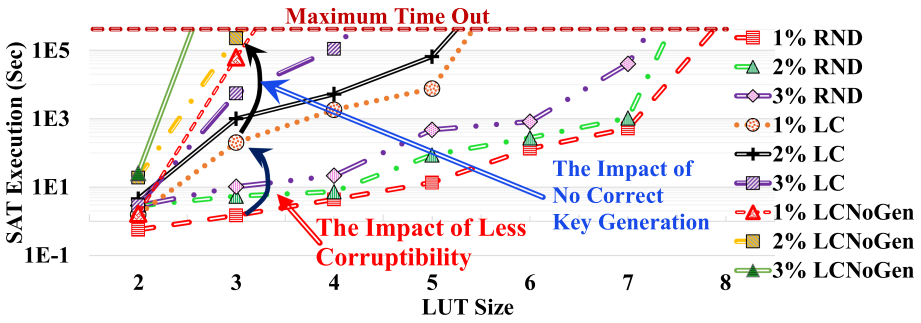

Fig. 6. De-obfuscation time using SAT-attack for different (1) replacement strategy, (2) LUT size (scale up), and (3) number of LUTs (scale out) in ISCAS-85 c7552 [14].

while traversing the graph, and thus, the overall complexity for traversing the graph and dictionary creation is given as $O(V + E)$. The optimization problem, however, is a combinatorial optimization problem. While finding the optimal set of gates for obfuscation is an NP-hard problem, we use off-the-shelf ILP solvers to get the sub-optimal solution using an approximation algorithm. The problem of gate selection with minimum output corruptibility is treated as a *variant* of a classical problem of minimum vertex cover. The task at hand is to choose a minimum number of nodes, that maximizes the cover. The gates selected by the optimizer are replaced with the LUT, and We demonstrate that by considering the *LC_NoGen*, the Security per PPA overhead footprint is improved. The proposed algorithm can also be scaled to other obfuscation techniques for inserting the obfuscation as this algorithm finds the optimal place for inserting a gate that minimizes the output corruptibility.

Figure 6 illustrates the performance of the discussed obfuscation strategies against the SAT attack. With the improved obfuscation strategies, the de-obfuscation time of the SAT solver is increased, and in most of the cases, obfuscation using *LC_NoGen*, strategy outweighs the Random and *LC* obfuscation, which shows that high security can be obtained with a lower number of gates obfuscated. The lower number of gates used for obfuscation results in lower PPA overhead. However, while providing higher security at the cost of lower PPA overheads, these obfuscation schemes tend to have low output corruptibility.

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

The experiment also shows that using a larger LUT size overweigh the benefits of using a better gate replacement policy. Another observation from the experiment is that change in the obfuscation coverage from 1% to 3% (i.e., changing the number of LUTs inserted in the circuit) also increases the runtime of SAT-solver. Thus, in the next section, we compare the effect of the size of LUT versus the number of gates obfuscated (obfuscation coverage) on the circuit resiliency in more detail.

3.3 LUT Size Versus Number of LUTs

To get the best security results using LUT-based obfuscation, it is wise to leverage the large size of the LUT, because increasing the LUT's size can thwart the SAT attack and increase the output

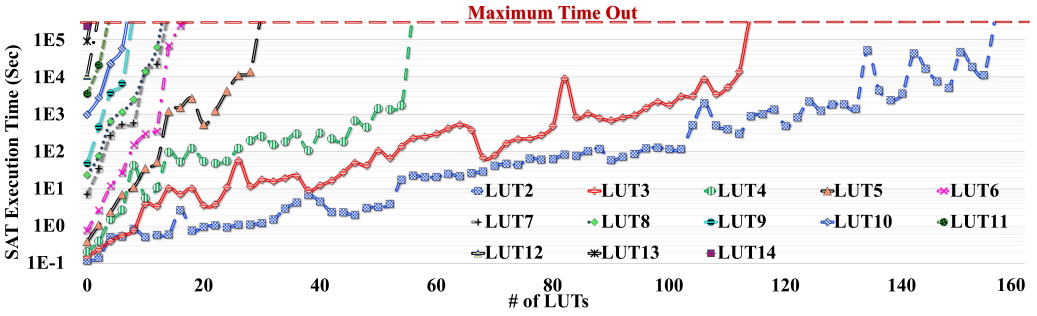

Fig. 7. De-obfuscation time of C7552 benchmark with different number of LUTs and different sizes of LUT using SAT-attack [14].

corruptibility. This advantage of using larger LUT sizes is that LUTs are modeled using the $\log_2(n)$-level MUX-based structure, and with the increasing size of LUT, the SAT attack will replace them with the deep MUX trees. As discussed, the SAT-attack leverages the CDCL algorithm for finding the distinguish input. However, when a symmetrical structure of the MUX-tree is used for the obfuscation, there is no shortleaf in finding the conflict clause while running the SAT-attack. The increasing size of LUT grows the MUX tree deep, and consequently, the search space and the efforts for finding the conflicting clause are increased, making the resulting instance an SAT-hard. In the following example, we show how the de-obfuscation time rises exponentially, with larger LUTs to find the keys' value.

For replacement using larger LUT sizes, combinations of gates are replaced using large LUT. Figure 1(a) shows an example of the large-sized LUTs used for the obfuscation. The 2-input gates (G_4, G_6, G_7, G_9,) are replaced using LUT of size 4. For SAT-simulation, the LUTs are represented using MUX-tree as shows in Figure 1(b). To evaluate the effect of LUT size versus the number of LUTs, we obfuscate the multiple gates with varying LUT sizes in a similar manner.

Figure 7 shows SAT execution time with more details on the ISCAS-85 C7552 benchmark for different sizes of LUTs used for obfuscation. This experiment shows that a similar trend of leveraging larger LUT size provides higher resiliency than utilizing more LUTs for obfuscation. We can see that SAT execution time increases at the near exponential rate in both directions, i.e., scale-up (increasing the size of LUTs) and scale-out (increasing the number of LUTs). However, obfuscating only a single gate with an LUT of size 13 can render a time-out state for the SAT solver.

To further substantiate these results and understand and compare the impact of LUT scale up versus scale out on SAT execution time, we use a regression model to demonstrate the relationship between SAT execution time and these two parameters. Figure 8 provides two different scenarios to accurately model the relationship between SAT deobfuscation time and the size of LUTs, as well as the number of LUTs. As shown in Figure 8(a), one factor (LUT size or number of LUTs) is fixed in each curve. LUT size is constant in one of them, while the number of LUTs is swept from 1 to 29. In another curve, the number of gates obfuscated is constant, and the size of LUTs has been swept from 2 to 8. Based on the independent (one-variable) exponential regression model illustrated on curves, it is clear to observe that the LUT scale-up has significantly more influence on SAT execution time compared to the LUT scale-out.

Figure 8(b) shows another similar situation for C7552 that proves that the LUT scale-up is more effective than the LUT scale-out. Also, according to a multi-dimensional linear regression, the impact coefficient of the number of gates obfuscated using LUT on SAT execution time is 72.347. However, the impact coefficient of using a large size of LUT is 1,969.25. This regression coefficient demonstrates that LUT size is the most important factor for security purposes than the number

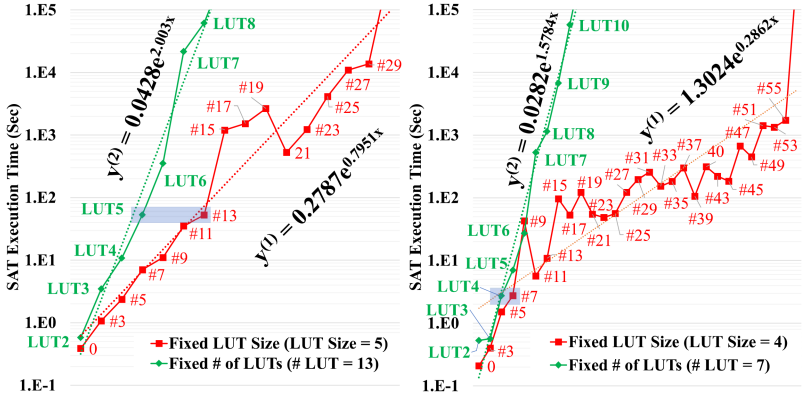

Fig. 8. LUT scale up vs. scale out: Comparison between the impact of LUT size and number of LUTs on SAT execution time on (a) ISCAS-85 C2670, (b) ISCAS-85 C7552 [14].

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

The thorough experiment on LUT-based obfuscation by considering (1) LUT technology, (2) LUT size, (3) number of LUTs, and (4) replacement strategy, it can be concluded that scaling up the size of the LUTs for obfuscation yields the excellent resiliency against state-of-the-art attacks. However, the investigation on PPA shows a sharp increase in the design overheads when using the large size of LUT. Figure 9 shows the Area and Power overhead for LUT-based obfuscation. Every point from Figure 7 was synthesized using Synopsys Design Compiler using TSMC 65nm library to obtain the power and area overheads. It can be observed that to render the timeout state for SAT solver using the LUT-based obfuscation, the PPA incurred is at least 10× the baseline version of C7552 with no obfuscation. **The enormous obfuscation overhead due to leveraging the traditional LUT-based obfuscation makes the LUT-obfuscation an idealistic method for hardware security.** This experiment also lets us find the smallest LUT size that results in SAT-resilient obfuscation with lower PPA. For this example, the LUT of size 8 results in SAT timeout with the lowest PPA among all samples, and thus in the following experiments, we use LUT of size 8 as the baseline for comparing our proposed novel LUT.

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

4 PROPOSED NOVEL LUT CONFIGURATION

Using large LUT sizes for obfuscation offers increased SAT resiliency as it creates a SAT-hard instance. The resulting instance is SAT-hard as the CDCL algorithm, which is responsible for finding the DIP, needs to consider the increased search space. However, with the larger LUT sizes, the LUT obfuscation renders an idealistic method, and to make it an efficient method of obfuscation, we must break the trade-off between security offered by the LUT-based obfuscation and the imposed PPA overheads. In the proposed method, we study the ***Davis-Putnam-Logemann-Loveland* (*DPLL*)** algorithm (or one of its derivatives), which is used to perform CDCL, and how

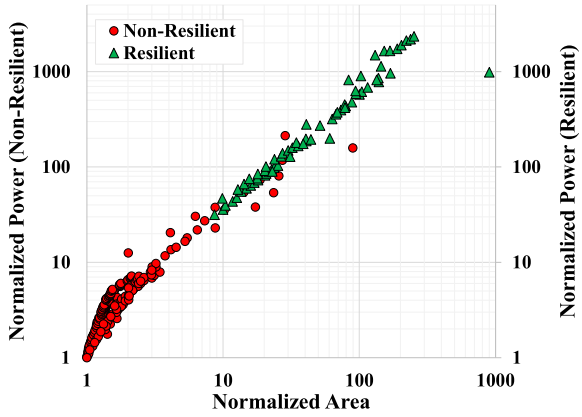

Fig. 9. Normalized area and power overhead of LUT-based obfuscation. Points that result in SAT execution time out are marked as SAT-resilient configuration [14].

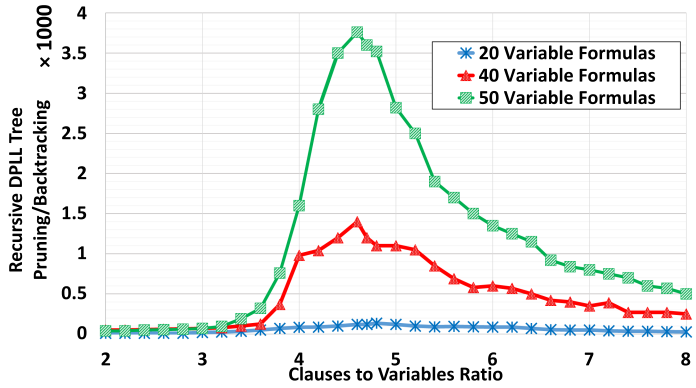

Fig. 10. Median number of recursive DPLL tree pruning/backtracking for random 3-SAT formulas, based on the ratio of clauses to variables [12, 22].

we can create SAT-hard problem using the small size of the LUT such that increased security can be obtained at the lower PPA footprint compared to the traditional LUT-based obfuscation.

For the SAT attack modeling, the obfuscated netlist is represented using the MUX tree, as discussed in Section 3. In the SAT solver, the ratio of the clause to variable can be used to evaluate the obfuscation strategy's security quantitatively. The resulting obfuscation instance can be called SAT-hard if the clause to variable ratio is around 4.2 [23]. Figure 10 shows that the ratio from 3 to 6 provides much higher DPLL calls, and 4.3 clauses per variable are the best ratio for generating the most computational SAT-hard instance, as it generates the highest number of DPLL calls. For example, a 100-variable and 300-clause instance (clause/variable = 3 is called as "under-constrained," because there are many satisfying assignments), or a 100-variable and 5000-clause instance (clause/variable = 50 is called as "over-constrained," because the contradictions can often be easily found) can be solved using the SAT solver very quickly. However, the SAT solver takes a long time to solve a 3-SAT instance, constructed with 100 variables and 450 clauses.

Table 1 shows the Tseytin transformation [44] of various logic gates into their respective CNF expression. From this table, only XOR/XNOR and MUX have 4 clauses per gate. This is when the clauses to variables ratio are 1 and 4/3 in MUX and XOR/XNOR, respectively. Despite the

Table 1. Tseytin Transformation of Basic Logic Gates [12]

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

observation that the XOR/XNOR has a larger clause to variables ratio for a single gate, MUXes provides a better building block for constructing SAT-hard circuits. This is because: (1) with no unit propagation and purification, for having four variables, a MUX can make the recursive DPLL tree one level deeper, (2) unit propagation and purification steps in the DPLL algorithm provide more simplified and smaller formula using enhanced Gaussian elimination while the contribution of XOR/XNOR gates is much higher [42]. Hence, MUXes needs more DPLL recursive tree prunings/backtrackings compared to XORs/XNORs. Moreover, since unit propagation and purification satisfy fewer equations or clauses, the clause to variable ratio will increase.

The next step for building an SAT hard problem is to push the clause to the variable ratio in the desired range of 3 to 6 (4.3 being the best). This prevents the propagation and purification algorithm from simplifying the circuit before branching into a recursive DPLL tree. This agenda of pushing the clause to the variable ratio in the desired range can be achieved by building a MUX tree. This property can be utilized to reinforce security offered by the conventional LUT-based obfuscation. We combine the traditional LUTs with an extra layer of LUT whose input size is fixed to 2. The size of the small LUTs is restricted to 2 to impose lower PPA overheads. The resulting novel LUT combines a large LUT with the addition of the smaller LUTs at its input. The scenario can be visualized as adding MUX-tree by large LUT, supplemented by another 2-input LUT, which resides at the select line of the previously inserted MUX-tree as shown in Figure 11(b). By adding 2-input LUT, the MUX tree can be imagined to grow in 2D space. (For example, a traditional LUT can be represented by the MUX tree growing in the horizontal direction, and the addition of MUXes on their select lines grows the MUX-tree in the vertical direction.) The proposed modification of adding another layer of LUTs to the LUT-based obfuscation benefits the IP from both reconfigurable and routing obfuscation. The additional layer of LUT increases the possible function of the large LUTs, thus increasing the search space for the SAT-solver. To restore the functionality of the IP, one has to find the correct functionality of both the small and large LUT simultaneously. For the n-LUT$_2$ followed by LUT$_n$, the number of possible functions implemented is

$$F_{possible} = n \times \underbrace{2^{2^2}}_{\text{Functions implemented by LUT}_2} \times \underbrace{2^{2^n}}_{\text{Functions implemented by LUT}_n}. \tag{1}$$

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

$$F_{total} = F_{LUT1} \times F_{LUT2}. \tag{2}$$

Moreover, the attacker's difficulty increases as the change in keys of the previous LUT is masked by the current LUT. Indirectly cascading LUTs in *LC_NoGen* further decreases the observability. Increased difficulty due to MUX-tree increases the search space, while reduced observability elevates

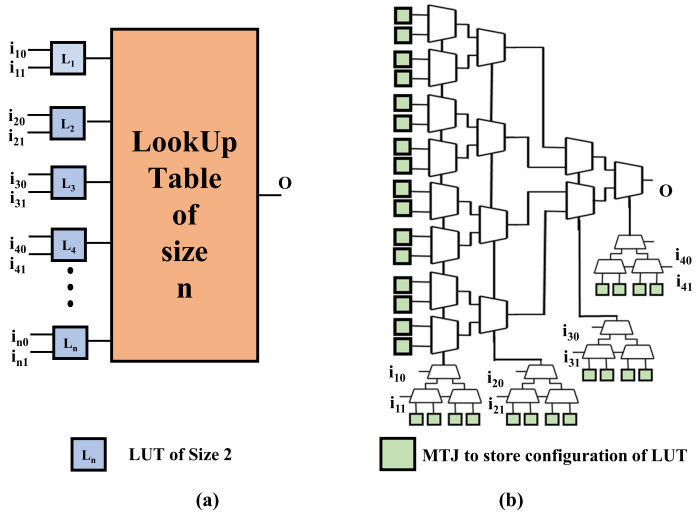

Fig. 11. (a) Proposed novel (LUT_n+n:LUT_2). (b) SAT-representation of proposed obfuscation with size 4 (LUT_4+4:LUT_2).

the security offered by the proposed novel LUT. With the growing search space, the probability of finding a correct key is significantly reduced. The probability of finding a single key when 2 LUT_7 + 7:LUT_2 are added in the circuit is given by

$$P = \frac{1}{1.452496e + 81}, \tag{3}$$

where

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

As the security obtained by the novel LUT is superior compared to the traditional LUT, we can reduce the LUT size required for the obfuscation, resulting in lower PPA and thus breaking the trade-off between security and the PPA.

Figure 11(a) shows the generic version of the proposed novel LUT. For the process of obfuscation using traditional LUT, the gates are identified and replaced using LUT_8 as LUT_8 renders the time-out scenario as shown in Figure 9 with the lowest possible PPA. On the contrary, in the novel LUT, the gate is always replaced with the combination of 2-input LUT and LUT_n such that $n < 8$ where n is the size of LUT. Part (b) of the figure shows LUT_4, which is preceded by 4-LUT_2. For the de-obfuscation using the SAT-attack, one can model the novel LUT block with the equivalent circuit shown in Figure 11(b).

Once we have replaced the gates with the proposed novel LUT, we empirically validated the clause to variable ratio of the obfuscated block. Figure 12 denotes the clause to variable ratio of both pre-obfuscation (Original) circuit and obfuscated circuit. It is evident that, after the LUT insertion, the Clause to Variable ratio falls in the range of 4–5, which creates a SAT-hard instance.

ASIC Iterative Security-driven Design Flow

Here, we provide an overview of the proposed methodology used to obfuscate the design using the proposed LUT. As seen earlier, **spin-transfer torque (STT)** MTJ-based LUTs have demonstrated higher PPA efficiency than the CMOS-based LUT. Due to this fact, we leverage the STT-based

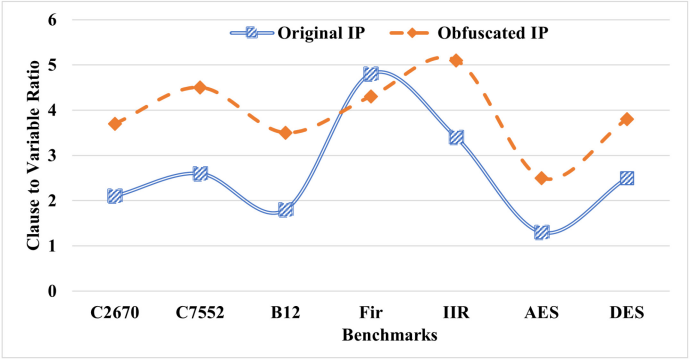

Fig. 12. Effect of obfuscation on the Clause to Variable Ratio.

LUT to construct the novel LUT in this work. The STT provides state-of-the-art performance and features such as (1) near-zero leakage, (2) soft error resiliency, (3) improved endurance and retention time, and finally (4) high integration density than SRAMs [45]. Besides enabling on-die re-configurability and providing high performance, the STTs are also highly integrative in the CMOS fabrication process.

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

Figure 13 illustrates the proposed concept of the iterative security-driven ASIC design flow for overhead optimization.[1] The main aim of this iterative flow is to find optimal gates for obfuscation such that overheads are mitigated. As per our study, the LUT's size is the most influential factor for SAT-resiliency, and even for a random gate selection, LUT-obfuscation results in SAT timeout, as seen in Figure 7. This gives the user flexibility to choose gates, as they do not have to abide by a particular obfuscation rule or policy. Even if the gate-selection policy (*LC_NoGen*) discussed in our work is used, the proposed flow is non-disruptive to the industrial design flow. In the proposed flow, the netlist is passed as the input along. Any High-level synthesis tools or RTL-syntheses tools can be used to generate the netlist after the insertion of the novel LUT. We use Python scripts as a wrapper around industrial tools such as Synopsys DC, PrimeTime, and VCS to automate the flow. The given flow supports the netlist generated with any library and does not restrict the designer in library support. The script first creates a **Verilog Module Object (VMO)** data structure of the standard cell library, which is then used to define the synthesized netlist abstractly. The result is a complete map of the flattened netlist, defined as interconnected VMO instances. The script reads a configuration file, which contains the list of the gates to be replaced and the configuration of the LUT to generate the **Configurable Logic Elements (CLE)**. The gates to be replaced are given by the *LC_NoGen*, or by the user. The configuration of a given LUT is defined by the number of inputs to both the primary and preceding LUT. The generated CLE is added to the standard cell VMO collection and replaces the VMO instances of the gates in

[1]Gray part in Figure 13 is iterative to obtain the best result.

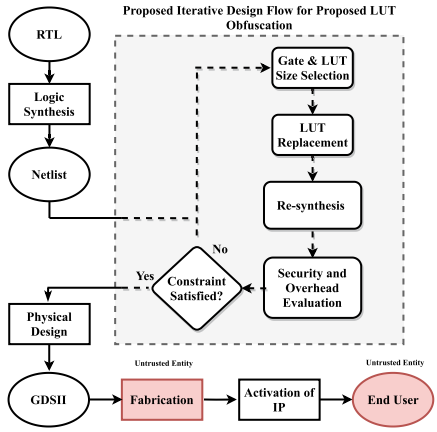

Fig. 13. Iterative-based security-driven design flow for PPA optimization with optimal security solution.

the top layer selected for the replacement. The program creates and runs an exhaustive test bench to extract the individual CLE configuration keys in parallel using a logic simulator (i.e., VCS). The output configuration keys are then chained together to form the top-level configuration key. The script finally generates RTL blocks that are inserted within the synthesized netlist, and after a final synthesis stage, the synthesized obfuscated design is provided along with its configuration key.

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

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

To verify logical equivalency between the original target benchmark and the obfuscated version, we use the Synopsys *Formality*, which is a formal verification tool. Moreover, to avoid missing functional bugs introduced by altering the gate-level netlist, the test benches are also designed such that the logical function provided by the target gates is exercised. In the case of the LUT-mapped design, an initialization task is required to load the configuration key. Once loaded, the original testbench can be run. This initialization-dependent testbench is run after re-synthesis as part of the constraint checking phase depicted in Figure 13.

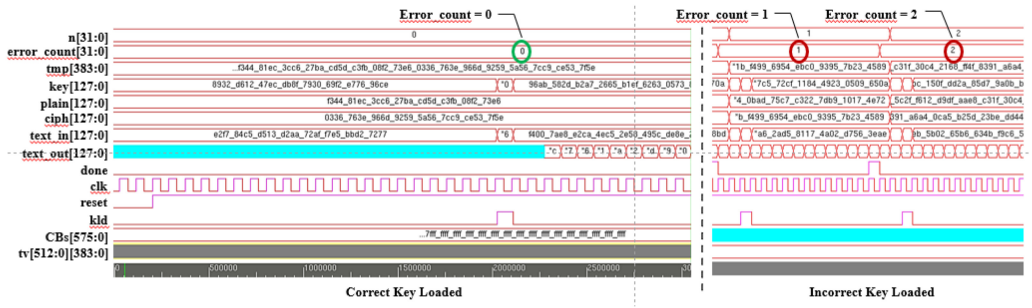

Fig. 14. Properly configured AES encryption benchmark, with successful validation on the left. Error count is 0, when the correct keys are loaded. With the improperly configured AES encryption benchmark, we can see the error count to be a non-zero number on the right.

As discussed earlier, implementing large LUT sizes adds an enormous amount of PPA overheads. However, using the proposed LUT-based obfuscation, we can use smaller LUT sizes than the LUT sizes required in traditional obfuscation methods. The reason being, the resiliency per area offered by the proposed LUT is far superior to that of the traditional LUT designs.

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

As a case study, Figure 14 shows the waveform of a successful test of an AES encryption benchmark after obfuscation has been completed. The gates in the design are replaced with 2 LUT of size 8, preceded by LUT of size 2 at each input. This accounts for a total of

$$2 \times (2^8 \ primary\ keys + 8 \times 2^2 \ input\ keys) = 576 \ keys$$

This resulting output is an obfuscated netlist with the keys as top-level inputs. This particular benchmark's testbench applies hundreds of directed test vectors (encryption key and input text combinations) and then compares the resulting cipher-text to the expected cipher-text (also included in the test vector). In this design, an incorrectly configured obfuscation key made all test vectors fail, as shown in Figure 14. Upon a test vector failure, the error count is incremented.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

To explore the design space and determine the impact of LUT size, the number of LUTs, and the replacement strategy, we used a cluster computing environment with 53 Dell computing nodes, each with dual Intel Xeon CPUs. The total number of cores ranging from 16 to 24, with RAM varying from 64 to 512 GB.

For the experimental evaluation, we use benchmarks from ISCAS-85,[2] ISCAS-89,[3] and Common Evaluation Platform (CEP).[4] The benchmarks are listed as part of Table 2. The CEP is a system on a chip design that represents typical microelectronics used by the Department of Defense (DoD) and includes instrumentation and government-specific benchmarks. These benchmarks are synthesized and flattened using the Synopsys DC Compiler.

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

Table 2. Benchmarks used for Experimentation with their Gate Counts

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

The goal of the adversary is to retrieve the key to unobfuscated the IP using SAT-attack. We use SMT-attack [4], which is not only the super-set of SAT-attack but also the newly developed state-of-the-art attack. The newly developed SMT attack uses a primary SAT-solver with extra theory solvers. Combining two solvers allows the SMT-solver to a model more complex problem, thus resulting in a strong attack. The SMT attack tries to find the correct key (LUT-configuration bits in the context of LUT-based obfuscation) that can restore the circuit's correct functionality. **We consider the runtime of the SAT attacks as an empirical yet essential metric for security evaluation of the obfuscated circuit.** Each SAT-runtime presented in work is the averaged run of 10 different SAT-solver execution. The obfuscation technique aims to render maximum security at the lower PPA overheads, and we show that the proposed LUT can deliver the security at minimal PPA footprints, increasing the viability of the reconfigurable-based obfuscation.

We empirically track and explore the execution time of the SMT solver by sweeping the size of large LUT in novel LUT from 4 to 7 to demonstrate the impact of novel LUT for security design. Also for SMT attack, a run-time limit of 30 days ($2{,}592 \times 10^3$ s) is set to demonstrate *time-out* states. The time-out state of 30 days is chosen to demonstrate how we can break the trade-off between the security and the imposed PPA compared with traditional LUT. This is the first work that has extended SAT-execution to 30 days for empirically proving the security offered by novel LUT.

For this experiment, the identified gates are replaced with STT-LUT technology to produce an obfuscated netlist as described in Section 4. For the overhead estimation, we again used Synopsys DC with the TSMC 65 nm technology library. The reported overhead in this work is reported with reference to the unobfuscated design and includes the overhead incurred due to the scan-chain mechanism used to load the values in MTJs. With the automation provided using the *ASIC Iterative Security-driven Design Flow*, the delay overhead is eliminated in almost all the cases for novel LUT-based obfuscation, and thus only area and power overhead are discussed in fine granularity. In the iterative flow, the *LC_NoGen* algorithm gets the timing report from the PrimeTime and removes the nodes from the graph, which is used for finding the gates for obfuscation. The SAT attack/SMT attack does not support the Verilog files for the SAT simulation. Therefore, the in-house developed python script was developed that converts the obfuscated Verilog files to the SAT-supported bench files.

5.2 Security Analysis Against SAT-based Attack

Figure 15(a) illustrates the design space exploration performed on "C7552" benchmark by leveraging novel LUT-based obfuscation. By varying the size and the number of LUTs used for novel LUT-based obfuscation, the de-obfuscation time against the SMT-solver is plotted. The SMT-solver time-out states can be rendered using traditional LUT-based obfuscation by obfuscating 14 gates with LUT size 8. LUT size 8 is considered per the experimental results obtained in Figure 9; recall that the combination of obfuscating 14 gates with LUT size 8 renders the lowest PPA with time-out state for SAT solver among all the configurations tested in Figure 9. However, with the proposed LUT. replacing just over 2 gates with LUT_7 + 7:LUT_2[5] or replacing 6 gates with LUT_6 + 6:LUT_2

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

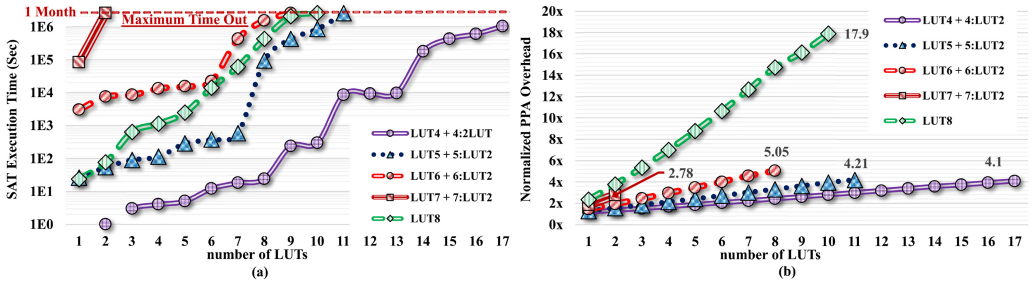

Fig. 15. Comparison of the proposed primitive where the size of the LUT is varied from 4 to 7 with the traditional LUT-based obfuscation. Panel (a) shows the de-obfuscation time of various obfuscated samples using SMT-solver and panel (b) shows the PPA incurred by the proposed primitive [14].

renders the obfuscation resiliency to create time-out scenario. This increased resiliency with the smaller size and number of the LUT breaks the trade-off between PPA and security. This finding justifies the fact that the resiliency of the novel LUT provided by LUT sizes 6 and 7 is on par with that of the traditional LUT with size 8. This added security is due to the virtue of the SAT-hard instance created by the LUT preceded by the LUT configuration. The results further show that we can yield significant computational challenges for SAT-based attacks, which grow exponentially with the increasing size of LUT.

While leveraging the small size of the LUT, such as LUT size 4 and 5 for novel LUT, the number of gates required to achieve SAT-resiliency (time-out states) is more. Nevertheless, the overheads imposed by LUT of size 4 and 5 is far less than that of LUT size 8, and thus it can be concluded that the novel LUT with size 4 and 5 provides a high ratio of security per PPA overhead footprint than the traditional LUT of size 8. It is worth noting that all instances of obfuscation using novel LUT render SAT-attack time-out, resulting in the on par resiliency level using traditional obfuscation using LUT size 8. Figure 15(b) validates the previous statement by showing that obfuscating 14 gates with novel LUT using size 4 adds 3.58× overhead compared to 14.76× overhead added by traditional LUT of size 8.

The reduction in area and power is applicable for all of the experiments conducted as part of this work. The normalized area and power overhead from Figure 15(b) for novel LUT incurred lower overheads than the LUT obfuscation using LUT_8. It also warrants that using a large size of fewer LUT, i.e., using just 2 LUT_7 results in the lowest PPA and on par resiliency.

As the resiliency provided by novel LUT increases as the function of the size of the LUT, one should replace a few gates with large LUT sizes rather than using small LUTs in large quantities. Moreover, our results suggest that it renders in SAT-resilient obfuscation while incurring permissible overheads. These results are also consistent with the experimental results obtained from Figure 7.

To further reinforce our conclusion regarding using large LUT sizes in fewer quantities for obfuscation, we performed another experiment where the key size is constrained. The number of key bits is indicative of the overheads imposed. More key bits require more fuses to store configuration bits and also require a large selection tree. Increasing keys thus results in increased power and area overhead. By adding the constraint on the key bits, we are indirectly constraining the area and power and the number and sizes of the LUT that can be leveraged for the obfuscation. We use the "AES" benchmark for this experiment while keeping the key lengths constrained to size 110, 160, 360, and 400, respectively. The key lengths let us use LUT of size 4 up to 6, and the number of gates replaced using the LUT is shown in the Figure 16 over the bars in the graph. For

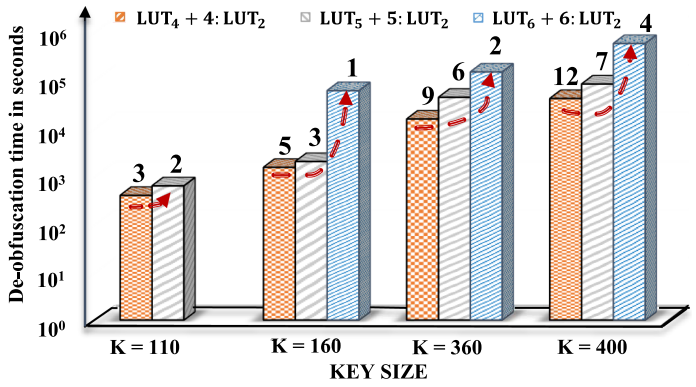

Fig. 16. SMT-solver execution time to find the unlocking key when the benchmark is obfuscated with different size and number of LUTs. The number and size of the LUTs to be replaced are determined by the key size, which is constrained in this experiment and denoted by the number above each individual bar [14].

example when the key size is 360, we can have at max 2 novel LUT_6 + 6:LUT_2 or 6 LUT_5 + 5:LUT_2 or 9 LUT_4 + 4:LUT_2. For each configuration, we plot the SMT-solver's de-obfuscation time, and it can be visualized that the time required for de-obfuscation using a large size of LUT in fewer quantities is greater in all 4 key lengths. Using just 4 novel LUTs of size 6, we encounter the time-out, which is better than leveraging 12 novel LUTs of size 4 for obfuscation. This increased resiliency is created due to the virtue of using large LUT sizes and the large MUX trees that are added to the circuit. Leveraging the large size of the LUT obfuscates the actual function in the space, which grows exponentially as the function of LUT size. When the key lengths are equal, the overhead added is roughly equal. With the same overhead footprint or key size, using the large LUT size provides maximum resiliency. Thus, one should use large LUT sizes in small quantities for obfuscation using the novel LUT. Furthermore, the experiment concludes that security grows faster than the added overhead, or the additional security comes with lower PPA overheads when the large LUT size in small quantities is used. The trade-off between security and design overhead is mitigated in the proposed novel LUT.

Figure 17 shows the power and area overhead for the different benchmarks using the proposed LUT of size 7. Size 7 is used, because LUT of size 7 resulted in the lower PPA, as seen in Figure 15(b). We omit timing results, because all designs maintained their initial timing specifications. As LUT_7 + 7:LUT_2 is the optimal PPA configuration, it incurred a small timing overhead while providing significant security performance. While we did not encounter a timing violation due to gate selection, in the event a timing violation did occur, we could select another gate that does not contribute to critical path delays. The overheads imposed by the novel LUT are compared with the traditional LUT-based obfuscation with LUT_8. LUT_8 is used for comparison, as it resulted in the SAT-resilient obfuscation while incurring the lowest PPA overhead as seen in Figure 6. Compared to the LUT_8 based on traditional obfuscation, the novel LUT with LUT_7 comes with 8× and 2× average reductions in area and power overheads without sacrificing the security. While one can argue that overheads for circuits such as "C7552" are very high, the circuit size of "C7552" is minimal (only 1,290 gates). However, with the larger circuits like "AES," "DES," or "GPS," representing real-world IPs, the incurred overheads are justifiable, making this technique a more efficient solution.

5.3 Comparison with other Obfuscation Methodologies

In this section, we compare the proposed methodology with the present state-of-the-art works. SFLL-HD [53] and Anti-SAT attack [47] are known to be SAT-resilient attacks, as the SAT-attack

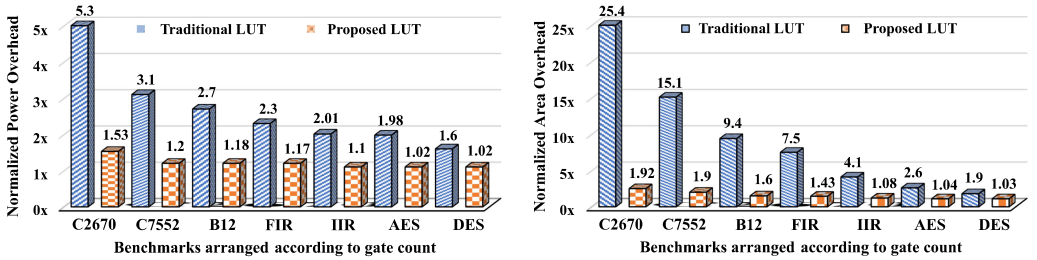

Fig. 17. Comparison of proposed LUT obfuscation with the traditional LUT-based obfuscation in terms of (a) design area overhead, (b) design power overheads. The number of LUTs and the size of the LUTs are chosen such that the obfuscated netlist results in the SMT-attack timeout with minimal PPA overheads. The iterative security-driven flow discussed in Figure 13 can be leveraged to obtain these netlists. The added overhead for both traditional and obfuscated netlist is provided in reference to each benchmark's unobfuscated version. The figures indicate that in proposed LUT-based obfuscation, the resiliency rendered by each LUT comes at much lower costs of design overheads [14]. The number on the top of the bars denotes the overheads where 1× denotes no overhead and 1.03× means 3% overhead.

Table 3. Overhead Comparison of Various Obfuscation Techniques

	Power overhead analysis					Area overhead analysis					SMT-Attack runtime (seconds)				
Benchmarks	FLL	**Proposed**	SFLL-HD	AntiSAT	InterLock	FLL	**Proposed**	SFLL-HD	AntiSAT	InterLock	FLL	**Proposed**	SFLL-HD	AntiSAT	InterLock
C2670	73%	53.27%	49.5%	62.1%	49.2%	142.2%	91.53%	91.55%	147.5%	82.5%	2468.1	∞	∞	∞	∞
C7552	52%	20.4%	24.2%	35.2%	21.2%	115.4%	91.53%	83.1%	105.1%	78.9%	4956.1	∞	∞	∞	∞
B12	31%	18.5%	20.1%	24.5%	19.9%	99.75%	60.52%	70.14%	87.1%	63.1%	8446.4	∞	∞	∞	∞
FIR	28%	17.3%	17.8%	18.6%	14%	72.95%	43.05%	40.21%	38.7%	35.9%	12224	∞	∞	∞	∞
IIR	25%	10.08%	5.4%	6.4%	9.9%	42.87%	8.44%	15.7%	11.6%	12.1%	11350	∞	∞	∞	∞
AES	11.2%	2.75%	3.8%	4.1%	2.9%	22.4%	4.94%	4.6%	2.1%	2.87%	15440	∞	∞	∞	∞
DES	9%	2.46%	2.2%	3.1%	2.1%	19.3%	3.27%	3.91%	2.8%	3.11%	17664	∞	∞	∞	∞

∞ denotes SMT-attack timeout, which is 1 month (or) 2,592,000 s.

requires an exponential amount of queries to retrieve the keys. We also compared our novel LUT against Fault Logic Locking [28]. *We obfuscate various designs such that* **SAT-attack results in a time-out state** *while trying to retrieve the keys with minimum obfuscation overhead.* We required larger key sizes for the obfuscation primitives presented here for comparison against the proposed primitive to result in time-out states. Recall that our time out is set to 30 days, while most work uses a few hours or days as their runtime.

For Fault Logic Locking, the key size used was relatively large and thus resulted in a significant overhead figure, but it still fails to resist the SAT attack. Interestingly, the time required to obfuscate the circuit is more than the de-obfuscation time. For obfuscating the circuit using FLL, we used off the shelf obfuscation tool provided by Reference [43].

From Table 3, we can observe that the SFLL-HD and AntiSAT result in about the same Area and Power overhead when compared to novel LUT-based obfuscation. This is true for the small as well as large benchmarks. However, with the increasing size of the benchmark, the overheads for all of these methods are more practical. However, it can be noted that Anti-SAT and SFLL-HD leverage a one-point function and thus has lower output corruptibility. **This makes the SFLL-HD and Anti-SAT vulnerable to many approximations SAT-attacks [34] and removal attacks [51]. Moreover, Fall-attack demonstrated in Reference [40] shows that the key for SFLL-HD can be retrieved without having access to the oracle design.**

Contrary to the compared algorithms, the proposed LUT-based obfuscation allows the user to randomly replace the gates in the circuitry, thereby increasing the output corruptibility while being resilient to SAT-attack, the approximation attack as well as removal attacks. Moreover, we compare the state-of-the-art InterLock obfuscation, which is a unified routing and logic

obfuscation technique [12]. The overhead for leveraging the Key programmable routing block size 64 is comparable to the proposed LUT-based obfuscation. It requires around 320 key programmable routing blocks where each block consists of 2 LUT_2 and 4 2:1 MUX. Moreover, using the size of 64 for yielding SAT-resiliency requires 10-stages of routing, making this obfuscation hard to deploy in a circuit. Proposed LUT-based obfuscation, however, can be deployed quickly and provides comparable overheads when compared to the state-of-the-art obfuscation primitives.

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

The proposed obfuscation was also compared against AppSAT [34]. For AppSAT, the termination criteria are determined by the error rate, which is one of the inputs to the attack. We performed 50 random queries (default setting of AppSAT) on the oracle attack after the key, which is given to us after the 20 iterations of the AppSAT. It is misleading to calculate the error rate using such a small amount of input patterns for the obfuscation with high output corruptibility [53]. When the number of queries has been increased to 1,000, AppSAT resulted in a time-out state. Nonetheless, the key given by the AppSAT, when the number of queries was varied between 50 and 1,000, did not fully unlock the obfuscated circuit.

5.4 Security Against Other Adversaries

While we show that proposed LUT-based obfuscation is resilient against SAT-based methods such as traditional SAT-attack, SMT-attack, and AppSAT, it also thwarts other hardware security attacks such as removal attacks, ATPG-based attacks, and also scan-chain-based attacks.

For ATPG-based attack, *HackTest* [54], reveals the true identity of the gates by utilizing the test data. The attack leverages the fact that the test pattern used for testing the IP generally has full coverage. Since the attacker has access to the test pattern, he/she finds the gate assignment of the LUT such that test coverage is maximized. The LUT-combination that results in reduced coverage upon applying the test patterns can be avoided, and as such, this attack is even powerful than SAT-attack. Compared to the SAT-attack, this method does not require the oracle and does not require the attacker to find the distinguishing inputs. However, leveraging the reconfigurability of the LUTs, one can generate the test patterns with maximum fault coverage for the incorrectly configured LUTs, thus retaining the ability to test the circuit. For example, if the LUT is supposed to function as "OR," then for the sake of testing, LUT can be configured to be anything but "OR," and the test patterns can be obtained and sent to the testing facility accordingly. The attacker uses the test patterns provided and will evaluate the LUT functionality to be anything but an "OR" gate. When the IP is back in the trusted regime, the true functionality of the LUT can again be restored by programming the MTJ's, and thus attacks like *HackTest* can be avoided. Further, work in Reference [29] shows that efficacy *HackTest* depends on the functions implemented by the reconfigurable gate. As the LUT used in work can implement multiple functionalities, the success of the *HackTest* is greatly reduced.

The proposed LUT-based obfuscation can also thwart the Scan and Shift attack. The Scan and Shift attack [17] tries to capture the **Secure Cell (SC)** value using the scan chain. However, in our proposed method, the mechanism uses a separate scan chain to load the values into the LUT configuration bits. By modifying this scan chain to block the scan out port, the attacker fails to retrieve the content stored in the LUT. Since this modified scan chain with a blocked scan out port

is only used to load the keys into the design, the other scan chains in the design are still fully functional and thus facilitate the IP's routine testing. Moreover, the programming of key is only done once in trusted regime, the scan chain is never used thereafter and scan-chain does not hold the key value.

5.5 Exploration of Large Benchmark

To show the efficiency of the proposed LUT insertion strategy, a large GPS benchmark (≈170,000 gates) from the CEP benchmark is chosen. The benchmark is widely used in many industrial and commercial gadgets and consists of hundreds of sub-IPs and Subsystems. The benchmark is obfuscated with the optimal LUT_7 + 7:LUT_2 LUT configuration discussed in previous sections. The obfuscation resulted in 0.4% power and 0.3% area overhead. This design achieved an SAT-attack timeout with both an area and power overhead of less than half a percent. This result affirms the trend established in the survey of small and medium-sized designs in the preceding sections: the power and area overheads required to implement the optimal SAT resilient LUT configuration amortizes as the design size increases.This experiment, along with others, shows that two LUT7 + 7: LUT2 were able to withstand SAT attack for a month and provides superior quality when compared to the state-of-the-art obfuscations while incurring permissible overheads.

6 CONCLUSIONS

In this work, we studied reconfigurable logic locking. We performed a comprehensive analysis of LUT-based obfuscation using four crucial design factors, i.e., (1) LUT technology, (2) LUT size, (3) number of LUTs, and (4) replacement strategy as they have a strong influence on the PPA overhead and security. Among the studied parameters for the LUT-based obfuscation, the LUT (LUT scale-up) size is the most influential and straightforward factor in achieving SAT resiliency. However, using a large size of LUTs for obfuscation results in hefty PPA overheads. Leveraging the STT-based LUT alleviates the PPA overheads. However, the incurred overheads still render LUT-based obfuscation an idealistic method for obfuscation. To mitigate the overhead that occurred due to large LUT, we introduce novel LUT, which breaks the trade-off between security and the design overheads. The novel LUT aims to create an SAT-hard design to reduce the design overhead without compromising security. The proposed LUT can be used with random gate placement to increase the output corruptibility without sacrificing SAT resiliency. The proposed LUT is resilient against approximate attacks, ATPG-based attacks, removal attacks, and Scan-chain-based attacks. We propose an iterative solution, a non-disruptive procedure to standard ASIC design flows that render the efficient and secure obfuscated design layout. The proposed technique is evaluated against the state-of-the-art SMT attack and removal attacks. Our experimental results show that resiliency against various attack vectors can be achieved with the proposed novel LUT-based obfuscation. Furthermore, compared to the traditional LUT-based obfuscation, nearly 2× power, and 8× area overheads can be reduced on average. These results warrant that the proposed methodology can be seamlessly integrated into IC design flow while ensuring permissible design overhead and security against today's state-of-the-art attacks.

REFERENCES

[1] Yousra M. Alkabani and Farinaz Koushanfar. 2007. Active hardware metering for intellectual property protection and security. In *Proceedings of the 16th USENIX Security Symposium on USENIX Security Symposium (SS'07)*. USENIX Association, Article 20, 16 pages. Retrieved from http://dl.acm.org/citation.cfm?id=1362903.1362923.

[2] A. Attaran, T. D. Sheaves, P. Mugula, and H. Mahmoodi. 2018. Static design of spin transfer torques magnetic look up tables for ASIC designs. In *Proceedings of the Great Lakes Symposium on VLSI*. ACM, 507–510. http://doi.acm.org/10.1145/3194554.3194651

[3] Aliyar Attaran, Tyler David Sheaves, Praveen Kumar Mugula, and Hamid Mahmoodi. 2018. Static design of spin transfer torques magnetic look up tables for ASIC designs. In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI'18)*. Association for Computing Machinery, New York, NY, 507–510. https://doi.org/10.1145/3194554.3194651

[4] K. Azar, H. Kamali, H. Homayoun, and A. Sasan. 2018. SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2019, 1 (Nov. 2018), 97–122. Retrieved from https://tches.iacr.org/index.php/TCHES/article/view/7335.

[5] A. Baumgarten, A. Tyagi, and J. Zambreno. 2010. Preventing IC piracy using reconfigurable logic barriers. *Design Test Comput.* 27, 1 (Jan. 2010), 66–75.

[6] Zhiqian Chen, Gaurav Kolhe, Setareh Rafatirad, Chang-Tien Lu, Sai Manoj P. D., Houman Homayoun, and Liang Zhao. 2020. Estimating the circuit de-obfuscation runtime based on graph deep learning. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'20)*. IEEE, 358–363. https://doi.org/10.23919/DATE48585.2020.9116544

[7] Ronald P. Cocchi, Lap Wai Chow, James P. Baukus, and Bryan J. Wang. 2013. Method and apparatus for camouflaging a standard cell based integrated circuit with micro circuits and post processing. https://patents.google.com/patent/US8510700B2/en.

[8] Rakibul Hassan, Gaurav Kolhe, Setareh Rafatirad, Houman Homayoun, and Sai Manoj Dinakarrao. 2020. SATConda: SAT to SAT-hard clause translator. In *Proceedings of the 21st International Symposium on Quality Electronic Design (ISQED'20)*. IEEE, 155–160. https://doi.org/10.1109/ISQED48828.2020.9137052

[9] Rakibul Hassan, Gaurav Kolhe, Setareh Rafatirad, Houman Homayoun, and Sai Manoj Pudukotai Dinakarrao. 2021. A neural network-based cognitive obfuscation towards enhanced logic locking. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* (2021), 1. https://doi.org/10.1109/TCAD.2021.3138686

[10] Frank Imeson, Ariq Emtenan, Siddharth Garg, and Mahesh V. Tripunitara. 2013. Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation. In *Proceedings of the 22nd USENIX Conference on Security (SEC'13)*. USENIX Association, 495–510. Retrieved from http://dl.acm.org/citation.cfm?id=2534766.2534809.

[11] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe. 1998. Watermarking techniques for intellectual property protection. In *Proceedings of the 35th Design and Automation Conference (DAC'98)*. IEEE, 776–781. https://doi.org/10.1145/277044.277240

[12] Hadi Mardani Kamali, Kimia Zamiri Azar, Houman Homayoun, and Avesta Sasan. 2020. InterLock: An intercorrelated logic and routing locking. In *Proceedings of the 39th International Conference on Computer-Aided Design (ICCAD'20)*. Association for Computing Machinery, New York, NY, Article 78, 9 pages. https://doi.org/10.1145/3400302.3415667

[13] R. Karri, J. Rajendran, K. Rosenfeld, et al. 2010. Trustworthy hardware: Identifying and classifying hardware trojans. *Computer* 43, 10 (Oct. 2010), 39–46.

[14] Gaurav Kolhe, Hadi Mardani Kamali, Miklesh Naicker, Tyler David Sheaves, Hamid Mahmoodi, Sai Manoj P. D., Houman Homayoun, Setareh Rafatirad, and Avesta Sasan. 2019. Security and complexity analysis of LUT-based obfuscation: From blueprint to reality. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'19)*. IEEE, 1–8. https://doi.org/10.1109/ICCAD45719.2019.8942100

[15] Gaurav Kolhe, Sai Manoj P. D., Setareh Rafatirad, Hamid Mahmoodi, Avesta Sasan, and Houman Homayoun. 2019. On custom LUT-based obfuscation. In *Proceedings of the Great Lakes Symposium on VLSI (GLSVLSI'19)*. Association for Computing Machinery, New York, NY, 477–482. https://doi.org/10.1145/3299874.3319496

[16] Gaurav Kolhe, Soheil Salehi, Tyler David Sheaves, Houman Homayoun, Setareh Rafatirad, Sai Manoj P. D., and Avesta Sasan. 2021. Securing hardware via dynamic obfuscation utilizing reconfigurable interconnect and logic blocks. In *Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC'21)*. IEEE, USA, 229–234. https://doi.org/10.1109/DAC18074.2021.9586242

[17] N. Limaye, A. Sengupta, M. Nabeel, and O. Sinanoglu. 2019. Is robust design-for-security robust enough? Attack on locked circuits with restricted scan chain access. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'19)*. IEEE, 1–8. https://doi.org/10.1109/ICCAD45719.2019.8942047

[18] B. Liu and B. Wang. 2014. Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'14)*. IEEE, 1–6. https://doi.org/10.7873/DATE.2014.256

[19] M. Mohamed, S. Garg, and M. V. Tripunitara. 2015. Integrated circuit decamouflaging: Reverse engineering camouflaged ICs within minutes. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'15)*.

[20] Hadi Mardani Kamali, Kimia Zamiri Azar, Kris Gaj, Houman Homayoun, and Avesta Sasan. 2018. LUT-Lock: A novel LUT-based logic obfuscation for FPGA-bitstream and ASIC-hardware protection. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'18)*. IEEE, 405–410. https://doi.org/10.1109/ISVLSI.2018.00080

[21] Vivek V. Menon, Gaurav Kolhe, Andrew Schmidt, Joshua Monson, Matthew French, Yinghua Hu, Peter A. Beerel, and Pierluigi Nuzzo. 2019. System-level framework for logic obfuscation with quantified metrics for evaluation. In *Proceedings of the IEEE Cybersecurity Development (SecDev'19)*. IEEE, 89–100. https://doi.org/10.1109/SecDev.2019.00020

[22] David Mitchell, Bart Selman, and Hector Levesque. 1992. Hard and easy distributions of SAT problems. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI'92)*. AAAI Press, 459–465.

[23] E. Nudelman, K. Leyton-Brown, H. H. Hoos, et al. 2004. Understanding random SAT: Beyond the clauses-to-variables ratio. In *Proceedings of the 10th International Conference on Principles and Practice of Constraint Programming (CP'04)*. Springer-Verlag, Heidelberg, 438–452. https://doi.org/10.1007/978-3-540-30201-8_33

[24] S. Patnaik, N. Rangarajan, J. Knechtel, et al. 2018. Advancing hardware security using polymorphic and stochastic spin-hall effect devices. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition*. IEEE, Germany, 97–102.

[25] J. Rajendran, Y. Pino, O. Sinanoglu, et al. 2012. Security analysis of logic obfuscation. In *Proceedings of the Design Automation Conference*. IEEE, 83–89.

[26] J. Rajendran, M. Sam, O. Sinanoglu, et al. 2013. Security analysis of integrated circuit camouflaging. In *Proceedings of the ACM SIGSAC Conference on Computer Communications Security (CCS'13)*. ACM, 709–720. http://doi.acm.org/10.1145/2508859.2516656

[27] J. Rajendran, H. Zhang, C. Zhang, et al. 2015. Fault analysis-based logic encryption. *IEEE Trans. Comput.* 64, 2 (Feb 2015), 410–424.

[28] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri. 2015. Fault analysis-based logic encryption. *IEEE Trans. Comput.* 64, 2 (Feb. 2015), 410–424. https://doi.org/10.1109/TC.2013.193

[29] N. Rangarajan, S. Patnaik, J. Knechtel, R. Karri, O. Sinanoglu, and S. Rakheja. 2022. Opening the doors to dynamic camouflaging: Harnessing the power of polymorphic devices. *IEEE Trans. Emerg. Top. Comput.* 10, 1 (2022), 137–156. https://doi.org/10.1109/tetc.2020.2991134

[30] A. Rezaei, Y. Shen, S. Kong, et al. 2018. Cyclic locking and memristor-based obfuscation against CycSAT and inside foundry attacks. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition*. IEEE, 85–90.

[31] S. Roshanisefat, H. Mardani Kamali, and A. Sasan. 2018. SRCLock: SAT-resistant cyclic logic locking for protecting the hardware. In *Proceedings of the Great Lakes Symposium on VLSI*. ACM, 153–158. http://doi.acm.org/10.1145/3194554.3194596

[32] M. Rostami, F. Koushanfar, and R. Karri. 2014. A primer on hardware security: Models, methods, and metrics. *Proc. IEEE* 102, 8 (Aug. 2014), 1283–1295. https://doi.org/10.1109/JPROC.2014.2335155

[33] J. A. Roy, F. Koushanfar, and I. L. Markov. 2010. Ending piracy of integrated circuits. *Computer* 43, 10 (Oct. 2010), 30–38. https://doi.org/10.1109/MC.2010.284

[34] K. Shamsi, M. Li, T. Meade, et al. 2017. AppSAT: Approximately deobfuscating integrated circuits. In *Hardware Oriented Security and Trust*. IEEE, 95–100.

[35] K. Shamsi, M. Li, T. Meade, et al. 2017. Cyclic Obfuscation for creating SAT-unresolvable circuits. In *Proceedings of the Great Lakes Symposium on VLSI*. ACM, 173–178. http://doi.acm.org/10.1145/3060403.3060458

[36] Sanket Shukla, Gaurav Kolhe, Sai Manoj P. D., and Setareh Rafatirad. 2019. MicroArchitectural events and image processing-based hybrid approach for robust malware detection: Work-in-progress. In *Proceedings of the International Conference on Compliers, Architectures and Synthesis for Embedded Systems Companion (CASES'19)*. Association for Computing Machinery, New York, NY, Article 10, 2 pages. https://doi.org/10.1145/3349569.3351538

[37] Sanket Shukla, Gaurav Kolhe, Sai Manoj P. D., and Setareh Rafatirad. 2019. Stealthy malware detection using RNN-based automated localized feature extraction and classifier. In *Proceedings of the IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI'19)*. IEEE, USA, 590–597. https://doi.org/10.1109/ICTAI.2019.00088

[38] Sanket Shukla, Gaurav Kolhe, Sai Manoj P. D., and Setareh Rafatirad. 2019. RNN-based classifier to detect stealthy malware using localized features and complex symbolic sequence. In *Proceedings of the 18th IEEE International Conference On Machine Learning And Applications (ICMLA'19)*. IEEE, 406–409. https://doi.org/10.1109/ICMLA.2019.00076

[39] Sanket Shukla, Sai Manoj P. D., Gaurav Kolhe, and Setareh Rafatirad. 2021. On-device malware detection using performance-aware and robust collaborative learning. In *Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC'21)*. IEEE, 967–972. https://doi.org/10.1109/DAC18074.2021.9586330

[40] D. Sirone and P. Subramanyan. 2019. Functional analysis attacks on logic locking. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'19)*. IEEE, Italy, 936–939.

[41] J. P. Skudlarek, T. Katsioulas, and M. Chen. 2016. A platform solution for secure supply-chain and chip life-cycle management. *Computer* 49, 8 (Aug. 2016), 28–34. https://doi.org/10.1109/MC.2016.243

[42] Mate Soos, Karsten Nohl, and Claude Castelluccia. 2009. Extending SAT solvers to cryptographic problems. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing (SAT'09) (Lecture Notes in Computer Science, Vol. 5584)*, Oliver Kullmann (Ed.). Springer, 244–257. https://doi.org/10.1007/978-3-642-02777-2_24

[43] P. Subramanyan, S. Ray, and S. Malik. 2015. Evaluating the security of logic encryption algorithms. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust*. IEEE, 137–143.

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

[45] T. Winograd, H. Salmani, H. Mahmoodi, et al. 2016. Hybrid STT-CMOS designs for reverse-engineering prevention. In *Proceedings of the 53nd ACM/EDAC/IEEE Design Automation Conference*. IEEE, 1–6.

[46] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor. 2016. Hardware trojans: Lessons learned after one decade of research. *ACM Trans. Des. Autom. Electron. Syst.* 22, 1, Article 6 (May 2016), 23 pages.

[47] Y. Xie and A. Srivastava. 2019. Anti-SAT: Mitigating SAT attack on logic locking. In *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 38. IEEE, 199–207.

[48] J. Yang, X. Wang, Q. Zhou, et al. 2019. Exploiting spin-orbit torque devices as reconfigurable logic for circuit obfuscation. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 38, 1 (Jan. 2019), 57–69.

[49] M. Yasin, B. Mazumdar, J. Rajendran, et al. 2016. SARLock: SAT attack resistant logic locking. In *Hardware Oriented Security and Trust*. IEEE, 236–241.

[50] M. Yasin, B. Mazumdar, O. Sinanoglu, et al. 2016. Camoperturb: Secure IC camouflaging for minterm protection. In *Proceedings of the 35th International Conference on Computer-Aided Design*. ACM, 29.

[51] M. Yasin, B. Mazumdar, O. Sinanoglu, et al. 2017. Security analysis of anti-SAT. In *Proceedings of the Asia and South Pacific Design Automation Conference*. IEEE, Japan, 342–347.

[52] Muhammad Yasin, Abhrajit Sengupta, Mohammed Thari Nabeel, Mohammed Ashraf, Jeyavijayan Rajendran, and Ozgur Sinanoglu. 2017. Provably-secure logic locking: From theory to practice. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. ACM, New York, NY, 1601–1618. https://doi.org/10.1145/3133956.3133985

[53] M. Yasin, A. Sengupta, M. Thari Nabeel, et al. 2017. Provably-secure logic locking: From theory to practice. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1601–1618.

[54] M. Yasin, O. Sinanoglu, and J. Rajendran. 2017. Testing the trustworthiness of IC testing: An oracle-less attack on IC camouflaging. *IEEE Trans. Info. Forensics Secur.* 12, 11 (2017), 2668–2682. https://doi.org/10.1109/TIFS.2017.2710954

Received May 2021; revised September 2021; accepted January 2022