EvoLUTe: Evaluation of Look-Up-Table-based Fine-Grained IP Redaction

Rui Guo*, M Sazadur Rahman*, Hadi M Kamali, Fahim Rahman, Farimah Farahmandi, Mark Tehranipoor
Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA.
{guor, mohammad.rahman, h.mardanikamali}@ufl.edu, {fahimrahman, farimah, tehranipoor}@ece.ufl.edu

***Abstract*—Recent studies on intellectual property (IP) protection techniques demonstrate that engaging embedded reconfigurable components (e.g., eFPGA redaction) would be a promising approach to concealing the functional and structural information of the security-critical design. However, detailed investigation reveals that such techniques suffer from almost prohibited overhead in terms of area, power, delay, and testability. In this paper, we introduce *EvoLUTe*, a distinct and significantly more fine-grained redaction methodology using smaller reconfigurable components (such as look-up-tables (LUTs)). In *EvoLUTe*, we examine both eFPGA-based and LUT-based design spaces, demonstrating that a novel cone-based and fine-grained universal function modeling approach using LUTs is capable of providing the same degree of resiliency at a much lower area/power/delay and testability costs.**

***Index Terms*—Logic Locking, Re-configurability, eFPGA IP redaction, Universal Function, IP Protection.**

*: Equivalent Contribution by the first and the second author.

I. Introduction

Due to the complexity of modern System-on-Chips (SoCs) and the prevalence of design reuse, it is common for IC designers to license high-level intellectual property (IP) cores in the form of soft (RTL), firm (netlist), or hard (GDSII) IPs for their SoC designs [1]. As a result of globalization, costs and time-to-market is improved. However, the inclusion of third-party IPs (3PIPs) in a cooperative model involves multiple entities around the world and introduces multiple security threats at different semiconductors IC supply chains stages, such as IP piracy, Trojan insertion, counterfeiting, and overproduction [2]. In response to these security threats, active design-for-trust (DfTr) strategies have become increasingly prominent since passive countermeasures like patents, copyrights, and watermarking have failed to provide any protection. Amongst them, the logic locking [3] is an effective solution, and its main principle is to insert additional key gates in the circuit to achieve the purpose of hiding the function of IP/IC. Typically after the chip is manufactured, key inputs are configured by tamper-proof memories on the chip. As a result, the design will only work properly if the correct key input value is provided.

Over the last decade, logic locking has made tremendous advances [4]. However, all of them are vulnerable to *Boolean satisfiability* (SAT)-based key extraction attack [5]. Several SAT-resistant logic obfuscations have recently been proposed, including point-function based [6], [7] and routing-based [8], [9]. However, logical locking techniques are often vulnerable to other attacks, either structural [10]–[12] or ML-based [13], [14], showing that the adversary can effectively define different attack procedures. Also, sequential or timing-based logic locking of IP/IC has also been investigated in the literature [15], [16]. Nevertheless, these countermeasures are vulnerable to other derivatives of the SAT attacks [17], [18], showing the critical need to develop countermeasures to such attacks, particularly SAT and its derivatives.

In recent years, a coarse-grained form of logic locking has gained momentum, in which SoC-level redaction is performed using an embedded FPGA (eFPGA) [19]–[22]. In this case, the user selects the whole [19] or a portion [20], [21] of the asset IP (e.g., security-critical modules) for redaction and implements them in an eFPGA embedded into the SoC. The bitstream that programs the eFPGA becomes the secret key, and the attacker must restore the complete bitstream to make the whole system and the eFPGA function correctly. Concurrently, researchers challenged the practical applicability of traditional logic locking methods considering the plethora of attacks [23], [24] and formally proved that universal circuits [23] can only achieve the indistinguishable security of logic locking at the cost of $O(nlogn)$ area expansion. The area overhead of the proposed eFPGA-based IP redaction methods [19]–[22] also validates the required key size [23]. Hence the challenges with eFPGA-based IP protection greatly lie in optimizing the area. In this paper, we propose *EvoLUTe*, a unique and completely fine-grained IP redaction method using unit reconfigurable components, such as look-up tables (LUTs). In *EvoLUTe*, as its contribution is listed below, we deploy an in-house developed conflict coloring graph-based circuit splitting tool that dissects the target RTL IP into sub-circuits and replaces the selected sub-circuits using LUTs:

(1) We propose the first ever fine-grained IP redaction method using universal circuits. Unlike the existing methods that select the whole or modules of the asset IP, we redact individual logic cones and replace them with reconfigurable LUTs.

(2) We develop a graph-based circuit splitting tool that disintegrates the asset IP into smaller sub-circuits and utilizes a conflict sub-circuit coloring method to efficiently select which sub-circuits should be redacted for better SAT resiliency.

(3) We vary the number of sub-circuit replacements by EvoLUTe to achieve a timeout (24 hours) and compare the achieved area overhead with that of eFPGA-based IP redaction. Our results show that EvoLUTe can achieve *2 orders of magnitude* area gain over eFPGA.

978-3-9819263-7-8/DATE23/© 2023 EDAA

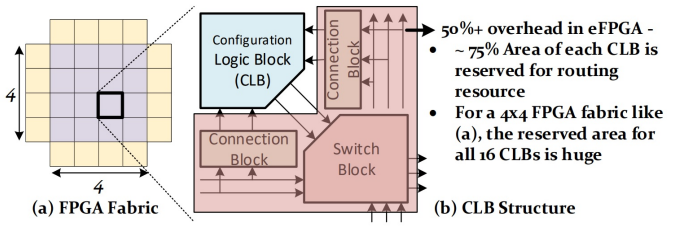

Fig. 1: eFPGA Fabric Area Overhead. (a) Simplified View of FPGA fabric, (b) CLB Structure (75% area of CLB is the reconfigurable routing).

II. Background

A. Threat Model

Similar to the eFPGA-based redaction technique, to assess the robustness of our proposed approach, we employ the traditional Oracle-guided threat model by conducting a SAT attack [5], in which (1) The adversary (either an untrusted foundry or a malicious end-user) has access to the locked netlist. The locked netlist can be obtained by either extracting gate-level netlists from the GDSII or de-processing and reverse-engineering the chip. (2) To generate correct outputs, the adversary must have access to one unlocked IC (oracle). It is possible to acquire such an IC from the open market, an in-field system, or a rogue insider within a supply chain. (3) In addition, scan chains can be used to partition sequential circuits into combinations of smaller circuits, then the attacker can use SAT attacks independently on each. In this strong threat model, an attacker has no restrictions on information other than the correct unlocking key.

B. Programmable Fabrics-Based Redaction

Researchers have explored programmable fabrics to improve the security of obfuscation strategies, primarily as a countermeasure against state-of-the-art attacks, including SAT (and its variants), structural-based attacks, machine learning (ML)-based attacks, etc. [4]. As stated in the previous section, the designer implements only a portion of the design into a programmable fabric in this technique. At the same time, the remaining part follows a conventional ASIC design and fabrication flow. As shown in Fig. 1(a), eFPGAs consist of Configurable Logic Blocks (CLBs) that contain look-up tables (LUTs), flip-flops, and routing logic (MUX-based crossbars) programmed to perform any function. To implement configurable routing, FPGAs contain MUX-based switch boxes and connection blocks. This reconfigurability allows the designer to configure the bitstream to implement critical parts of the design after manufacturing, where the bitstream is the key to the design. In this way, the design can be manufactured by any untrusted offshore foundry without the eFPGA being programmed, which makes the overall design obfuscated. Once the design is fabricated, a trusted facility programs the eFPGA to recover the correct functionality of the whole design.

C. Inapplicability of eFPGA-based Logic Redaction

Redactions that utilize programmable fabric-based technologies pose many challenges to designers, such as customizing/selecting fabric configurations, deciding which blocks of IP should be implemented on programmable fabrics, as well as the overhead involved in integration and implementation. [22] takes the approach of selecting to redact from a high-level (C-based) design to maximize safety metrics until an area overhead threshold is reached. [19] implement the generation of the eFPGA fabric by selecting individual modules of the design. Considering the trade-offs between security and other design factors such as area and time, in IP redaction, the selected module ("redaction module") needs to fit into the eFPGA fabric. So, the designer needs to know the resources available for a specific fabric size, and limited resources make it impossible for the designer to choose a target arbitrarily. Alternatively, after determining the redaction module, select the fabric that can meet the size requirements. However, the minimum fabric size can be affected by different factors. The interface of the module (number of inputs and outputs) will affect the number of I/O blocks required, while the number of state elements (registers/flip-flops) will affect the number of CLBs. Either factor will cause the size of the final eFPGA to change, resulting in low utilization of the internal resources of the structure for redaction. Furthermore, eFPGA-based coarse-grained reconfigurability has a significant impact on the area since the number of CLBs and I/O blocks grow nonlinearly with eFPGA fabric size. When a designer makes one of the CLB or I/O blocks meet the minimum requirements, there is often a waste of space for the other. Further, a large amount of area must be dedicated to routing, as shown in Fig. 1, resulting in a smaller effective logic area ratio.

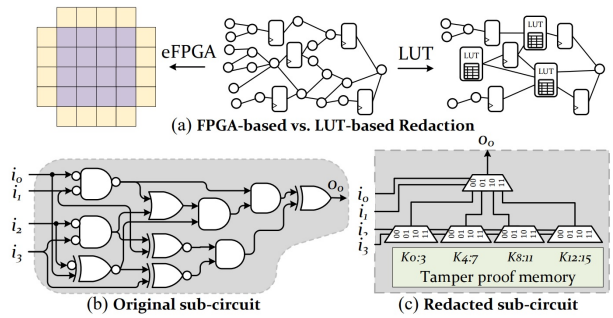

Fig. 2: Cone-based Logic Redaction in EvoLUTe. (a) FPGA-based vs. LUT-based redaction, (b) A Sub-circuit instance, (c) MUX-based LUT counterpart of the Sub-circuit in EvoLUTe.

III. EvoLUTe: Fine-Grained Redaction

Fig. 2 shows a big picture of the differences between eFPGA-based and LUT-based redaction. From eFPGA (consisting of hundreds to thousands of LUTs) to individual LUTs, coarse granularity to fine granularity can be realized. Fig. 2(b) and 2(c) show how each logic sub-circuit will be mapped to a LUT. In this case, the sub-circuit with 4/1 inputs/output will be redacted by a LUT-4 with 16 configuration bit ($K_{0:15}$), and similar to eFPGA-based redaction, this configuration is treated as the secret stored in tamper-proof memory. When eFPGA-based redaction is in place, hundreds of these (from logic to LUT) conversions will be accomplished surrounded by a huge routing crossbar. However, the main aim of EvoLUTe is to demonstrate that **instead of mapping a module/IP into an**

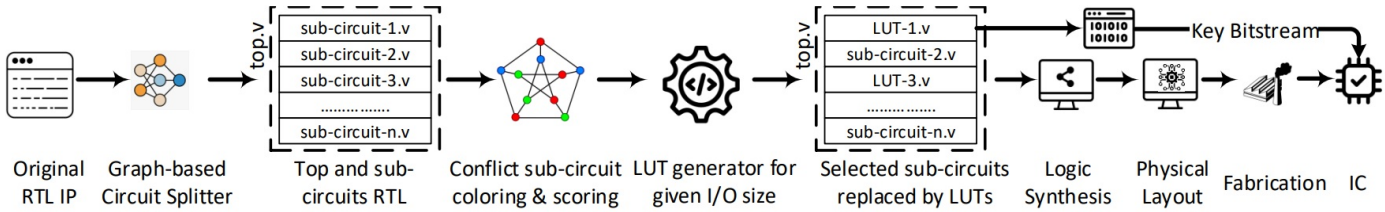

Fig. 3: IC Design Flow with EvoLUTe Powered LUT-based Selective Redaction of Subcircuits.

eFPGA fabric, it can be reduced to the sub-circuits of modules/IPs with selective cone-to-LUT redaction.

Compared to eFPGA-based redaction, acting selectively for LUT-based redaction allows us to have a much more area-efficient redaction. For instance, in Fig. 2, a sub-circuit of ten gates is replaced by a MUX-based universal circuit of only five gates (∼50% area gain). We investigated several benchmarks from [25], which boils down to the following observations: (i) Splitting the original IP into fine-grained smaller sub-circuits keeps the number of inputs to the sub-circuits at a minimum. Therefore, the size of the replacement universal circuit can be smaller than the original sub-circuit, and eventually, the IP redaction may reduce area (further validated in Section V). (ii) As the number of replacements increases, the redacted IP becomes more SAT resilient (stronger obfuscation). However, the area gain from individual replacement adds up and can eventually increase the overall area overhead. Therefore, a "break-even" point must be achieved for expected resiliency.

A. Overview of EvoLUTe Flow

EvoLUTe aims to split the original circuit as fine-grained as possible so that a number of sub-circuits can be selected for LUT-based redaction, thereby reducing area overhead while keeping the SAT resiliency. The whole process consists of three main parts - (i) circuit splitter, (ii) conflict circuit coloring and scoring, and (iii) reconfigurability generation. Fig. 3 shows a high-level overview of EvoLUTe. It begins with the processing of the original RTL of the IP asset. As highlighted previously, splitting the original IP into smaller sub-circuits keeps the number of inputs to the sub-circuits small, which can make the replacement universal circuit smaller than the original sub-circuit. Hence, EvoLUTe deploys a graph-based circuit splitting tool that disintegrates the original RTL IP into smaller sub-circuits in the form of RTLs and creates an RTL in which the sub-circuits are instantiated to reconstruct the whole IP functionality (Section III-B).

To achieve IP redaction-based hardware obfuscation while keeping the area cost low, EvoLUTe proposes a selected number of sub-circuits to be replaced by MUX-based LUTs as shown in Fig. 2. To select which sub-circuits to be replaced, EvoLUTe performs a conflict graph coloring to identify sub-circuits with non-interference. This approach ensures that the attacker encounters each sub-circuit individually and cannot gain any leverage on another sub-circuit by compromising one. Later, EvoLUTe assigns a score to each non-interference sub-circuit based on their interaction with the other sub-circuits. A higher score defines more interaction with the neighboring sub-circuits (Section III-C). An in-house developed python parser is then used to generate the MUX-tree-based universal circuit for the given I/O bandwidth of the sub-circuits and their associated bitstream (key inputs in Fig. 2(b)). After that, the top-level RTL generated by the circuit splitter tool is utilized to integrate all the sub-circuits and LUTs to realize the whole IP. Later, conventional ASIC design flow is carried out. Once individual ICs are packaged, a bitstream generated earlier is used to unlock the LUT-based redacted IP functionality.

It is worth mentioning that, to the best of the authors' knowledge, the state-of-the-art LUT-based obfuscation techniques have concentrated on the gate to LUT replacement (individual logic gate to individual LUT) that incurs significant overhead [26]. However, EvoLUTe focuses on sub-circuit (logic cone) to LUT replacement by using the proposed splitter and conflict graph coloring techniques, which consistently improve resiliency per incurred overhead.

B. Graph-based Circuit Splitter

EvoLUTe splits a circuit into several sub-circuits and replaces one or more with MUX-tree-based LUTs. After the initialization step (defining splitting cutting edges and building sub-RTLs), the circuit splitter merges the sub-circuits with their adjacent sub-circuits iteratively. EvoLUTe accomplishes these steps very efficiently to maximize the scalability of the framework. To do that, the processing is accomplished in $O(n)$ (n: sub-circuits) where each sub-circuit is processed at most once in an iteration. Additionally, it is enabled with a straightforward structural and functional verification, to guarantee the correctness of split and merge. The left side of Fig.4 shows the detailed steps of the splitter, and Alg. 1 demonstrates the details of the splitting algorithm.

The splitting in EvoLUTe (Alg. 1) has been accomplished iteratively to support multiple degrees of (fine-)granularity. It starts from the initial iteration ($iter_0$) to the final iteration ($iter_N$), in each the granularity is in a specific range. From

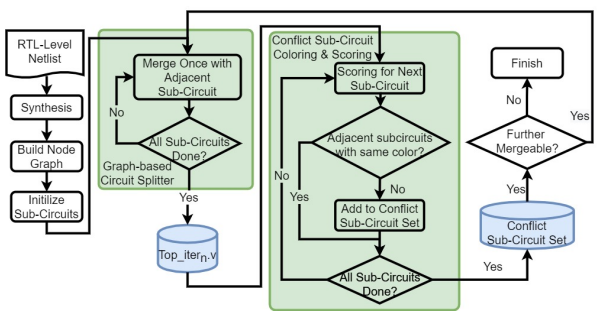

Fig. 4: Circuit Splitter & Conflict Sub-circuit Coloring & Scoring in EvoLUTe.

iter$_0$ to (*iter*$_N$), the size of each sub-circuit will be increased, and accordingly, the number of sub-circuits that built the whole circuit will decrease. So, from *iter*$_0$ to (*iter*$_N$), splitting provides a trend from fine-granularity to coarse-granularity. Fig. 5 shows an example of iterations (*iter*$_{1-3}$) on *ISCAS-85 c17*. As shown, for *iter*$_1$ $\rightarrow$ *iter*$_3$, the number of gates per each sub-circuit is {2-3}, {5-6}, and 11, respectively, showing that granularity is getting coarse. On the other side, with coarse-granularity, we see that the number of sub-circuits that build the whole circuit decreases from 4 to 2 and then from 2 to 1. These iterations draw the granularity range from excessively fine-grained (*iter*$_0$) to excessively coarse-grained (*iter*$_3$) splitting. In literature, the initial iteration (*iter*$_0$), where each sub-circuit is an individual gate, is used for LUT-based obfuscation, and the final iteration (*iter*$_N$), where the sub-circuit is the whole IP/module, is used for eFPGA-based redaction. In EvoLUTe, we initiate an exploration by redaction of sub-circuits from *iter*$_{1-2}$, showing that the definition of granularity can affect the overhead significantly while it still guarantees robustness[1].

C. Sub-circuits Coloring and Scoring

Moving from coarse-grained to fine-grained redaction opens the possibility of acting selectively for the redaction point. So, after splitting, we engage a unique strategy selection in EvoLUTe based on graph coloring [27] to find the suited sub-circuits, called redaction-oriented conflict graph coloring (*RoCGC*). Two sub-circuits have conflict in EvoLUTe if they are directly connected. We use *RoCGC* to color sub-circuits in a way that sub-circuits with a conflict must have different colors. Fig. 5 shows how coloring may be applied on *ISCAS-85 c17* throughout different iterations. The main aim of *RoCGC* is to prioritize LUT-based redaction. This conflict graph coloring method helps identify the set of sub-circuits with less interference among themselves. When sub-circuits with minimal interference are redacted, the attacker must encounter each sub-circuit individually diverging towards a more brute-force scenario (non-correlative-based) [28].

[1] N (the final iteration) depends on the size of the circuit. In larger circuits, more iteration is required for splitting and N becomes larger.

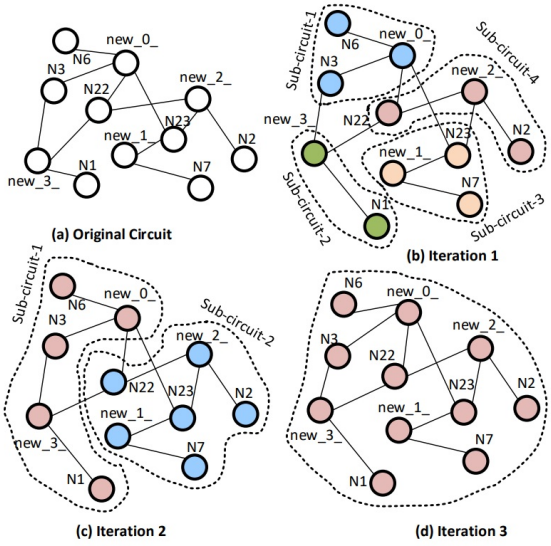

Fig. 5: Circuit Splitting and Conflict Coloring Example in EvoLUTe (c17).

Algorithm 1 Circuit Split Algorithm

```
procedure GNENRATESUBCIRCUITS
Input: {struc_graph_model}
Output: subcircuits
    InitializeCategory(struc_graph_model)
    for sub_circuit in struc_graph_model
      if sub_circuit.processed != 1 then
        minCat = GetMinConnectCate(sub_circuit)
        sub_circuit.category = minCat
        sub_circuit.processed = 1
    // define and process boundary nodes
    for nCategory in struc_graph_model
      for boundary_node in nCategory
        if node.fan_in_nodes.category! = nCategory then
          NewInNode(node)
        if node.fan_out_nodes.category! = nCategory then
          DefOutNode(node)
    return subcircuits
```

This is also highly possible (almost in all cases) that the designer may have more than enough sub-circuits for redaction. Hence, EvoLUTe deploys a structural interference-based scoring mechanism that captures the interaction of the conflict sub-circuits with the other neighboring sub-circuits. The score of a given sub-circuit k in *iter*$_j$ can be calculated by equation 1. Both *RoCGC* and *score*$_{k|j}$ are defined in a way that the selected sub-circuits provide a reasonable corruptibility (not too high to be against the SAT and not too low to be against structural attacks), while the correlation between sub-circuits are low enough making any form of (either structural or non-structural) pre-processing for de-obfuscation impractical.

$$score_{k|j} = nodes_{total} \times w_{ckt} \frac{subckt_{connected}}{n} + \frac{subckt_{connected}}{n}(1 - w_{ckt}) \sum_{i=0}^{n} \frac{nodes_{connected}}{nodes_{ckt_i}} \quad (1)$$

In Eq. 1, w_{ckt} is weight of circuit, $subckt_{connected}$ is number of circuits connected to circuit k, $nodes_{connected}$ is number of nodes in circuit i connected with k, $nodes_{ckt_i}$ is number of nodes in circuit i, n is the total number of circuits in *iter*$_j$. Note that w_{ckt} is a coefficient defining the attribute to focus on for redaction. When w_{ckt} is high, the score of sub-circuit k is high if sub-circuit k has more connections with other neighboring sub-circuits (sub-circuit-level connection). Conversely, when the w_{ckt} is low, the score of sub-circuit k is high if the nodes of sub-circuit k have more connections with other sub-circuits' nodes (node-level connection). This unique definition configured by coefficient w_{ckt} allows the designer to select the sub-circuits based on both node-level and circuit-level attributes[2].

D. Granular Redaction

After splitting, coloring, and scoring the sub-circuits, a sorted list of disintegrated sub-circuits are available for redaction. Unlike eFPGA-based redaction that employ only 1-2

[2] To avoid loss of generality, our experiments in this study aim to draw the impact of granularity on redaction overhead and robustness, and this metric (w_{ckt}) will be swept for detailed analysis in future work.

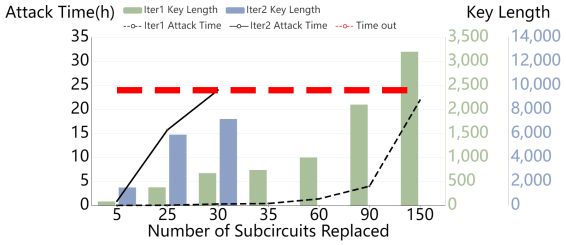

Fig. 6: Sweeping the Number of Redacted Sub-Circuits vs. the SAT Attack Time vs. Key Length for Redaction (TO: 24 Hours).

eFPGA instances (due to huge overhead), since sub-circuit to LUT replacement incurs less overhead[3], multiple (5-150) sub-circuits are selected for the redaction.

IV. Security Analysis of EvoLUTe

Since EvoLUTe is a redaction-based technique, even though it is fine-grained, the assumptions and requirements of the security analysis are in line with other redaction techniques, such as eFPGA-based redaction [19], [21]. Hence, the focus of the security analysis and attack evaluation is on the SAT attack as it is the only applicable state-of-the-art attack on the redaction-based techniques [19], [21][4]. By defining the range of granularity, the main aim of this work is to investigate how the number (and the size) of selected sub-circuits affect SAT resilience as well as area overhead.

Fig. 6 shows a simple yet crucial sweep in LUT-based redaction using EvoLUTe. Here for benchmark circuit *Arbiter* [25], after applying the splitting, coloring, and scoring, we swept the number of redactions to see its impact on SAT attack time as well as the key size. Following are some of the important takeaways from this observation:

(i) As the number of redactions increases, the SAT attack time increases in a way that is somewhere between quadratic and exponential regression models (acceptable accuracy). As it follows fine-granularity, it allows the designer to tune the redaction using a more systematic method in order to achieve the desired robustness. This is the opposite of eFGPA-based redactions, where the designers must insert at least one huge fabric to determine whether it is robust enough.

(ii) The level of granularity has a crucial impact on area overhead, attack time, as well as key length. Fig. 6 shows two different iterations of EvoLUTe ($iter_1$ and $iter_2$). Higher iterations (more coarse redaction) can achieve robustness with a very small number of sub-circuits redacted using LUTs (from 175 in $iter_1$ to only 20 in $iter_2$). It is evident for $iter$>2, the total number of redactions (to be resilient) becomes less and less, leading to the most coarse-grained (whole circuit), which requires only 1 (or 2) to be resilient. However, it affects the overhead negatively (making the higher iterations less efficient).

(iii) Key length is completely dependent on the level of granularity. For higher iterations in which the sub-circuits are bigger, the configuration size increases exponentially, and accordingly, when the redaction is more coarse-grained, the key length is much higher. As shown in Fig. 6, for $iter_1$, the minimum number of redaction resilient against the SAT requires ∼3.2K key bits. However, in $iter_2$, ∼7.2K key bits are required for the redacted sub-circuits. This observation is consistent with eFPGA-based redaction as a coarse-grained model, where >200K key bits are required for the same circuit.

[3]In some cases, based on the topological structure of selected sub-circuit, the area of redacted design is less (evaluated in Section V).

[4]For instance, (i) since the circuit is redacted (specification and functional parts are redacted), none of the feature-based ML-based attacks are applicable; (ii) removal-based and structural-based are not applicable as the structural and functional are concealed together, and removing LUT(s) will result in malfunctioning; (iii) re-synthesis-based attacks are not applicable as applying the key to LUTs does not reduce it to any form close to the original circuit.

TABLE I: Benchmark Circuits Specification

Small				Large			
Circuit	Area	#Inputs	#Outputs	Circuit	Area	#Inputs	#Outputs
Sine	6465.5	24	25	Arbiter	5759	256	129
Square	25439	64	128	Log2	42220	32	332
Multiplier	34586	128	128				

V. Experimental Results

To investigate the security resiliency and performance overhead of EvoLUTe, we redacted a set of widely used combinational benchmarks [25] shown in Table I by following EvoLUTe-based fine-grained redaction and compared their results with eFPGA-based redaction. The experimental results are shown in Table II. All SAT attack experiments were run with a timeout of 24 hours on a server with a 32-core 2.6 GHz Intel Xeon CPU and 64GB RAM.

For each benchmark shown in Table I, we first applied the circuit splitter tool discussed in Section III-B to disintegrate the whole circuit into smaller subcircuits with different levels of granularity ($iter_1$, $iter_2$, ..., etc.). Afterward, to assess the impact of granularity in redaction towards SAT resiliency and area overhead, we picked $iter_1$ and $iter_2$ outcomes of the circuit splitter tool. As discussed in Section III, at $iter_1$, splitting is more granular, generating a larger number of smaller subcircuits, while at $iter_2$ splitting is more coarse, generating a smaller number of larger subcircuits. For each iteration ($iter_1$ and $iter_2$) across different benchmarks, as mentioned in the first two rows of Table II, we varied the number of selected subcircuits for redaction (first column of Table II), replaced the selected subcircuits by the same number of LUTs based on their I/O bandwidth, and performed SAT attack [5] until a timeout (24 hours) is reached. For each of the different number of redactions as presented in different rows of Table II, their associated area overhead ($a\%$), key size ($\|\mathrm{k}\|$), and SAT attack execution time ($t(h)$) is reported in different columns. We kept increasing the number of redactions until a timeout was met. Once a timeout is met, we replicated a similar scenario in eFPGA-based coarse-grained redaction by implementing an exactly same size redaction in eFPGA and reported the associated area overhead, bitstream size, and SAT attack execution time. For instance, we met timeout for $iter_1$ of *Sine* benchmark for 60 subcircuits redaction. Hence we implemented a circuit in eFPGA whose area is equivalent to the total area of those 60 subcircuits (see Table II).

TABLE II: SAT Attack and Overhead (Area) Comparison between LUT-based and eFPGA-based Redaction.

Redact	$Iter_1$ (more fine-grained)															$Iter_2$ (more coarse-grained)													
	Sine			Square			Log2			Multiplier			Arbiter			Sine			Square			Log2			Multiplier			Arbiter		
	a%	‖k‖	t(h)	a%	‖k‖	t(h)	a%	‖k‖	t(h)	a%	‖k‖	t(h)	a%	‖k‖	t(h)	a%	‖k‖	t(h)	a%	‖k‖	t(h)	a%	‖k‖	t(h)	a%	‖k‖	t(h)	a%	‖k‖	t(h)
5 LUTs	-14.5	24	0.41	-21	96	2.07	-13	28	0.59	-25.8	54	0.44	17	84	0.01	-18	304	1	-18.06	504	3.54	-9.85	68	1	31	17K	1	33	1.5K	0.9
25 LUTs	-3.22	356	2.76	-8	236	4.52	-5.4	184	2.54	-17.6	130	1.90	28	378	0.05	14	3K	2	-15.51	1.1K	6.25	-8.18	254	4.21	39	18.9K	1.9	41	5.9K	15.74
30 LUTs	1	648	4.05	-1	296	19.5	-4.7	292	3.32	-17.4	316	2.76	40	674	0.31	71	6.5K	5.1	-14	1.4K	TO	-2	312	6.5	42	21K	2.5	75	7.2K	TO
35 LUTs	9	736	5.83	-18.3	440	TO	-4	356	4.05	-17.1	384	3.08	47	738	0.41	152	12K	TO	-	-	TO	1	424	7.26	47	27K	4.39	-	-	-
60 LUTs	17	1K	15.3	-	-	TO	1	648	6.94	-15.8	640	5.62	53	1K	1.38	-	-	TO	-	-	TO	9	918	13.5	83	43K	13.9	-	-	-
90 LUTs	20	1.1K	TO	-	-	TO	7	812	23.2	-12.4	1.2K	22.7	57	2.1K	4.01	-	-	TO	-	-	TO	17	1.1K	TO	94	46K	TO	-	-	-
150 LUTs	26	-	TO	-	-	TO	18	1.3K	TO	-12	13K	TO	71	3.2K	21.9	-	-	TO	-	-	TO	-	-	-	-	-	-	-	-	-
1 eFPGA	354	9K	TO	201	14K	TO	129	16K	TO	151	17K	TO	311	11K	TO	323	7.5K	TO	189	11.7K	TO	105	14.6K	TO	138	16.3K	TO	258	10.2K	TO

(i) *Number of Redactions:* For each benchmark and different iterations, area overhead, key size, and SAT attack time increase as more subcircuits are redacted.
(ii) *Granularity:* For every benchmark, the timeout was met at $iter_2$ with fewer redactions than $iter_1$, due to the larger subcircuit redactions at $iter_2$; however, it came out at the cost of more area overhead. Hence, better SAT resiliency can be achieved with more fine-grained redactions.
(iii) *eFPGA-based redaction:* Being the most coarse-grained form of redaction in the spectrum, requires less number of subcircuits to be redacted for SAT resiliency but incurs maximum overhead similar to the existing literature [19]–[22].

The above observations are also eminent for later iterations (more coarse-grained redaction) leading up to the whole circuit redaction. A critical takeaway from this experiment is that fine-grained redactions offer better area overhead with less SAT resiliency which is the opposite of coarse-grained redaction. Therefore a balance must be satisfied to achieve the target area overhead and SAT resiliency while utilizing circuit redaction against IP piracy using universal circuits.

VI. Conclusion

In this paper, we explored the impact of real fine to coarse granularity on the efficacy of IP redaction techniques. Through this exploration, we introduced *EvoLUTe*, a distinct and significantly more fine-grained redaction methodology using smaller reconfigurable components (LUTs). In EvoLUTe, we defined a splitting, coloring, and redaction mechanism that efficiently allows us to apply redaction at an appropriate granularity. We demonstrated that IP redaction at an appropriate granularity provides the same resiliency at significantly lower overhead.

References

[1] M. S. Rahman *et al.*, "LL-ATPG: Logic-Locking Aware Test Using Valet Keys in an Untrusted Environment," in *IEEE Int'l Test Conference (ITC)*, 2021, pp. 180–189.
[2] B. Shakya *et al.*, "Introduction to hardware obfuscation: Motivation, methods and evaluation," in *Hardware Protection through Obfuscation*. Springer, 2017, pp. 3–32.
[3] J. Roy *et al.*, "Epic: Ending piracy of integrated circuits," in *Design, Automation & Test in Europe Conference (DATE)*, 2008, pp. 1069–1074.
[4] H. M. Kamali *et al.*, "Advances in logic locking: Past, present, and prospects," *Cryptology ePrint Archive*, 2022.
[5] P. Subramanyan *et al.*, "Evaluating the security of logic encryption algorithms," in *Int'l Symposium on Hardware Oriented Security and Trust (HOST)*, 2015, pp. 137–143.
[6] M. Yasin *et al.*, "Sarlock: Sat attack resistant logic locking," in *Int'l Symposium on Hardware Oriented Security and Trust (HOST)*, 2016, pp. 236–241.
[7] Y. Xie, Yang *et al.*, "Anti-sat: Mitigating sat attack on logic locking," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 199–207, 2018.
[8] H. M. Kamali *et al.*, "Full-Lock: Hard Distributions of SAT Instances for Obfuscating Circuits using Fully Configurable Logic and Routing Blocks," in *Design Automation Conference (DAC)*, 2019, pp. 1–6.
[9] H. M. Kamali *et al.*, "InterLock: An Intercorrelated Logic and Routing Locking," in *Int'l Conf. On CAD (ICCAD)*, 2020, pp. 1–9.
[10] X. Xu *et al.*, "Novel Bypass Attack and BDD-based Tradeoff Analysis against All known Logic Locking Attacks," in *Int'l Conference on CHES*, 2017, pp. 189–210.
[11] M. Yasin *et al.*, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517–532, 2017.
[12] K. Shamsi *et al.*, "Appsat: Approximately deobfuscating integrated circuits," in *Int'l Symposium on Hardware Oriented Security and Trust (HOST)*, 2017, pp. 95–100.
[13] D. Sisejkovic *et al.*, "Challenging the security of logic locking schemes in the era of deep learning: A neuroevolutionary approach," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 17, no. 3, pp. 1–26, 2021.
[14] K. Z. Azar *et al.*, "Nngsat: Neural network guided sat attack on logic locked complex structures," in *Int'l Conference on Computer-Aided Design (ICCAD)*, 2020, pp. 1–9.
[15] M. S. Rahman *et al.*, "O'clock: lock the clock via clock-gating for soc ip protection," in *Design Automation Conf. (DAC)*, 2022, pp. 775–780.
[16] R. S. Chakraborty *et al.*, "HARPOON: An Obfuscation-based SoC Design Methodology for Hardware Protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.
[17] K. Z. Azar *et al.*, "Smt attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the sat attacks," *IACR Trans. on CHES*, pp. 97–122, 2019.
[18] S. Roshanisefat *et al.*, "RANE: An Open-Source Formal De-obfuscation Attack for Reverse Engineering of Logic Encrypted Circuits," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2021, pp. 221–228.
[19] P. Mohan *et al.*, "Hardware redaction via designer-directed fine-grained efpga insertion," in *Design, Automation & Test in Europe Conference (DATE)*, 2021, pp. 1186–1191.
[20] J. Bhandari *et al.*, "Exploring efpga-based redaction for ip protection," in *Int'l Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–9.
[21] J. Bhandari *et al.*, "Not all fabrics are created equal: Exploring efpga parameters for ip redaction," *arXiv preprint arXiv:2111.04222*, 2021.
[22] B. Hue*t al.*, "Functional obfuscation of hardware accelerators through selective partial design extraction onto an embedded fpga," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2019, pp. 171–176.
[23] P. Beerel *et al.*, "Towards a formal treatment of logic locking," *Cryptology ePrint Archive*, 2022.
[24] Chhotaray, Animesh *et al.*, "Hardening circuit-design ip against reverse-engineering attacks," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1672–1689.
[25] Amarú, Luca *et al.*, "The epfl combinational benchmark suite," in *Proceedings of the 24th International Workshop on Logic & Synthesis (IWLS)*, no. CONF, 2015.
[26] H. M. Kamali *et al.*, "Lut-lock: A novel lut-based logic obfuscation for fpga-bitstream and asic-hardware protection," in *IEEE Symposium on VLSI (ISVLSI)*, 2018, pp. 405–410.
[27] T. Jensen*et al.*, *Graph coloring problems*. John Wiley & Sons, 2011.
[28] J. Rajendran *et al.*, "Security analysis of integrated circuit camouflaging," in *ACM SIGSAC conference on CCS*, 2013, pp. 709–720.