Using OFF-set Only for Corrupting Circuit to Resist Structural Attack in CAC Locking

Hsiang-Chun Cheng, RuiJie Wang, TingTing Hwang
Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
{wrj651121, a0920203860}@gmail.com, tingting@cs.nthu.edu.tw

***Abstract*—Corrupt-and-Correct (CAC) Logic Lockings [1]–[4] are state-of-the-art hardware security techniques designed to protect IC/IP designs from IP piracy, reverse engineering, overproduction, and unauthorized use. Although these techniques are resilient to SAT-based attacks, they remain vulnerable to structural attacks, which exploit structural traces left by the synthesis tool to recover the original form.**

In this paper, we will propose a novel method that uses only the OFF-set to corrupt the circuit. This approach helps the added circuitry better merge with the original circuit, thereby thwarting structural attacks while maintaining resilience to SAT-based attacks. Additionally, we demonstrate that our proposed method can incur less area overhead compared to previous locking methods in HIID [5]. Compared to SFLL-rem [4], our method can achieve comparable area overhead while effectively resisting structural attacks, including Valkyrie [6] and SPI attacks [7].

***Index Terms*—Hardware security, logic locking, structural attack**

I. Introduction

Logic locking is one of the most effective techniques to protect IC/IP designs from IP piracy, reverse engineering, overproduction, and unauthorized use. It protects IP privacy by inserting additional key gates, such as XORs, XNORs, MUXs, or LUTs. The secret keys are stored in a tamper-proof memory [8]. The circuit behaves normally only if the correct key is applied, otherwise, the circuit does not function properly. In the early years of logic locking, random logic locking (RLL) [9], fault analysis-based logic locking (FLL) [10] and strong interference-based logic locking (SLL) [11] were developed.

Nevertheless, these locking techniques were overcome with the advent of the Boolean satisfiability (SAT) attack [12]. The SAT attack can prune out incorrect key(s) using SAT solver to find distinguishing input patterns (DIPs). To resist SAT attack, point function techniques force the SAT attack to eliminate only one key per iteration, causing the number of iterations to increase exponentially with the length of the key. Locking methods such as SARLock [13] and AntiSAT [14] are designed as a point function. However, these techniques face the problem that the added circuitry may not integrate seamlessly with the original circuit, leaving structural traces detected by structural analysis attacks. For example, SARLock [13] and Anti-SAT [14] can be broken by removal [15] and bypass [16] attacks.

To address this challenge, stripped functionality logic locking (SFLL), also known as corrupt-and-correct (CAC) logic locking, was developed. A *perturb/corrupt unit* first flips the output of the original circuit, and then a *restore/correct unit* flips the output back when the correct key is applied. TTLock [1], SFLL-HD [2], SFLL-flex [2], SFLL-fault [3] and SFLL-rem [4] all belong to this category. These approaches were effective in thwarting SAT attacks and structural attacks such as removal, bypass and signal probability skew (SPS) attacks [17]. However, two powerful state-of-the-art structural attacks have been introduced recently: the sparse prime implicant (SPI) [7] and Valkyrie attacks [6]. These attacks are capable of exploiting all existing single flip locking (SFLTs) and double flip locking techniques (DFLTs). Table I shows the summarization.

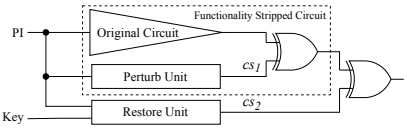


Fig. 1: Corrupt-and-Correct (CAC) logic locking

TABLE I: State-of-art locking methods and structural attacks

Attack \ Defense	Anti-SAT	SARLock	TTLock	SFLL-HD	SFLL-flex	SFLL-fault	SFLL-rem	Our
SPI	✗	✗	✗	✗	✗	✗	✗	✔
Valkyrie	✗	✗	✗	✗	✗	✗	✗	✔

✔: successful defense ✗: successful attack

In this paper, based on CAC structure, we propose a novel method to use OFF-set only to corrupt the original circuit to thwart the state-of-art structural attacks: SPI [7] and Valkyrie [6]. We prove that our method can help the extra added circuitry better mix with the original circuit, therefore, increasing the resilience to structural analysis attack. The experimental results also show that our method can achieve greater area reduction compared to previous methods.

In summary, the following are our contributions: 1) We propose a method to use OFF-set only for corrupting circuit. 2) We prove that our proposed method increases the resilience to structural analysis attacks while maintaining resilience to SAT-based attacks. 3) We demonstrate that our proposed method can help reduce the area as compared to the previous methods.

The rest of this paper is organized as follows. Section II and III introduce the definitions and properties of proposed methods. Section IV details the design flow and the algorithms. Section V and VI present a security analysis and our experimental results. Finally, Section VII concludes this work.

978-3-9826741-0-0/DATE25/© 2025 EDAA

II. Preliminaries

Definition 1 (Corrupt-and-Correct (CAC) Logic Locking)**.** *Given a Boolean function $F : I^n \rightarrow O^m$, where n is the size of the input, m is the size of the output, $I \subseteq \{0,1\}^n$ is the primary input, and $O \subseteq \{0,1\}^m$ is the primary output. For the encrypted circuit F_{enc}, let $K \subseteq \{0,1\}^k$ be denoted as the key input, where k is the size of the key. The protected input pattern/cube, denoted as PIP, serves as the secret key of the encrypted circuit. We define the functionality stripped circuit F_{fsc} and the encrypted/locked circuit, F_{enc} as follows:*

F_{fsc} differs from the original circuit F only when the input pattern equals the protected input pattern PIP:

$$F_{fsc}(I) = F(I) \oplus PIP$$

F_{enc} is constructed by integrating a restore unit into F_{fsc}. This restore unit functions as a comparator that outputs 1 to flip the output back to its correct state when $I = K$:

$$F_{enc}(I,K) = F_{fsc}(I) \oplus Restore(I,K)$$

The circuits F and F_{enc} are equivalent only if the correct key is applied:

$$\begin{cases} \forall I, \quad F(I) = F_{enc}(I,K) \quad when \quad K = PIP \\ \exists I, \quad F(I) \neq F_{enc}(I,K) \quad when \quad K \neq PIP \end{cases}$$

Figure 1 shows the structure of CAC logic locking.

Definition 2 (ON-set and OFF-set)**.** *Given a circuit F. Let $ON(F)$ and $OFF(F)$ denote the ON-set of F and the OFF-set of F.*

$$\begin{cases} \forall I \in ON(F), \quad F(I) = 1 \\ \forall I \in OFF(F), \quad F(I) = 0 \end{cases}$$

Definition 3 (Prime Implicant Table (PIT))**.** *The Prime Implicant Table (PIT) is used in logic optimization for Boolean functions to find the minimal set of implicants that cover all minterms of the function. A prime implicant is an implicant that cannot be covered by any other implicant in the table.*

Definition 4 (Distance-1 Merging)**.** *[7], [18] Given two implicants of a Boolean function, they can only be merged to a bigger implicant by uniting theorem of Boolean Algebra if their hamming distance is one.*

Implicant, product term, and cube are used interchangeably.

III. Properties

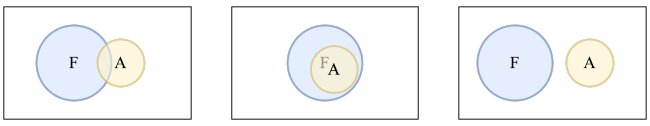
As defined in Definition 1, the functionality stripped circuit F_{fsc} is given by:

$$F_{fsc}(I) = F(I) \oplus PIP$$

where PIP is a hard-coded AND-tree serving as the corrupt/perturb unit. Given a Boolean function F and a random cube A, there are three cases of XOR operation to consider, as shown in Figure 2: (i) the general case in Figure 2(a); (ii) the case where $A \subseteq ON(F)$ in Figure 2(b); and (iii) the case where $A \subseteq OFF(F)$ in Figure 2(c). In the general case, the expression for the XOR operation is:

$$F \oplus A = F \cdot \overline{A} + \overline{F} \cdot A.$$

However, in special cases where $A \subseteq ON(F)$ or $A \subseteq OFF(F)$, the XOR operation can be simplified, thus making signals easily hidden after synthesis. For these two special cases, we have developed the following propositions.



(a) General case (b) $A \subseteq ON(F)$ (c) $A \subseteq OFF(F)$

Fig. 2: Three cases of $F \oplus A$

Proposition 1 (PIP Using ON-set Cube)**.** *Given a Boolean function F, for any cube $A \subseteq ON(F)$, the corrupted circuit: $F \oplus A = F \cdot \overline{A}$*

Proof. The XOR of F and A is given by: $F \oplus A = F \cdot \overline{A} + \overline{F} \cdot A$ Since $A \subseteq ON(F)$, it follows that $\overline{F} \cdot A = 0$. Therefore: $F \oplus A = F \cdot \overline{A}$ □

Proposition 2 (PIP Using OFF-set Cube)**.** *Given a Boolean function F, for any cube $A \subseteq OFF(F)$, the corrupted circuit $F \oplus A = F + A$*

Proof. The XOR of F and A is given by: $F \oplus A = F \cdot \overline{A} + \overline{F} \cdot A$ Since $A \subseteq OFF(F)$, it follows that $F \cdot \overline{A} = F$ and $\overline{F} \cdot A = A$. Therefore: $F \oplus A = F + A$ □

Proposition 3 (Effect of PIP Using ON-set Cube on the Size of Product Terms of F_{fsc})**.** *Given a Boolean function F with an ON-set cube A, using A as a PIP to corrupt the circuit will increase the number of product terms in the Prime Implicant Table (PIT) of $F_{fsc} = F \cdot \overline{A}$.*

$$|F| - 1 \leq |F \cdot \overline{A}| \leq |F| + S - 1$$

where $|F|$ and $|F \cdot \overline{A}|$ denote the minimum number of product terms in F and $F \cdot \overline{A}$, respectively, and S represents the increase in the number of product terms. In some cases, the size of increased product terms S can be as large as the size of literals in A.

Proof. Corrupting the circuit using an ON-set cube means removing it from the ON-set and adding it to the OFF-set. For the lower bound, if $PIP = A$, a product term in the PIT, the number of products is reduced to $|F| - 1$. For the upper bound, this process may split other product terms that overlap with PIP, thereby increasing the number of product terms in the PIT. To prove that S can be as large as the size of the literals in A. Let $F(x_1, x_2, \ldots, x_n) = 1$, representing a single cube, and $A = \{(0,0,\ldots,0)\}$, which is a minterm of F with n literals. We rewrite F using $x + \overline{x} = 1$ as follows:

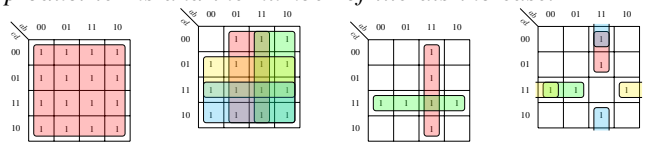
$$\begin{aligned} F &= x_1 + \overline{x_1} \\ &= x_1 + \overline{x_1}(x_2 + \overline{x_2}) \\ &= x_1 + \overline{x_1}x_2 + \overline{x_1 x_2} \\ &\vdots \\ &= x_1 + \overline{x_1}x_2 + \cdots + \overline{x_1 x_2 x_3 \cdots x_{n-1}}x_n + \overline{x_1 x_2 x_3 \cdots x_{n-1} x_n} \end{aligned}$$

Then, $F \oplus A = x_1 + \overline{x_1}x_2 + \cdots + \overline{x_1 x_2 x_3 \ldots x_{n-1}}x_n$

Since $(0,0,0,\ldots,0)$ is removed from F, the rest of the product terms cannot be merged because no pairs of product terms have a distance of 1. Hence, $S = n$. □

Example 1. *As shown in Figure 3, Figure 3(a) and Figure 3(c) display the original PITs of the Boolean functions $F1$ and $F3$ respectively. Figure 3(b) shows the PIT after removing the cube $\overline{abcd}$ from $F1$. Consequently, the number of product*

terms increases by $4-1$. Figure 3(d) illustrates the PIT after removing the cube $abcd$ from $F3$. Here, both the number of product terms and the number of literals increase.



(a) Original $F1$ (b) Removing $\overline{abcd}$ from $F1$ (c) Original $F3$ (d) Removing $abcd$ from $F3$

Fig. 3: Removing ON-set cube from the original function

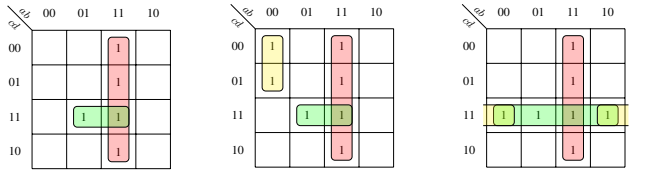
Proposition 4 (Effect of PIP Using OFF-set Cube on the Size of Product Terms of F_{fsc})**.** *Given a Boolean function F with an OFF-set cube A, using A as a PIP to corrupt the circuit increases the number of product terms in the Prime Implicant Table (PIT) of $F_{fsc} = F + A$ by at most one.*

$$|F| - S + 1 \leq |F + A| \leq |F| + 1$$

where S represents the decrease in the number of product terms.

Proof. Corrupting the circuit F using an OFF-set cube A means moving PIP from the OFF-set to the ON-set. For the upper bound, if the added OFF-set cube has any distance-1 ON-set cubes, it can merge with cubes in ON-set. Then, either A is partially merged in the cube of ON-set and reduce the number of literals, or A is completely merged and reduce the number of cubes. Otherwise, A stays alone and consequently, the maximum number of product terms increases by at most one. For the lower bound, similar to Proposition 3, if A is merged with S split prime implicants, the number of product terms can be reduced by $S-1$. □

Example 2. *As shown in Figure 4, Figure 4(a) represents the original Boolean function F. Figure 4(b) shows the PIT after adding the cube $\overline{abc}$ to F. In this case, the number of product terms increases by at most one. Figure 4(c) displays the PIT after adding the cube $\bar{b}cd$. In this case, the number of product terms remains the same.*



(a) Original F (b) Adding $\overline{abc}$ to F (c) Adding $\bar{b}cd$ to F

Fig. 4: Adding OFF-set cube to the original function

Proposition 5 (Hiding the Critical Signal of the Corrupt/Perturb Unit Using OFF-set)**.** *Given a Boolean function F with an OFF-set cube A, to conceal the critical signal of the corrupt/perturb unit, the added cube A needs to merge with the cubes in the original function F.*

Proof. As shown in Figure 4(c), adding the cube A to the original function F and merging it with other cubes ensures that the critical signal is effectively hidden. The merging process integrates the cube A into the original function, resulting in the critical signal no longer being present. □

Proposition 6 (Finding OFF-set Cubes with ATPG)**.** *Test patterns for a stuck-at-1 fault that propagate the faulty value through paths containing an even number of inverting elements (inverters, NAND gates, or NOR gates) to the primary output, and test patterns for a stuck-at-0 fault that propagate the faulty value through paths containing an odd number of inverting elements (inverters, NAND gates, or NOR gates) to the primary output are referred to as OFF-set cubes.*

Proof. To test a fault, we must first activate it and then propagate it to the primary output. For a stuck-at-0 fault, we need to make the fault-free value equal to 1 to activate it since the faulty value is 0, while for a stuck-at-1 fault, we need to make the fault-free value equal to 0 since the faulty value is 1. To propagate the fault-free/faulty value, the value will be inverted when encountering inverting elements (inverters, NAND gates, or NOR gates). Therefore, if the fault-free value equals 1 and is inverted an odd number of times, or if the fault-free value equals 0 and is inverted an even number of times, the final fault-free value will be 0. Test patterns meeting these conditions are referred to as OFF-set cubes. □

Proposition 7 (Finding ON-set Cubes with ATPG)**.** *Test patterns for a stuck-at-0 fault that propagate the faulty value through paths containing an even number of inverting elements (inverters, NAND gates, or NOR gates) to the primary output, and test patterns for a stuck-at-1 fault that propagate the faulty value through paths containing an odd number of inverting elements (inverters, NAND gates, or NOR gates) to the primary output are referred to as OFF-set cubes.*

It can be proven similarly as in Proposition 4.

IV. Using OFF-set to Corrupt Circuit

In this section, we present our proposed method of using OFF-set cubes to corrupt the circuit. Based on Propositions 3 and 4, using an ON-set cube as PIP may increase the number of increased product terms by as large as the number of literals in PIP, whereas using an OFF-set cube as PIP increases the number of product terms by at most one. Furthermore, even if encrypting a prime implicant in ON-set reduces the number of product terms by 1, the effectiveness of the encryption is generally limited by the number of specified bit of the cube [19]. Therefore, we choose to use an OFF-set cube in our proposed method. Our approach leverages Proposition 2, which allows using $F + A$ instead of $F \oplus A$ to perturb the circuit. This helps seamlessly integrate the corrupt/perturb unit with the original circuit, concealing the critical signal of the corrupt/perturb unit and thereby increasing resilience against structural attacks. We will begin by introducing the overall design flow of our proposed method.

A. Design Flow

As illustrated in Figure 5, the first step is to select a primary output PO from F and extract its logic cone, denoted as F_{PO}. It is essential at this stage to ensure that the number of primary inputs in the logic cone exceeds the required key size. For the second step, we generate ON-set and OFF-set cubes using the ATPG tool with Propositions 6 and 7. Following this, in the third step, we identify a *mergeable* OFF-set cube to serve as PIP. Using this PIP, we corrupt the extracted logic cone F_{PO} with Proposition 2, resulting in $F_{PO} + PIP$. Subsequently, we use a logic synthesis tool to resynthesize

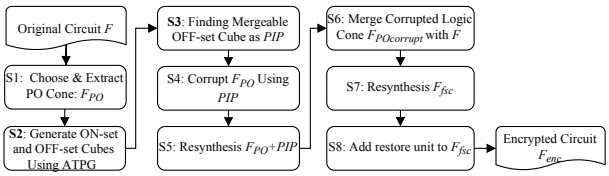


Fig. 5: Flow of the proposed method

it, ensuring seamless integration into the logic cone. Next, we merge the corrupted logic cone with the original circuit F to obtain F_{fsc}. This circuit is then resynthesized again to optimize its structure. Finally, a restore unit is added to F_{fsc} to obtain the encrypted circuit F_{enc}. The restore unit consists of a comparator that outputs 1 when the input pattern equals the key input, ensuring the correct functionality of F_{enc} when the correct key is applied.

In the following section, we will provide a detailed explanation of steps 2 and 3 while the remaining steps are straightforward.

B. Finding Protected Input Pattern PIP

The second step is to use the ATPG tool to generate test patterns T_f of the extracted logic cone F_{PO}. By applying Propositions 6 and 7, we obtain a set of ON-set and OFF-set cubes, denoted as $Cubes_{ON}$ and $Cubes_{OFF}$. We begin by filtering out the cubes in $Cubes_{OFF}$ that do not meet the required key size in terms of specified bits. To increase the likelihood of the added OFF-set cube effectively merging with the original circuit, we pairwise compare the cubes in $Cubes_{ON}$ and $Cubes_{OFF}$ to calculate their Hamming distance. For each OFF-set cube in $Cube_{OFF}$, we record its distance-1 ON-set cubes. We then sort these OFF-set cubes in descending order based on the number of their distance-1 ON-set cubes. These sorted OFF-set cubes are considered as candidate PIP.

The third step is for each cube in $Cubes_{OFF}$ to check if it can be merged with the extracted logic cone F_{PO} by using a logic synthesis tool to perform two-level optimization with its distance-1 ON-set cubes. If the cube merges successfully, it will not appear in the optimized prime implicant table (PIT). Additionally, we need to ensure that the cube meets the required key size. If the cube passes this check and meets the security requirements, we select it as PIP.

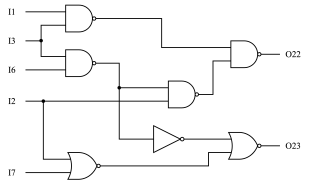
The pseudocode of the third step of our proposed method to find a PIP is provided in Algorithm 1.

Algorithm 1: Finding Mergeable OFF-set Cube as PIP

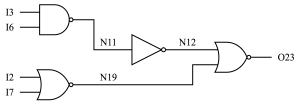
Input: ON-set, $Cubes_{ON}$, Sorted OFF-set, $Cubes_{OFF}$
Output: Protected Input Pattern PIP

```
foreach C_OFF ∈ Cubes_OFF do
    Cubes_dist1 ← Get_Distance1_Cubes(C_OFF);
    PIT ←
      Two-Level_Optimization(C_OFF, Cubes_dist1);
    if C_OFF does not exist in PIT then
        return C_OFF;
    end
end
```

Example 3. *As shown in Figure 6, we use c17 as an example. The original circuit of c17 is depicted in Figure 6(a). Figure 6(b) shows the extracted logic cone O23. Figures 6(c)(d)(e) display the test patterns T_f of this extracted logic cone F_{O23}, generated using the ATPG tool, the ON-set and OFF-set cubes derived from T_f respectively. After calculating the Hamming distance pairwise with the ON-set cubes, we sorted the OFF-set cubes based on their number of distance-1 ON-set cubes. In this example, the cube $0xx0$ has the maximum number of distance-1 ON-set cubes, while $x111$ is the second. These cubes serve as candidates for PIP. Figure 6(f) presents the K-map of the original circuit. Figure 6(g) illustrates the K-map when using $0xx0$ as PIP. Despite changes in F, the added OFF-set cube PIP remained unmerged. Figure 6(h) illustrates the K-map when using $x111$ as PIP. Here, both F and the added OFF-set cube PIP changed, demonstrating a seamless merge of PIP and F. Thus, we use $x111$ as PIP for F_{O23}.*



(a) Original circuit F



(b) Extracted logic cone F_{O23}

	I2	I3	I6	I7	O23
N3 /1	1	0	1	x	1
N6 /1	1	1	0	x	1
N12 /0	1	1	1	x	0
	0	1	1	1	0
N19 /0	0	0	x	0	0
	0	1	0	0	0
O23 /0	1	0	x	x	1
	1	1	0	x	1
	0	0	x	1	1
	0	1	0	1	1
N2 /0	1	0	x	0	1
N7 /0	0	0	x	1	1
O23 /1	0	x	x	0	0
	1	1	1	0	0
	x	1	1	1	0

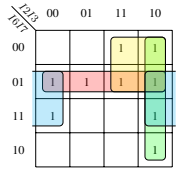
(c) T_f generated from ATPG tool.

I2	I3	I6	I7	O23
1	0	1	x	1
1	1	0	x	1
1	0	x	x	1
1	1	0	x	1
0	0	x	1	1
0	1	0	1	1
1	0	x	0	1
0	0	x	1	1

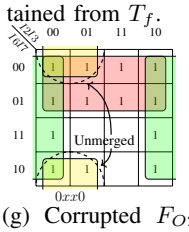
(d) ON-set cubes obtained from T_f.

I2	I3	I6	I7	O23
1	1	1	x	0
0	1	1	1	0
0	0	x	0	0
0	1	0	0	0
0	x	x	0	0
1	1	1	0	0
x	1	1	1	0

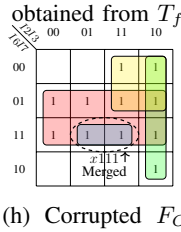
(e) OFF-set cubes obtained from T_f.



(f) Original F_{O23}



(g) Corrupted F_{O23} using $0xx0$ as PIP



(h) Corrupted F_{O23} using $x111$ as PIP

Fig. 6: Finding OFF-set cubes as PIP.

V. Security Analysis

In this section, we provide a security analysis to show that our proposed method enhances resilience against structural attacks including SPI [7] and Valkyrie [6] attacks while maintaining robustness against SAT attacks.

A. Boolean Satisfiability (SAT) Attack

The probability of a successful SAT attack on corrupt-and-correct (CAC) logic locking is determined by the likelihood of encountering a protected input pattern [1]–[4]. If SAT attack

selects DIP within the PIP, it can eliminate all the wrong keys in a single iteration. In our proposed method if we select an OFF-set cube with k specified bits for a n input Boolean function F. The number of input patterns in the PIP is equal to 2^{n-k} since the number of unspecified bits of PIP is $(n-k)$. The probability of the SAT attack successfully finding the PIP in a single iteration is equal to the error rate of the corrupted circuit. This probability is given by:

$$P = \frac{2^{n-k}}{2^n}$$

For a sufficiently large key size k, this probability becomes very low, making the circuit secure against SAT attacks.

B. Sparse Prime Implicant (SPI) Attack

The SPI attack analyzes the prime implicant table (PIT) of the corrupted circuit, as the added or removed PIP may or may not merge with the PIT of the original circuit. As described in the paper by Han *et al.* [7], four cases highlight the vulnerabilities of logic locking: (1) adding a PIP where the PIP does not merge, (2) adding a PIP where the PIP merges, (3) removing a PIP where the PIP does not merge, and (4) removing a PIP where the PIP merges. In cases (2) and (4), PIP is merged with other cubes and hence resists the SPI attack.

In our proposed method, we use a *mergeable* OFF-set cube as the PIP, which is case (2) as depicted in Figure 4(c). This means that the search space includes all the minterms in the ON-set of the corrupted circuit F_{corrupt}. The number of minterms in the ON-set of the corrupted circuit is $|\{m \mid \forall m \in \text{ON}(F_{\text{corrupt}})\}|$ [7], thereby enhancing resistance against SPI attacks.

C. Valkyrie

In CAC shown in Figure 1, there are two critical signals. The first critical signal cs_1 represents the output signal of the corrupt/perturb unit, while the second one cs_2 represents the output signal of the correct/restore unit. The Valkyrie attack aims to identify these critical signals and their correct values to manipulate the circuitry, thereby restoring the circuit to its original functionality.

It begins by leveraging specific properties to identify the encrypted logic cone and its corresponding critical signals. After finding these critical signals, the attack injects stuck-at faults for these two signals:

$$(cs_1, cs_2) = \{(0,0), (0,1), (1,0), (1,1)\}$$

Here, the two critical signals can each either be 0 or 1, resulting in four possible cases of their values. Subsequently, an EDA tool is employed to verify the equivalence between the original circuit and the fault-injected circuit. This process continues iteratively until the correct critical signals are identified or no further candidates remain.

In traditional CAC, the additional corrupt/perturb unit does not merge with the original circuit well and remains isolated, making it easier to identify its critical signal. However, by using OFF-set only for the corrupt/perturb unit, the added OFF-set cube merges with the cubes in the original function. Consequently, the critical signal cs_1 of the corrupt/perturb unit ceases to exist, thereby thwarting the Valkyrie attack.

TABLE II: Area evaluation using Genus with key = 32

Benchmark	Original (μm²)			F_{fsc} (μm²)			F_{enc} (μm²)			F_{fsc} Overhead (%)			F_{enc} Overhead (%)		
	7ns	6ns	5ns	7ns	6ns	5ns	7ns	6ns	5ns	7ns	6ns	5ns	7ns	6ns	5ns
b14_C	2660.27	2669.31	2823.32	2461.30	2540.30	2660.53	2581.58	2593.02	2706.86	-7.48	-4.83	-5.77	-2.96	-2.86	-4.12
b15_C	4088.42	4043.20	4167.16	3963.67	4039.74	4121.40	4058.68	4125.71	4222.27	-3.05	-0.09	-1.10	-0.73	2.04	1.32
b17_C	13658.57	13793.16	14123.27	13233.77	13485.67	13856.47	12496.73	13597.17	13875.94	-3.11	-2.23	-1.89	-8.51	-1.42	-1.75
b20_C	6022.51	6750.81	6890.46	5873.55	6594.41	6256.05	6143.05	6247.59	6162.47	-2.47	-2.32	-9.21	2.00	-7.45	-10.57
b21_C	5885.78	6452.36	6696.28	6284.78	6160.03	6119.06	5656.01	5756.29	5815.07	6.78	-4.53	-8.62	-3.90	-10.79	-13.16
b22_C	9211.58	9236.05	9204.93	8850.09	8544.98	9264.51	8995.37	8678.56	9482.15	-3.92	-7.48	0.65	-2.35	-6.04	3.01
Average										-2.21	-3.58	-4.32	-2.74	-4.42	-4.21

TABLE III: Power evaluation using Genus with key = 32

Benchmark	Original (mW)			F_{fsc} (mW)			F_{enc} (mW)			F_{fsc} Overhead (%)			F_{enc} Overhead (%)		
	7ns	6ns	5ns	7ns	6ns	5ns	7ns	6ns	5ns	7ns	6ns	5ns	7ns	6ns	5ns
b14_C	0.55	0.62	0.78	0.52	0.61	0.78	0.54	0.61	0.74	-4.87	-1.95	-0.51	-1.58	-1.71	-5.27
b15_C	0.44	0.52	0.64	0.45	0.52	0.66	0.46	0.55	0.63	2.25	1.07	2.00	4.49	6.02	-2.38
b17_C	1.67	1.88	2.23	1.59	1.79	2.11	1.42	1.82	2.15	-4.76	-4.90	-5.11	-14.51	-3.28	-3.35
b20_C	1.31	1.70	2.09	1.39	1.88	2.08	1.45	1.79	2.05	6.59	10.44	-0.66	10.62	5.36	-2.13
b21_C	1.33	1.65	2.01	1.59	1.80	2.03	1.38	1.62	1.79	19.26	8.99	0.92	3.68	-1.73	-11.01
b22_C	1.96	2.27	2.64	2.02	2.21	2.82	1.99	2.19	2.86	2.84	-2.61	7.10	1.57	-3.39	8.31
Average										3.55	1.84	0.62	0.71	0.21	-2.64

VI. Experimental Results

In this section, we study the effectiveness of our proposed method to resist the SAT attack and the Valkyrie attack (SPI is not tested since it is not accessible in public domain). We also examine the impact of our method on Power, Performance, and Area (PPA) overhead, we employ the commercial EDA tool Cadence Genus along with the NanGate FreePDK45 Open Cell library [20] to conduct a precise evaluation of the PPA overhead of our proposed method. Additionally, we compare the PPA overhead of our method with existing locking techniques, including Anti-SAT, SARLock, TTLock, and CAC, using the open source HIID tool [5]. Our experiments are conducted on a platform equipped with an AMD EPYC 7543 32-Core Processor running at 2.8 GHz and 251 GB of RAM. Test patterns are generated using the Atalanta ATPG tool [21] in our flow. The benchmarks are sourced from ITC'99 [22].

A. Effectiveness of PPA Overhead Using Genus

The experiments are conducted with different timing constraints, the clock periods ranging from 5 to 7 ns, and key sizes set to 32 bits. The synthesis results of the corrupted and locked circuits are assured to have no timing violations. We also consider the area of the tamper-proof memory (TPM) in the locked circuit, assuming TPM occupies 0.699 μm^2 per bit.

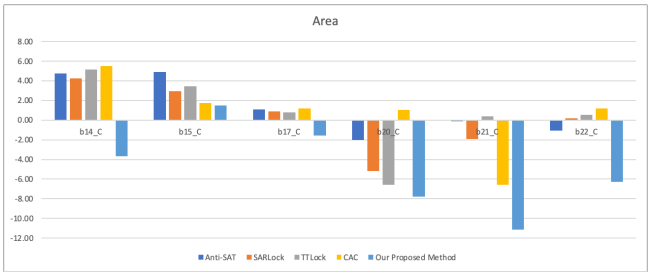
Table II shows the area overhead of F_{fsc} and F_{enc}. For F_{fsc}, the average area overhead is negative in all cases, indicating that the added OFF-set cube is mergeable and helps reduce the area through logic optimization. When the clock period becomes tighter, the area reduction compared to the original circuit increases. As described in [23], tighter design constraints can direct the synthesis tool to work more intensively. For F_{enc}, the average area overhead is also negative in all cases, indicating that despite the addition of a restore unit, the overall area is still reduced compared to the original circuit.

Table III shows the power overhead of F_{fsc} and F_{enc}. For F_{fsc}, the average power overhead is 3.55%, 1.84%, and 0.62% respectively. For F_{enc}, the average power overhead is 0.71%, 0.21%, and -2.64% respectively.

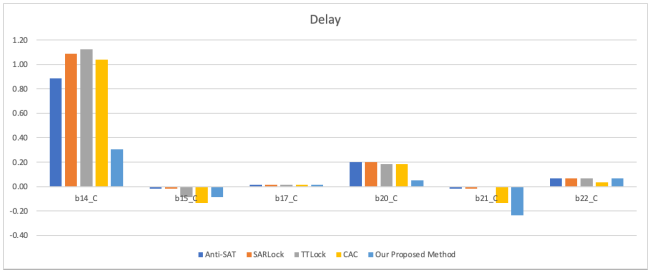
B. Comparison Using HIID

We compare the results of our proposed method with other existing CAC locking techniques. Figure 7 presents the

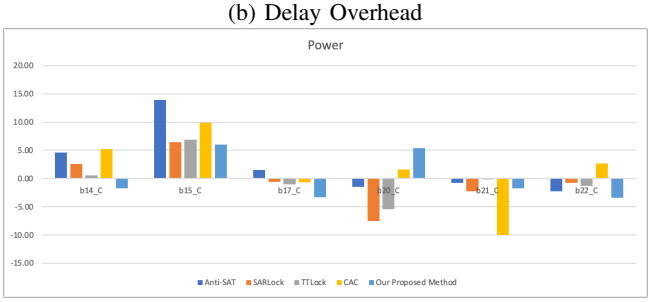
original synthesized results for circuits locked by Anti-SAT, SARLock, TTLock, and CAC using the open-source HIID tool [5], as well as our proposed method. For consistency, we used the same synthesis tool and library as described previously. The percentages relative to the original circuit in Figures 7(a), 7(b), and 7(c) show the comparisons of area overhead, delay overhead, and power overhead, respectively. Our proposed method outperforms other CAC locking techniques in area overhead across all benchmarks. For delay overhead, our method does not surpass others only in benchmarks b15_C and b22_C. However, the difference is minor and negligible. In terms of power overhead, our method excels in most cases, except for benchmarks b20_C and b21_C. These results highlight the effectiveness of our method compared to Anti-SAT, SARLock, TTLock, and CAC.



(a) Area Overhead



(b) Delay Overhead



(c) Power Overhead

Fig. 7: Comparison with existing CAC locking techniques

C. Comparison with SFLL-rem [4]

SFLL-rem is the latest SFLL method, differing from earlier SFLL methods [2] that adds extra logic to construct corrupt circuit. Instead, it introduces a stuck-at fault to remove logic and uses an ECO tool to restore all test patterns except protected input pattern [4]. Since SFLL-rem did not outsource its encrypted circuits or tools, we borrowed the experimental results from their paper [4]. The keysize was set to 80. SFLL-rem utilized the Global Foundries (GF) 65-nm LPe Library for synthesizing the locked circuits, whereas we employed the NanGate FreePDK45 Open Cell Library. Therefore, we scaled down the area occupied by each bit in the tamper-proof memory and reduced the clock period from 9 ns to 5-7 ns for comparison. We ensured that all locked circuits were free from any timing violations.

Table IV and V present the comparison of area and power overhead with SFLL-rem, where REM represents SFLL-rem in the table. While the power overhead results of our proposed method do not surpass those of SFLL-rem, our method demonstrates superior performance in terms of area overhead with a clock period set to 6 ns, averaging 2.32%. At a clock period of 5 ns, our approach achieves nearly equivalent performance, with an overhead of 4.46%. These findings indicate that our proposed method can better reduce the area of the locked circuits. Compared to SFLL-rem [4], our method can achieve comparable area overhead while effectively resisting structural attacks, including Valkyrie [6] and SPI attack [7].

TABLE IV: Area comparison with SFLL-rem (key = 80)

Benchmark	Original (μm^2)				F_{enc} (μm^2)				Overhead (%)			
	REM (9ns)	Our (7ns)	Our (6ns)	Our (5ns)	REM (9ns)	Our (7ns)	Our (6ns)	Our (5ns)	REM (9ns)	Our (7ns)	Our (6ns)	Our (5ns)
b14_C	4530.60	2660.27	2669.31	2823.32	5002.00	2767.59	2872.92	3014.17	10.40	4.03	7.63	6.76
b15_C	7362.36	4088.42	4043.20	4167.16	7587.16	4330.60	4378.22	4491.27	3.05	5.92	8.29	7.78
b17_C	23285.16	13658.57	13793.16	14123.27	23008.12	13880.53	14007.42	14447.65	-1.19	1.63	1.55	2.30
b20_C	10150.56	6022.51	6750.81	6890.46	10633.12	6696.14	6659.17	7080.24	4.75	11.19	-1.36	2.75
b21_C	10011.60	5885.78	6452.36	6696.28	10563.64	6851.48	6513.13	6802.81	5.51	16.41	0.94	1.59
b22_C	15541.00	9211.58	9236.05	9204.93	15541.00	9555.64	9109.03	9717.63	2.21	3.74	-1.38	5.57
Average	-	-	-	-	-	-	-	-	4.13	7.15	2.61	4.46

TABLE V: Power comparison with SFLL-rem (key = 80)

Benchmark	Original (mW)				F_{enc} (mW)				Overhead (%)			
	REM (9ns)	Our (7ns)	Our (6ns)	Our (5ns)	REM (9ns)	Our (7ns)	Our (6ns)	Our (5ns)	REM (9ns)	Our (7ns)	Our (6ns)	Our (5ns)
b14_C	0.09	0.55	0.62	0.78	0.09	0.52	0.62	0.81	2.43	-5.36	-1.01	4.14
b15_C	0.08	0.44	0.52	0.64	0.08	0.48	0.53	0.64	-3.27	10.13	3.64	-0.88
b17_C	0.25	1.67	1.88	2.23	0.24	1.63	1.81	2.23	-3.48	-1.96	-3.67	0.04
b20_C	0.25	1.31	1.70	2.09	0.25	1.53	1.75	2.11	-0.69	17.20	2.87	1.10
b21_C	0.24	1.33	1.65	2.01	0.24	1.60	1.82	2.12	-0.21	20.24	10.36	5.19
b22_C	0.37	1.96	2.27	2.64	0.36	2.10	2.31	2.89	-3.76	7.14	1.72	9.49
Average	-	-	-	-	-	-	-	-	-1.50	7.90	2.32	3.18

D. Effectiveness of SAT and Valkyrie Attacks

To evaluate the security of our proposed method, we conducted SAT and Valkyrie attacks on the locked circuits. We set a time limit of 2 days for the SAT attack, and TLE is used to indicate "Time Limit Exceeded" when the attack was unsuccessful. Additionally, we use X to denote the Valkyrie attack terminates when no possible candidate critical signals are found. 32-bit key size was tested, and the outcomes are summarized in Table VI.

TABLE VI: Results of SAT and Valkyrie attacks

Benchmark	Key size (bit)	#SAT iteration	Valkyrie	Valkyrie runtime (s)
b14_C	32	TLE	X	293
b15_C	32	TLE	X	374
b17_C	32	TLE	X	561
b20_C	32	TLE	X	387
b21_C	32	TLE	X	764
b22_C	32	TLE	X	443

VII. Conclusions

Based on corrupt-and-correct (CAC) logic locking, we have proposed a novel method that utilizes the OFF-set only to corrupt the circuit. This approach enhances the integration of the additional circuitry with the original design, thereby increasing resilience against structural attacks while maintaining protection against SAT-based attacks. Additionally, our method demonstrates superior performance in reducing the area of the functionality stripped circuit compared to previous methods.

REFERENCES

[1] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "Ttlock: Tenacious and traceless logic locking," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 166–166, 2017.

[2] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, (New York, NY, USA), p. 1601–1618, Association for Computing Machinery, 2017.

[3] A. Sengupta, M. Nabeel, M. Yasin, and O. Sinanoglu, "Atpg-based cost-effective, secure logic locking," in *2018 IEEE 36th VLSI Test Symposium (VTS)*, pp. 1–6, 2018.

[4] A. Sengupta, M. Nabeel, N. Limaye, M. Ashraf, and O. Sinanoglu, "Truly stripping functionality for logic locking: A fault-based perspective," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4439–4452, 2020.

[5] L. Aksoy, "Hiid: A logic locking tool." https://github.com/leventaksoy/HIID.

[6] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability assessment tool and attack for provably-secure logic locking techniques," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 744–759, 2022.

[7] Z. Han, M. Yasin, and J. J. Rajendran, "Does logic locking work with EDA tools?," in *30th USENIX Security Symposium (USENIX Security 21)*, pp. 1055–1072, USENIX Association, Aug. 2021.

[8] P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters, "Read-proof hardware from protective coatings," in *Cryptographic Hardware and Embedded Systems - CHES 2006* (L. Goubin and M. Matsui, eds.), (Berlin, Heidelberg), pp. 369–383, Springer Berlin Heidelberg, 2006.

[9] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *2008 Design, Automation and Test in Europe*, pp. 1069–1074, 2008.

[10] J. Rajendran, H. Zhang, C. Zhang, G. S. Rose, Y. Pino, O. Sinanoglu, and R. Karri, "Fault analysis-based logic encryption," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 410–424, 2015.

[11] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *DAC Design Automation Conference 2012*, pp. 83–89, 2012.

[12] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 137–143, 2015.

[13] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "Sarlock: Sat attack resistant logic locking," in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 236–241, 2016.

[14] Y. Xie and A. Srivastava, "Anti-sat: Mitigating sat attack on logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 199–207, 2019.

[15] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517–532, 2020.

[16] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel bypass attack and bdd-based tradeoff analysis against all known logic locking attacks," in *Cryptographic Hardware and Embedded Systems – CHES 2017* (W. Fischer and N. Homma, eds.), (Cham), pp. 189–210, Springer International Publishing, 2017.

[17] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Security analysis of anti-sat," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 342–347, 2017.

[18] A. Sarabi, N. Song, M. Chrzanowska-Jeske, and M. A. Perkowski, "A comprehensive approach to logic synthesis and physical design for two-dimensional logic arrays," in *Proceedings of the 31st Annual Design Automation Conference*, DAC '94, (New York, NY, USA), p. 321–326, Association for Computing Machinery, 1994.

[19] R. Wang, L.-N. Hsu, Y.-C. Chen, and T. Hwang, "Expanding in-cone obfuscated tree for anti sat attack," in *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, 2023.

[20] NanGate, "Nangate freepdk45 open cell library." http://www.nangate.com/?page_id=2325.

[21] H. K. Lee and D. S. Ha, "On the generation of test patterns for combinational circuits," Tech. Rep. 12_93, Virginia Polytechnic Institute and State University, Department of Electrical Engineering, 1993.

[22] F. Corno, M. Reorda, and G. Squillero, "Rt-level itc'99 benchmarks and first atpg results," *IEEE Design & Test of Computers*, vol. 17, no. 3, pp. 44–53, 2000.

[23] F. Almeida, L. Aksoy, Q.-L. Nguyen, S. Dupuis, M.-L. Flottes, and S. Pagliarini, "Resynthesis-based attacks against logic locking," in *2023 24th International Symposium on Quality Electronic Design (ISQED)*, pp. 1–8, 2023.