PhaseCamouflage: Leveraging Adiabatic Operation to Thwart Reverse Engineering

Ivan Miketic, *Student Member, IEEE*, and Emre Salman, *Senior Member, IEEE*

***Abstract*—This paper focuses on thwarting reverse engineering attacks and IP theft by leveraging charge-recycling adiabatic circuits. Adiabatic circuit operation has recently received attention for Internet-of-things (IoT) applications due to high energy efficiency and enhanced security characteristics. Such applications typically consist of resource-constrained designs and are often deployed in the field, making them particularly vulnerable to malicious attacks. PhaseCamouflage is a circuit obfuscation technique that leverages the inherent phase differences in power supply voltage of adiabatic logic gates and exhibits strong resistance against structural/removal attacks. The proposed method relies on inserting camouflaged phase differences in the power supply voltage of subsequent logic gates while still producing a functional netlist. PhaseCamouflage is a unique logic obfuscation technique with low overhead, particularly applicable to pervasive computing applications where both efficiency and security are of primary concern.**

***Index Terms*—Logic obfuscation, camouflaging, layout-level, logic locking, IP theft, reverse engineering, adiabatic logic, charge-recycling logic, power-clock signals**

I. Introduction

The recent exponential increase in wirelessly connected devices has resulted in an urgent need for low overhead security primitives. Counterfeiting and hardware Trojan insertion are facilitated via reverse engineering attacks, thus raising significant concern for integrated circuits (ICs). Reverse engineering involves an attacker delayering a die and scanning the various layers to rebuild a gate-level netlist [1]. Once a functional netlist is generated, counterfeit designs that are potentially insecure and unreliable can be fabricated. Not only does reverse engineering pose a significant economic risk to the IC industry in the form of lost profits and reputation, but it also presents a significant risk to consumers and private data [2], [3]. There are two major techniques that have been developed to obstruct reverse engineers: *camouflaging* and *logic locking*.

IC camouflaging involves disguising a design through layout-level techniques. For example, dummy contact based camouflaging involves using a mix of dummy vias and real vias to disguise a standard cell library. After reverse engineering, a camouflaged OR gate may look identical to an AND gate [1]. This uncertainty leads to the reverse engineer guessing the gate functionality and potentially leads to incorrect extraction of the gate-level netlist. In order for camouflaging schemes to be most effective, they should ensure that an incorrect guess of logic gate results in an incorrect output. There are also various other camouflaging methods that rely on altering the doping of transistors [4] and intentional hot carrier injection to affect threshold voltage of transistors [5].

Camouflaging causes significant overhead since each camouflaged gate should look as if it can produce multiple Boolean logic functions; resulting in more nets and vias. Camouflaging each gate within an IC leads to unreasonable increase in power and area. As a result, various techniques were developed to insert camouflaged gates that add the most ambiguity and be most resistant to attacks [1].

Logic locking is another technique used to protect against reverse engineering, IP theft, and counterfeiting. Logic locking involves the addition of *key gates* into a design to affect the functionality of the netlist. A secret key should be supplied either from inputs or (preferably tamper-proof) memory to those gates in order for the design to function correctly. Thus, the entire scheme relies on the confidentiality of this key. Furthermore, these key gates should be inserted in such a manner where the key inputs are not easily attacked [6], [7]. In PhaseCamouflage, the camouflaged phase differences in the power supply voltage of the subsequent adiabatic gates behave similarly to a "key".

PhaseCamouflage consists of two steps: 1) camouflaging the power supply connections to an adiabatic gate with *dummy* vias and 2) inserting *obfuscated gates* into the netlist, where the phase difference in the power supply voltage of the obfuscated gate and the previous gate is different than the conventional phase difference (which is 90° for efficient charge recovery logic, an adiabatic family that is considered in this paper [8]). If an attacker attempts to reverse engineer the hardened design and applies a 90° phase difference in the power supply voltages of subsequent gates, it would result in incorrect functionality, thus protecting the IP/design. If the attacker then attempts to determine the correct phase difference for all of the camouflaged gates, they would need to perform a brute force attack on the design, guessing the phase difference for each gate with camouflaged power supply connections.

It is important to note that unlike logic locking, PhaseCamouflage does not protect against supply chain attacks [9] since foundries are aware of which vias are dummy vias. As such, PhaseCamouflage does not protect against overproduction of counterfeit ICs. Instead, PhaseCamouflage provides a similar amount of security that camouflaging techniques provide against reverse engineering and unauthorized modification of an obfuscated netlist. Therefore, PhaseCamouflage is introduced in this paper as a logic obfuscation technique (rather than logic locking) as it does not need to be "unlocked" with a secret key [9]. Since IoT devices are often left *in situ*, they are particularly vulnerable to attacks by users, making

*The authors are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794 USA (e-mail: emre.salman@stonybrook.edu)

This research was supported in part by the National Science Foundation under grant number 1717306 and in part by Semiconductor Research Corporation under contract number 2017-TS-2767.

PhaseCamouflage a promising approach for power-constrained IoT applications. To the authors' best knowledge, this work represents the first study where inherent phase differences in adiabatic logic are leveraged for logic obfuscation. The power and area overhead are investigated. An analysis of output corruptibility is performed by measuring Hamming distances. The resistance of the proposed approach to modern attacks against camouflaging and logic locking is also discussed.

The rest of this paper is organized as follows. Background on charge-recycling adiabatic operation, logic locking, and camouflaging is provided in Section II. The proposed methodology is detailed in Section III. Results are presented in Section IV. Finally, the paper is concluded in Section V.

II. Background and Related Work

The particular charge-recycling adiabatic logic family used in this work is efficient charge recovery logic (ECRL), as summarized in Section II-A. Background on logic locking and camouflaging, various attacks and modern countermeasures are covered in Section II-B.

A. Adiabatic/Charge-Recycling Logic

Adiabatic logic utilizes a time-varying (either a trapezoidal or sinusoidal) power supply voltage rather than the conventional DC voltage. Significant reduction in power consumption is achieved via (a) minimizing resistive loss across the transistors by ensuring that the voltage difference (therefore current) across the transistor is very small during charging and (b) the charge stored at the output node is partially recycled back to the power supply when the supply voltage is reduced. Since one full cycle of the time-varying power supply signal is divided into multiple phases, the power supply signal also acts as the global clock signal and typically referred to as power-clock signal. Due to this multi-phase operation, adiabatic logic is inherently pipelined.

ECRL is utilized in this work, where a 4-phase sinusoidal power-clock signal is used, as illustrated in Figs. 1(a) and (b) [10]. In this adiabatic logic family, the phase difference between the power-clock signals of adjacent gates (i.e. among *pc1*, *pc2*, *pc3*, and *pc4*) should be 90°. As such, each power-clock signal is divided into four stages: evaluation (E), hold (H), recovery (R), and wait (W). For example, during *evaluation* (when the power-clock signal is rising), the logic gate turns on and evaluates the input signal. Then, as the power-clock signal reaches the *hold* stage, the following logic gate uses the output of the first stage for evaluation (due to 90° phase difference). The *recovery* stage is when the charge at the output node of the gate is recycled back to the power-clock signal (when the power-clock signal is decreasing). Finally, the *wait* stage is for clock symmetry and for the time when inputs are being prepared in the previous gate [8]. For example, in Fig. 2, each shaded region represents a particular clock phase. In the first stage, the AND and XOR gates are connected to $pc1$ (since both of these gates have the same logic depth) whereas, in the next stage, the AND gate and OR gate are connected to $pc2$ (which has 90° phase difference with respect to $pc1$).

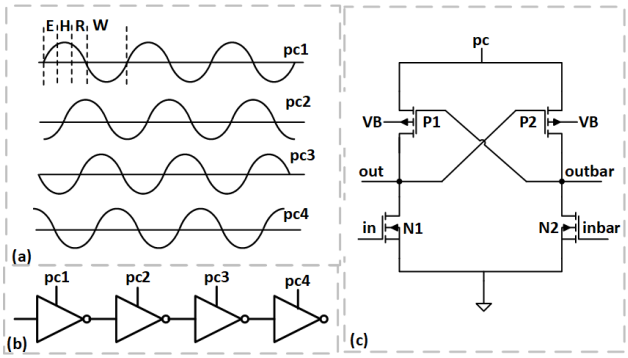


Fig. 1. Overview of efficient charge recovery logic (ECRL): (a) four required power-clock signals with 90° phase difference. Each power-clock signal is divided into 4 stages (evaluation, hold, recovery, wait), (b) four cascaded inverters where the power-clock signal of each inverter has 90° phase difference with respect to the adjacent inverters, (c) transistor-level schematic of an inverter in ECRL.

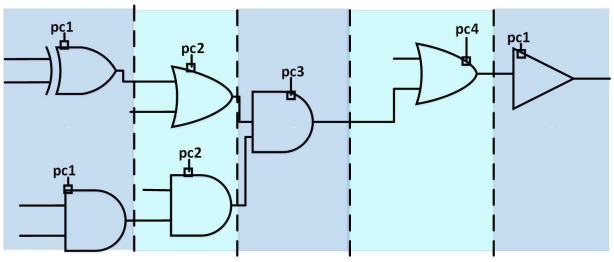


Fig. 2. An example gate-level netlist in conventional ECRL, illustrating inherent pipelining and the 90° phase difference among the gates.

An inverter designed with ECRL is illustrated in Fig. 1(c). Each ECRL gate consists of a cross-coupled pMOS pair and a pull-down network that uses complementary inputs and outputs. The operation of an ECRL inverter is described as follows: assume *in* is high and the power-clock signal is connected to *pc1*. As *pc1* rises during the *evaluation* stage, *out* goes to logic-low since N1 is turned on. Alternatively, *outbar* remains at logic-high since P2 is turned on and N2 is off. During the *hold* stage, the output of the inverter remains constant as the input of the subsequent gate that is connected to *pc2*. Once *pc1* enters the recovery stage, charge is recovered from *outbar* back to *pc1*. As *pc1* reaches the *wait* stage, power to the gate is turned off, resulting in logic-low at both *out* and *outbar*.

Adiabatic logic has recently received interest in resource-constrained security applications. For example, new charge-recycling logic families have been developed to maximize energy efficiency and increase resistance against power-based side-channel attacks, where secret key bits are extracted by analyzing power consumption for different input patterns [11], [12]. It was demonstrated that charge-recycling logic has relatively uniform current consumption, thus the same power-based side channel attacks on CMOS designs are not sufficiently successful on designs implemented in adiabatic logic [12]. In [10], [13], adiabatic operation was leveraged in an AC computing paradigm for wirelessly powered devices. In this methodology, overhead related to AC-to-DC conversion (required in traditional RF-powered devices) is eliminated

since harvested signal is sinusoidal, as needed by ECRL. Recently, novel communication protocols have been developed that are compatible with charge-recycling based AC computing methodology [14], [15]. Unlike these existing studies that focus on efficiency and side-channel resistance of adiabatic logic, in this paper, the primary emphasis is on leveraging adiabatic operation for lightweight circuit camouflaging.

B. Attacks and Countermeasures against Logic Locking and Circuit Camouflaging

Early logic locking schemes used various forms of combinational logic such as XOR gates [16], look-up tables (LUTs) [17], and MUXs [9] as key gates. Initially, these key gates were inserted into a netlist at random points, but this was found to be easily broken through sensitization attacks [18]. Since key gates were randomly inserted, large parts of the key could easily be deciphered by using traditional automatic test pattern generation (ATPG) tools where key values are propagated to primary outputs of the design [19]. The computational complexity of retrieving the key bits was reduced to linear time, which is not a sufficient level of security [18]. This led to key gates being inserted in such a way that they mutually interfere with each other [19]. Similarly, a sensitization attack against camouflaging involves producing a truth table of a target camouflaged gate by justifying the output of the target gate and *sensitizing* it to a primary output of the netlist [1]. As a result, the Boolean logic of the camouflaged gate would no longer be ambiguous. A defense to this kind of attack is a clique based selection method that inserts mutually interfering camouflaged gates [1].

When SAT attack techniques on logic locking and circuit camouflaging were developed, the security level of existing logic locking and camouflaging methods was significantly compromised [20], [21]. The SAT attack relies on running various input patterns through a functional device, obtaining the corresponding correct outputs, and using each input pattern to remove potential key values (in the case of logic locking), and sets of potential Boolean functionalities (in the case of circuit camouflaging) of the locked/obfuscated netlist. Since each input pattern is capable of removing multiple key guesses and multiple sets of camouflaged gate functionalities from the search space, it greatly reduces the theoretical exponential difficulty that was thought to be needed to reverse engineer locked and camouflaged netlists. Defense mechanisms against SAT attacks of camouflaged circuits generally rely on flipping output bits so that a distinguishing input pattern (DIP) does not decipher multiple incorrect keys [22], [23].

Attackers can also perform removal attacks when they are able to discern the actual locking hardware/logic used in the netlist from the original circuit. This technique renders the locking/camouflaging scheme useless since the attacker can simply cut out those locked/camouflaged portions of the netlist. It is therefore important that the locking/camouflaging mechanism of designs is intertwined with the original netlist. Examples of removal attacks are demonstrated in [24], where modern SAT resilient locking schemes (such as anti-SAT) are compromised [6].

Newer developments in camouflaging and logic locking have introduced the idea that functionality can also rely on the timing constraints of the design. For example, TimingCamouflage uses wave-pipelined paths that require attackers to guess whether a path uses single period clocking or pipelining with two data waves [25]. Delay locking introduces tunable key gates, where the key value alters the delay of combinational paths by introducing additional capacitance [26]. Reverse engineers therefore must not only recreate a netlist, but also correctly guess the timing of the logical paths. PhaseCamouflage, as introduced in this paper, alters the phase differences within power-clock signal connections of adiabatic logic. Similar to TimingCamouflage, PhaseCamouflage achieves combinational obfuscation through non-combinational means.

III. Proposed Methodology: PhaseCamouflage

In the proposed methodology, dummy vias are used to camouflage the power-clock connections of adiabatic gates. An attacker needs to guess the correct phase difference among the gates to obtain a functional netlist. By changing the number of camouflaged gates, PhaseCamouflage can provide the desired security level at significantly lower overhead as compared to existing techniques, making it particularly applicable to resource-constrained applications.

The concepts of *camouflaged* and *obfuscated* gates required for PhaseCamouflage are introduced in Section III-A. In Section III-B, circuit operation with non-standard power-clock phase differences (different than 90°) is described. Methods to insert obfuscated gates in a netlist are discussed in Section III-C. Finally, in Section III-D, the equivalent security level achieved via PhaseCamouflage is quantified.

A. Camouflaged vs. Obfuscated Gates in PhaseCamouflage

Dummy vias are the root of security in PhaseCamouflage since an attacker needs to guess the correct power-clock signal connections of the gates to obtain a functional netlist. Referring to Fig. 3, in a camouflaged gate, there are four power-clock signal connections to the gate (*pc1* to *pc4*) with 90° phase difference (see Fig. 1). Three of the four power-clock connections are made with a dummy via/contact. From an attacker's perspective, the physically correct power-clock connection could be any of the four possible power-clock signals in ECRL operation.

It was demonstrated that it is very difficult for an attacker to etch the fine geometries (tens of nanometers thick) necessary to discern between a dummy via and a real via [1]. Furthermore, it is infeasible for an attacker to reverse engineer every via in a design. Also, the bottom layers of a reverse engineered die could potentially be partially eroded from the chemicals used to etch through the upper metal layers, making it difficult to distinguish a dummy via from a partially eroded one [27].

The gates that have camouflaged power-clock signal connections, as depicted in Fig. 3, are referred to as *camouflaged gates*. These camouflaged gates can be created by modifying already existing gates within the original netlist or by inserting new gates. The true power-clock signal of some of these camouflaged gates can have a phase difference (with respect to

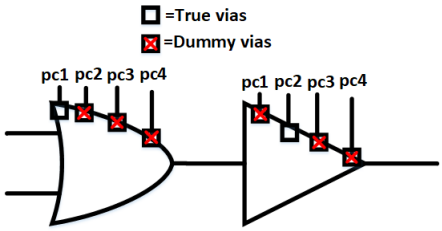


Fig. 3. Two adjacent ECRL gates with camouflaged power-clock connections.

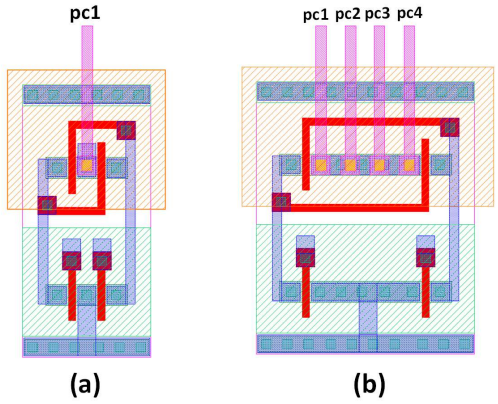


Fig. 4. Layout views of (a) a conventional ECRL gate with a single power-clock connection, (b) camouflaged ECRL gate with four power-clock signals.

the power-clock signal of the previous gate) that is *different* than the conventional 90° (either 0° or 180°, as described later). These camouflaged gates where the phase difference between the power-clock signals is not 90°, are referred to as *obfuscated gates*. Note that by this definition, any *obfuscated gate* also has to be a *camouflaged gate* (otherwise the reverse engineer would not have to guess the correct power-clock connection), but a *camouflaged gate* does not have to be an *obfuscated gate*, since it is possible for the camouflaged gates to have a 90° phase difference with respect to the previous gate. Note that the phase difference here refers to the difference in the phase between the power-clock signal of the current gate and the gate immediately preceding it.

An attacker cannot distinguish an obfuscated and camouflaged gate. Thus, it is possible to introduce a varying degree of security (and therefore overhead) by changing the number of obfuscated and camouflaged gates that are introduced to the netlist. More obfuscated gates may lead to more overhead than simply changing existing gates in the netlist to camouflaged gates. This is described more in detail in Section III-D.

A conventional ECRL buffer is compared with a camouflaged ECRL buffer in Fig. 4. Three of four power-clock nets are connected to the sources of the cross-coupled pMOS devices with dummy vias. The area overhead incurred by camouflaging a single conventional gate is approximately 70%. Note that PhaseCamouflage incurs much less overhead at the chip-level than dummy-based camouflaging [28], [29] since high level of security can be achieved with significantly less number of camouflaged gates.

B. Obfuscating with Phase Differences

This section focuses on utilizing non-standard phase differences within the netlist and how a functional netlist can be maintained despite these phase differences that normally do not exist in ECRL operation. Note that in traditional ECRL operation, the phase difference between any two consecutive gates is 90°. In the following sections, two potential phase differences (0° and 180°) in addition to the conventional 90° are introduced as viable options for producing functional logic in an ECRL netlist. For clarity, the following convention will be used in the figures detailing the methodology of PhaseCamouflage:

- Red logic gates indicate *obfuscated gates*.
- Orange logic gates indicate *camouflaged gates*
- Red interconnect lines indicate a path that has been obfuscated with a 180° or 0° obfuscated gate.
- Orange interconnect lines indicate a path that has been altered or inserted as part of PhaseCamouflage technique.
- The combination of all these items form a *camouflaged path*, referred to as a connected set/group of camouflaged gates that mutually interfere with each other (details are described in Section III-C).

1) 90° Phase Difference: This scenario represents the conventional case in ECRL operation, as shown in Fig. 2, where there is 90° phase difference between $pc1$ and $pc2$. In PhaseCamouflage with 90° phase difference, a camouflaged gate is in the $evaluation$ stage while the previous gate is in the $hold$ stage, resulting in expected logical computation. Note that by definition, an obfuscated gate cannot have 90° phase difference whereas a camouflaged gate could. Since an attacker cannot tell the difference between camouflaged and obfuscated gates, this ambiguity makes 90° phase difference a viable/necessary guess from attacker's perspective. Thus, the conventional 90° phase difference can be used to make the reverse engineering more difficult, as further described in Section III-D.

2) 180° Phase Difference: In this section, the feasibility of using a 180° phase difference between the power-clock signals is demonstrated. Specifically, assume that there is 180° phase difference between the power-clock signals of an obfuscated gate and the previous gate (such as $pc1$ and $pc3$). Then, when the obfuscated gate is in $evaluation$ stage, the previous gate is in $wait$ stage [see Fig. 1(a)]. Due to the relationship between $evaluation$ and $wait$ stages of a sinusoidal power-clock signal, both inputs of the obfuscated gate are at logic-low, as depicted in Fig. 5(a). This produces a race condition between the two cross-coupled pMOS transistors of the obfuscated gate because the nMOS transistors are off. This race condition produces a constant output of either 0 or 1 for the obfuscated gate, depending upon which pMOS within the cross-coupled pair "wins" the race condition, as shown in Fig. 5(b) where the output of the obfuscated gate goes to logic-high during each $evaluation$ stage. In PhaseCamouflage, this constant output is utilized as a *non-controlling value* and strategically inserted into the functional netlist to maintain correct operation. Thus, functionally correct operation is ensured despite a non-standard 180° phase difference.

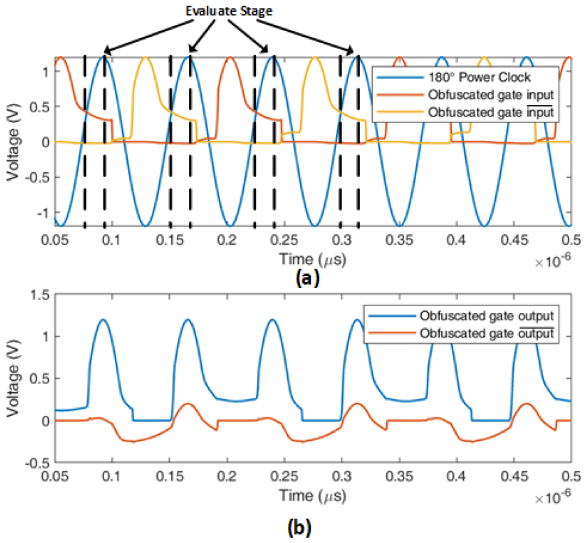


Fig. 5. Illustration of the effect of 180° phase difference on ECRL operation: (a) input signals to an obfuscated gate being evaluated during the wait stage of the previous gate, (b) corresponding constant output signal of the obfuscated gate.

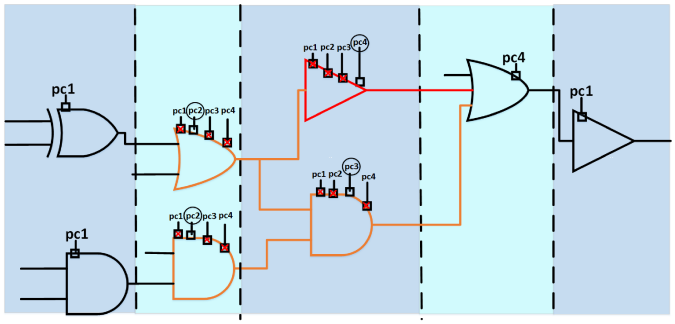


Fig. 6. Functional netlist with a camouflaged path using a 180° obfuscated gate (shown in red) that outputs a constant logic-low. Physically correct power supply connections are circled.

For example, if Fig. 2 represents the original netlist, Fig. 6 shows an obfuscated netlist of this circuit where the inserted red buffer is the obfuscated gate with 180° phase difference with respect to the previous gate ($pc2$ and $pc4$). The existing OR gate is modified from 2 inputs to 3 inputs. Three existing gates are modified into camouflaged gates (orange color). This obfuscated netlist remains functionally correct provided that the output of the obfuscated gate (red buffer) is at logic-low. Furthermore, there is only a single set of valid power-clock connections that results in correct functionality (circled connections in the figure). Thus, the circuit produces incorrect output if a conventional 90° phase difference is applied between all of the gates. The combination of red obfuscated gates and orange camouflaged gates represents the camouflaged path. It should be noted that if the constant output of the obfuscated gate were logic-high, then this non-controlling value could be used with an existing AND gate.

To reliably use 180° phase difference for obfuscation, the previously described race condition should result in a predetermined value (logic-low in the example shown in Fig. 6).

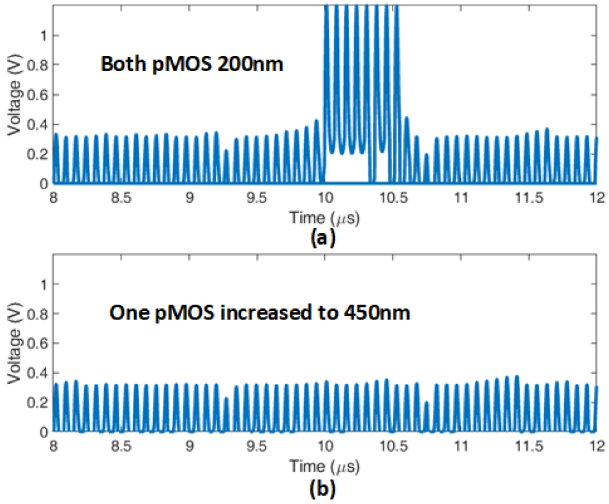


Fig. 7. Illustration of the mismatch on cross-coupled pMOS transistors to ensure a constant output for 180° obfuscated gates: (a) both pMOS transistors have the same size, (b) one of the pMOS transistor size is increased.

This objective can be achieved via introducing mismatch (in, for example, size or threshold voltage) between the two cross-coupled pMOS transistors. An example is shown in Fig. 7 where the output node of a 180° obfuscated gate is plotted for different pMOS transistor sizes. Specifically, if the size of both pMOS transistors is equal to 200 nm, the output voltage varies between logic-high and logic-low. Alternatively, if one of the pMOS size is increased to 450 nm (while maintaining the size of the other pMOS), the output voltage is maintained at logic-low, which can be used as a non-controlling value. Since mismatch is intentionally inserted into the design, corner simulations are performed on inserted 180° obfuscated gates to ensure reliability. Correct functionality of the netlist and the constant output of 180° gates is maintained at all process corners, provided that sufficient mismatch is introduced.

An intentional mismatch in the two cross-coupled pMOS transistors of an obfuscated gate, however, can leak information to attackers about which gate is a 180° obfuscated gate, thus exponentially decreasing the search space (explained further in Section III-D). A countermeasure to this leakage issue is to introduce mismatched pMOS pairs to other camouflaged gates, which are a very small percentage of the overall number of gates in a design. Extending this mismatch to other camouflaged gates does not affect functionality for 0° and 90° gates since in those cases, the pull-down network of the gates is not turned off.

It is important to note that any Boolean logic gate can be used as an obfuscated gate with 180° phase difference as long as there is intentional mismatch within the cross-coupled pMOS pair. This freedom to choose any type of cell as the obfuscated gate makes it highly challenging for reverse engineers to narrow the valid search space and therefore prevents removal attacks.

3) 0° Phase Difference: An obfuscated gate can also have a phase difference of 0° with respect to the previous gate (i.e. same power-clock signal is used for both gates). When there

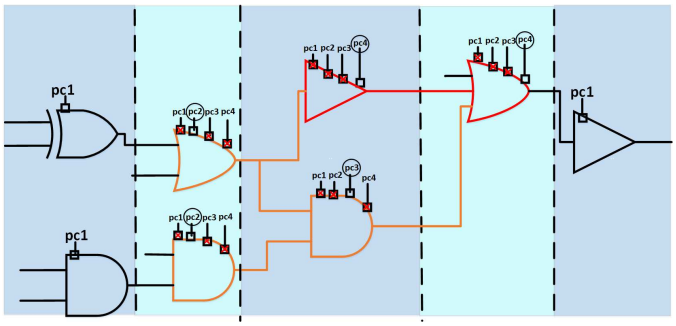


Fig. 8. Functional netlist with a camouflaged path that includes a 180° obfuscated gate and a 0° obfuscated gate.

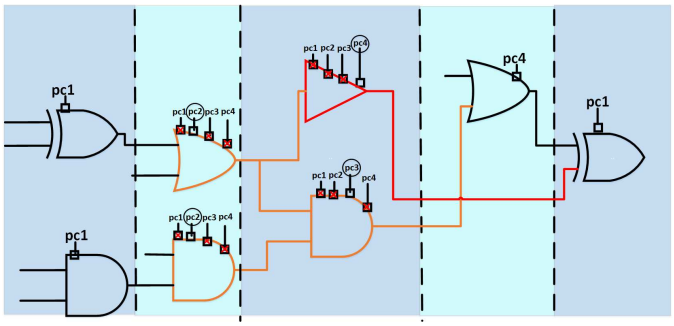


Fig. 9. Example of replacing a buffer in the original netlist with an XOR gate and utilizing a constant 0 output from a 180° obfuscated buffer.

is no phase difference, the input data signal passes through both consecutive gates during the same *evaluation* stage of the power-clock signal. This scheme sacrifices the inherent power savings of the adiabatic switching since the second gate computes with a power-clock signal that does not start from zero volt. The increase in overall power consumption, however, is negligible for large circuits since the number of obfuscated cells is sufficiently small, as further described in Section III-D.

0° obfuscated gates should be inserted in a way that they cannot be simply substituted with a 90° phase difference since that would reduce the search space for the attacker. For example, in Fig. 8, the existing OR gate after the 180° obfuscated gate has been modified to become a 0° obfuscated gate. If an attacker guesses 90° phase difference (*pc1*), the netlist would not function correctly because there would then be a 180° phase difference between the OR gate and preceding AND gate. Thus, for 0° obfuscation, it is important to choose a multi-input gate where the preceding stage has 180° phase difference.

Note that a phase difference of 270° is not a feasible option for obfuscation in adiabatic ECRL circuits because in that case, the obfuscated gate would be in the *evaluation* stage while the previous gate would be in *recovery* stage. Since the output of a gate during the *recovery* stage is input dependent, it is not possible to use this output as a non-controlling value (as proposed for 180°). Thus, 270° is not a viable guess for an attacker and can be removed from the search space.

To summarize, an attacker should correctly guess the power-clock connection of each camouflaged gate, of which there are three viable options: 0°, 90°, and 180°. More discussion is provided in the following section about how camouflaged paths should be created to look like they are part of the original netlist, thereby significantly strengthening the obfuscation capability.

C. Selection Process Guidelines for Removal Attacks

The strength of PhaseCamouflage stems from the potential for any logic cell to be used as an obfuscated gate. Thus, there are many options of inserting obfuscated gates into a netlist or transforming existing gates into obfuscated gates.

It is important to diversify camouflaged paths when obfuscating a netlist to ensure a sufficiently large search space. If most of the camouflaged paths appear identical, the attacker may "unlock" the entire netlist by guessing the same set of phase differences for all of the camouflaged paths, or even worse, the obfuscated gates within the camouflaged paths may become blatant to the attacker and they could be susceptible to removal attacks, as explained in Section II-B. The same approach applies when inserting individual obfuscated gates. These gates should be indistinguishable from normal logic so that the method is not susceptible to removal/structural attacks. The structural dependencies of 180° and 0° camouflaged gates do not leak information to attackers because of the variety of ways these gates can be inserted. For example, 180° camouflaged gates do not always have to be followed by 3 input AND or OR gates. As illustrated in Fig. 9, an existing buffer can be replaced with an XOR gate and be routed to a constant 0 (output of a 180° obfuscated gate), thus maintaining functionality. Additionally, the constant output can be used as a controlling value on inserted logic and used as a non-controlling value at a later stage within the netlist, at the cost of additional overhead. Even though it is true that a 0° camouflaged gate should be preceded by a 180° camouflaged gate, a 180° camouflaged gate does not always have to be followed by a 0° camouflaged gate. These options of using the constant output as a controlling value for multiple stages; routing to multi-input OR/AND gates; and changing an existing buffer/inverter to an OR/AND/XOR gate represent sufficient variety to prevent information leakage through structural dependencies of PhaseCamouflage.

To summarize, the following guidelines should be followed when inserting camouflaged paths in order to thwart removal and ATPG based sensitization attacks:

- 0° obfuscated gates should be used in combination with 180° obfuscated gates.
- Camouflaged and obfuscated gates should be used to form non-resolvable gates using existing methods [1].
- Different camouflaged paths should consist of contrasting logic cells.
- All of the proposed phase differences (0°, 90°, and 180°) should be used throughout the design, however they do not all have to be used within the same camouflaged path.
- Multiple fan-in gates should be used as obfuscated gates to connect multiple camouflaged paths together.
- 180° and 0° obfuscated gates should be inserted in diverse ways (as exemplified in Figs. 6, 8 and 9) so

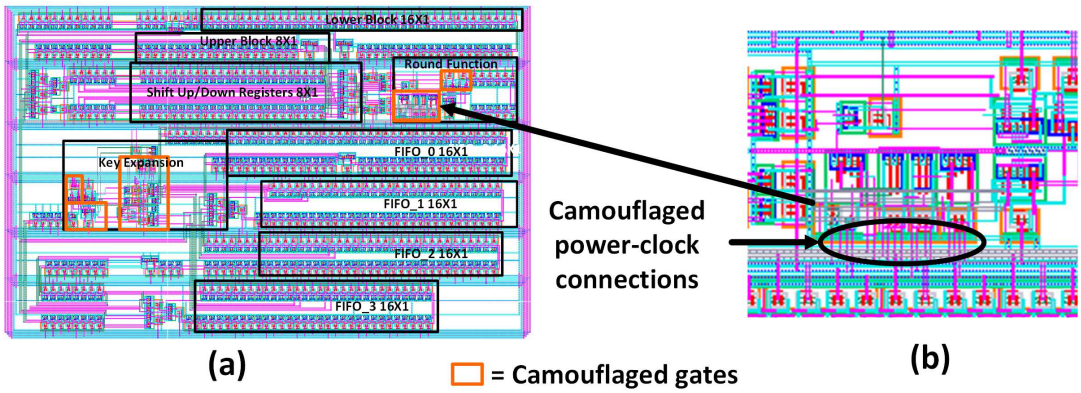


Fig. 10. The layout view of camouflaged SIMON core: (a) full layout illustrating the primary blocks (black rectangles) and inserted camouflaged gates (orange rectangles), (b) zoomed in view of a portion of the layout illustrating the camouflaged gates and corresponding power-clock connections.

that the correct phase difference is not revealed via the physically observable structures within the netlist.

D. Equivalent Security Level

In the proposed methodology, an attacker needs to guess the phase differences among camouflaged gates, where the phase difference between two consecutive gates can be 0°, 90°, or 180°. If n is the number of camouflaged gates within a design, 3^n is the overall complexity an attacker faces when attempting to guess the phase differences in a brute force manner. This exponential relationship is due to the independent characteristic of the camouflaged gates. Thus, correctly guessing a phase difference of a particular camouflaged gate does not provide any hints for the phase differences of other camouflaged gates.

Compared to traditional logic locking techniques that rely on a secret key, PhaseCamouflage has a larger base in the complexity (3^n vs 2^m where m is the number of key bits in conventional logic locking). Thus, PhaseCamouflage requires smaller number of camouflaged gates (compared to the number of obfuscated gates) in order to realize the same security level. Specifically, to achieve the same level of security as an m bit key, the number of camouflaged gates n in PhaseCamouflage should be equal to $\lceil \log(2^m)/\log(3) \rceil$. For example, to achieve a security level that is equivalent to a 128-bit key, a designer should insert only 81 camouflaged gates into the design.

In PhaseCamouflage, output corruptibility (the likelihood that an incorrect guess in a camouflaged netlist results in an incorrect output) is sufficiently high due to the strong dependence of functionality on phase difference of the power-clock signals in ECRL circuits. For a 180° obfuscated gate, any guess other than 180° produces an incorrect output, as discussed in Section III-B2. For a 0° obfuscated gate, however, either only a 0° guess or both a 0° and 90° guess can produce a correct output, depending upon how the obfuscated gate is inserted, as discussed in Section III-B3. Similar to existing logical camouflaging schemes, there may be cases where unintended combinations of phase differences for camouflaged gates result in correct functionality. The guidelines described in Section III-C should be followed to minimize the set of valid power-clock connections. For example, in Fig. 8, the netlist has only one set of valid power-clock connections, where the search space consists of 243 different phase differences (3^5 since there are 5 camouflaged gates). If PhaseCamouflage is implemented as described in this paper, the attacker needs to perform exhaustive simulations, equivalent of brute force, in order to recover the functional and correct netlist. Thus, depending upon the application, an appropriate security level should be chosen to ensure that an exhaustive search is computationally infeasible.

This discussion on equivalent security levels is based on brute force attacks. In future work, once more sophisticated attacks such as SAT and sensitization are formulated with revised tools for adiabatic logic, quantitative metrics (such as number of input patterns and overall time to break the camouflaged circuit) can be determined to have a more applicable comparison. The transformations and discussion in Section IV-C serve as the preliminary investigation into such real-time attacks.

IV. Results

An 8-bit multiplier, a SIMON encryption core, and the ISCAS-85 benchmark circuit c432 are implemented in static CMOS, conventional adiabatic ECRL, and ECRL with the PhaseCamouflage methodology in a 65 nm commercial technology node. The operating frequency is 13.56 MHz. The circuits in this work are designed with a full custom methodology in Cadence Virtuoso, and results are obtained through Spectre simulations [30] [31]. The SIMON encryption core is designed to encrypt a 32-bit plaintext with a 64-bit key, where one bit of plaintext goes through one round function in one clock cycle, representing a bit-serial implementation [32]. In circuit camouflaging literature, the number of camouflaged gates inserted to a design is either a constant number [33]–[35] or a percentage of the total gates [1], [34], [36]. Both designs are obfuscated with an equivalent security level of a 32-bit key, requiring 21 camouflaged gates. For the SIMON core, 8 of the 21 camouflaged gates are obfuscated gates. For the multiplier, there are 10 obfuscated gates. When inserting these camouflaged gates into SIMON core and multiplier, different strategies were adopted for evaluation. Specifically,

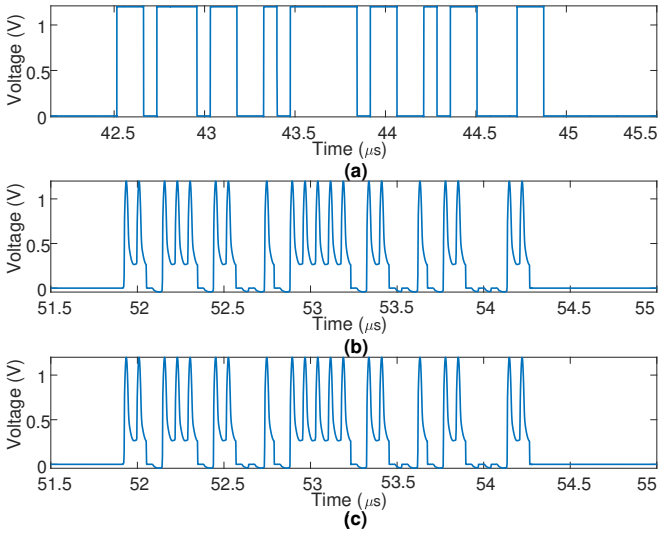


Fig. 11. SIMON ciphertext serial output: (a) static CMOS, (b) unprotected ECRL, (c) camouflaged ECRL using PhaseCamouflage with correct phase differences.

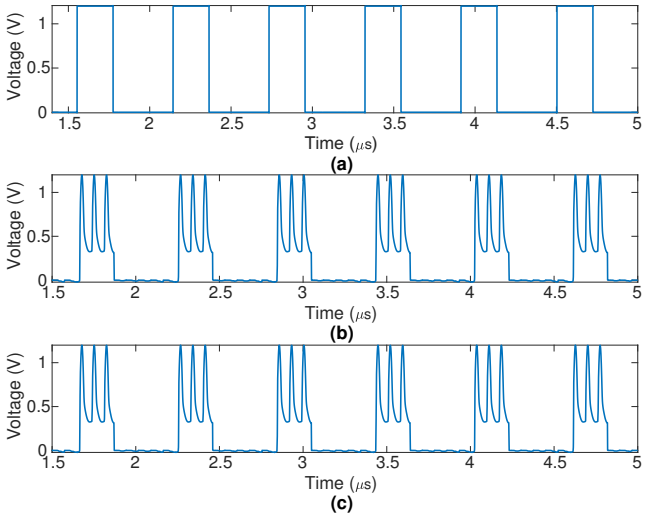


Fig. 12. Most significant bit of 8-bit multiplier: (a) static CMOS, (b) unprotected ECRL, (c) camouflaged ECRL using PhaseCamouflage with correct phase differences.

for the SIMON core, the primary objective was to minimize the overhead whereas for the multiplier, the design guidelines described in Section III-C were followed to strengthen the protected design against removal and sensitization attacks. Note that PhaseCamouflage would be weaker if an attacker could reverse engineer one set of obfuscated/camouflaged gates and simply look for similar sets of gates within the netlist. As an example, the camouflaged SIMON layout illustrating the camouflaged gates and camouflaged power-clock signal connections is shown in Fig. 10.

Since the SIMON encryption core is implemented in a bit-serial fashion, a large portion of the design consists of FIFOs. Obfuscated gates were inserted into the round and the key expansion blocks since the majority of the combinational logic exists in these two blocks.

The obfuscation of the multiplier focused on demonstrating the versatility of PhaseCamouflage by using multiple types of logic gates (AND, OR, XOR) as obfuscated gates. Since the 8-bit multiplier is an entirely combinational design, obfuscated gates were inserted throughout the entire netlist. A variety of 180° obfuscated gates (which require more overhead to maintain functionality) were also used.

Correct functionality is demonstrated among the three implementations (static CMOS, unprotected ECRL, protected ECRL using PhaseCamouflage with correct phase differences) of both the SIMON core and 8-bit multiplier. The serial output for SIMON core and the most significant output bit of the 8-bit multiplier are shown, respectively, in Figs. 11 and 12 for each implementation. Note that the ECRL implementations of the SIMON core exhibit slightly higher latency than static CMOS due to synchronization overhead of charge-recycling logic.

An incorrect guess in the phase difference of only one camouflaged gate (out of 21 camouflaged gates) produces a wrong output in both the SIMON core and the multiplier, as shown in Fig. 13. According to this figure, a single incorrect guess drastically alters the functionality of the circuit, thereby

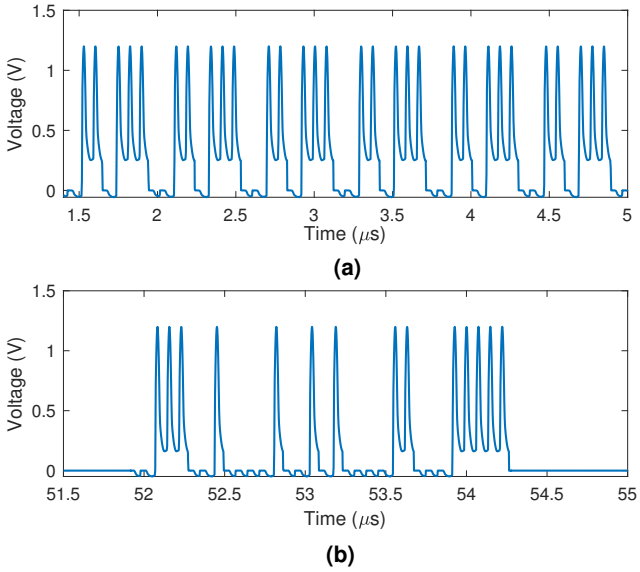


Fig. 13. Illustration of incorrect output when one of the phase difference guesses (out of 21 camouflaged gates) is wrong: (a) most significant bit of 8-bit multiplier and (b) SIMON ciphertext serial output.

demonstrating that camouflaged gates can produce large output corruptibility.

A. Power and Area Overhead

As mentioned in the previous section, the SIMON core was camouflaged to minimize overhead whereas the multiplier was camouflaged according to the guidelines described in Section III-C to thwart removal attacks. This difference can be observed in overhead results, even though both designs achieve the same security level of a 32-bit key.

Area and power results are listed in Table I for three implementations of the SIMON core. Compared to static CMOS implementation, an ECRL based SIMON reduces power consumption by 76.5%. The power savings slightly degrade to

75.9% for the protected ECRL via PhaseCamouflage. The area of the unprotected ECRL and protected ECRL cores are, respectively, 2.2% and 5.8% larger than the static CMOS core. For the 8-bit multiplier, the overhead is greater, as listed in Table II. The area of the camouflaged ECRL increases by approximately 11% as compared to unprotected ECRL. The power consumption increases by approximately 24%, thereby decreasing the power savings (as compared to static CMOS) from approximately 58% to 47%. It is important to note that for larger designs, the area and power overhead is expected to be much less since the required number of camouflaged gates remains the same to achieve the same security level.

TABLE I
OVERHEAD IN AREA AND POWER FOR THE SIMON ENCRYPTION CORE.

	Area (μm^2)	Power (μW)
Static CMOS	4068 (N/A)	21.99 (N/A)
Unprotected ECRL	4159 (+2.2%)	5.17 (-76.5%)
PhaseCamouflage ECRL	4303 (+5.8%)	5.30 (-75.9%)

TABLE II
OVERHEAD IN AREA AND POWER FOR THE 8-BIT MULTIPLIER.

	Area (μm^2)	Power (μW)
Static CMOS	1275 (N/A)	3.35 (N/A)
Unprotected ECRL	1599 (+25.4%)	1.42 (-57.6%)
PhaseCamouflage ECRL	1780 (+39.6%)	1.76 (-47.5%)

B. Corruptibility Analysis

The output corruptibility of PhaseCamouflage is quantified by measuring the Hamming distance between the outputs of a correct netlist (i.e. correct phase differences) and incorrect netlist (i.e. incorrect phase difference guesses). Hamming distance was computed over 100 different input patterns, with 16 different sets of phase differences in random locations of the ISCAS'85 c432 benchmark circuit, which consists of 160 gates and has 7 output bits [37]. The benchmark circuit is obfuscated with an equivalent security level of a 32-bit key (21 camouflaged gates, of which 8 are obfuscated gates). Ten random camouflaged gates are given an incorrect phase difference for the incorrect netlist. The Hamming distance averaged 57% over the 16 phase difference sets, ranging from 14% to 86%, as shown in Fig. 14. Since the ideal Hamming distance percentage is 50%, it is highly likely for an incorrect phase difference guess to produce an incorrect output, which means that utilizing a constant logic-low or logic-high as an outcome (instead of a Boolean function as in traditional circuit camouflaging schemes) is effective.

The average Hamming distance of 57% is largely due to the high number of camouflaged gates in relation to the total number of gates in the c432 benchmark circuit. Furthermore, output corruptibility is highly dependent on circuit topology and the location of camouflaged gates, since camouflaged gate outputs can be masked by other converging paths. The results reported in this section are based on random selection of camouflaged gates within the netlist. The wide range of

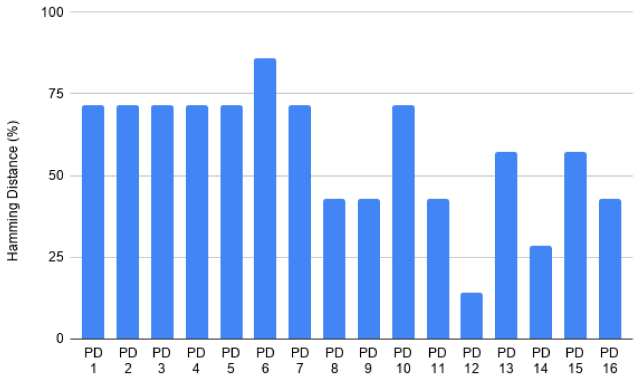


Fig. 14. Average Hamming distance percentages between the output bits obtained with correct phase differences and output bits obtained with incorrect phase differences. Hamming distance is calculated for 16 different sets of phase differences (PDs) where each set is evaluated for 100 different inputs.

Hamming distances for different phase difference guesses demonstrates that the location of camouflaged gates plays an important role in output corruptibility. The high average output corruptibility typically means that the camouflaged circuit is more susceptible to SAT attacks (since less DIPs are needed to prune away the wrong combinations of phase differences [38]). This susceptibility can be mitigated by selecting camouflaged gates such that these gates have logical paths that converge to a smaller subset of the primary outputs.

C. Resistance of PhaseCamouflage to Existing Attacks

1) Threat Model: In the assumed threat model, attackers have access to the following information:

1) A functional IC that can be used to give correct input/output pairs as a black box
2) A camouflaged gate-level netlist, which can be obtained through reverse engineering
3) Means to view the layout-level characteristics of the netlist (most notably, the sizing of the cross-coupled pairs needed for the mismatch of 180° obfuscated gates)
4) Boolean functionalities that the camouflaged gates perform (the functional behavior of 0°, 90°, and 180° phase differences).

According to this threat model, attackers are assumed to have the typical capability for conventional circuit camouflaging scenarios (items 1, 2 and 4), and additionally are able to observe layout-level characteristics of the mismatch between the cross-coupled pair (item 3). This threat model is considered so that the attacker has additional reverse engineering capability to discern 180° obfuscated gates via layout analysis. To thwart removal/structural attacks of the 180° obfuscated gates, the countermeasure discussed in Section III-B-2 should be used.

2) Sensitization Attack: Existing techniques for selecting the camouflaging gates apply to PhaseCamouflage since these techniques are independent of the particular camouflaging methodology used. For example, the clique based selection process introduced in [1] can be used to choose the location of camouflaged gates in PhaseCamouflage. This technique makes

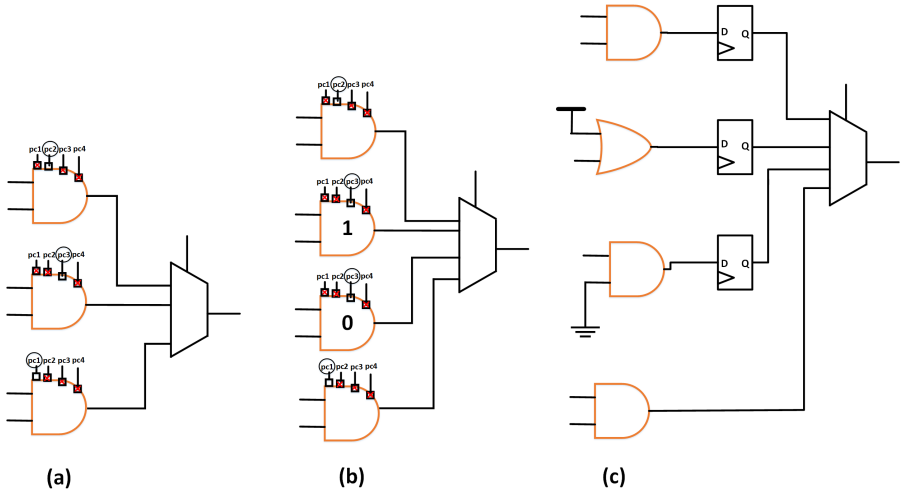


Fig. 15. Transformation unit for a potential SAT attack on PhaseCamouflage: (a) first step of transformation with the three possible phase difference options, (b) expanded transformation unit where 180° obfuscated gate outputs a constant logic-high or logic-low, and (c) final transformation unit that fully converts the dependence of phase difference to Boolean functionality by also considering the timing differences between 0° and other obfuscated gates.

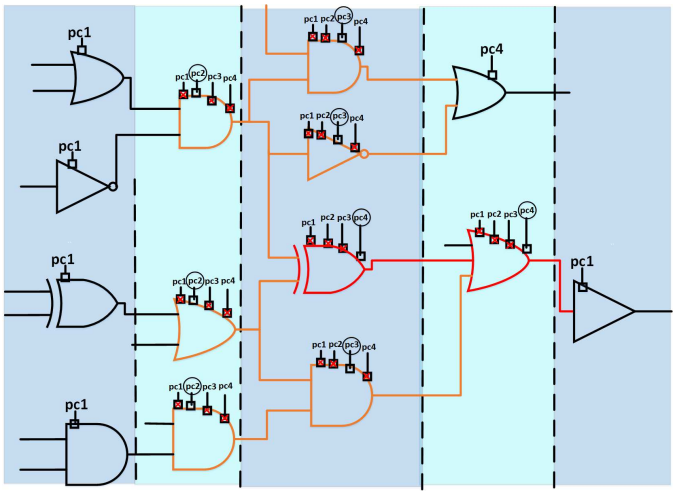


Fig. 16. Insertion of a camouflaged path using a 2-input 180° obfuscated gate.

it highly challenging to generate test vectors and determine the functionality of camouflaged gates by creating mutual interferences. Similar to existing techniques for dummy-based camouflaging, the proposed camouflaged (including obfuscated) gates should be inserted to ensure that they mutually interfere with each other and therefore result in non-resolvable gates. In the proposed methodology, mutual interference is realized via camouflaged paths that contain several interconnected camouflaged gates. This can be achieved by using multiple fan-in gates as obfuscated gates, as shown with the 180° obfuscated XOR gate in Fig. 16. Specifically, with a multiple fan-in gate, camouflaged paths are connected to other camouflaged paths. Thus, the number of gates that interferes with each other (non-resolvable gates) is increased, making sensitization attacks more difficult.

3) SAT Attack: To formulate an SAT attack, the netlist in question should be reconstructed to a conjunctive normal form (CNF)-encoded SAT problem [39]. SAT attacks work on existing camouflaging schemes by guessing possible Boolean functionality, which would not be sufficient for the proposed approach without significant effort. For example, 0° obfuscated gates in PhaseCamouflage affect the timing characteristics by evaluating logic one phase (a quarter clock cycle) sooner.

A specific SAT attack [40], TimingSAT, has been developed that works on both the camouflaged timing profile and camouflaged functionality provided by the TimingCamouflage scheme proposed in [25]. TimingSAT can detect wave-pipelined paths by inserting "transformation units" (TU) (which consist of a flip-flop and a MUX) into the camouflaged netlist. The select line of the MUX acts as a 1-bit key where one value represents a single-cycle path and the other value represents a wave-pipelined path. TUs essentially convert the locked timing profile to depend on Boolean logic functionality through these new key bits. After TU insertion, TimingSAT relies on unrolling these inserted flip-flops and converting the netlist into a combinational circuit so that an SAT attack can be performed.

Similar to TimingSAT [40], in order to perform an SAT attack on a camouflaged adiabatic netlist, the obfuscation should be transformed so that all of the dependencies are represented in Boolean form instead of phase differences. This is because a CNF representation of the circuit should be generated as an input to the SAT solver. In order to achieve this, the camouflaged ECRL design should be converted to static CMOS with a transformation unit (TU), as illustrated in Fig. 15. First, all possible phase differences are connected to a multiplexor, as shown in Fig. 15(a). However, 180° obfuscated gates can output a constant logic-low or logic-high, so the transformation unit is edited as in Fig. 15(b). Finally, Fig. 15(c) shows the complete transformation unit without phase difference dependency. The constant logic-low is represented with an AND gate that has an input connected to ground, and the constant logic-high is represented with an OR gate that has an input connected to VDD. Furthermore, flip-

flops are placed after the gates that represent 90° and 180° obfuscation to accurately reflect the timing characteristic of the 0° obfuscated gate (which achieves computation without consuming any clock phase, unlike other obfuscated gates). Thus, flip-flops ensure that the outputs along those paths are delayed one cycle with respect to the output of the 0° obfuscated gate.

However, some challenges exist when implementing the SAT attack on the proposed method with adiabatic gates. Even though non-camouflaged ECRL gates can be directly replaced with static CMOS versions, each camouflaged gate using the proposed method should be replaced with the transformation unit shown in Fig. 15(c), while maintaining correct functionality. It is difficult for an attacker to ensure the correct functionality and timing of the netlist with the additional flip-flops needed in the transformation unit. Attackers would also need to consider that in camouflaged ECRL, 180° and 90° obfuscated gates consume a quarter of a cycle (one phase) whereas in the transformed unit, they consume one cycle since a quarter cycle cannot be represented in static CMOS based circuits. In addition, current SAT solvers, such as MiniSAT [41], do not currently have the capability to represent the inputs tied to ground and VDD since the inputs are in the form of propositional logic. Thus, while it may be possible to successfully mount an SAT attack on PhaseCamouflage, attackers would need to spend more effort than the conventional Boolean-based camouflaging.

Recently a more powerful attack, the Satisfiability Modulo Theory (SMT) attack, was shown to break obfuscation schemes that are not based on Boolean logic, such as delay locking [26]. SMT attack is able to use theory solvers that allow the attacker to express constraints that cannot be represented with CNF, such as power and delay [42]. Potentially, an SMT attack can be formulated against PhaseCamouflage, but SMT attacks are typically more complex because constraint clauses that represent the non-Boolean functionality of the phase differences must be formulated for the chosen theory solver. The resistance against such attacks and the complexity of these attacks for PhaseCamouflage in adiabatic gates require additional investigation.

V. Conclusion

PhaseCamouflage is a logic obfuscation technique that prevents reverse engineers from extracting a functional netlist and protects against IP theft. The inherent phase difference between gates in adiabatic/charge-recycling logic is leveraged for lightweight obfuscation that is particularly useful for resource-constrained applications. Due to the exponential complexity of the methodology, strong obfuscation can be achieved with a relatively small number of camouflaged cells. Design guidelines are also developed to ensure that PhaseCamouflage is protected against modern removal attacks. The proposed methodology is demonstrated in two circuits with a security level equivalent to 32-bit key. The overhead in power consumption and area is analyzed. The corruptibility achieved with the proposed approach is also quantified. In future work, PhaseCamouflage can be combined with traditional circuit camouflaging methods to further enhance security. Specifically, the dependence of adiabatic logic on both phase and Boolean functionality can be leveraged to develop a more secure camouflaging approach. Theoretically, an attacker would have to guess the type of the Boolean gate and the phase difference (90°, 180°, 0°), thereby increasing the base complexity required to de-camouflage the circuit.

References

[1] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 709–720.

[2] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[3] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2011, pp. 333–338.

[4] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware trojans: extended version," *Journal of Cryptographic Engineering*, vol. 4, no. 1, pp. 19–31, 2014.

[5] N. E. C. Akkaya, B. Erbagci, and K. Mai, "A secure camouflaged logic family using post-manufacturing programming with a 3.6 ghz adder prototype in 65nm cmos at 1v nominal v dd," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 128–130.

[6] 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, 2018.

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

[8] Y. Moon and D.-K. Jeong, "An efficient charge recovery logic circuit," *IEICE transactions on electronics*, vol. 79, no. 7, pp. 925–933, 1996.

[9] S. M. Plaza and I. L. Markov, "Solving the third-shift problem in ic piracy with test-aware logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 961–971, 2015.

[10] T. Wan, Y. Karimi, M. Stanaćević, and E. Salman, "Ac computing methodology for rf-powered iot devices," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 5, pp. 1017–1028, 2019.

[11] S. D. Kumar, H. Thapliyal, A. Mohammad, and K. S. Perumalla, "Design exploration of a symmetric pass gate adiabatic logic for energy-efficient and secure hardware," *Integration*, vol. 58, pp. 369–377, 2017.

[12] S. D. Kumar, H. Thapliyal, and A. Mohammad, "Finsal: Finfet-based secure adiabatic logic for energy-efficient and dpa resistant iot devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 110–122, 2017.

[13] T. Wan, E. Salman, and M. Stanacevic, "A new circuit design framework for iot devices: Charge-recycling with wireless power harvesting," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 2046–2049.

[14] H.-V. Tran and G. Kaddoum, "Robust design of ac computing-enabled receiver architecture for swipt networks," *IEEE Wireless Communications Letters*, 2019.

[15] V.-D. Nguyen and O.-S. Shin, "An efficient design for noma-assisted miso-swipt systems with ac computing," *IEEE Access*, vol. 7, pp. 97 094–97 105, 2019.

[16] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*. ACM, 2008, pp. 1069–1074.

[17] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing ic piracy using reconfigurable logic barriers," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[18] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 83–89.

[19] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 9, pp. 1411–1424, 2015.

[20] 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)*. IEEE, 2015, pp. 137–143.

[21] M. El Massad, S. Garg, and M. V. Tripunitara, "Integrated circuit (ic) decamouflaging: Reverse engineering camouflaged ics within minutes." in *NDSS*, 2015, pp. 1–14.

[22] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Camoperturb: secure ic camouflaging for minterm protection," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2016, pp. 1–8.

[23] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, "Provably secure camouflaging strategy for ic protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2017.

[24] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, 2017.

[25] G. L. Zhang, B. Li, B. Yu, D. Z. Pan, and U. Schlichtmann, "Timingcamouflage: Improving circuit security against counterfeiting by unconventional timing," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 91–96.

[26] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against ic counterfeiting and overproduction," in *Proceedings of the 54th Annual Design Automation Conference 2017*. ACM, 2017, p. 9.

[27] L. W. Chow, J. P. Baukus, B. J. Wang, and R. P. Cocchi, "Camouflaging a standard cell based integrated circuit," Apr. 3 2012, uS Patent 8,151,235.

[28] C. Yan, J. Dofe, S. Kontak, Q. Yu, and E. Salman, "Hardware-efficient logic camouflaging for monolithic 3-d ics," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 6, pp. 799–803, 2017.

[29] L.-W. Chow, J. P. Baukus, and W. M. Clark Jr, "Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide," Nov. 13 2007, uS Patent 7,294,935.

[30] "Cadence virtuoso layout suite ," https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/layout-design/virtuoso-layout-suite.html, accessed: 2021-01-09.

[31] "Cadence spectre simulation platform," https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-simulation-platform.html, accessed: 2021-01-09.

[32] T. Wan and E. Salman, "Ultra low power simon core for lightweight encryption," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5.

[33] M. Yasin, O. Sinanoglu, and J. Rajendran, "Testing the trustworthiness of ic testing: An oracle-less attack on ic camouflaging," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2668–2682, 2017.

[34] N. Rangarajan, S. Patnaik, J. Knechtel, R. Karri, O. Sinanoglu, and S. Rakheja, "Opening the doors to dynamic camouflaging: Harnessing the power of polymorphic devices," *arXiv preprint arXiv:1811.06012*, 2018.

[35] M. El Massad, S. Garg, and M. V. Tripunitara, "The sat attack on ic camouflaging: Impact and potential countermeasures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 8, pp. 1577–1590, 2019.

[36] V. C. Patil and S. Kundu, "On leveraging multi-threshold finfets for design obfuscation," in *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2020, pp. 108–113.

[37] "Iscas benchmark circuits," http://web.eecs.umich.edu/ jhayes /iscas.restore/benchmark.html, accessed: 2021-01-09.

[38] K. Juretus and I. Savidis, "Characterization of in-cone logic locking resiliency against the sat attack," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 8, pp. 1607–1620, 2019.

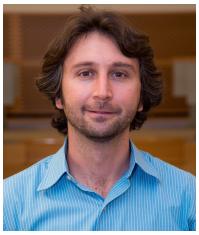
[39] C. Yu, X. Zhang, D. Liu, M. Ciesielski, and D. Holcomb, "Incremental sat-based reverse engineering of camouflaged logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 10, pp. 1647–1659, 2017.

[40] M. Li, K. Shamsi, Y. Jin, and D. Z. Pan, "Timingsat: Decamouflaging timing-based logic obfuscation," in *2018 IEEE International Test Conference (ITC)*. IEEE, 2018, pp. 1–10.

[41] N. Sorensson and N. Een, "Minisat v1. 13-a sat solver with conflict-clause minimization," *SAT*, vol. 2005, no. 53, pp. 1–2, 2005.

[42] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "Smt attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the sat attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 97–122, 2019.

Ivan Miketic Ivan received the B.Sc. degree in physics from Adelphi University, Garden City, NY, USA in 2017. He is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering at Stony Brook University, Stony Brook, NY, USA. His research interests are in hardware security specifically emerging applications and devices.



Emre Salman (S'03-M'10-SM'17) received the B.S. degree in microelectronics engineering from Sabanci University, Istanbul, Turkey, in 2004, and the M.S. and Ph.D. degrees in electrical engineering from the University of Rochester, NY, USA, in 2006 and 2009, respectively.

He was previously with STMicroelectronics, Synopsys, and Freescale Semicondutor (now NXP Semiconductors), where he was involved in research in the fields of custom circuit design, timing, and noise analysis. Since 2010, he has been with the Department of Electrical and Computer Engineering, Stony Brook University (SUNY), NY, USA, where he is currently an Associate Professor and the Director of the Nanoscale Circuits and Systems Laboratory. He is the leading author of a comprehensive tutorial book *High Performance Integrated Circuit Design* (McGraw-Hill, 2012, Chinese translation, 2015). His broad research interests include analysis, modeling, and design methodologies for integrated circuits and VLSI systems with applications to low power and secure computing, Internet of things with energy harvesting, and implantable devices.

Dr. Salman was a recipient of the National Science Foundation Faculty Early Career Development Award in 2013, the Outstanding Young Engineer Award from IEEE Long Island, NY, USA, in 2014, and the Technological Innovation Award from IEEE Region 1 in 2018. He also received multiple outreach initiative awards from the IEEE Circuits and Systems Society. He served on the Editorial Board of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS. He currently serves as the Americas Regional Editor for the Journal of Circuits, Systems and Computers, on the Editorial Board of IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, on the organizational/technical committees of various IEEE and ACM conferences, and as the Chair for the VLSI Systems and Applications Technical Committee (VSA-TC) of the IEEE Circuits and Systems Society.