http://dx.doi.org/10.1109/ICCAD.2017.8203758

Obfuscating the Interconnects: Low-Cost and Resilient Full-Chip Layout Camouflaging

Satwik Patnaik[†], Mohammed Ashraf[‡], Johann Knechtel[‡], and Ozgur Sinanoglu[‡]
[†] Tandon School of Engineering, New York University, New York, USA
[‡] New York University Abu Dhabi, Abu Dhabi, United Arab Emirates
{sp4012, ma199, johann, ozgursin}@nyu.edu

***Abstract*—Layout camouflaging (LC) is a promising technique to protect chip design intellectual property (IP) from reverse engineers. Most prior art, however, cannot leverage the full potential of LC due to excessive overheads and/or their limited scope on an FEOL-centric and accordingly customized manufacturing process. If at all, most existing techniques can be reasonably applied only to selected parts of a chip—we argue that such "small-scale or custom camouflaging" will eventually be circumvented, irrespective of the underlying technique.**

In this work, we propose a novel LC scheme which is low-cost and generic—full-chip LC can finally be realized without any reservation. Our scheme is based on obfuscating the interconnects (BEOL); it can be readily applied to any design without modifications in the device layer (FEOL). Applied with split manufacturing in conjunction, our approach is the first in the literature to cope with both the FEOL fab and the end-user being untrustworthy. We implement and evaluate our primitives at the (DRC-clean) layout level; our scheme incurs significantly lower cost than most of the previous works. When comparing fully camouflaged to original layouts (i.e., for 100% LC), we observe on average power, performance, and area overheads of 12%, 30%, and 48%, respectively.

Here we also show empirically that most existing LC techniques (as well as ours) can only provide proper resilience against powerful SAT attacks once at least 50% of the layout is camouflaged—only large-scale LC is practically secure. As indicated, our approach can deliver even 100% LC at acceptable cost. Finally, we also make our flow publicly available, enabling the community to protect their sensitive designs.

I. Introduction

Ensuring the security and trustworthiness of hardware has become a major concern in recent years [1]–[3]. One reason for protecting the hardware is that intellectual property (IP) can otherwise be duplicated without consent, resulting in a financial loss for the IP owner. Furthermore, understanding the gate-level implementation of a chip may advance other attacks such as side-channel analysis or Trojan insertion [2], [3]. A malicious end-user, i.e., an adversary without direct access to the design and fabrication process, has to resort to *reverse engineering (RE)* of chips to obtain the IP. The tools and know-how for RE attacks are becoming more advanced and widely available, thus rendering RE a practical threat [2]–[6].

The goal of *layout camouflaging (LC)* is to mitigate RE attacks.[1] The fundamental idea is to alter the behavior or appearance of a chip such that it is arduous or even impossible for the RE attacker to infer the chip's real functionality. This can be achieved, e.g., by "look-alike" or ambiguous gates [8], [9], by secretly configured multiplexers (MUXes) [10], [11], or by threshold-dependent camouflaging of gates [12]–[14]. We discuss the prior art further in Sec. II; besides, a comprehensive overview on LC is given in [15].

On the cost of prior art: Most existing LC schemes have a high layout cost and are accordingly limited for practical use. For example, the ambiguous XOR-NAND-NOR gate proposed in [9] has 5.5× power, 1.6× delay, and 4× area cost in comparison to a conventional NAND gate. Even promising works such as the threshold-dependent, full-chip LC proposed in [14] induces overheads of 14%, 82%, and 150% in power, performance, and area (PPA), respectively. Besides, FEOL-based LC techniques are typically only available as customized IP, and/or require some alterations for the FEOL manufacturing process, incurring financial cost on top of PPA overheads.

[1] *Hardware/layout obfuscation* are wide-spread synonyms for LC. We use the term *obfuscation* purposefully in the context of *obfuscating the interconnects*, which is the key principle of our work. Besides, LC is closely related to the concept of *logic locking* [7], which itself is also known as *logic encryption*.

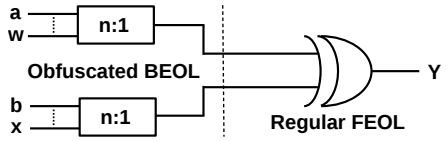


Fig. 1. Our camouflaging concept is based on secret n:1 mappings in the BEOL, which obfuscate the inputs for any regular gate (not only two-input XOR). The set of possible functionalities depends on n:1, the selection of the input nets, and the gate type.

In practice, existing LC schemes can be applied only selectively—if at all—due to their inherent PPA overheads and their impact on the FEOL processing. As a result, the constrained application of these techniques may lead to a compromise in their security.

On the resilience of prior art: When LC techniques are applicable only to parts of a chip, the challenge is where and to what extent camouflaging shall be effected. Ideally, an attacker's effort to obtain the design from a carefully camouflaged netlist would be exponential in the number of camouflaged gates [16]. Advances on the *Boolean satisfiability problem (SAT)*, however, have enabled powerful attacks on LC (and on logic locking) [16]–[18]. The previously unforeseen success of such SAT attacks stems from the typically small number of input/output (I/O) patterns required in practice for de-camouflaging. Recent works [19]–[21] aim for exponentially scaling and *provably secure LC* but are still prone to other advanced attacks (Sec. II).

Accounting for the recent advances of analytical and invasive attacks (Sec. II), we make the following case: to remain resilient, at least as long as foreseeable, LC has to be applied at a large scale, i.e., more than 50% of the layout should be camouflaged.

On this paper: Here we promote full-chip LC with high resilience and low cost. To do so, we propose and evaluate simple but effective LC primitives which are based on obfuscating (parts of) the interconnects. Our contributions can be summarized as follows:

1) We enable resilient and full-chip LC, based on a novel, security- and cost-driven approach for *obfuscating the interconnects*.
2) Our novel BEOL-centric LC primitives are tailored for regular gates (Fig. 1). That is, our primitives do not require any modification at the FEOL device layer and can, therefore, be easily integrated into any (industrial) design flow. We make our flow (based on *Cadence Innovus*) publicly available in [22].
3) The fact that our primitives implement BEOL-centric LC suggests the (optional) application of *split manufacturing* [1], [3] in conjunction. Doing so allows us for the first time to hinder fab adversaries in addition to malicious end-users, all while imposing only low commercial cost, which may be even compensated for. Note that the implementation effort and cost of

http://dx.doi.org/10.1109/ICCAD.2017.8203758

protecting against a malicious fab *and* malicious end-users have been considered as mutually exclusive so far.

4) We assess the resilience of different flavors of our proposed primitives and compare them against previous works on LC. In that process, we employ powerful SAT attacks on traditional benchmarks (which are relatively small) while we also show—for the first time—attacks on large VLSI benchmarks. Besides, we introduce the notion of *practically secure LC*, which seeks to impose an excessive computational cost on SAT-based attacks without inserting additional, dedicated circuit structures.
5) We conduct a thorough evaluation of camouflaged, DRC-clean layouts. In contrast, most of the previous works investigate their LC primitives only as stand-alone devices, without applying them in actual layouts; we argue that this is overly optimistic. Our work is one of the very few in the literature providing comprehensive layout-level evaluation in general, and the first to do so for large VLSI benchmarks with up to 39,014 gates.

II. Background

Next, we discuss the recent progress on LC (along with demonstrated and potential attacks), which is typically focused on FEOL-centric camouflaging. Further, we discuss an early study on BEOL-centric camouflaging, which also inspired our work to some degree.

A. Camouflaging at the FEOL and Vulnerabilities

As already mentioned, powerful SAT attacks have challenged most prior art on LC (and logic locking) [16]–[18]. Thus, several recent works on provably secure LC (and logic locking) [19]–[21] seek to mitigate SAT attacks by inserting dedicated (but high-cost) structures which in theory necessitate to consider an exponential number of I/O patterns. However, we argue that the tailored structures of [19]–[21] can be easy to identify during RE; these structures (i) are typically applied only in a few places, due to their relatively high cost, and (ii) rely on arrays of possibly camouflaged gates with their outputs converging in large combinatorial trees. Moreover, the output wires of these trees have been successfully identified by signal probability analysis [23]. These critical wires may then be cut to circumvent the security features.[2] Besides, advanced SAT attacks have also been demonstrated to mitigate provably secure or "cyclic" logic locking techniques very recently [23], [25]–[28]. In short, a sophisticated attacker may learn which gates to ignore, replace or cut off while (nearly) recovering the original netlist of such SAT-hardened chips.

RE measures may render FEOL-centric LC also directly void, without the assistance of analytical techniques. LC schemes such as "look-alike" and ambiguous gates [8], [9] or secretly configured MUXes [10], [11] rely on dummy contacts or dummy channels. While it is often claimed by the authors of these studies that simple etching cannot reveal these features, other powerful techniques/tools are available. Specifically, the use of *scanning electron microscopy* in the *passive voltage contrast* mode (*SEM PVC* in short) allows for accurate and efficient measurement of charge accumulations; this has been recently demonstrated by Courbon *et al.* [6] in an unprecedented case study for reading out secured Flash memories. Now, SEM PVC may break the above LC techniques as well, since dummy contacts/channels will accumulate charges to a much lower degree than real contacts/channels. Threshold-dependent camouflaging of gates [12]–[14] can also be revealed by SEM PVC, as successfully demonstrated by Sugawara *et al.* [5]. Besides, as the authors in [12] indicate themselves, monitoring the etch rates can reveal different doping levels which are at the heart of threshold-dependent gates.

[2]Wire cutting has been successfully demonstrated in the past, for example by Helfmeier *et al.* [24], enabling such invasive attacks in principal.

B. Towards Camouflaging at the BEOL

Chen *et al.* [29] suggest implementing vias either as real, conductive vias (using magnesium, Mg) or as dummy, non-conductive vias (using magnesium oxide, MgO). That is, the authors advocate BEOL-centric camouflaging besides the "classical LC" at the FEOL. Despite their pioneering work on RE-resilient vias, Chen *et al.* did not succeed to propose a resilient LC application, as we show in Sec. VI. Also, note that our concept is different from [29]; we only leverage their notion of Mg/MgO vias for obfuscation.

Chen *et al.* [29] elaborated that the use of Mg/MgO is practical from both the perspectives of (i) manufacturability and (ii) RE mitigation. For (i), it is noted that Mg has been traditionally used to facilitate the bonding of copper interconnects to the dielectric layer. For (ii), the authors fabricated samples and observed that Mg was completely oxidized (into MgO) within a few minutes. That is, the real Mg vias became indistinguishable from the dummy MgO vias during RE. Independently, Swerts *et al.* [30] and Hwang *et al.* [31] have used Mg and MgO during CMOS-centric BEOL processing (without LC in mind). Hwang *et al.* have shown that Mg not only oxidizes but also dissolves quickly—as does MgO—when surrounded by fluids, which is inevitable in classical RE etching procedures.

Although one can argue that RE of Mg/MgO vias is somehow possible nevertheless, such an attack is yet to be demonstrated.[3] In any case, our work relies only on the generic concept of RE-resilient interconnects, and not on particular materials currently available for high-volume manufacturing. Future interconnects, e.g., based on carbon/graphene or spintronics [32], [33], may hinder RE as well.

It is important to note that RE-resilient interconnects are implemented not only at the designer's choice but also at the manufacturer's discretion. That is, this concept requires either split manufacturing [1], [3] or a trusted BEOL process in conjunction.

Finally, another aspect of this concept is that its commercial cost may be easily compensated for, even when split manufacturing is applied. That is because it only requires additional BEOL masks which are, e.g., for M5/M6 3.5–4× cheaper than the Poly masks at 16nm, according to industry experts. In contrast, most prior FEOL-centric LC (but not threshold-dependent LC) incur a relatively high cost for the different FEOL masks required. In general, any FEOL-centric approach demands some alterations of IP libraries and/or for the manufacturing process—such alterations are likely more costly than camouflaging at the BEOL.

III. Our Concept and Threat Model

Our key idea is the following: we leverage the concept of RE-resilient interconnects to design novel, BEOL-centric LC primitives which are applicable to any type of regular gates.

Our concept: We implement n wires for each input of the gates to be camouflaged, and each wire has—without loss of generality—some of its vias obfuscated. In other words, we obfuscate the real driving wires of any gate via *secret n:1 mappings in the BEOL* (Figs. 1–3). As a result, the actual function which a gate implements is obfuscated in a simple yet effective manner; the set of possible functionalities depends on the selection of the wires, on n, and on the gate itself. Note that our concept is generic and directly applicable for any multi-input gates, and not only for two-input gates.

Since our concept employs obfuscation in the BEOL layers, it can be readily applied in conjunction with split manufacturing, even with

[3]For example, we do *not* expect the powerful SEM PVC attack [6] to be successful here. Once an attacker seeks to measure charges in the individual BEOL layers, she/he will inevitably rip up all the interconnects during the layer-wise RE process. Hence, a localized lack of charges will hint on any non-functional wire, be it an obfuscated dummy or a ripped-up regular wire.

low commercial cost on top (i.e., at least when splitting at higher metal layers—which we do by splitting at M5, see also Sec. VI).

Our threat model: We assume both the end-user and the fab to be untrusted. The latter is in strong contrast to most prior art on LC that *has to* trust the fab because of their FEOL-centric techniques.

To hinder fab adversaries, i.e., in particular to protect the secret mappings in the BEOL layers, we leverage split manufacturing.[4] The goal of malicious end-users is to RE the chip's gate-level layout and identify its secret mappings in the BEOL layers—the latter is challenging and yet to be demonstrated (Sec. II). Ultimately, both adversaries want to reconstruct the original netlist and its IP. Towards this end, end-users can use another working chip copy as an oracle for SAT attacks, whereas fab workers can launch *proximity attacks* [34].

IV. On Different Flavors of Our Primitive

Here, we shed light on several flavors of our LC scheme. We assess all flavors by (i) their impact on PPA and (ii) their SAT attack resilience. To do so, we (i) conduct a thorough GDSII-level analysis and (ii) employ powerful SAT attacks as proposed in [17]. For the latter, the authors made their attack tool publicly available [35]; other recent attacks proposed in, e.g., [18], [25]–[28] have not been made available to us. We camouflage the layouts in the range of 10% to 100%, in steps of 10%. Further setup details are given in Sec. VI.

Without loss of generality, we exemplarily discuss our primitives when applied for two-input gates in the following.

A. Basic Flavor, with Simple n:1 Mappings

2:1 mapping: We first explore the most basic flavor with two wires for each gate's input, i.e., one dummy wire and one real wire.

This flavor comprises only four functionalities. For example, with a, w being the wires for one input, and b, x being the wires for the other input (see also Fig. 1), the gate can implement either $f(a, b)$, $f(a, x)$, $f(w, b)$, or $f(w, x)$, where f is the functionality of the gate. It is easy to see that even less than four functions are realized when some wires are driven by the same net. We avoid this—for all flavors—by selecting unique nets for each gate's wires (see Sec. V).

We expect and observe this flavor to break relatively easily against the attacks we leverage from [17]. For benchmarks *apex4* and *des*, e.g., the attack terminates within 100–300 seconds, even for large-scale LC (i.e., for 50%). Compared to our other flavors, this one has the weakest resilience to SAT attacks. Hence, we did not consider the further application or layout-level evaluation of this basic primitive.

3:1 mapping: We extend the basic scheme by adding one more dummy wire for each input (Fig. 2). This primitive cloaks one out of nine possible functionalities; hence, we can expect a higher resilience when compared to the 2:1 mapping primitive. In fact, we observe that the resilience scales well across all ranges of LC for this 3:1 primitive and, thus, is enforcing reasonably high efforts for SAT attacks, especially for larger benchmarks such as *b15*. On average, the attack runtime scales by ≈3× when compared to the 2:1 primitive.

On the flip side, we observe relatively high PPA cost for this primitive, especially for performance/delay; on average, the delay overhead approaches 60% when "only" 50% of the layout is camouflaged. That is because we need to choose all the wires such that they are unique for any gate (as already indicated) and, as a result, the more the camouflaging, the higher the routing congestion. In turn, this increases wirelength and capacitive loads, which simultaneously aggravates delay and power, imposing practical limitations for large-scale LC using this basic 3:1 mapping primitive.

[4]In a weaker threat model where the fab is trusted, our technique can still be directly applied, just without split manufacturing.

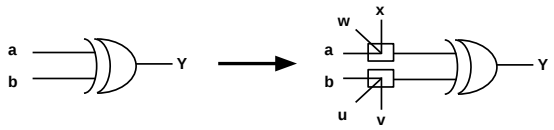


Fig. 2. Our basic primitive with secret 3:1 mappings, resulting in up to 9 possible functions for the gate.

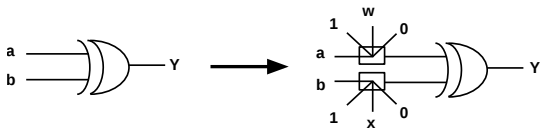


Fig. 3. Our extended, final primitive, where the fixed logic values of *0* and *1* are employed along with two regular signal nets/wires. Depending on the gate type, 10 or 14 functions are possible.

B. Extended Flavor, with Fixed Logic Values

A promising option is to additionally employ the fixed logic values *0* and *1* along with regular nets/wires (Fig. 3). According to Boolean algebra and considering the underlying gate, this extended primitive provides either $5m$ or $7m$ functionalities, where m is the number of regular wires for each input (in addition to the two wires with the fixed values *0* and *1*). For example for $m = 2$, with a, w and b, x as the respective regular wires for the two inputs, and with XOR as the underlying gate (Fig. 3), we obtain the following 14 functionalities: $0, 1, a, w, b, x, \overline{a}, \overline{w}, \overline{b}, \overline{x}$, $a \oplus b$, $a \oplus x$, $w \oplus b$, and $w \oplus x$.

We note that this approach enables significantly lower PPA cost. That is, for $m = 1$, the average delay overhead is only ≈10% when 50% of the layout is camouflaged, and for $m = 2$, the average delay overhead is still only ≈20–30%, even when 50–100% of the layout is camouflaged. When compared to employing only regular nets/wires, the additional use of *0* and *1* offers a fundamental benefit: their fixed-value wires are not switching, thus exhibiting only negligible power consumption and imposing no timing overhead.

We expect the SAT attack resilience of this extended primitive to be "in between" the two basic primitives, i.e., at least for $m = 1$. Our experiments corroborate this expectation; for $m = 1$, this primitive offers on average a lower resilience than 3:1 mapping (effecting 25–30% less attack runtime), but a higher resilience than 2:1 mapping (≈35% more runtime) when the same set of gates are camouflaged.

To further strengthen the resilience, we may add more regular wires (i.e., increase m), thereby extending the set of possible functionalities. This will directly impact the overall search space and, as a result, increase the average effort required for SAT attacks [19]. Adding more wires (to be driven by unique nets), however, notably contributes to routing congestion which aggravates PPA cost in turn. In short, we empirically choose $m = 2$ for our final primitive (Fig. 3). We elaborate on this final primitive in Sec. VI in detail.

V. Our Camouflaging Methodology

A. Protecting Fixed Values and "Implausible Functions"

As indicated above, our final primitive appears attractive due to its relatively low PPA cost and its adequate resilience (see also Sec. VI).

For large-scale LC, however, the ubiquitous wires relating to fixed values may give away clues to an attacker; fixed values are typically used only for special registers or "hardware mode flags." An attacker observing a vast number of fixed-value wires—which itself may be easy, given that such wires are typically connected to distinct *TIE cells*—might rightfully assume that these wires have been introduced for obfuscation. Thus, unless we make these wires essential for large parts of the design, an attacker may readily and safely disregard them.

We have to address another, complementary challenge at once. That is, a mindful attacker may also try to rule out all the "implausible"

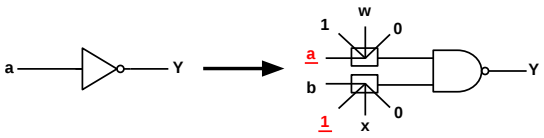


Fig. 4. An inverter transformed into a two-input NAND gate. The real nets/wires are underlined and shown in red.

functions (i.e., INV, BUF, *0* and *1*) which are those beyond any gate's original functionality. Since these additional functions arise only due to the fixed values being part of the obfuscated inputs to begin with, they can only become effective once the fixed values are an essential part of the design (and vice versa).

In short, the fixed-value wires have to be rendered essential, while also protecting all the "implausible functions" at the same time.[5] To do so, we perform simple *netlist transformations* as follows:

1) We transform some inverters (INVs) and buffers (BUFs) into gates of other types (e.g., see Fig. 4). Nowadays, around 50% of all gates are repeaters (INV/BUF) [37], [38], offering ample opportunities for large-scale transformations/camouflaging. Here one can freely choose the number of INVs/BUFs to camouflage, and the type of gate (AND, NAND, OR, NOR, XOR, XNOR) to transform them into. The best strategy, which we also apply, is to randomly transform 50% of INVs/BUFs. This way, an attacker cannot easily infer a direct correspondence between any of those gates and their functionality. Note how this transformation renders the fixed values essential—they cannot be ignored without misinterpreting the transformed gates. Also, the functionalities INV and BUF remain plausible now throughout the layout.
2) We insert some additional gates (into randomly selected regions of whitespace) with their real inputs tied to fixed values. These gates act as TIE cells in disguise; they "drive" other camouflaged gates in turn. Thus, also the functionalities *0* and *1* cannot be ignored anymore without misinterpreting the transformed netlist.

B. Overall Flow

Here we provide an overview of our camouflaging methodology (Fig. 5), which can be easily integrated into any design flow. In this work, we implement our methodology for *Cadence Innovus*. We also provide open access to our flow in [22].

Given an HDL netlist, we initially synthesize, place, and route the design. On this original layout, we then apply our transformations outlined above. Next, we insert and wire *obfuscating cells* for all the inputs of each gate to be camouflaged. It is essential to understand that these custom cells do not impact the FEOL layer—their sole purpose is to enable the routing of all dummy and real wires (Fig. 6). Hence, the physical design of these custom cells is tailored for routability, while their arrangement remains flexible and unconfined regarding the already placed standard cells (see also Sec. V-C).

For our final primitive, recall that there are four obfuscated wires for each input (i.e., for each obfuscating cell, Fig. 6): two wires connect to *1* and *0* (either randomly with regular TIE cells or with gates acting as TIE cells in disguise), one wire connects with the real net, and one wire with a *dummy net*. In case *1* or *0* is the desired input—i.e., when the gate shall implement INV, BUF, or a TIE cell in disguise—the real net is either replaced with another unique dummy net or "driven" by another of the disguised TIE cells.

As indicated in Sec. IV, we have to choose dummy nets carefully such that they are unique with respect to each gate to camouflage.

[5]Independent from our work, Keshavarz *et al.* [36] recently called for maintaining the plausibility of all (chip-level) viable functionalities, albeit with a focus on logic synthesis and technology mapping, and without assuming that a working chip is available as an oracle for the (SAT-centric) attacks.

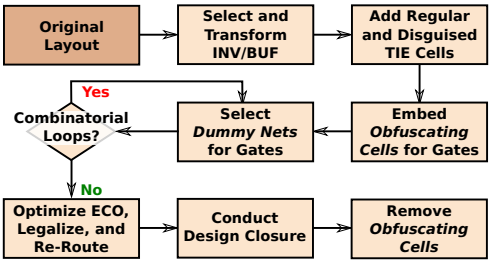
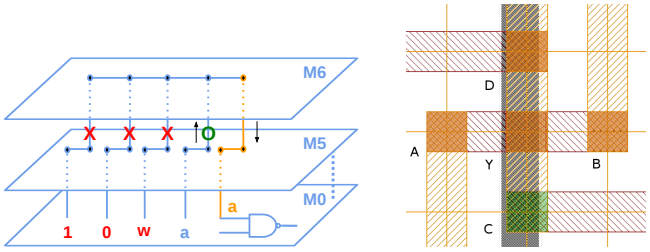


Fig. 5. Flow of our layout-level, BEOL-centric camouflaging methodology.



(a) (b)

Fig. 6. Wiring and vias for our obfuscating cell; the concept for the BEOL is illustrated in (a) and the physical-design view in (b). Note that the actual wiring in M5/M6 depends on which vias are dummy and which are real. In (a), the dummy/real vias are indicated as red crosses/green circle. In this example, the real net is labeled a, and it connects to pin C in (b); the camouflaged gate's input in (a) is wired with pin Y in (b). As for (b), the pins A, B, and Y reside in M6 (orange, vertical wiring), whereas pins C and D are set up in M5 (red, horizontal wiring). The vias (all between M5 and M6) are represented by the pins and either colored in orange (dummy vias) or in green (real via). Also note the cell's grey outline beneath the pins; the latter are partially located outside the cell, which is feasible/supported. The minimal width of the cell is solely to ease its visual differentiation from standard cells at design time.

We do so by applying a local spatial search around each gate's inputs; nearby nets/wires are preferably selected to limit the routing congestion. We also check for combinatorial loops which may have resulted during that process, and re-select dummy nets as required.

After embedding and connecting the obfuscating cells, we perform an *ECO optimization* and legalization; the latter is also based on custom constraint rules (Sec. V-C). At the same time, we re-route the design—now with all the dummy wires along with the real wires. We perform final design closure, remove the obfuscating cells from the design, re-extract the *RC* data, and finally gather PPA numbers.

C. Physical Design of the Obfuscating Cell

We implemented the obfuscating cell as a custom cell and extended the LIB/LEF files. Here we elaborate on the cell's physical design.

1) The cell has four input pins and one output pin (Fig. 6(b)). The pins have been set up in two metal layers: pins A, B, and Y reside in M6, whereas pins C and D are set up in M5. We have chosen two different layers to minimize the routing congestion; in exploratory experiments with all pins in M6, we observed overly high congestion, especially for LC beyond 50%. Note that one can easily tailor the pins for different layers as well (e.g., M7/M8), if considered useful for particular designs.
2) The dimensions of the pins ($0.14 \times 0.14 \mu m$) and their offsets are chosen such that the pins can be placed directly on the respective metal layer's tracks (thin yellow grid in Fig. 6(b)); this is to further minimize the routing congestion.
3) We define custom constraint rules which prevent the pins of different obfuscating cells to overlap during legalization. These obfuscating cells can, however, freely overlap with any standard cell without inducing routing conflicts. That is because standard cells have their pins exclusively in the lower metal layers.

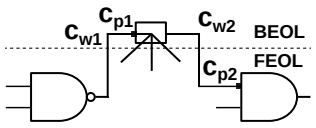


Fig. 7. The capacitance c_{p1} for any input pin of the obfuscating cell is to be annotated, to account for both the wire capacitance c_{w2} and the camouflaged gate's input-pin capacitance c_{p2}. Otherwise, the respective driver's load would be underestimated during ECO optimization.

4) We leverage the timing and power characteristics of BUFX2, i.e., a buffer with driving strength of 2. Note that a detailed library characterization is not required as the obfuscating cell only implements BEOL wires and vias. However, we have to set up an annotation regarding the pin capacitances (Fig. 7). This is essential to enable proper ECO optimization and to evaluate the final PPA numbers.

VI. Experimental Investigation

Setup for layout evaluation: We implement our methodology as custom scripts for *Cadence Innovus 15.1*; all our procedures incur negligible runtime cost. We employ the public *NanGate 45nm Open Cell Library* [39] with ten metal layers. The PPA analysis is carried out for 0.95V, 125°C, and the slow process corner, along with a default input switching activity of 0.2—note that this is a rather conservative setup. Power and timing results are obtained by *Innovus* as well. We configure the initial utilization rates (i.e., for the original layouts) such that the routing congestion remains below 1%.

Recall that we seek to safeguard our camouflaged layouts also from fab adversaries and that we employ split manufacturing towards this end (Sec. III). We suggest splitting at M5, i.e., just beneath the Mg/MgO vias which represent our secret assets to be protected.[6]

Setup for security evaluation: We implement all LC techniques proposed in [9]–[11], [41] for the sake of comparison with our work. For a fair evaluation, the same sets of gates are camouflaged across all LC techniques: for a given benchmark, gates are randomly selected once and then memorized.[7] Ten different sets are generated for each benchmark, ranging from 10% to 100% LC, in steps of 10%.

Recall that we evaluate the LC primitives against powerful SAT attacks [17] which are publicly available [35]. Note that the tool [35] was developed for logic locking but is still applicable for our study; logic locking and LC are closely related and can be transformed into one another [42]. All the SAT attacks are executed on a server with five compute nodes, where each node has two 14-core Intel Broadwell processors, running at 2.4 GHz with 128 GB RAM. The CPU time-out ("t-o") is set to 48 hours.

We attribute both the runtime and the *growth trend for clauses* as primary indicators for a design's resilience. The latter is helpful for large-scale LC when monitoring the attack runtimes becomes prohibitive. Specifically, the trend of clauses indicates whether the SAT solver can simplify the structures in the camouflaged design at all [42]. In case the number of clauses is continuously (and linearly) increasing, i.e., the saturation of clauses is not reached, we can conclude that the attack will not finalize in foreseeable time.

[6]Xiao *et al.* [40] noted that splitting at higher metal layers imposes relatively low efforts; higher layers have rather large pitches, which are easy and cheap to manufacture by the trusted (low-end) BEOL facility. Besides cost and practicability, however, we acknowledge that Wang *et al.* [34] argued that splitting at M5 may not be secure, based on their advanced *proximity attack*. Here it is important to note that our approach allows splitting beneath M5 without any restriction. Moreover, we observe in exploratory experiments that our LC scheme inherently helps mitigating such proximity attacks. More specifically, after fully camouflaging 14 selected designs, splitting their layouts at M5, and running the attack of [34] against the FEOL layouts, we observe *correct connections* of only 22.5% on average. We believe that is because all camouflaged gates are routed through M6 and above. Especially for large-scale LC, thus, our scheme induces a plethora of open nets (i.e., nets that are cut across FEOL and BEOL) which is challenging for any such proximity attack. A more detailed study will be the scope of our future work.

[7]Besides random selection, any other technique such as *maximum clique* [9] can be applied as well. Somewhat surprisingly, Massad *et al.* [16] observe that random selection is on average almost as effective.

TABLE I
Characteristics of Selected Benchmarks (Those in Italics are from the EPFL Suite [43], Others are from Traditional Suites)

Benchmark	Inputs	Outputs	Gate Count
aes_core	789	668	39,014
b14	277	299	11,028
b15	485	519	10,354
b17	1,452	1,512	36,770
b22	767	757	33,110
c7552	207	108	4,045
des	256	245	6,473
diffeq1	354	289	30,584
square	64	127	28,148

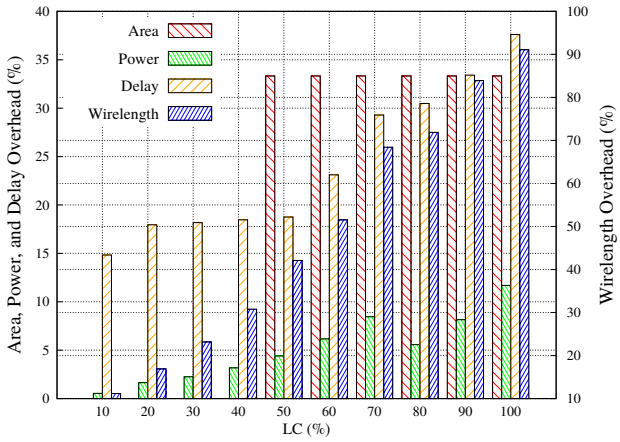


Fig. 8. Layout cost for *aes_core* [43] using our final LC primitive, with the baseline being the original layout. The discrete and monotonous area cost is due to a step-wise up-scaling of die outlines as needed; see also below.

Benchmarks: We conduct extensive experiments on traditional benchmarks suites (*ISCAS-85*, *MCNC*, and *ITC-99*) and, for the first time, also on the large-scale *EPFL MIG suite* [43]. For the latter, we compile the original circuits, not the MIG versions. Selected benchmarks are reviewed in Table I; we consider 34 benchmarks in total, however. Note that all benchmarks are combinatorial, but our approach can be directly applied to sequential designs as well.

A. Layout Evaluation

Here we report on general trends for PPA cost regarding our final LC primitive, and we also compare to previous works. Figure 8 illustrates the cost for benchmark *aes_core* [43], and Table II reports the cost for this and other benchmarks used in this work.

On die area: Recall that our obfuscating cells do not tangent the standard-cell area; the reported cost is thus concerning the *die outline*. Especially for large-scale camouflaging, we have to scale up the die outlines to mitigate routing/DRC errors. For the sake of simplicity, we scale up the outlines in view of decreasing the utilization rate in steps of 0.1. For example, while relaxing the utilization from 0.5 to 0.4, the die area has to be increased by 25%. As this technique can be rather profuse, we like to note that a more conservative up-scaling may enable less area overhead.

The overheads are typically not more than 25% for up to 60% LC, whereas for 100% LC, we note an average area cost close to 50%.

TABLE II
OUR GDSII-LEVEL COST IN % FOR LARGE-SCALE LC (IN % OF ALL GATES) ON SELECTED BENCHMARKS

Benchmark	Utilization for Original Layout	20% LC			40% LC			60% LC			80% LC			100% LC		
		Area	Power	Delay	Area	Power	Delay	Area	Power	Delay	Area	Power	Delay	Area	Power	Delay
aes_core	0.4	0	1.6	17.9	0	3.1	18.4	33.3	5.5	23.1	33.3	6.1	30.4	33.3	11.6	37.6
b14	0.5	0	2.3	8.1	0	5.3	9.2	0	11.4	17.9	25.0	14.6	21.1	25.0	15.1	22.1
b15	0.5	0	2.3	13.8	0	6.1	21.8	25.0	7.2	25.1	25.0	8.5	27.4	25.0	12.7	41.5
b17	0.5	0	2.8	15.5	25.0	2.7	28.6	25.0	4.9	31.2	66.7	6.6	30.3	66.7	8.7	36.9
b22	0.6	0	3.7	9.3	20.0	4.6	16.4	20.0	6.1	17.4	50.0	11.2	26.2	50.0	19.0	31.9
diffeq1	0.5	0	3.5	14.9	25.0	8.6	25.8	25.0	6.0	12.3	66.7	8.2	13.6	66.7	10.2	16.3
square	0.5	0	1.8	10.1	25.0	3.6	13.9	25.0	4.6	15.1	66.7	5.4	16.4	66.7	8.2	24.8
Average	0.5	0	2.6	12.8	13.6	4.9	19.2	21.9	6.5	20.3	47.6	8.7	23.6	47.6	12.2	30.2

Again, these overheads enable DRC-clean layouts even for full-chip camouflaging—we believe that this is a justifiable achievement.

On power and performance: Recall that our primitive has all its wires routed in higher metal layers; see also Fig. 9. Hence, an impact on power and performance is expected.

Interestingly, for camouflaging up until 60% of the layout, the average overheads are relatively small, typically in the range of 2–7% for power and 12–21% for delay. The overheads increase for larger and full-scale LC, but still follow a linear growth. We believe that these two trends are due to the following:

1) Besides the transformed INV/BUF gates, all gates remain as is; we experience no inherent overheads for the majority of gates.
2) The relatively low resistance of the higher metal layers used by our primitive helps to limit the delay cost for LC.
3) The ubiquitous nets for the fixed values *0* and *1* are not switching and, thus, they neither increase power nor delay.
4) For LC beyond 60%, the positive effects above are offset by the steady increase of the wiring for camouflaged gates, thereby raising the routing congestion. Since routing congestion can only be managed by re-routing in some detours, this lengthens parts of the wires further. In turn, this also impacts power and delay.

Comparison on layout level: Recall that our work is one of the very few to evaluate LC on placed and routed GDSII designs. When contrasting to a previous study, conducted by Malik *et al.* [41], we observe significantly lower overheads; such a qualitative comparison is fair as the authors use the same *NanGate* library [39]. Specifically, Malik *et al.* reported overheads of 7.09×, 6.45×, and 3.12× for area, power, and delay, respectively. It is also important to note that Malik *et al.* implement and evaluate their approach for one *AES S-box*, which has a far lower number of gates—namely only 421—when compared to all the benchmarks we consider. The authors indicate themselves that the cost will increase for larger circuits [41].

For Zhang's work [11], as its MUX-based primitive is not publicly available, we implement it ourselves, and perform a detailed layout-level evaluation. For example, we observe 464%, 638%, and 63% increase in area, power, and delay, respectively, when camouflaging all the gates for the *ISCAS-85* benchmark *c7552*. These numbers are 8.3×, 55.4×, and 2.18× higher than ours for the same scenario.

Besides advocating a provably secure LC scheme, Li *et al.* [19] also propose two different LC primitives, called *STF-type* and *XOR-type*. As the authors report only gate-level cost, we conduct a layout-level evaluation ourselves as well (Table III). Here we map the numbers reported for their primitives to regular gates of the respective type—doing so is fair and conservative as it implies only a linear scaling.

Comparison on gate level: Previous works typically report their cost only for small-scale LC, which arguably is their sole scope. For example, the MUX-based approach of [10] exhibits on average 50% delay overhead and 15% area overhead already for 5% LC.

We also like to note that most prior art report numbers based on RTL simulations, which seems too optimistic, especially for large-scale LC. Conservatively assuming that these numbers would still scale only linearly, the primitive of [9] would incur ≈600%, ≈400%, and ≈300% cost on area, power, and delay, i.e., with 50% of the gates camouflaged. For the same scenario, our technique incurs only 16.7%, 6.9%, and 12.8% overheads for area, power, and delay, respectively. That is, we attain 36×, 58×, and 23× lower overheads.

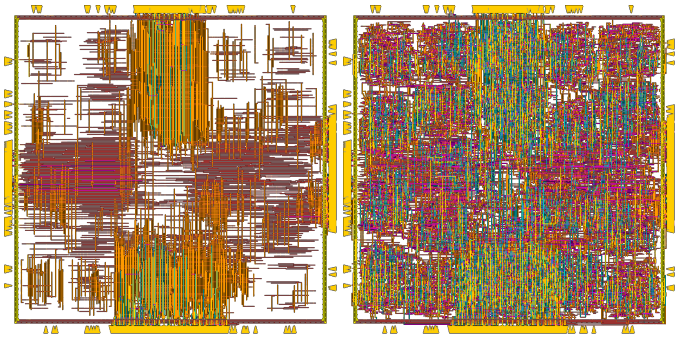


Fig. 9. Metal layers M5 to M10 for benchmark *aes_core* [43]; the original layout is on the left and the layout with 100% LC is on the right.

TABLE III
COMPARISON OF AREA (A), POWER (P), AND DELAY (D) FOR 100% LC ON SELECTED BENCHMARKS (N/A MEANS NOT AVAILABLE)

Benchmark	XOR-type [19]			STF-type [19]			[14]			Ours		
	A	P	D	A	P	D	A	P	D	A	P	D
apex4	59.7	5.9	45.1	51.3	18.8	45.3	N/A	N/A	N/A	50.0	17.5	41.8
c432	51.1	19.3	9.8	38.9	32.6	24.3	140.0	8.0	96.0	62.5	25.3	26.8
c5315	96.3	27.4	51.1	70.1	44.3	59.8	200.0	10.0	76.0	55.6	17.5	42.2
c7552	96.6	30.0	66.8	59.7	40.0	63.8	175.0	9.0	90.0	55.6	11.5	28.9
b14	74.8	14.2	39.6	57.6	36.6	39.2	N/A	N/A	N/A	25.0	15.1	22.1
b17	71.3	3.3	53.0	53.4	20.9	58.9	N/A	N/A	N/A	66.7	8.7	36.9
b20	78.5	16.4	46.0	62.6	40.9	72.2	N/A	N/A	N/A	50.0	15.1	30.1
Average	75.5	16.6	44.5	56.2	33.4	51.9	171.7	9.0	87.3	52.2	15.8	32.7

Comparison with threshold-dependent LC: Nirmala *et al.* [13] recently proposed a promising concept of threshold-dependent LC switches. Also here we apply a thorough layout-level evaluation, across 34 benchmarks, based on their numbers reported for their primitives. As a result, we find that this approach will incur significant layout cost for full-chip LC: ≈1,360%, ≈1,266%, and ≈100% for area, power, and delay, respectively. For [12], we observe a linear trend in the reported layout cost; extrapolating those numbers for full-chip LC would translate to ≈78% and ≈147% for power and delay, respectively. Finally, for the work of Erbagci *et al.* [14], we report their numbers in Table III for comparison.

Comparison with provably secure LC: Recall that [19]–[21] rely on additional circuitry to protect individual, selected gates/wires. Such circuitry can incur a high cost, especially for area. For example in Xie *et al.*'s work [21], when protecting one gate/wire of the *c7552* benchmark using 64 key bits, the die-level area overhead we observe

TABLE IV
RUNTIME FOR OUR SAT ATTACKS (USING [35]) ON SELECTED DESIGNS, IN SECONDS (TIME-OUT T-O IS 172,800 SECONDS, I.E., 48 HOURS)

Benchmark	10% LC					20% LC					30% LC					40/50–100% LC				
	[9]	[41]	[10]	[11]	Our	[9]	[41]	[10]	[11]	Our	[9]	[41]	[10]	[11]	Our	[9]	[41]	[10]	[11]	Our
aes_core	703	t-o	t-o	162	6,732	4,779	t-o	t-o	470	t-o	6,783	t-o	t-o	2,304	t-o	t-o	t-o	t-o	t-o	t-o
ac97_ctrl	102	t-o	3,271	37	1,723	346	t-o	t-o	97	8,678	4,211	t-o	t-o	589	t-o	t-o	t-o	t-o	t-o	t-o
b15	423	t-o	t-o	65	583	5,163	t-o	t-o	331	10,012	5,163	t-o	t-o	3,306	t-o	t-o	t-o	t-o	t-o	t-o
b17	7,894	t-o	t-o	1,340	t-o	t-o	t-o	t-o	6,041	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
c7552	39	1,686	2,429	12	68	160	t-o	t-o	103	8,486	1,589	t-o	t-o	746	t-o	t-o	t-o	t-o	t-o	t-o
diffeq1	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o
square	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o	t-o

in our layout-level implementation of their work is close to 106%.[8] If we were to protect more gates/wires, the overhead would scale up accordingly. In contrast, when camouflaging *all gates* of the same circuit, our approach incurs only ≈56% overhead for the die area.

B. Security Evaluation

Recall that our primary objective is large-scale LC; an important observation is that this achieves *practically secure LC*. That is, by camouflaging up to 100% of the layout, we seek to induce the highest computational effort possible for SAT attacks, at least without inserting additional *provably secure* structures.

On the notion of practically secure LC: It is not straightforward to prove beforehand to what extent large-scale LC will render a layout (practically) secure without leveraging a SAT solver's capabilities for de-camouflaging. Li *et al.* [19] have shown that the effort for de-camouflaging scales *on average* with both (i) the solution space C concerning all the possible functionalities for the whole, camouflaged design and (ii) the Hamming distance among those different functionalities. In turn, the key reason why a further theoretical evaluation of large-scale LC is so difficult is that the space of C (and thereby the Hamming distance as well) depends on (a) the types and count of possible functionalities for all employed LC primitives, (b) the count and selection of gates to camouflage, and (c) the connectivity among all gates in the design, all at the same time. On the one hand, for example, designs containing XOR/XNOR and/or multipliers are harder to de-camouflage in practice [17], [42]. On the other hand, as with classical logic minimization, the solution space of some interconnected camouflaged gates (but not for [N]AND-trees) may be largely simplified during the SAT search.

In short, while we can easily estimate the upper bound of C, it may be far from the actual number of functions required to consider, depending on the design. This implies that without conducting SAT attacks (or similar techniques), there is no basis for a fair evaluation of different LC schemes. Hence we resort to such an empirical but comprehensive study, also to contrast our work to prior art.

As we observe in this study, we can indeed expect prohibitive runtimes for large-scale LC. More specifically, e.g., we extrapolate that de-camouflaging *aes_core* (when all 39,014 gates are camouflaged using our primitive) can take approximately 108 years.[9] Similar observations have also been made by Yu *et al.* [42], albeit for a different primitive and the much smaller benchmark *c432* (209 two-input gates); even this layout could not be resolved within three days.

Comparative study on the resilience of LC: In Table IV, we list the runtime for SAT attacks on camouflaged designs, contrasting our LC primitive and those of [9]–[11], [41]. Note that these prior studies did not report on any SAT attacks. Thus, we model their primitives ourselves as outlined in [17], [42]. Also, recall that we camouflage the same sets of gates across all techniques for a fair comparison.

[8]Since these studies [19]–[21] are all based on the same principle (inserting combinatorial trees), we can also expect similar trends for the other schemes.

[9]For a meaningful regression, we sample the average attack runtimes across 70 different sets of randomly camouflaged gates.

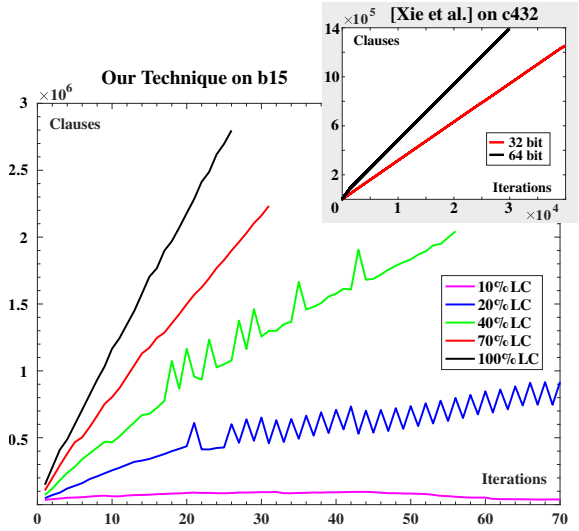


Fig. 10. The progress of our SAT attacks (using [35]) on our LC implementation (main plot) and [21] (inset). For ours, 10% and 20% LC is resolved; the larger setups incur time-out (48 hours) and excessive numbers of clauses. For [21], neither setups (32- and 64-bit keys) are resolved before time-out.

It is noteworthy that none of the layouts can be de-camouflaged within 48 hours once full-chip LC is applied. In fact, all layouts remain already resilient just beyond 40/50% LC. We ran further exploratory attacks for 7 days on large-scale LC using our primitive—without observing any improvement. We can reasonably expect other prior art, such as the XOR-type/STF-type gates of [19], to perform comparably well for large-scale LC (i.e., at least on the same benchmarks). That is because the set of possible functionalities which these prior studies implement are typically within the same range.

For a more meaningful comparison, we also consider relatively small scales for LC (10% up to 30%). Here, the primitive advocated by Zhang [11] appears as the weakest, and the one proposed by Malik *et al.* [41] seems the most resilient. Our primitive is next only to that of Wang *et al.* [10] and that of Malik *et al.* [41]. Now, it is also important to recall that our proposed technique incurs significantly smaller PPA cost than these schemes [10], [41].

In short, we argue that our proposed technique is currently the only resilient yet practical solution towards large-scale LC.

On the SAT-attack effort against large-scale LC: Recall that the growth of clauses hints on any SAT attack's progress [42]. When employing our primitive for small-scale LC, the number of clauses saturates at some point, and the layout can be de-camouflaged within 48 hours (Fig. 10). On the contrary, for large-scale LC (again, which has not been resolved), we observe that the number of clauses increases continuously. We found a similar trend when conducting attacks on [21] (Fig. 10, inset). This similarity indicates that attacks on our primitive—once it is applied at large scales—require enormous efforts, similar to attacks on SAT-hardened primitives such as [21].

TABLE V
RUNTIME FOR SAT ATTACKS ON [29], ENABLED BY [35], IN SECONDS (COLUMNS N1–N3 DENOTE THE NUMBER OF DUMMY WIRES ADDED WHILE LIMITING THE DELAY OVERHEAD TO 0%, 3%, AND 5%, RESPECTIVELY, AS PROPOSED IN [29])

Benchmark	N1	Time	N2	Time	N3	Time
b14	30	7	36	9	55	11
b15	38	7	44	8	84	15
b17	92	149	198	170	272	214
b18	265	2,964	334	3,223	518	3,816
b19	438	4,685	583	5,393	893	7,684
b20	48	35	85	46	166	70
b21	54	29	76	56	168	63
b22	76	58	113	79	191	128

Overall, we are *not* claiming that large-scale LC (using our primitive or prior art) cannot be resolved *eventually* using SAT attacks. Rather, we provide strong empirical evidence that such practically secure LC (i.e., once 50–100% of all gates are camouflaged) imposes a prohibitive computational cost on SAT solvers. Besides, while the SAT-solver capabilities are only going to increase, physical limits for computation will remain in any case [16], [38].

On the resilience of [29]: Since our work is inspired by [29] to some degree, it seems imperative to investigate its resilience as well. We model the attack on [29] as outlined in Fig. 3 of [42]. It can be seen from Table V that the SAT attacks can distinguish between real and dummy interconnects within a relatively short time, no matter how many interconnects are obfuscated (i.e., at least concerning the delay constraints defined in [29]). That is, an overly constrained obfuscation as proposed in [29] cannot withstand powerful SAT attacks. Similar observations have been reported by Yu *et al.* [42].

VII. SUMMARY AND OUTLOOK

We show in this work that most techniques for layout camouflaging (LC) are only resilient to powerful SAT attacks once more than 50% of the entire chip is camouflaged. We extend this call for large-scale LC also towards SAT-hardened and advanced LC techniques, to thwart up-and-coming customized and/or invasive attacks.

We promote the obfuscation of interconnects as another promising avenue for LC. Towards this end, we propose and implement BEOL-centric but generic LC primitives (which are applicable to any FEOL node), thoroughly contrast them to the prior art, and make our design flow publicly available in [22]. We have leveraged powerful SAT attacks and LC schemes proposed in the literature and provided a comprehensive and fair evaluation. Further, we strive for practically relevant layout evaluation; all our camouflaged layouts are DRC-clean at the GDSII level. The proposed scheme is the first (to the best of our knowledge) that can deliver low-cost *and* resilient full-chip camouflaging, i.e., when considering both the layout and manufacturing cost as well as the readily deployable protection against fab adversaries enabled by split manufacturing.

In future work, we will elaborate in more detail on the cost and security for split manufacturing enabled by our LC primitive.

ACKNOWLEDGMENTS

We would like to thank Pramod Subramanyan (University of California, Berkeley) and Sergei Skorobogatov (University of Cambridge, UK) for their valuable inputs. Furthermore, we are grateful to Tri Cao and Jeyavijayan (JV) Rajendran (University of Texas at Dallas) for providing their network-flow attack of [34]; this helped exploring the resilience of our primitive in the context of split manufacturing.

This work was supported in part by the Army Research Office (ARO) under Grant 65513-CS, and the Center for Cyber Security (CCS) at New York University NY/Abu Dhabi (NYU/NYUAD). Besides, this work was carried out in part on the High Performance Computing resources at New York University Abu Dhabi.

REFERENCES

[1] C. McCants, "Trusted integrated chips (TIC)," Intelligence Advanced Research Projects Activity (IARPA), Tech. Rep., 2011. [Online]. Available: https://www.iarpa.gov/index.php/research-programs/tic
[2] S. Skorobogatov, "Physical attacks and tamper resistance," in *Introduction to Hardware Security and Trust*. Springer, 2012, pp. 143–173.
[3] M. Rostami *et al.*, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.
[4] P. Subramanyan *et al.*, "Reverse engineering digital circuits using structural and functional analyses," *Trans. Emerg. Top. Comp.*, vol. 2, no. 1, pp. 63–80, 2014.
[5] T. Sugawara *et al.*, "Reversing stealthy dopant-level circuits," *J. Cryptogr. Eng.*, vol. 5, no. 2, pp. 85–94, 2015.
[6] F. Courbon *et al.*, "Direct charge measurement in floating gate transistors of flash EEPROM using scanning electron microscopy," in *Proc. ISTFA*, 2016, pp. 1–6.
[7] J. A. Roy *et al.*, "Ending piracy of integrated circuits," *Computer*, vol. 43, no. 10, pp. 30–38, 2010.
[8] R. P. Cocchi *et al.*, "Circuit camouflage integration for hardware IP protection," in *Proc. DAC*, 2014, pp. 1–5.
[9] J. Rajendran *et al.*, "Security analysis of integrated circuit camouflaging," in *Proc. CCS*, 2013, pp. 709–720.
[10] X. Wang *et al.*, "Secure and low-overhead circuit obfuscation technique with multiplexers," in *Proc. GLSVLSI*, 2016, pp. 133–136.
[11] J. Zhang, "A practical logic obfuscation technique for hardware security," *Trans. VLSI Syst.*, vol. 24, no. 3, pp. 1193–1197, 2016.
[12] M. I. M. Collantes *et al.*, "Threshold-dependent camouflaged cells to secure circuits against reverse engineering attacks," in *Proc. ISVLSI*, 2016, pp. 443–448.
[13] I. R. Nirmala *et al.*, "A novel threshold voltage defined switch for circuit camouflaging," in *Proc. ETS*, 2016, pp. 1–2.
[14] B. Erbagci *et al.*, "A secure camouflaged threshold voltage defined logic family," in *Proc. HOST*, 2016, pp. 229–235.
[15] A. Vijayakumar *et al.*, "Physical design obfuscation of hardware: A comprehensive investigation of device- and logic-level techniques," *Trans. Inf. Forens. Sec.*, vol. 12, no. 1, pp. 64–77, 2017.
[16] M. E. Massad *et al.*, "Integrated circuit (IC) decamouflaging: Reverse engineering camouflaged ICs within minutes," in *Proc. NDSS*, 2015, pp. 1–14.
[17] P. Subramanyan *et al.*, "Evaluating the security of logic encryption algorithms," in *Proc. HOST*, 2015, pp. 137–143, see also [35].
[18] D. Liu *et al.*, "Oracle-guided incremental SAT solving to reverse engineer camouflaged logic circuits," in *Proc. DATE*, 2016, pp. 433–438.
[19] M. Li *et al.*, "Provably secure camouflaging strategy for IC protection," in *Proc. ICCAD*, 2016, pp. 28:1–28:8.
[20] M. Yasin *et al.*, "CamoPerturb: Secure IC camouflaging for minterm protection," in *Proc. ICCAD*, 2016, pp. 29:1–29:8.
[21] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," in *Proc. CHES*, 2016, pp. 127–146.
[22] (2017) IC camouflaging – DfX Lab, NYUAD. [Online]. Available: http://sites.nyuad.nyu.edu/dfx/research-topics/design-for-trust-ic-camouflaging/
[23] M. Yasin *et al.*, "Removal attacks on logic locking and camouflaging techniques," *Trans. Emerg. Top. Comp.*, vol. PP, no. 99, 2017.
[24] C. Helfmeier *et al.*, "Breaking and entering through the silicon," in *Proc. CCS*, 2013, pp. 733–744.
[25] K. Shamsi *et al.*, "AppSAT: Approximately deobfuscating integrated circuits," in *Proc. HOST*, 2017, pp. 95–100.
[26] Y. Shen and H. Zhou, "Double DIP: Re-evaluating security of logic encryption algorithms," in *Proc. GLSVLSI*, 2017, pp. 179–184.
[27] X. Xu *et al.*, "Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks," in *Proc. CHES*, 2017.
[28] H. Zhou *et al.*, "CycSAT: SAT-based attack on cyclic logic encryptions," in *Proc. ICCAD*, 2017.
[29] S. Chen *et al.*, "Chip-level anti-reverse engineering using transformable interconnects," in *Proc. DFTS*, 2015, pp. 109–114.
[30] J. Swerts *et al.*, "BEOL compatible high tunnel magneto resistance perpendicular magnetic tunnel junctions using a sacrificial Mg layer as CoFeB free layer cap," *Applied Physics Letters*, vol. 106, no. 26, pp. 262 407:1–262 407:4, 2015.
[31] S.-W. Hwang *et al.*, "A physically transient form of silicon electronics," *Science*, vol. 337, no. 6102, pp. 1640–1644, 2012.
[32] A. Todri-Sanial *et al.*, Eds., *Carbon Nanotubes for Interconnects*. Springer, 2017.
[33] A. Naeemi *et al.*, "BEOL scaling limits and next generation technology prospects," in *Proc. DAC*, 2014, pp. 26:1–26:6.
[34] Y. Wang *et al.*, "The cat and mouse in split manufacturing," in *Proc. DAC*, 2016, pp. 165:1–165:6.
[35] P. Subramanyan. (2017) Evaluating the security of logic encryption algorithms. [Online]. Available: https://bitbucket.org/spramod/host15-logic-encryption
[36] S. Keshavarz *et al.*, "Design automation for obfuscated circuits with multiple viable functions," in *Proc. DATE*, 2017, pp. 886–889.
[37] R. S. Shelar and M. Patyra, "Impact of local interconnects on timing and power in a high performance microprocessor," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, vol. 32, no. 10, pp. 1623–1627, 2013.
[38] I. L. Markov, "Limits on fundamental limits to computation," *Nature*, vol. 512, no. 7513, pp. 147–154, 2014.
[39] (2011) NanGate FreePDK45 Open Cell Library. Nangate Inc. [Online]. Available: http://www.nangate.com/?page_id=2325
[40] K. Xiao *et al.*, "Efficient and secure split manufacturing via obfuscated built-in self-authentication," in *Proc. HOST*, 2015, pp. 14–19.
[41] S. Malik *et al.*, "Development of a layout-level hardware obfuscation tool," in *Proc. ISVLSI*, 2015, pp. 204–209.
[42] C. Yu *et al.*, "Incremental SAT-based reverse engineering of camouflaged logic circuits," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, vol. PP, no. 99, 2017.
[43] L. Amarù. (2015) Majority-inverter graph (MIG) benchmark suite. [Online]. Available: http://lsi.epfl.ch/MIG