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

Satwik Patnaik, Student Member, IEEE, Mohammed Ashraf, Ozgur Sinanoglu, Senior Member, IEEE, and Johann Knechtel, Member, IEEE

Abstract—Layout camouflaging can protect the intellectual property of modern circuits. Most prior art, however, incurs excessive layout overheads and necessitates customization of active-device manufacturing processes, i.e., the front-end-of-line (FEOL). As a result, camouflaging has typically been applied selectively, which can ultimately undermine its resilience. Here, we propose a low-cost and generic scheme—full-chip camouflaging can be finally realized without reservations. Our scheme is based on obfuscating the interconnects, i.e., the back-end-of-line (BEOL), through design-time handling for real and dummy wires and vias. To that end, we implement custom, BEOL-centric obfuscation cells, and develop a CAD flow using industrial tools. Our scheme can be applied to any design and technology node without FEOL-level modifications. Considering its BEOL-centric nature, we advocate applying our scheme in conjunction with split manufacturing, to furthermore protect against untrusted fabs. We evaluate our scheme for various designs at the physical, DRC-clean layout level. Our scheme incurs a significantly lower cost than most of the prior art. Notably, for fully camouflaged layouts, we observe average power, performance, and area overheads of 24.96%, 19.06%, and 32.55%, respectively. We conduct experiments to infer the chip IP directly, as the IP is hidden within an obfuscated physical design. The traditional goal of layout camouflaging (LC), or simply camouflaging, is to mitigate reverse-engineering attacks by end-users. Therefore, camouflaging alters the physical appearance of an IC such that it is arduous for the attacker to understand the true IC design (Fig. 1). In other words, when facing camouflaged ICs, an attacker shall be rendered unable to infer the chip IP directly, as the IP is hidden within an obfuscated physical design.

Index Terms—Hardware security, IP protection, Reverse engineering, IC camouflaging, Interconnects, Split manufacturing, Boolean satisfiability, Proximity attacks.

I. INTRODUCTION

Ensuring the integrity, security, and trustworthiness of integrated circuits (ICs) have become major concerns in recent years [2], [3]. One important aspect here is that the intellectual property (IP) of modern ICs can be duplicated without consent, resulting in financial losses for IP vendors, which are estimated to amount to several billion dollars per year. This is because modern ICs are the products of distributed and globalized supply chains. If an IC design is not protected, any of the various entities involved in such outsourced supply chains may reconstruct/pirate the underlying IP. Protecting the IP can also help to mitigate other hardware-centric threats, e.g., hardware Trojans [3]. Besides, a malicious end-user, i.e., an adversary without direct access to the design and fabrication process, can resort to reverse engineering of ICs to obtain the IP. The tools and know-how for such reverse-engineering attacks are becoming more advanced and widely available, rendering this a practical and severe threat [3], [4].

The traditional goal of layout camouflaging (LC), or simply camouflaging, is to mitigate reverse-engineering attacks by end-users. Therefore, camouflaging alters the physical appearance of an IC such that it is arduous for the attacker to understand the true IC design (Fig. 1). In other words, when facing camouflaged ICs, an attacker shall be rendered unable to infer the chip IP directly, as the IP is hidden within an obfuscated physical design.

Camouflaging is implemented traditionally at the device layer, e.g., by “look-alike” or ambiguous gates [6], [7], by secretly configured multiplexers [8], [9], or by the threshold-dependent configuration of gates [10]–[12]. We note that prior art requires a fully trustworthy fabrication process, as the device obfuscation has to be realized by the foundry. We also note that most schemes incur a high layout cost. For example, the XOR-NAND-NOR primitive proposed in [7] is expected to incur 4× area, 5.5× power, and 1.6× delay cost in comparison to a conventional NAND gate. For another scheme by Akkaya et al. [13], demonstrated in a 65nm chip, an equivalent primitive incurs even higher APD cost, namely 7.3× area (A), 9.2× power (P), and 6.6× delay (D).

This work is an extension of [1]. This work was supported in part by the Army Research Office under Grant 65513-CS, the Center for Cyber Security (CCS) at NYU/NYU AD, and the NYU AD Research Enhancement Fund (REF) under Grant RE218. The work of S. Patnaik was supported by the Global Ph.D. Fellowship at NYU/NYU AD. Besides, this work was carried out in part on the HPC facility at NYU AD. This paper was recommended by Associate Editor Y. Makris.

S. Patnaik is with the Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University (NYU), Brooklyn, NY, 11201, USA (e-mail: sp4012@nyu.edu).

M. Ashraf, O. Sinanoglu, and J. Knechtel are with the Division of Engineering, New York University Abu Dhabi (NYU AD), Saadiyat Island, 129188, UAE (e-mail: ma199@nyu.edu; ozgursin@nyu.edu; johann@nyu.edu). © 2020 IEEE. This is the author’s version of the work. It is posted here for personal use. Not for redistribution. The definitive Version of Record is published in IEEE TCAD, DOI 10.1109/TCAD.2020.2981034
Accordingly, the application of such schemes is suggested to be limited, which in turn may compromise their resilience as discussed next.

Assuming that prior schemes are applicable only to selected parts of an IC in practice, an essential challenge is where and to what extent camouflaging shall be effected. Ideally, an attacker’s effort to resolve a carefully camouflaged netlist would be exponential in the number of camouflaged gates [14]. To achieve such resilience, among other approaches, Rajendran et al. [7] proposed to camouflage gates that are interfering, i.e., gates that cannot be resolved individually. Still, advances for Boolean satisfiability (SAT) solvers have enabled powerful attacks on camouflaging (and logic locking) [14]–[16], undermining the promises of [7] and other works. Even for hardened schemes such as [17]–[19], which explicitly aim for exponential attack resilience, tailored SAT and other attack frameworks have been proposed [20]–[23]. Therefore, the trade-off between resilience and cost/applicability remains an open challenge for camouflaging.

In Table I we provide a high-level comparison of the selected prior art and our work. We also discuss the prior art and other relevant security aspects further in Sec. II. Besides, a comprehensive overview of camouflaging is given in [24], and IP protection in general is reviewed in [25], [26].

In this paper, we make the following case: to remain resilient, at least as long as foreseeable, camouflaging should be applied at large scales, ideally for the full chip. Alternative approaches are required to enable large-scale camouflaging, especially when aiming for low cost and fewer dependencies on trustworthy fabrication. To that end, we propose and evaluate a novel concept based on obfuscating the interconnects (Fig. 2). Our contributions can be summarized as follows:

1) Based on the emerging notion of obfuscating the interconnects, we enable resilient and large-scale camouflaging, up to full-chip camouflaging. To do so, we develop a security- and cost-driven, end-to-end CAD methodology, and we propose novel back-end-of-line (BEOL)-centric camouflaging primitives which are applicable for all types of regular gates. That is, our primitives do not require any modification at the front-end-of-line (FEOL) layers and can, therefore, be easily integrated into any design, technology, and CAD flow. We leverage Cadence Innovus, and make our layouts publicly available in [28].

2) The fact that our work realizes BEOL-centric camouflaging suggests an application of split manufacturing [2], [29], [30]. Doing so allows us, for the first time for camouflaging, to hinder fab adversaries in addition to malicious end-users.

3) We conduct a thorough evaluation of our scheme concerning the APD (or PPA—power, performance, and area) cost for camouflaged, DRC-clean physical layouts. In contrast, most prior art investigate their camouflaging primitives only as stand-alone devices, without applying them in actual layouts; we argue that this is overly optimistic. Thus, our work is one of the very few providing a comprehensive layout-level evaluation, and we also provide an in-depth and comprehensive study on APD cost for a wide range of prior camouflaging schemes.

4) We assess the resilience of our scheme and compare it against previous works. For our scheme, we introduce the notion of practically secure camouflaging which seeks to impose high computational costs on SAT-based attacks while, unlike provably secure camouflaging, not requiring additional, dedicated circuit structures. For that assessment, we employ powerful SAT-based attacks [31], [32] on traditional benchmarks as well as large VLSI benchmarks. With regards to split manufacturing, we employ state-of-the-art proximity attacks [33], [34] and demonstrate that our scheme is also resilient against such fab-based adversarial activities.

II. BACKGROUND AND MOTIVATION

A. Split Manufacturing

A technique orthogonal to camouflaging is split manufacturing [2], [29], [30]—while camouflaging seeks to protect against malicious end-users, split manufacturing aims to hinder adversaries acting at manufacturing time. Typically, split manufacturing means to split the IC manufacturing flow into FEOL and BEOL processes, i.e., the front-end-of-line and back-end-of-line steps, where the FEOL is outsourced, and the BEOL is realized by some trusted in-house or on-shore facility.

Note that split manufacturing has been demonstrated: [30] describes promising measurement results for an IBM/GLOBALFOUNDRIES 130nm split process, and [29] summarizes further results, most notably for a 28nm split process run by Samsung across Austin and South Korea.

For the FEOL foundry, the resulting partial layout appears as “sea of gates,” with most of the interconnects missing. Therefore, it is argued that an adversary residing in the foundry cannot infer the full design easily, which hinders him/her from both IP piracy and Trojan insertion. The advantages of split manufacturing are as follows: 1) it allows for commissioning high-end, state-of-the-art, yet untrusted, FEOL facilities; 2) the BEOL is significantly less complex and cheap to implement compared to the FEOL and, thus, technology requirements for the trusted facility are moderate; 3) the supply chain can

| Work | LC Style | Trusted Foundry | LC Scale | APD | Resilience to Algorithmic Attacks |
|------|----------|-----------------|----------|-----|---------------------------------|
| [7]  | Contact  | Yes             | “Small”  | “Low” | Vulnerable [23]                 |
| [11] | Contact/Dopant | Yes            | “Small”  | “Moderate” | Vulnerable [23] |
| [12] | Dopant   | Yes             | “Small”  | “Moderate” | Vulnerable [23] |
| [14] | Interconnects | Yes            | “Small”  | “Low” | Vulnerable [This] |

Ours: Interconnects Untrusted FEOL, “Large” “Moderate” Superior; attacks yet to succeed

Fig. 2. Our concept is based on secret n:1 mappings at the BEOL, also using dummy vias and wires, thereby obfuscating the real inputs for any gate (not only two-input XOR gates). The set of obfuscated functionalities depends on n, the randomized selection of the dummy nets, and the gate type.

TABLE I

COMPARISON OF SELECTED LAYOUT CAMOUFLAGING (LC) SCHEMES
Since physical layouts are designed holistically, at least when using regular and security-unaware CAD tools, various hints on the BEOL can remain within the FEOL. Wang et al. [35] proposed a so-called proximity attack, where they formulated various FEOL-level hints within a network-flow attack model. For selected designs, they succeeded to infer the majority of missing BEOL wires. Magaña et al. [36] proposed various attacks that focus on FEOL-level routing, and they conclude that such efforts are more effective than relying solely on the placement. Zhang et al. [37] and Li et al. [38] leveraged machine learning for advanced and scalable attack frameworks. However, neither of those attacks [36–38] recovers an actual netlist; they instead provide sets of most probable BEOL connections. Therefore, an attacker requires further effort and knowledge-how before he/she can obtain a complete netlist. In any case, attacks which can achieve 100% correctness when inferring all BEOL wires of industrial, large-scale designs, possibly even with some placement- or routing-level perturbations introduced for protection (e.g., see [39–42]), are yet to be demonstrated.

**B. Camouflaging at the FEOL: Limitations and Vulnerabilities**

Prior art targets mainly at the active device layer, the FEOL, and, thus, requires (i) custom design of camouflaged gates, along with library characterization, and (ii) some modifications of well-established and optimized device manufacturing processes. It is easy to see that both will incur efforts and financial cost, that is on top of any layout overheads. Regarding layout overheads, as already indicated in Sec. II, most existing camouflaging schemes incur a high cost and are accordingly limited for practical use. For example for Zhang’s work [9], we observe 464%, 638%, and 63% overheads for area, power, and delay, respectively, when camouflaging all gates of the ISCAS-85 benchmark c7552; these numbers are 13.92×, 37.51×, and 4.47× higher than ours for the same scenario (see also Sec. VI-B for more comparisons).

Furthermore, it is important to note that prior art can be constrained to begin with, namely by synthesis limiting the extent or scale to which camouflaging is applicable. More specifically, most prior arts implement stealthily multiple Boolean functions within customized gate-level structures, but these primitives capture typically only some subsets of all the functionalities needed for modern IC designs. For example, with the XOR2-NAND2-NOR2 primitive proposed in [7], the camouflaging scale is inherently constrained (by synthesis) to the ratio of gates being mapped to XOR2, NAND2, and NOR2. We explore this limitation for selected schemes in Table III. We note that the limited scales of around 30–40% camouflaging observed for some schemes are not sufficient to protect against SAT-based attacks; see also Sec. VII-C for details. We also note that, as a countermeasure, one may conduct synthesis with constrained libraries that only provide the functions covered by the employed camouflaging primitives. Doing so, however, would incur considerable APD cost over full-library synthesis already for the baseline (Table III), with actual overheads incurred by the camouflaging primitives themselves coming further on top.

In short, existing schemes can be applied only selectively—if at all—due to their impact on FEOL processing steps, their inherent APD overheads, and their limited scale. As a result, the constrained application of these and other schemes can compromise their resilience, as discussed next.

2The hints are: (i) physical proximity of gates, (ii) avoidance of combinatorial loops, which are rare in practice, (iii) timing and load constraints, and (iv) orientation and routing direction of wires.
As mentioned in Sec. [1], powerful SAT-based attacks have challenged most prior art on camouflaging (and logic locking) [14]–[16]. The idea is that an obfuscated netlist can be modeled as satisfiability problem, where a working chip copy is leveraged as an oracle to iteratively revise the model of the camouflaged netlist, by pruning out step by step any infeasible assignments for the camouflaged gates. The previously unforeseen success of such SAT-based attacks stems from the fact that typically only a small number of I/O patterns are required when resolving various camouflaging schemes.

Several studies on provably secure camouflaging (and logic locking), e.g., see [17]–[19], [44], seek to mitigate SAT-based attacks by inserting dedicated structures which necessitate considering an exponential number of I/O patterns. However, the output signals related to such structures have been identified successfully, e.g., by signal-probability- or sensitization-driven algorithms [23], [45], [46]. These critical wires may be ignored or even cut to circumvent the security features. We note that wire cutting has been demonstrated in the past [47], enabling such invasive attacks in principle. Besides, these schemes are also vulnerable to other algorithmic attacks [20]–[23]. This is because a key limitation of these schemes is that they induce low output corruptibility, allowing an attacker to obtain relatively easily an approximate version of the IP. Note that this fact also implies that these schemes are less suitable to protect design IP at the chip scale.

Reverse engineering measures may also render FEOL-centric camouflaging directly void, without the assistance of algorithmic techniques. Schemes such as “look-alike” and ambiguous gates [6], [7], or secretly configured MUXes [8], [9] rely on dummy contacts or dummy channels. While yet to be demonstrated, we argue that scanning electron microscopy in the passive voltage mode (SEM PVC) might break these schemes. This is because dummy contacts/channels accumulate charges to a much lower degree than real contacts/channels. Threshold-dependent camouflaging of gates can also be revealed by SEM PVC, as demonstrated successfully by Sugawara et al. [4]. As Collantes et al. [10] indicate themselves, monitoring the etch rates can also reveal different doping levels, which are at the heart of their threshold-dependent gates. Besides, the tailored structures of provably secure schemes may be relatively easy to identify during reverse engineering, as these structures are typically applied only in few places due to their relatively high cost, and they rely on otherwise uncommon, very large combinatorial trees.

Overall, considering these various limitations and vulnerabilities for prior art on FEOL-centric camouflaging, we argue that alternative approaches are necessary. We believe that such novel approaches should foremost be large-scale camouflaging schemes, ideally camouflaging full chips, to 1) remain resilient as long as foreseeable against any algorithmic attacks, and 2) impose the highest efforts for any physical attacks.

C. Toward Camouflaging at the BEOL

An interesting approach was suggested by Chen et al. [27], namely to obfuscate the interconnects by implementing real, conductive vias using magnesium (Mg) along with dummy, non-conductive vias using magnesium oxide (MgO). Chen et al. elaborated that the use of Mg/MgO is practical from both the perspectives of (i) manufacturability and (ii) mitigation of reverse engineering. For (i), they noted that Mg had been used traditionally to facilitate the bonding of copper interconnects, and both Mg/MgO are compatible with standard processes, in particular sputtering, electron-beam evaporation, and Dual-Damascene. Independently, Swerts et al. [48], [49] and Hwang et al. [50] have used Mg and MgO for customized BEOL processes, albeit without camouflaging in mind. For (ii), Chen et al. 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 reverse engineering. Again independently, Hwang et al. [50] have shown that Mg not only oxidizes but also dissolves quickly—as does MgO—when surrounded by fluids. We note that such dissolving would be inevitable for etching and other reverse-engineering procedures.

Although one can argue that reverse engineering of Mg/MgO vias is possible nevertheless, this is yet to be demonstrated. In general, the notion of camouflaging at the BEOL is not limited to materials currently established for manufacturing; future interconnects, e.g., based on carbon/graphene or spintronics [51], [52], may become relevant as well.

It is important to note that obfuscated interconnects, similar to FEOL-centric camouflaging, are implemented not only at the security-enforcing designer’s choice but also at the manufacturer’s discretion. However, obfuscating the interconnects still allows to limit the dependency on outsourced, potentially untrustworthy manufacturing facilities, since only the less complex BEOL part has to remain protected within trusted facilities. We note that split manufacturing is particularly promising in this context (Fig. 3).

Finally, another argument for obfuscating the interconnects is that commercial cost may be compensated for, even when split manufacturing is applied. This is because one requires only additional BEOL processing steps, with BEOL masks being relatively cheap; e.g., for 16nm, M5/M6 masks are 3.5–4× cheaper than Poly masks [53]. In contrast, most FEOL-centric camouflaging (but not necessarily threshold-dependent camouflaging) incurs high cost for the additional
FEOL masks. In general, we would like to recall that FEOL-centric camouflaging demands some alterations for devices and their manufacturing process—such alterations are likely more complex and costly than camouflaging at the BEOL.

Despite their pioneering work, Chen et al. did not provide an SAT-resilient scheme; see Sec. VI-C. We also emphasize that our concept and methodology differ significantly from [27]; here, we only leverage their notion of Mg/MgO vias. Moreover, we advocate and enable the obfuscation of interconnects for large-scale, even full-chip camouflaging, by proposing dedicated design-time techniques to handle dummy vias and wires for any gate. In contrast, Chen et al. [27] considered only small-scale, selected use of additional dummy wires/vias.

III. CONCEPT AND CAMOUFLAGING PRIMITIVE

Our key idea is the following: we leverage reverse-engineering-resistant and obfuscated interconnects to design novel, BEOL-centric camouflaging primitives that apply to any regular gate (Fig. 2). We implement $n$ wires for each input of the gates to be camouflaged, where $n-1$ of those wires are randomly selected to be acting as dummy, and all $n$ wires have their vias obfuscated. In other words, we obfuscate the real inputs of any gate via secret and randomized $n$:1 mapping at the BEOL. As a result, the actual function which a gate implements is obscured in a simple yet effective manner. The set of possible functionalities depends on $n$, the selection of the $n$ wires, and on the gate itself; a representative example is given in Fig. 4.

We conducted an exploratory, yet thorough, study on different flavors of our camouflaging primitive. We assessed all flavors when applied at large scales for various benchmarks by 1) their APD cost, while taking our scheme end-to-end and evaluating GDSII layouts, and 2) their resilience against SAT-based attacks while running the seminal attack [15], [31]. The exploratory study is detailed in [1]; we summarize it next.

We first explored the most basic flavor with two wires for each gate’s input, i.e., one randomly selected dummy wire and one real wire. We observed that this flavor offers relatively weak resilience against SAT-based attacks, even for large-scale camouflaging. Therefore, we next extended our primitive for two dummy wires, which provided notably stronger resilience. However, we also noted considerable APD cost for this flavor, due to the following. To hinder an attacker from identifying dummy wires as such, we have to select all the nets for dummy wires randomly but also such that there are no combinatorial loops. Furthermore, dummy wires have to be driven by unique nets, as otherwise some of the possible functionalities could be ruled out directly. These requirements translated to significantly increased routing congestion in practice and, in turn, higher power consumption and delays. Therefore, we note that the use of dummy wires should be limited for large-scale applicability of our scheme. Finally, we explored an extended flavor comprising fixed logic values 0 and 1 along with regular nets (Fig. 4). When compared to employing only regular nets, the benefit of using fixed-value nets is that the latter are not switching, thus exhibiting only negligible power consumption and imposing no timing overhead. Also, many of these wires can have the same driver (but not all, see Sec. V-A), which helped to reduce the routing congestion. Regarding SAT-based attacks, we observed that this flavor provides good resilience.

In short, we select the flavor exemplified in Fig. 4 for the remainder of our study, as we found empirically that it offers the best security-cost trade-off [1].

Since our concept is based on camouflaging at the BEOL, it can be readily applied in conjunction with split manufacturing (Sec. II-A). To that end, we tailor our camouflaging primitive initially for higher metal layers, namely for M5 and M6 (Sec. V-C). We do so because split manufacturing is considered more practical for higher layers [41], [54], as those layers have rather large pitches, which are relatively easy and cheap to manufacture by the BEOL facility. We also explore our camouflaging primitive when tailored for lower metal layers (i.e., M3 and M4 in this work), and we support both types of primitives holistically in our flow (Sec. V-B).

IV. THREAT MODEL

We assume both the end-users and the fab to be untrusted. The latter is in direct contrast to the prior art on camouflaging that has to trust the fab because of their FEOL-centric techniques. To hinder fab adversaries, i.e., to protect the secret dummy/true vias and wires in the BEOL layers, we leverage split manufacturing; therefore, we require a trusted BEOL facility[4] The goal of malicious end-users is to reverse engineer the chip’s layout and identify its secret mappings in the BEOL layers—recall that the latter is challenging and yet to be demonstrated (Sec. II-C). Ultimately, both malicious end-users and fab adversaries want to reconstruct the original netlist and its IP. To that end, we assume that end-users employ SAT-based attacks, whereas fab workers launch proximity attacks. We evaluate both threat scenarios in Sec. VI-C and VI-D.

We assume that malicious end-users can get hold of working chips, which are used (i) as oracle for iterative SAT-based attack, and (ii) for reverse engineering of the chip’s layout (as required to model the obfuscated netlist as an SAT problem), but end-users cannot directly infer the secret BEOL mappings. We assume that malicious fab workers have access to the FEOL layers, but not to the BEOL layers. We further assume that both end-users and fab workers have the complete knowledge of our scheme, the used design tools, etc. However, the adversaries cannot, by nature, reproduce the randomized selection steps taken throughout our scheme.

[4] In a weaker threat model where the fab is trusted, our scheme and its primitives can still be applied as is, just without need for split manufacturing.
V. CAMOUFLAGING METHODOLOGY

A. Protecting Fixed Values and “Implausible Functionalities”

For large-scale camouflaging using our primitive (Fig. 5), we note that the ubiquitous wires relating to fixed values 0 and 1 may give away clues to an attacker. Such fixed-value wires would be driven by distinct, easy-to-identify TIE cells, and such cells and wires are typically used only selectively, e.g., for special registers or “hardware feature flags.” Therefore, an attacker observing a vast number of fixed-value wires might rightfully assume that these wires have been introduced only for obfuscation, and could try to disregard them. Besides, we have to address another challenge at once: a mindful attacker may also try to rule out all the “implausible functionalities” which are those beyond any gate’s original functionality, i.e., inverter, buffer, and TIE cells, from the search space. Since these functionalities arise only due to the fixed values being part of the obfuscated inputs, they can only become effective once the fixed values are made an essential part of the design, and vice versa.

In short, the fixed-value wires have to be rendered essential parts of the layout, while also protecting all the “implausible functionalities” of gates. To do so, we perform netlist transformations at the beginning of our methodology as follows:

1) We transform selected inverters (INVs) and buffers (BUFs) into camouflaged gates of other types (e.g., see Fig. 5). Nowadays, around 50% of all gates are repeaters using INV/BUF [55], offering ample opportunities for related transformations. One can freely choose (i) the number of INVs/BUFs to transform and (ii) the type of gate to transform them into following Boolean algebra. For a standard library, for example, INV (BUF) can be implemented using NAND (AND), NOR (OR), as well as both XOR and XNOR, all while using the fixed-value wires as one of the real inputs for those gates. This step renders the fixed-value wires essential—they cannot be ignored anymore without misinterpreting those transformed, camouflaged gates and, therefore, without misinterpreting the whole design. We also note that the “implausible functionalities” INV and BUF are now rendered plausible for the design and, independently, an attacker cannot simply infer a direct relationship between any of the camouflaged gates and their true functionality.

2) We insert additional gates of randomly selected types into randomly selected regions of whitespace. These additional gates are camouflaged as well, with their real inputs tied to fixed-value wires. These gates act as “TIE cells in disguise” and, in turn, they seemingly “drive” other camouflaged gates, which are randomly selected from the vicinity. Therefore, also the “implausible functionalities” 0 and 1 cannot be ruled out anymore without misinterpreting the design. As with the transformation of INV/BUF above, leveraging randomness here is essential to hinder attackers from simply inferring these camouflaged gates.

B. CAD Flow

Here we provide an overview of our methodology (Fig. 6), which can be easily integrated into any design flow. In this work, we implement our methodology for Cadence Innovus.

Given a netlist, we initially synthesize, place, and route the design. On this original layout, we then apply our netlist transformations outlined above, along with the insertion of TIE cells. Recall that, for our scheme, we define camouflaging scale, or LC scale for short, as the ratio of obfuscated nets over all the nets. Also recall that this is directly comparable to the ratio of camouflaged gates over all the gates, as we always obfuscate all the input nets of any gate selected for camouflaging. Next, depending on the camouflaging scale, we prepare for camouflaging in lower and higher metal layers, e.g., 30% camouflaging in lower layers and 70% camouflaging in higher layers for an camouflaging scale of 100% overall. That so-called metal-layer ratio for lower/higher layers and the camouflaging scale are provided by the user. Then we randomly assign nets to be camouflaged at lower/higher layers, subject to these two parameters.

Next, we insert and connect individual obfuscating cells for all the inputs of each gate to be camouflaged—these cells are essentially implementing our camouflaging primitives. It is essential to understand that these customized obfuscating cells do not impact the FEOL layers—their sole purpose is to enable the routing of all dummy and real wires (Fig. 7). Hence, the physical design of these customized cells is tailored for routability, while their arrangement remains flexible and unconfined concerning the already placed standard cells. See Sec. V-C for more details on the physical design, and see Fig. 8 for a layout snapshot of multiple obfuscating cells in M5/M6.

As already indicated in Sec. III, we have to choose dummy nets carefully such that they are unique concerning each gate to be camouflaged. 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
layers, which may also be more economical than enlarging die outlines [41].

At the same time, we advocate to enlarge the die outline carefully to gain more routing resources. That is because the related dummy wires can also incur additional wiring throughout the lower layers. Second, we note that obfuscating cells and their vias inserted into the lower layers tend to require even more resources. Regarding (ii), in case congestion remains after exploring various metal-layer combinations, which may have resulted during that process and re-select dummy nets as required.

After embedding and connecting the obfuscating cells, we perform trial routing—now with all the dummy wires along with the real wires—and we evaluate the congestion. In case of excessive congestion, the user is tasked to revise (i) the metal-layer ratio and, if need be, (ii) the die outline. Regarding (i), we advocate to revise it toward more utilization of higher layers and less utilization of lower layers for the following reasons. First, we note that lower layers tend to become more utilized in any case, even when obfuscating cells are assigned to higher layers. That is because the related dummy wires can also incur additional wiring throughout the lower layers. Second, we note that obfuscating cells and their vias inserted into the lower layers tend to require even more resources. Regarding (ii), in case congestion remains after exploring various metal-layer ratios, we advocate to enlarge the die outline carefully to gain more routing resources.

Once we obtain an uncongested solution, we perform legalization and ECO optimization; the latter is also based on custom constraint rules, see Sec. V-C. At the same time, we also re-route the design. Drivers are automatically up-scaled as needed to effectively/seemingly drive all the additional true/dummy wiring and the increased fanouts; see also Sec. V-C. We perform final design closure, remove the obfuscating cells from the design, re-extract the RC data, stream out the GDSII data, and gather the final APD/PPA numbers. Note that, after removal of the obfuscating cells, all the true/dummy vias and related wiring remain as such in place of those cells. Thus, the final APD/PPA numbers represent the nature of the camouflage layout accurately.

C. Physical Design of Obfuscating Cells

We implement the obfuscating cells as customized cells, which only pertain to the BEOL. To that end, and without loss of generality, we extend the LIB/LEF files of the NanGate 45nm Open Cell Library [43]. Next, we elaborate on the crucial aspects of the cells’ physical design. Besides the design described here for metal layers M5/M6, we also implemented obfuscating cells for M3/M4 in a similar manner.

1) The cells have four input and one output pin (Fig. 7(b)). These 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 camouflaging scales beyond 50%.

2) The dimensions of the pins (0.14 × 0.14µm) and their offsets are chosen such that the pins can be placed directly on the respective layer’s tracks (thin yellow grid in Fig. 7(b)); this is to minimize routing congestion further. Also, note the cell’s grey outline beneath the pins; the pins are partially located outside the outline, which is supported. The minimal width is solely to ease its visual differentiation from standard cells at design time.

3) We define custom constraint rules which prevent the pins of different obfuscating cells from overlapping during legalization. These obfuscating cells can, however, freely overlap with any standard cell without inducing routing conflicts. This is because standard cells have their pins exclusively in the lower metal layers. Therefore, we can camouflage the entire design without blocking any valuable standard-cell area.

4) We leverage the timing and power characteristics of BUFX2, i.e., a buffer with a driving strength of 2. Note that a detailed library characterization is not required as obfuscating cells implement BEOL wires and vias only. However, since obfuscating cells are essentially interposed within the paths, a capacitance annotation of subsequent loads is crucial (Fig. 9). Without such annotation, the respective driver might become undersized, as
TABLE IV
Runtime, in Seconds, for Our Camouflaging CAD Procedures

| Benchmark | Original | 20% LC Scale | 40% LC Scale | 60% LC Scale | 80% LC Scale | 100% LC Scale |
|-----------|----------|--------------|--------------|--------------|--------------|---------------|
| aes_core  | 498      | 1.323        | 2.469        | 2.102        | 3.075        | 4.019         |
| b14_C     | 178      | 217          | 258          | 313          | 403          | 527           |
| b15_C     | 256      | 316          | 376          | 445          | 514          | 639           |
| b17_C     | 603      | 901          | 2.497        | 2.640        | 2.930        | 4.378         |
| b22_C     | 428      | 573          | 1.105        | 1.771        | 2.221        | 3.652         |
| diff_eq   | 530      | 819          | 1.576        | 2.733        | 3.439        | 6.256         |
| square    | 451      | 787          | 1.964        | 2.118        | 4.606        | 5.343         |
| Average   | 420.57   | 705.14       | 1,463.57     | 1,731.71     | 2,385.43     | 3,544.86      |

Benchmarks in italics are from the EPFL suite [56], others are from the ITC-99 suite.

VI. EXPERIMENTAL INVESTIGATION

A. Setup

Setup for physical design: We implement our methodology as custom TCL scripts for Cadence Innovus 16.15. Our procedures incur some runtime costs, as presented in Table IV.

We employ the public NanGate 45nm Open Cell Library [43] with ten metal layers. Synthesis is performed using Synopsys Design Compiler (DC) for the slow process corner. The APD analysis is carried out for 0.95V, 125°C, and the slow process corner, along with a default input switching activity of 0.2—this is a rather pessimistic setup, providing conservative results with “safety margins” and “room for lowering cost.” We obtain power and timing results using Innovus. We configure the initial utilization rates, i.e., for the original layouts, such that the routing congestion remains close to 0%.

Recall that we implement the obfuscation cells as customized cells in two versions, one for higher layers in M5/M6, and one for lower layers in M3/M4. Unless stated otherwise, results are obtained with obfuscating cells embedded exclusively in M5/M6. For experiments with both higher and lower layers being used, we start with only lower layers being utilized, and revise that ratio, without loss of generality and for the sake of simplicity, in steps of 1% toward higher layers.

Setup for security evaluation: We model ours as well as all camouflaging techniques proposed in [7]–[9], [11], as outlined in [16]. More specifically, we model ours as individual 4-to-1 MUXes for all the inputs of each gate to be camouflaged, with all the related dummy/true wires connected to the MUXes’ inputs and with two select/key signals. For the netlist transformation steps explained in Sec. VA, we select 50% of all INV/BUFs randomly and transform them into various camouflaged gates randomly and the remaining INVs/BUFs are camouflaged as is; both selections are also subject to the overall camouflaging scale.

For a fair evaluation, the same sets of gates are camouflaged across all camouflaging techniques: for a given benchmark, gates are randomly selected once and then memorized [9]. Ten different sets are generated for each benchmark, ranging from 10% to 100% camouflaging scale, in steps of 10%.

For the scenario of malicious end-users, we evaluate our scheme against powerful SAT-based attacks [15], [21], [31], [32] in Sec. VI-C. We note that these attacks were developed for logic locking but are still applicable for our study—logic locking and camouflaging are closely related and can be transformed into one another [16]. We attribute the average runtime of the attacks as an empirical, yet essential, indicator for a design’s resilience; see also the discussion on the notion of practically secure camouflaging in Sec. VI-C.

For the scenario of malicious fab employees, we evaluate our scheme against the powerful open-source proximity attacks [33], [35] and [34], [36] in Sec. VI-D. We conduct experiments for layouts being split after M3 and M4, respectively, which helps us to evaluate the significance of the split layer for our scheme. For [33], [35], we leverage the commonly applied metrics of correct connection rate (CCR), i.e., the ratio of correctly inferred BEOL connections over total BEOL connections, and Hamming distance (HD) and output error rate (OER) [41]; the latter both quantify the functional correctness of the inferred netlist. HD and OER are computed using Synopsys VCS using 100,000 patterns. For [34], [36], since it does not provide any actual netlists, these metrics are not applicable; instead, we leverage the metrics defined in [36], namely virtual pins (vpins), E[LS], and figure of merit (FOM) [7].

All attacks are executed on a high-performance computing (HPC) facility. Each computational node has two 14-core Intel Broadwell processors (Xeon E5-2680), running at 2.4 GHz. Each node has 128 GB RAM in total, and 4 GB RAM are guaranteed (by the Slurm HPC scheduler) for each attack run. The CPU time-out (labeled as “t-o”) is set to 48 hours.

Benchmarks: We conduct extensive experiments on traditional benchmarks suites (ISCAS-85 and ITC-99) and, for the first time, also on the large-scale EPFL suite [56] (using the original circuits, not the MIG versions). Note that all benchmarks are combinatorial, but our approach can be directly applied to sequential designs as well. We employ custom scripts to convert (i) the original netlists to bench format with two-input gates, as required for the SAT-based attack [15], [31], and (ii) the post-layout netlists to rt and out format, as required for the proximity attack [34], [36]. We release the scripts for (ii) and other related materials in [57].

Any other technique such as maximum clique [7] could be applied here as well. However, El Massad et al. [44] observed that maximum-clique selection, on average, cannot offer a better resilience than random selection.

The term vpin or virtual pin refers to the top-most, open, or “dangling” end of a FEOL wire, which is a hint for an adversary that a via would be placed there to connect the FEOL with the undisclosed BEOL. By definition, [34], [59] considers only vpins related to two-pin nets. E[LS] is the number of candidate vpins to match with other vpins over a specific region, i.e., the number of possible pairings in the vicinity of some missing BEOL connection. FOM represents the number of possible pairings normalized over the area of the specific region considered. Thus, when applied step wise across the FEOL layout and averaged, FOM serves to quantify the complexity of exploring all possible BEOL connections. Since the attack [34], [59] does not recover any actual netlists, these metrics are essentially indicators for the complexity imposed on subsequent, yet-to-be-proposed attack steps to infer such netlists.
Layout cost are in percentage. Obfuscating cells are embedded exclusively in M5/M6. Benchmarks in italics are from the EPFL suite [56], others are from the ITC-99 suite. All layouts are DRC-clean and devoid of congestion.

### Table V

| Benchmark | Utilization for Original Layout | 20% LC Scale | 40% LC Scale | 60% LC Scale | 80% LC Scale | 100% LC Scale |
|-----------|---------------------------------|--------------|--------------|--------------|--------------|--------------|
|           | Area | Power | Delay | Area | Power | Delay | Area | Power | Delay | Area | Power | Delay |
| aes_core  | 0.5  | 0 | 9.02 | 0 | 13.39 | 27.46 | 0 | 14.29 | 15.44 | 29.72 | 0 | 25 | 20.22 | 30.94 | 33.33 | 27.76 | 35.72 |
| b14_C    | 0.5  | 0 | 6.43 | 0 | 9.88 | 5.01 | 0 | 11.11 | 16.92 | 9.75 | 19.04 | 21.39 | 8.74 | 19.04 | 38.46 | 14.76 |
| b15_C    | 0.5  | 0 | 10.78 | 0 | 11.19 | 10.14 | 0 | 11.11 | 14.39 | 11.45 | 25 | 17.27 | 15.74 | 25 | 28.78 | 20.77 |
| b17_C    | 0.5  | 11.11 | 7.16 | 7.19 | 19.04 | 11.13 | 11.22 | 25 | 16.93 | 14.64 | 42.85 | 17.14 | 14.21 | 42.85 | 28.29 | 16.92 |
| b22_C    | 0.5  | 0 | 7.24 | 11.41 | 0 | 13.11 | 14.37 | 0 | 11.11 | 15.11 | 14.81 | 19.04 | 25.27 | 15.11 | 25 | 37.48 | 22.41 |
| diff1    | 0.5  | 0 | 5.78 | 3.61 | 0 | 10.64 | 6.54 | 0 | 11.11 | 15.87 | 8.49 | 11.11 | 24.19 | 16.79 | 25 | 31.23 | 28.75 |
| square   | 0.5  | 0 | 8.19 | 7.14 | 0 | 11.11 | 9.98 | 13.06 | 0 | 11.11 | 13.21 | 11.47 | 19.04 | 18.3 | 16.42 | 25 | 25.15 | 22.45 |

Average 0.5 1.59 7.8 8.83 4.31 11.35 12.54 13.55 15.41 14.33 23.01 20.54 16.85 27.89 31.02 23.11

We first report in detail on APD cost for our scheme, and we then compare ours to various prior works. Figure 10 illustrates a step-wise up-scaling of die outlines as needed; see text below.

### B. Layout Evaluation

We first report in detail on APD cost for our scheme, and we then compare ours to various prior works. Figure 10 illustrates a step-wise up-scaling of die outlines as needed; see text below.

Fig. 10. Layout cost for EPFL benchmark aes_core [55]. The baseline is the original, unprotected layout. The discrete and monotonous area cost is due to a step-wise up-scaling of die outlines as needed; see text below.

On die area: Recall that our obfuscating cells impact only the metal layers; they do not increase the standard-cell area. Since wiring cannot be expressed by standard-cell area, the reported area cost is concerning die outlines as explained next.

Especially for large-scale camouflaging, with the additional wiring imposed by our scheme requiring significant routing resources, we can observe routing congestion and DRC issues. Recall that we first revise the metal-layer ratio, e.g., see Fig. 11 and, if need be, scale-up die outlines. In other words, we ultimately maintain that our large-scale camouflaged layouts are “DRC-clean” at the cost of larger outlines. For simplicity, we scale-up the outlines by decreasing the utilization rate in steps of 0.05 and 0.02, as appropriate. For example, while relaxing the utilization from 0.5 to 0.45, the die area has to be increased by 11.11%.

The average area overheads in Table V are not more than 14% for up to 60% camouflaging scale, whereas for full-chip camouflaging, we note an average area cost close to 28%. Again, these overheads enable DRC-clean layouts even for full-chip camouflaging; we believe that this is a justifiable cost.

Finally, while the transformation of 50% of INV/BUF gates to the camouflaging scale (LC) is, in general, a fair comparison, the same die outlines are considered for both the $H$ and $L&H$ setups. Note that the cost for the $H$ setup are the same as in Table VI.

Recall that our obfuscating cells impact only the metal layers; they do not increase the standard-cell area. Since wiring cannot be expressed by standard-cell area, the reported area cost is concerning die outlines as explained next.

Especially for large-scale camouflaging, with the additional wiring imposed by our scheme requiring significant routing resources, we can observe routing congestion and DRC issues. Recall that we first revise the metal-layer ratio, e.g., see Fig. 11 and, if need be, scale-up die outlines. In other words, we ultimately maintain that our large-scale camouflaged layouts are “DRC-clean” at the cost of larger outlines. For simplicity, we scale-up the outlines by decreasing the utilization rate in steps of 0.05 and 0.02, as appropriate. For example, while relaxing the utilization from 0.5 to 0.45, the die area has to be increased by 11.11%.

Fig. 11. Innovus snapshots for full-chip camouflaging of ITC-99 b15.C. For visibility, colors are inverted. (a) Camouflaging is implemented as 30%/70% in lower/higher layers, resulting in around 10,000 DRC issues (marked by crosses). (b) For the same die outline, camouflaging is implemented as 20%/80% in lower/higher layers, resulting in a DRC-clean layout. For (a), total over-congestion is reported as 10.4%, with worst gcell over-congestion of 12.71%, whereas for (b), these metrics are 0.19% and 0.43%.
On power and performance: As our camouflaging primitive incurs additional wiring, which becomes significant for large-scale camouflaging, an impact on power and performance is expected. See also Fig. [12] for wiring snapshots across camouflaging ranges.

For camouflaging up until 60% of the layout, the average power and delay overheads are moderate for most benchmarks, around 6–17% for power and 4–15% for delay (Table [V]). Only for EPFL benchmark aes_core, delays are consistently larger, by approximately 15%. The overheads increase naturally toward full-chip camouflaging, but still follow linear growth trends. We believe that this is due to the following:

1) Besides the transformed INV/BUF gates, all gates remain as is. Even for these transformed gates, cost are limited: for experiments on varying ranges of INVs/BUFS transformed for full-chip camouflaging of the large-scale EPFL benchmark aes_core [50], we observed variations for power and performance cost of less than 9%. That is, we experience no inherent overheads for the majority of gates and marginal overheads for transformed gates.

2) The lower resistance of higher metal layers, which are typically leveraged by most of the obfuscating-cell instances, helps to limit the delay cost.

3) The nets for fixed-values 0 and 1 are not switching and, thus, they neither increase power nor delay. Also, while dummy nets are switching, they drive only wiring loads.

4) For large-scale camouflaging, the positive effects are eventually offset by a steady increase of dummy nets and wiring for camouflaged gates, thereby raising the capacitive loads and power consumption as well as routing congestion. Since congestion can only be managed by rerouting in some detours, this lengthens parts of the wires further. In turn, this also impacts power and delay.

Regarding the use of obfuscating cells in both lower and higher metal layers (Table [VI]), we observe the following. First, utilizing lower and higher layers at once (L&H setup) allows to reduce cost, especially for power—in contrast to utilizing only the higher layers (H setup), we observe a reduction of 7.95% and 5.52% in power for the ITC-99 benchmarks b14_c and b15_c, respectively, and the delay reduction is 1.36% and 3.73% for the same benchmarks. To explain these trends, we also determine the load capacitance overheads for both setups: for the H setup, the capacitance overhead is about 62.78%, whereas this overhead is reduced to 29.99% for the L&H setup. Second, for large-scale camouflaging, the usability of lower layers becomes less; recall the related discussion in Sec. [V-B]. Finally, recall that splitting at higher layers is considered beneficial [41], [54], but here we cannot split above the lowest layer containing some camouflaging primitives. Therefore, while the holistic use of lower and higher layers can help to reduce layout cost, it limits the practicality of split manufacturing. For a weaker threat model with the fab being trusted, however, this approach would become more relevant.

Comparison at RTL level and for small-scale camouflaging: Previous works typically report their cost only for small-scale camouflaging and based on RTL simulations, which seems impractical. For example, Rajendran et al. [7] observe ≈60%, ≈40%, and ≈30% increase in area, power, and delay, respectively, when camouflaging 5% of gates in the ISCAS-85 benchmark e7552. The approach of Wang et al. [8] exhibits 50% delay and 15% area overhead already for 5% camouflaging scale. For another, more relevant example, Chen et al. [27] report 3.68% delay and 3.93% area overhead on average, i.e., when configuring their scheme for 3% power budget/overhead and while obfuscating only tens to few hundreds of wires for the ITC-99 benchmarks.

Comparison on physical-layout level: Recall that our work is one of the very few to evaluate camouflaging on placed-and-routed GDSII layouts. When contrasting to a previous study, conducted by Malik et al. [58], we notice significantly higher overheads for them. We note that Malik et al. implement and evaluate their approach exclusively for one AES S-box, which has a far lower number of gates, namely 421, when compared to all the benchmarks we consider. However, we argue that a qualitative comparison is still fair as Malik et al. use the same NanGate library [43]. They reported overheads of 7.09×, 6.45×, and 3.12× for area, power, and delay (APD), respectively, for their case study. Moreover, Malik et al. indicate themselves that cost will increase for larger circuits.

Besides a provably secure scheme, Li et al. [17] propose two “regular” camouflaging primitives (based on obfuscated contacts), called STF-type and XOR-type. As Li et al. report only gate-level cost for their primitives, we conduct a layout-level comparison for full-chip camouflaging ourselves (Table [VII]).
Here we first constrain synthesis accordingly (recall Sec. II-B), and then we map their gate-level cost to all the gates—this extrapolation is conservative as it implies only a linear scaling even for full-chip camouflaging. While the STF-type primitive becomes a contender to ours in terms of power and delays, it still incurs 23.88% larger die-area cost on average.

For Zhang’s work [9], as its MUX-based primitive is not made available, we implement it ourselves, by using MUXes and INVs for the primitives, and primary inputs for their key bits, and we perform a detailed layout-level evaluation. For example, we observe 464%, 638%, and 63% increase for APD, bits, and we perform a detailed layout-level evaluation. First, we conduct a layout-level analysis for all the camouflaged gate instances. Again, while this is an extrapolating approach, we argue that it is fair, as we reflect the different overheads for different types of gates camouflaged. We find that the approach by Nirmala et al. will incur significant APD cost for full-chip camouflaging: \( \approx 1.360\% \), \( \approx 1.266\% \), and \( \approx 100\% \), respectively. For the work of Collantes et al. [10], we observe a linear trend in the reported layout cost; simply extrapolating those numbers for full-chip camouflaging would translate to \( \approx 78\% \) and \( \approx 147\% \) for power and delay, respectively. Finally, for the work of Erbagci et al. [12], we quote their numbers in Table VII.

**Comparison with provably secure camouflage:** Recall that schemes like [17]–[19] rely on additional circuitry to protect individual, selected gates, wires, or outputs. Such circuitry can incur a high cost, especially for area and power. Using the benchmarks provided by Amir et al. [59] on trust-hub.org, we conduct a layout-level analysis for Anti-SAT [19], namely for Anti-SAT with random logic locking (RLL) and for Anti-SAT with secure logic locking (SLL) [60]. We perform a regular design flow here, using the Nangate 45nm library [43] at the slow corner and with timing constrained to 4ns. When comparing these schemes to our full-chip camouflaging (Table VIII), we observe that average overheads are notably larger, namely by 8.95× and 9.59× for area and by 23.66× and 26.05× for power, respectively; only delay overheads are less. We note that these cost arise for protecting single outputs and increase further if we would protect more outputs.

In any case, we emphasize again that our notion of full-chip camouflaging is different from such schemes (including functional obfuscation [22] discussed next), see also on the notion of practically secure camouflage further below.

**Comparison with functional obfuscation:** Xu et al. [22] proposed binary decision diagram (BDD)-based obfuscation which operates at the functional level, in contrast to other approaches working at the netlist level. It was shown that their scheme can be tailored to remain resilient against SAT-based attacks as well “removal attacks” [22], [23]. As above, we evaluate two approaches of [22] at layout level, namely random obfuscation and Anti-SAT-inspired obfuscation [59]. Again, expect for delay, we observe that average overheads are significantly larger, namely by 92.39× and 76.57× for area and 197.1× and 246.7× for power, respectively (Table IX).

Table VII

| Benchmark | XOR-type [17] | STF-type [17] | Threshold [12] | Ours |
|-----------|---------------|---------------|----------------|------|
| A | P | D | A | P | D | A | P | D |
| c5315 | 92.33 | 33.81 | 30.56 | 62.18 | 38.37 | 14.79 | 200 | 10 | 76 |
| c7552 | 24.10 | 31.96 | 40.64 | 62.5 | 54.32 | 43.86 | 200 | 10 | 76 |
| c1355 | 24.10 | 31.96 | 40.64 | 62.5 | 54.32 | 43.86 | 200 | 10 | 76 |

Table VIII

| Benchmark | Anti-SAT-RLL | Anti-SAT-SLL | Ours |
|-----------|--------------|--------------|------|
| A | P | D | A | P | D | A | P | D |
| c880 | 5.168 | 6.291 | 9.84 | 5.289 | 6.359 | 9.34 | 27.22 | 23.28 | 19.63 |
| c1908 | 5.727 | 5.973 | 9.15 | 5.289 | 6.359 | 9.34 | 27.22 | 23.28 | 19.63 |
| c3540 | 2.803 | 2.113 | 1.76 | 2.807 | 2.119 | 1.75 | 50 | 10.34 | 10.24 |
| c5315 | 76.89 | 45.49 | 7.26 | 51.63 | 39.59 | 6.67 | 50 | 10.19 | 9.94 |
| c7552 | 109.30 | 63.49 | 0.81 | 133.57 | 97.04 | -0.33 | 33.33 | 17.01 | 14.09 |

Table IX

| Benchmark | Random Obfus. | Anti-SAT Obfus. | Ours |
|-----------|--------------|----------------|------|
| A | P | D | A | P | D | A | P | D |
| c880 | 5.168 | 6.291 | 9.84 | 5.289 | 6.359 | 9.34 | 27.22 | 23.28 | 19.63 |
| c1908 | 5.727 | 5.973 | 9.15 | 5.289 | 6.359 | 9.34 | 27.22 | 23.28 | 19.63 |
| c3540 | 2.803 | 2.113 | 1.76 | 2.807 | 2.119 | 1.75 | 50 | 10.34 | 10.24 |
| c5315 | 76.89 | 45.49 | 7.26 | 51.63 | 39.59 | 6.67 | 50 | 10.19 | 9.94 |
| c7552 | 109.30 | 63.49 | 0.81 | 133.57 | 97.04 | -0.33 | 33.33 | 17.01 | 14.09 |

| Average | 3.706 | 4.040 | 9.47 | 3.702 | 4.028 | 9.41 | 40.12 | 16.38 | 14.07 |
C. Security Evaluation: Malicious End-Users, SAT-Based Attacks

On the notion of practically secure camouflage: Recall that the primary objective for this work is large-scale camouflage, and an important observation is that this achieves practically secure camouflage. That is, by camouflage up to 100% of the layout, we can induce significant computational efforts for SAT-based attacks, without leveraging additional provably secure structures.

Still, it is not straightforward to prove beforehand to what extent large-scale camouflage will render a layout resilient without actually leveraging a SAT solver’s capabilities for de-camouflage attacks. Li et al. [17] have shown that de-camouflage efforts scale on average with (i) the solution space $C$ concerning all possible functionalities of the whole design, and (ii) the Hamming distances among those different functionalities. A further theoretical evaluation of large-scale camouflage is difficult since $C$ depends on (i) the number and composition of possible functionalities supported by the camouflage primitives, (ii) the number of camouflaged gates, (iii) the selection of camouflaged gates, and (iv) the interconnectivity of the design, all at the same time. For example, designs containing XOR/XNOR and/or multipliers are harder to de-camouflage in practice than other designs [15], [16].

In short, while one can easily estimate the upper bound of $C$, this may not reflect on the actual efforts required for successful attacks. Hence, we next resort to an empirical but comprehensive study. As we observe in this study, we can indeed expect prohibitive runtimes for large-scale camouflage. More specifically, we observe a polynomial trend for attacking the benchmark aes_core, setting in around 14% camouflage (Fig. 13). Similar observations have also been made by Yu et al. [16], albeit for a different primitive and the much smaller benchmark c432 with 209 two-input gates; even such a small layout could not be resolved within three days.

Comparison with prior art on large-scale camouflage: In Table [X] we list runtimes for the seminal SAT-based attack [15], [31], evaluating our camouflage primitive and those of [7], [9], [11]. Note that these prior studies did not report on any SAT-based attacks for large-scale camouflage themselves. Also, recall that we camouflage the same sets of gates across all techniques for a fair comparison for each run, and that we report average runtimes over ten such runs.

For our scheme, we do not observe any significant differences for the average SAT-based attack runtimes when considering different ranges of INVs/BUFs being transformed (Sec. V-A). This is because, for cases with less INVs/BUFs being selected for transformation, more other gates are selected for camouflage, while the overall count of camouflaged gates remains the same; recall that we use the camouflage scaling as design-wide “knob” for all of our experiments.

We note that none of the layouts can be de-camouflaged after 48 hours once full-chip camouflage is applied. In fact, almost all layouts remain already resilient beyond 40% camouflage scale; the only exception arises for the ISCAS-85 benchmark c7552 when the most simple camouflage primitive [9] is leveraged. We ran further exploratory attacks for 7 days on large-scale camouflage using our primitive—without observing any improvement for the attack. For a more meaningful comparison, we also consider relatively small camouflage scales, i.e., 10% up to 30%. Here, the primitive by Zhang et al. [9] appears as the weakest and that by Wang et al. [8] as the most resilient. Our primitive is next only to those of [8], [11]. However, it is also important to recall that ours incurs significantly less APD cost (Sec. VI-B).

We also leverage the Double DIP attack, provided by Shen et al. [21], [32]. The crux of their attack is that it can rule out at least two incorrect keys during the application of one single distinguishing input pattern (DIP) for the oracle. While conducting the same set of experiments as above, we observe that the average runtimes are even higher across all benchmarks and camouflage schemes (Table [XI]). For example, for the ITC-99 benchmark b14_C with 10% camouflage applied using our primitive, we observe $\approx 137$ seconds for [31], but $\approx 402$ seconds for [32]. Thus, we conclude that Double DIP, while successful for attacking dedicated protection schemes such as SARLock cannot cope well with large-scale camouflage schemes in general.

Overall, we are not claiming that large-scale camouflage, whether based on prior art or our primitive, cannot be resolved eventually using SAT-based attacks. Rather, we provide strong empirical evidence that practically secure camouflage, i.e., once 50–100% of all gates are camouflaged, imposes prohibitive computational cost on SAT solvers.

On the (lack of) resilience for [27]: Since our work is inspired to some degree by that of Chen et al. [27], it seems imperative to investigate the resilience of their scheme as well, especially since Chen et al. did not consider any SAT-based attacks themselves. In Table [XII] we note that the SAT-based attack [15], [31] can distinguish between real and dummy interconnects within relatively short runtimes, no matter how many interconnects are obfuscated, i.e., at least concerning the delay constraints defined in [27]. That is, an overly constrained obfuscation as proposed in [27] cannot withstand powerful SAT-based attacks. Similar observations have been also reported by Yu et al. [16].

D. Security Evaluation: Fab Adversaries, Proximity Attacks

Similar to the observations made for SAT-based attacks, here we find that our scheme of obfuscating the interconnects—along with all the additional wiring going through both the FEOL and BEOL—serves also well to hinder proximity attacks. That is essentially due to vast numbers of wires being
TABLE X
AVERAGE RUNTIMES (IN SECONDS) FOR THE SEMINAL SAT-BASED ATTACK [15], [31] ON SELECTED ISCAS-85 AND ITC-99 BENCHMARKS

| Benchmark | 10% Camouflaging Scale | 20% Camouflaging Scale | Our | 10% Camouflaging Scale | 20% Camouflaging Scale |
|-----------|------------------------|------------------------|-----|------------------------|------------------------|
| b14_C     | 13.77                  | 36.12                  | 87.14 | 230.05                | 9,735.42               |
| b15_C     | 20.93                  | 77.24                  | 172.27 | 1,010.27          | 5,070.97               |
| b17_C     | 256.71                 | 1,174.39               | 2,103.08 | 23,406.4           | 1,580.73               |
| b20_C     | 56.35                  | 176.81                 | 323.81 | 1,909.9            | 8,755.17               |
| b22_C     | 121.25                 | 446.33                 | 1,128.02 | 6,289.34          | 25,814.6              |
| c3540     | 1.47                   | 3.32                   | 4.08   | 12.04              | 27.22                  |
| c5315     | 0.65                   | 1.85                   | 4.03   | 12.88              | 22.90                  |
| c7552     | 1.13                   | 21.39                  | 8.62   | 33.54              | 122.96                 |
|           |                        |                        |       |                     |                        |
| 30% Camouflaging Scale |                |                        |       |                     |                        |
| b14_C     | 115.78                 | 1,659.75               | 14,180.7 | 2,560.86          | 96,271.4              |
| b15_C     | 349.34                 | 2,481.25               | 9,165.07 | 759.22            | 24,442.9              |
| b17_C     | 3,046.33               | 68,028.1               | t-o    | t-o                | t-o                   |
| b20_C     | 2,406.48               | 6,027.25               | t-o    | t-o                | t-o                   |
| b22_C     | 3,754.7                | t-o                    | t-o    | t-o                | t-o                   |
| c3540     | 0.63                   | 35.25                  | 95.48  | 22,613.6         | 4,971.03              |
| c5315     | 5.88                   | 22.91                  | 87.42  | 323.62             | 4,971.03              |
| c7552     | 26.40                  | 91.07                  | 1,618.37 | 1,538.92       | 1,447.68              |
|           |                        |                        |       |                     |                        |
| 40% Camouflaging Scale and Beyond |                |                        |       |                     |                        |
| b14_C     | 115.78                 | 1,659.75               | 14,180.7 | 2,560.86          | 96,271.4              |
| b15_C     | 349.34                 | 2,481.25               | 9,165.07 | 759.22            | 24,442.9              |
| b17_C     | 3,046.33               | 68,028.1               | t-o    | t-o                | t-o                   |
| b20_C     | 2,406.48               | 6,027.25               | t-o    | t-o                | t-o                   |
| b22_C     | 3,754.7                | t-o                    | t-o    | t-o                | t-o                   |
| c3540     | 0.63                   | 35.25                  | 95.48  | 22,613.6         | 4,971.03              |
| c5315     | 5.88                   | 22.91                  | 87.42  | 323.62             | 4,971.03              |
| c7552     | 26.40                  | 91.07                  | 1,618.37 | 1,538.92       | 1,447.68              |

For a fair evaluation, the same sets of gates are camouflaged across all camouflaging techniques: for a given benchmark, gates are randomly selected once and then memorized. Ten such random sets are generated for each benchmark. Time-out (t-o) is 48 hours, i.e., 172,800 seconds. * Number of obfuscated functionalities we assume; refer to related publication for the actual sets of functionalities. + The runtime range reported for ISCAS-85 benchmark c7552 being obfuscated with c5315 is for 40–100% camouflaging scale. All other runtimes here correspond to 40% camouflaging scale; we found that 50% camouflaging scale occurs t-o for all other cases.

We execute the seminal network-flow attack [33], [35] on different layouts for major camouflaging scales and for two split layers, M3 and M4, on selected ISCAS-85 benchmarks (Table XIII). As indicated, we observe that our scheme is effective in mitigating this attack. The average CCR observed for original layouts is significantly reduced for fully camouflaged layouts, namely by 7.48× and 2.41× when split after M3 and M4, respectively. This significant reduction in CCR is corroborated by the increase of cut wires to handle the splitting at the split layer, creating a large solution space for the attacks to work on. Besides, recall that the metrics used in this section have been introduced in Sec VI-A.

Regarding the impact of the camouflaging scale and split layer, we observe the following. First, for both layers, there is a significant CCR reduction already for 20% camouflaging over original layouts, namely by ≈2.1×. This implies that a relatively small-scaled application of our scheme is sufficient to significantly weaken this attack [33], [35], and this holds true across split layers. Second, for M3, increasing the camouflaging scale helps to reduce the CCR notably further, whereas the CCR remains at ≈26.4% for M4 across camouflaging scales. This implies that our scheme is more effective when lower split layers are considered, which can be expected and has been discussed before in general, e.g., in [39].

We also study the quality (or rather, lack thereof) for the netlists recovered by [33], [35]. In Table XIV we report HD and OER values across camouflaging scales and split layers. The OER values reveal errors for all the netlists, irrespective of the camouflaging scale. Concerning HD, we observe many errors as well, already for 20% camouflaging scale, and for full-chip camouflaging, HD is approaching the ideal value (50%) for all the netlists recovered across both split layers. Therefore, aside from CCR, we also demonstrate that netlists recovered by [33], [35] deviate significantly in terms of functional behavior from the original layouts.

Finally, we also conduct the routing-congestion-aware attack crouting by Magaña et al. [34], [36] for the larger ITC-99 benchmarks (Table XV). We observe a significant increase of vpins once larger camouflaging scales are effected: across all benchmarks, on average, the increase is 12.55× for 100% camouflaging when compared to original, unprotected layouts. The number of candidates E[LS], which represents the possible pairings for each cut net to consider, increases accordingly as well. As a result, the attack complexity, represented by FOM, increases on average by 9.30× and 24.15× for split layers M3 and M4, respectively. Considering the impact of the camouflaging scale and split layer, we observe similar trends as above. That is, we note already for 20% camouflaging a significant increase in the FOM, for both split layers. Furthermore, our scheme remains more resilient at M3, while FOM for M4 increase well with the camouflaging scale here. Overall, we find that our scheme weakens the prospects of the crouting attack [34], [36] significantly.

VII. CONCLUSION

Given the manifold limitations of the prior art for camouflaging—regarding applicability, layout overheads, as well as resilience—we argue that new avenues are called for. Note that this argument also applies to provably secure schemes, where important fallacies have been demonstrated, namely algorithmic attacks tailored for identification and/or removal of the additional, dedicated circuitry required by such schemes, as well as their inherently low output corruptibility. Here, we promote the obfuscation of interconnects as such a
promising avenue going beyond the prior art, especially for efficient IP protection at large scales.

Toward this end, we propose and implement BEOL-centric camouflaging primitives which are applicable to any FEOL node, and integrate them within a commercial-grade CAD framework. We strive for practically relevant layout evaluation—all our camouflaged layouts are DRC-clean at the GDSII level, and we consider traditional but also modern, large-scale benchmark suites. We thoroughly contrast ours to the prior art, both in terms of layout cost and resilience. For the latter, we have leveraged powerful SAT-based attacks and proximity attacks. Our scheme hinders both threats ultimately by virtue of scale and entanglement. For SAT-based attacks, we show that tackling our large-scale camouflaging scheme becomes computationally too expensive; for proximity attacks, we show that our large-scale camouflaging scheme becomes computationally too expensive; for proximity attacks, we show that tackling our large-scale camouflaging scheme becomes computationally too expensive. That is especially true when also considering the protection offered against fab adversaries, enabled by split manufacturing applied in conjunction with our notion of obfuscating the interconnects.

As for future work, we will study in more detail the formal underpinnings for computational cost induced on algorithmic attacks by our scheme. We also plan to demonstrate that our scheme suffers neither from low output corruptibility, rendering it resilient against approximate SAT-based attacks like AppSAT [20], nor from removal attacks [22, 23], which is both in contrast to provably secure schemes. A straightforward intuition here is that our scheme intertwines camouflaging within the netlist at large scales, where any false inferences can propagate throughout the whole netlist, whereas provably secure schemes protect only particular patterns/outputs.

| Benchmark | N1 Time for N1 | N2 Time for N2 | N3 Time for N3 |
|-----------|----------------|----------------|----------------|
| b14_C     | 30             | 7              | 36             |
| b15_C     | 38             | 7              | 44             |
| b17_C     | 92             | 149            | 198            |
| b18_C     | 265            | 2,964          | 334            |
| b19_C     | 438            | 4,685          | 583            |
| b20_C     | 48             | 35             | 85             |
| b21_C     | 54             | 29             | 76             |
| b22_C     | 76             | 58             | 113            |

We model the attack as outlined in Fig. 3 of [15], i.e., by inserting 2-to-1 MUXes fed by real and dummy wires. Columns N1, N2, and N3 quote the number of dummy wires, while limiting the delay overheads to 0%, 3%, and 5%, respectively, as proposed in [27]. Since the exact locations of dummy wires and other details are not reported in [27], we insert N1-N3 wires randomly into the netlists available at our end. Time for N1-N3 denote the total runtime for attacking the related netlists.

### Table XII

| Benchmark | LC Scale | Original | 10% Camouflaging Scale | 20% Camouflaging Scale | 30% Camouflaging Scale | 40% Camouflaging Scale and Beyond |
|-----------|----------|----------|------------------------|------------------------|------------------------|----------------------------------|
| b14_C     | 30       | 55       | 11                     | 11                     | 11                     | 11                               |
| b15_C     | 38       | 44       | 8                      | 8                      | 8                      | 8                                |
| b17_C     | 92       | 198      | 170                    | 170                    | 170                    | 170                              |
| b18_C     | 265      | 334      | 212                    | 212                    | 212                    | 212                              |
| b19_C     | 438      | 583      | 359                    | 359                    | 359                    | 359                              |
| b20_C     | 48       | 85       | 46                     | 46                     | 46                     | 46                                |
| b21_C     | 54       | 76       | 56                     | 56                     | 56                     | 56                                |
| b22_C     | 76       | 113      | 79                     | 79                     | 79                     | 79                                |

We model the attack as outlined in Fig. 3 of [15], i.e., by inserting 2-to-1 MUXes fed by real and dummy wires. Columns N1, N2, and N3 quote the number of dummy wires, while limiting the delay overheads to 0%, 3%, and 5%, respectively, as proposed in [27]. Since the exact locations of dummy wires and other details are not reported in [27], we insert N1-N3 wires randomly into the netlists available at our end. Time for N1-N3 denote the total runtime for attacking the related netlists.

### Table XIII

| Benchmark | LC Scale | Split after M3 | Split after M4 |
|-----------|----------|----------------|---------------|
| b14_C     | 55       | 52             | 41            |
| b15_C     | 44       | 21             | 10            |
| b17_C     | 336      | 52             | 41            |
| b18_C     | 32       | 23             | 10            |
| b19_C     | 893      | 52             | 41            |
| b20_C     | 66       | 23             | 10            |
| b21_C     | 336      | 52             | 41            |
| b22_C     | 52       | 23             | 10            |

We model the attack as outlined in Fig. 3 of [15], i.e., by inserting 2-to-1 MUXes fed by real and dummy wires. Columns N1, N2, and N3 quote the number of dummy wires, while limiting the delay overheads to 0%, 3%, and 5%, respectively, as proposed in [27]. Since the exact locations of dummy wires and other details are not reported in [27], we insert N1-N3 wires randomly into the netlists available at our end. Time for N1-N3 denote the total runtime for attacking the related netlists.

### Table XIV

| Benchmark | LC Scale | Original | 10% Camouflaging Scale | 20% Camouflaging Scale | 30% Camouflaging Scale | 40% Camouflaging Scale and Beyond |
|-----------|----------|----------|------------------------|------------------------|------------------------|----------------------------------|
| c1908     | 55       | 52       | 41                     | 41                     | 41                     | 41                               |
| c3540     | 55       | 52       | 41                     | 41                     | 41                     | 41                               |
| c5315     | 55       | 52       | 41                     | 41                     | 41                     | 41                               |
| c7552     | 55       | 52       | 41                     | 41                     | 41                     | 41                               |

We model the attack as outlined in Fig. 3 of [15], i.e., by inserting 2-to-1 MUXes fed by real and dummy wires. Columns N1, N2, and N3 quote the number of dummy wires, while limiting the delay overheads to 0%, 3%, and 5%, respectively, as proposed in [27]. Since the exact locations of dummy wires and other details are not reported in [27], we insert N1-N3 wires randomly into the netlists available at our end. Time for N1-N3 denote the total runtime for attacking the related netlists.
TABLE XIV
HD, OER for netlists obtained by proximity attack [33, 35]

| Benchmark | LC Scale: 20% | 40% | 60% | 80% | 100% |
|-----------|---------------|-----|-----|-----|-------|
|           | HD OER | HD OER | HD OER | HD OER | HD OER |
| c1908     | 30.91 99.98 | 40.14 99.99 | 45.67 99.98 | 47.71 99.99 | 49.83 99.99 |
| c3540     | 42.74 99.99 | 44.59 99.99 | 45.86 99.99 | 46.97 99.99 | 49.79 99.99 |
| c5315     | 43.23 99.99 | 46.81 99.99 | 47.89 99.99 | 49.51 99.99 | 50.78 99.99 |
| c7552     | 45.62 99.99 | 47.55 99.99 | 49.04 99.99 | 47.78 99.99 | 48.73 99.99 |
| Average   | 42.63 99.99 | 44.76 99.99 | 47.11 99.99 | 48.67 99.99 | 49.79 99.99 |
|           | Split after M3 |     |     |     |       |
| c1908     | 35.49 99.99 | 38.49 99.99 | 43.89 99.99 | 46.31 99.99 | 47.43 99.99 |
| c3540     | 40.34 99.99 | 43.19 99.99 | 46.56 99.99 | 47.91 99.99 | 48.33 99.99 |
| c5315     | 42.87 99.99 | 43.87 99.99 | 45.91 99.99 | 47.17 99.99 | 48.78 99.99 |
| c7552     | 44.12 99.99 | 45.52 99.99 | 46.78 99.99 | 47.89 99.99 | 48.79 99.99 |
| Average   | 42.74 99.99 | 44.59 99.99 | 46.78 99.99 | 47.89 99.99 | 48.79 99.99 |

TABLE XV
RESULTS FOR CROWTING PROXIMITY ATTACK [34, 36] ON SELECTED ITC-99 BENCHMARKS

| Benchmark | LC Scale | Split after M3 | Split after M4 |
|-----------|----------|----------------|----------------|
|           | vpins ELS FOM | vpins ELS FOM | vpins ELS FOM |
| Original  | 224 1.78 0.04 | 25 1.32 0.01 | 25 1.32 0.01 |
| b14_C     | 114 10.3 0.27 | 950 12.36 0.26 | 12.36 0.26 |
| b15_C     | 1342 9.88 0.27 | 143 10.44 0.29 | 12.36 0.26 |
| b17_C     | 1508 9.58 0.25 | 68 10.32 0.27 | 12.36 0.26 |
| b20_C     | 4382 36.78 0.23 | 3130 40.65 0.26 | 3130 40.65 0.26 |
| b21_C     | 448 3.21 0.03 | 100 0.82 0.01 | 100 0.82 0.01 |
| b22_C     | 2714 28.19 0.3 | 2068 27.72 0.3 | 2068 27.72 0.3 |
|           | 4116 35.18 0.37 | 3600 33.17 0.35 | 3600 33.17 0.35 |
|           | 5386 24.45 0.39 | 5182 23.9 0.38 | 5182 23.9 0.38 |
|           | 6616 37.39 0.43 | 6738 37.23 0.43 | 6738 37.23 0.43 |
|           | 7880 51.61 0.43 | 7343 48.86 0.43 | 7343 48.86 0.43 |
|           | 12586 63.53 0.43 | 1234 63.69 0.33 | 1234 63.69 0.33 |
|           | 448 3.21 0.03 | 100 0.82 0.01 | 100 0.82 0.01 |
|           | 2714 28.19 0.3 | 2068 27.72 0.3 | 2068 27.72 0.3 |
|           | 4116 35.18 0.37 | 3600 33.17 0.35 | 3600 33.17 0.35 |
|           | 5386 24.45 0.39 | 5182 23.9 0.38 | 5182 23.9 0.38 |
|           | 6616 37.39 0.43 | 6738 37.23 0.43 | 6738 37.23 0.43 |
|           | 7880 51.61 0.43 | 7343 48.86 0.43 | 7343 48.86 0.43 |
|           | 12586 63.53 0.43 | 1234 63.69 0.33 | 1234 63.69 0.33 |

The metric vpin is number of open two-pin nets, ELS is the number of candidates over a specific region, and FOM is the figure of merit for attack complexity, as defined in [34] and explained in Section IV of [35] (N/A denotes attack failures. For a fair comparison across different benchmarks and layouts, we setup each run with a bounding-box size equal to 1/8 of the half perimeter of the respective die outlines.

REFERENCES
[1] S. Patnaik et al., “Obfuscating the interconnects: Low-cost and resilient full-chip layout camouflage,” in Proc. ICCAD, 2017, pp. 41–48.
[2] M. Rostami et al., “A primer on hardware security: Models, methods, and metrics,” Proc. IEEE, vol. 102, no. 8, pp. 1283–1295, 2014.
[3] M. Tehranipoor et al., Eds., Introduction to Hardware Security and Trust. Springer, 2014.
[4] M. Tehranipoor et al., “Reversing stealthy dopant-level circuits,” J. Cryptogr. Eng., vol. 5, no. 2, pp. 85–94, 2015.
[5] Y. Lao et al., “Obfuscating DSP circuits via high-level transformations,” Trans. VLSI Syst., vol. 23, no. 5, pp. 819–830, 2015.
[39] A. Sengupta et al., “Rethinking split manufacturing: An information-theoretic approach with secure layout techniques,” in Proc. ICCAD, 2017, pp. 329–336.

[40] L. Feng et al., “Making split fabrication synergistically secure and manufacturable,” in Proc. ICCAD, 2017.

[41] S. Patnaik et al., “Concerted wire lifting: Enabling secure and cost-effective split manufacturing,” in Proc. Asia South Pa. Des. Autom. Conf., 2018, pp. 251–258.

[42] ——, “Raise your game for split manufacturing: Restoring the true functionality through BEOL,” in Proc. Des. Autom. Conf., 2018, pp. 140:1–140:6.

[43] (2011) Nangate FreePDK45 Open Cell Library. Nangate Inc. [Online]. Available: http://www.nangate.com/?page_id=2325.

[44] M. Yasin et al., “Provably-secure logic locking: From theory to practice,” in Proc. Comp. Comm. Sec., 2017, pp. 1601–1618.

[45] S. Jiang et al., “An efficient technique to reverse engineer minterms protection based camouflaged circuit,” J. Comp. Sci. Tech., vol. 33, no. 5, pp. 998–1006, 2018.

[46] M. Yasin et al., “Security analysis of Anti-SAT,” in Proc. Asia South Pa. Des. Autom. Conf., 2017, pp. 342–347.

[47] C. Helfmeier et al., “Breaking and entering through the silicon,” in Proc. Comp. Comm. Sec., 2013, pp. 733–744.

[48] J. Swerts et al., “BEOL compatible high tunnel magnetoresistance perpendicular magnetic tunnel junctions using a sacrificial Mg layer as CoFeB free layer cap,” Appl. Phys. Lett., vol. 106, no. 26, pp. 26407:1–26407:4, 2015.

[49] ——, “Solving the BEOL compatibility challenge of top-pinned magnetic tunnel junction stacks,” in Proc. Int. Elect. Devices Meeting, 2017, pp. 38.6.1–38.6.4.

[50] S.-W. Hwang et al., “A physically transient form of silicon electronics,” Science, vol. 337, no. 6102, pp. 1640–1644, 2012.

[51] A. Todri-Sanial et al., Eds., Carbon Nanotubes for Interconnects. Springer, 2017.

[52] A. Naemi et al., “BEOL scaling limits and next generation technology prospects,” in Proc. Des. Autom. Conf., 2014, pp. 261–266.

[53] Confidential communication with industry partners, 2017.

[54] K. Xiao et al., “Efficient and secure split manufacturing via obfuscated built-in self-authentication,” in Proc. HOST, 2015, pp. 14–19.

[55] I. L. Markov, “Limits on fundamental limits to computation,” Nature, vol. 512, no. 7513, pp. 147–154, 2014.

[56] L. Amarú. (2015) Majority-inverter graph (MIG) benchmark suite. [Online]. Available: http://ls1.epfl.ch/MIG

[57] Tools for SM working on DEF files — DIX Lab, NYUAD. [Online]. Available: https://github.com/DIX-NYUAD/SM_DEF_tools

[58] S. Malik et al., “Development of a layout-level hardware obfuscation tool,” in Proc. Comp. Soc. Symp. VLSI, 2015, pp. 204–209.

[59] S. Amir et al., “Development and evaluation of hardware obfuscation benchmarks,” J. Hardw. Sec. Sys., vol. 2, no. 2, pp. 142–161, 2018.

[60] M. Yasin et al., “On improving the security of logic locking,” Trans. Comp.-Aided Des. Integ. Circ. Sys., vol. 35, no. 9, pp. 1411–1424, 2016.

Mohammed Ashraf is a Senior Physical Design engineer from India. He obtained his Bachelor's degree in electronics and telecommunication engineering from College of Engineering Trivandrum, Kerala, in 2005. He carries an experience of 10 years in the VLSI industry. He has worked with various multi-national companies like NVIDIA Graphics, Advanced Micro Devices (AMD), and Wipro Technologies. He worked also with Dubai Circuit Design, Dubai Silicon Oasis, UAE. Mr. Ashraf is currently a Research Engineer at Center for Cyber Security (CCS) at New York University Abu Dhabi.

Ozgur Sinanoglu (M’11–SM’15) is a Professor of Electrical and Computer Engineering at New York University Abu Dhabi. He earned his B.S. degrees, one in Electrical and Electronics Engineering and one in Computer Engineering, both from Bogazici University, Turkey in 1999. He obtained his MS and PhD in Computer Science and Engineering from University of California San Diego in 2001 and 2004 respectively. He has industry experience at TI, IBM and Qualcomm, and has been with NYU Abu Dhabi since 2010. During his PhD, he won the IBM PhD fellowship award twice. He is also the recipient of the best paper awards at IEEE VLSI Test Symposium 2011 and ACM Conference on Computer and Communication Security 2013.

Prof. Sinanoglu’s research interests include design-for-test, design-security and design-for-trust for VLSI circuits, where he has more than 180 conference and journal papers, and 20 issued and pending US Patents. Prof. Sinanoglu has given more than a dozen tutorials on hardware security and trust in leading CAD and test conferences, such as DAC, DATE, ITC, VTS, ETS, ICCD, ISQED, etc. He is serving as track/topic chair or technical program committee member in about 15 conferences, and as (guest) associate editor for IEEE TIFS, IEEE TCAD, ACM JETC, IEEE TETC, Elsevier MEJ, JETTA, and IET CDT journals.

Prof. Sinanoglu is the director of the Design-for-Excellence Lab at NYU Abu Dhabi. His recent research in hardware security and trust is being funded by US National Foundation, US Department of Defense, Semiconductor Research Corporation, Intel Corp and Mubadala Technology.

Johann Knechtel (M’11) received the M.Sc. in Information Systems Engineering (Dipl.-Ing.) in 2010 and the Ph.D. in Computer Engineering (Dr.-Ing., summa cum laude) in 2014, both from TU Dresden, Germany. He is a Research Scientist at the New York University, Abu Dhabi, UAE. Dr. Knechtel was a Postdoctoral Researcher in 2015–16 at the Masdar Institute of Science and Technology, Abu Dhabi. From 2010 to 2014, he was a Ph.D. Scholar with the DFG Graduate School on “Nano- and Biotechnologies for Packaging of Electronic Systems” hosted at the TU Dresden. In 2012, he was a Research Assistant with the Dept. of Computer Science and Engineering, Chinese University of Hong Kong, China. In 2010, he was a Visiting Research Student with the Dept. of Electrical Engineering and Computer Science, University of Michigan, USA.

His research interests cover VLSI Physical Design Automation, with particular focus on Emerging Technologies and Hardware Security.