Concerted Wire Lifting: Enabling Secure and Cost-Effective Split Manufacturing

Satwik Patnaik, *Member, IEEE*, Mohammed Ashraf, Haocheng Li, Johann Knechtel, *Member, IEEE*, and Ozgur Sinanoglu, *Senior Member, IEEE*

***Abstract*—In this work, we advance the security promise of split manufacturing through judicious handling of interconnects. First, we study the cost-security trade-offs underlying for split manufacturing, which are limiting its adoption. Next, aiming to resolve these concerns, we propose three effective and efficient strategies to dedicatedly lift nets to higher metal layers. Towards this end, we design custom "elevating cells" and devise procedures for routing blockages. All our techniques are employed in a commercial-grade computer-aided design (CAD) framework. For our security analysis, we leverage various state-of-the-art attacks (network flow-based attack, routing-congestion-aware attack, and deep learning-based attack), established metrics (CCR, OER, and HD), and advanced metrics (percentage of netlist recovery and mutual information). Our extensive experiments show that our scheme provides superior protection. Simultaneously, we induce reasonably low and controllable overheads on power and performance, without any silicon area costs. Besides, we support higher split layers, which helps to alleviate concerns on the practicality of split manufacturing.**

***Index Terms*—Split manufacturing, Hardware security, IP protection, Reverse engineering, Routing perturbation**

I. Introduction

NOWADAYS, integrated circuit (IC) manufacturing is a complicated and costly process where, more often than not, third-party entities are involved. As a result, protecting the intellectual property (IP) of chip designs and ensuring trust in the ICs have become serious concerns.

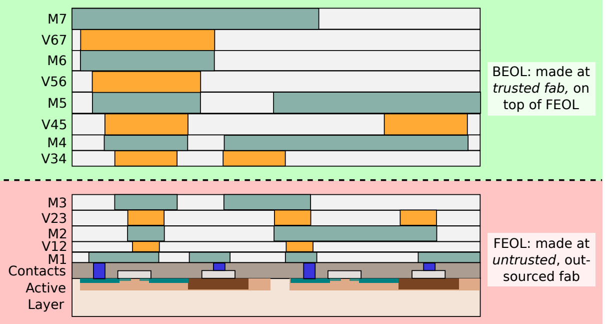
M7
V67
M6
V56
M5
V45
M4
V34
BEOL: made at *trusted fab*, on top of FEOL
M3
V23
M2
V12
M1
Contacts
Active Layer
FEOL: made at *untrusted*, out-sourced fab

Fig. 1. Conceptual illustration of split manufacturing, i.e., the separation of a layout into the front-end-of-line (FEOL) and back-end-of-line (BEOL).

The Intelligence Advanced Research Projects Activity agency proposed *split manufacturing* as a protection technique to ward off threats like IP piracy, unauthorized overproduction, and targeted insertion of hardware Trojans [2]. In the simplest embodiment of split manufacturing, the front-end-of-line (FEOL) is manufactured by a high-end, competitive off-shore fab which is potentially *untrustworthy*, while the back-end-of-line (BEOL) is fabricated on top of the incomplete FEOL layout at a low-end, *trustworthy* facility (Fig. 1). Note the different pitches of the metal layers; this disparity facilitates the practicality of split manufacturing. Still, as the FEOL is outsourced, it may require additional protection to deter fab-based adversaries from recovering the missing BEOL connections. Hill *et al.* [3] successfully demonstrated split manufacturing on a 1.3 million-transistor asynchronous FPGA. Further studies also bear testament to the applicability of split manufacturing [2], [4]; [4] describes promising measurement results for an *IBM/GlobalFoundries* 130nm split process, and [2] summarized further results, most notably for a 28nm split process run by *Samsung* across Austin and South Korea. However, the overall acceptance of split manufacturing remains behind expectations due to concerns about cost and yield.

The protection offered by split manufacturing is based on the fact that the FEOL fab does not have access to the complete design, and an attacker is hindered from malicious activities. The threat models for split manufacturing are accordingly focused on FEOL fab-based adversaries, which either seek to (i) retrieve the design and/or its IP, or (ii) insert hardware Trojans. Some studies also consider both at the same time [5]–[7]. In this work, we address (i).

Prior art suggests splitting after metal layer M1,[1] as such a scenario forces an attacker to tackle a "vast sea of gates" with only a few transistor-level interconnects remaining [4]. Although splitting after M1 offers the best protection in principle, it also necessitates a high-end BEOL fab for the

This work was supported in part by the Center for Cyber Security at NYU New York/Abu Dhabi (NYU/NYUAD). The work of S. Patnaik was supported by the Global Ph.D. Fellowship at NYU/NYUAD. This work was carried out in part on the HPC facility at NYUAD. This work is an extension of [1].

Satwik Patnaik was with the Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University, Brooklyn, NY 11201, USA. He is currently with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: satwik.patnaik@tamu.edu).

Haocheng Li is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong (CUHK), NT, Hong Kong (e-mail: hcli@cse.cuhk.edu.hk).

Mohammed Ashraf, Johann Knechtel, and Ozgur Sinanoglu are with the Division of Engineering, New York University Abu Dhabi (NYUAD), Saadiyat Island, 129188, United Arab Emirates (e-mail: ma199@nyu.edu; johann@nyu.edu; ozgursin@nyu.edu).

[1] We advocate the terminology "split after," instead of the commonly used "split at" or similar wording. For example, to "split at M2" remains ambiguous whether M2 is still within the FEOL or already in the BEOL. Similar uncertainty applies to the vias V12 (between M1 and M2) and V23 (between M2 and M3), respectively. Our definition for "split after M2" is that M2 and V12 are present in FEOL, while the vias V23 is present in the BEOL.

trustworthy fabrication of the remaining metal layers, including some lower layers with very small pitches. Since this requirement is not cost-effective, some studies propose splitting after M4 [5], [8]. However, splitting after higher metal layers can undermine security by revealing more structural information to an opportune attacker [9]–[14].

The key challenge we seek to address in this work is: *how to render split manufacturing practical regarding both security and cost?* We address this challenge with a secure and effective approach for split manufacturing. Our fundamental principle is to lift wires to the BEOL in a controlled and concerted manner, considering both cost and security. The primary contributions of our work can be summarized as follows:

1) We revisit the cost-security trade-offs for split manufacturing. We explore the prospects of lifting wires to higher metal layers and find that although naive lifting can improve security, it comes with high layout costs/PPA. Thus, we proclaim the need for cost- and security-aware wire lifting techniques.
2) We introduce an information-theoretic metric that quantifies a given layout's resilience against any proximity attack. This metric (based on the mutual information) operates on the fundamental relation of distances between connected logic gates, more specifically between open pins/vias, which are to be connected in the BEOL. We also integrate this metric within the layout generation tool. We also promote a new attack-based metric, the *percentage of netlist recovery (PNR)*, which quantifies the resilience of any split manufacturing protection scheme against varyingly effective attacks.
3) We propose multiple strategies to identify and lift nets. The key principles to achieve strong resilience are to (i) increase the number of protected/lifted nets and (ii) dissolve hints of physical proximity for those nets.
4) Based on our strategies and our evaluation technique, we propose a methodology and workflow for the *concerted lifting of wires* with a controllable impact on PPA. We lift wires to the higher metal layers (M6 and M8), thereby lowering the commercial cost incurred by split manufacturing. For the actual layout-level lifting, we design custom "elevating cells" and independently leverage routing blockages.
5) We conduct a thorough security evaluation of our protected layouts against the network-flow attack [11], [15], a recent deep learning-based attack [13], and the open-source routing-congestion-aware attack [8], [16]. Our detailed analysis demonstrates the strength of the proposed techniques when compared to seven placement perturbation and four routing perturbation techniques. We also conduct a detailed layout-level PPA study for various benchmarks, including the industrial *IBM-superblue* suite. We make our layouts publicly available in [17].

II. Background and Prior Art

In this section, we discuss the relevant background and selected prior art. The broader efforts on split manufacturing are also reviewed in more detail in [18], [19].

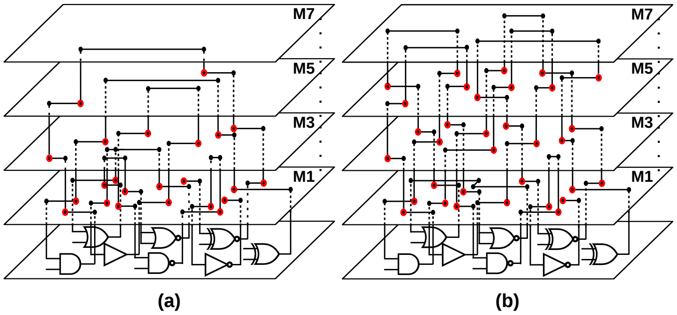

Fig. 2. (a) Conceptional illustration of a regular, unprotected layout. (b) Conceptional illustration of a layout protected by wire lifting. The red dots represent open pins, which induces dangling wires once the layout is split.

Concept and Terminology. The general notion of split manufacturing was introduced in Fig. 1. When a net is cut across FEOL and BEOL by split manufacturing, at least two *dangling wires* arise in the topmost layer of the FEOL.[2] Dangling wires remain unconnected at one end; these open ends indicate the locations where the BEOL fab manufactures the vias linking the FEOL and BEOL. We refer to those via locations as *open pins* (Fig. 2), as also defined in [11]. The routing of dangling wires is observable in the FEOL, but the correct mapping of drivers to sinks (through open pins) is comprehensible only with the help of BEOL. Note that the majority of nets for a regular, unprotected layout are completed in lower layers; thus, fewer open pins are observed when layouts are split after higher layers (Fig. 2(a)). Conversely, Fig. 2(b) illustrates the layout protected with naive wire lifting. In this scenario, the majority of nets are completed in M7 (without loss of generality). Consequently, any split below M7 induces many open pins to be tackled by an attacker.

Attack Schemes. Naive split manufacturing (i.e., splitting the layout as is) fails to avert skillful attackers. This is because physical-design tools tend to arrange gates to be connected as close as possible, subject to available routing resources and other constraints like power, timing, and wirelength, etc. The concept of a *proximity attack* was first introduced by Rajendran *et al.* [9], where this insight, as mentioned above, is exploited to infer missing interconnects. Wang *et al.* [11] proposed a network-flow-based attack that utilizes additional hints, such as the direction of dangling wires, non-formation of combinatorial loops, constraints on both load capacitances and delays. Magaña *et al.* [8] proposed five different versions of proximity attacks where they leverage notions of proximity based on placement data, routing data, routing congestion (denoted as *crouting*), union of placement and routing data, and spatial overlap of placement and routing data. The authors found that *crouting* is the most effective among these notions of proximity [8]. The main idea for *crouting* is to limit and guide the spatial search for the problem of open pins to be matched based on routing patters, including congestion-

[2] The reverse is not necessarily true, i.e., not all dangling wires represent a cut net—dangling wires may also be used dedicatedly for obfuscation. Such wires are routed in the FEOL but would remain open in the BEOL; see also Sec. VI. Besides, the number of dangling wires depends on the net's pin count and how/where precisely it is cut. See also Fig. 5 for an illustrative example.

dominated patterns. These attacks were evaluated on industrial *IBM-superblue* benchmarks, albeit using academic tools.

Recently, machine learning-based attacks have also been demonstrated in split manufacturing. Zhang *et al.* [12] leveraged a random-forest classifier to analyze the security of split manufacturing for *IBM-superblue* benchmarks. However, the authors do not predict the BEOL connections directly but instead generate only a list of candidates, which can be of considerable size. For instance, while attacking layouts split after M4, their most successful classifier provides on average several hundred or even thousands of candidates for each broken connection. Hence, it can become practically impossible to correctly retrieve all broken connections from those candidates. Li *et al.* [13], [14] propose a sophisticated deep neural network (DNN) while capturing various layout-level placement and routing hints as vector-based as well as image-based features. This DNN attack was demonstrated in [13], [14] to be able to infer missing BEOL connections with higher accuracy than the network-flow attack [11].

Protection Schemes. Various techniques have been proposed to protect split manufacturing-based layouts from proximity attacks. Swapping of pins between different partitions within hierarchical designs was proposed by Rajendran *et al.* [9] to impose a Hamming distance (HD; see also Sec. IV) of 50% between the outputs of the original netlist and the attacker's reconstructed netlist under simulation. Wang *et al.* [11] proposed gate-level placement perturbation within an optimization framework to maximize resilience and minimize wirelength overhead. Wang *et al.* [20] also proposed a routing-centric protection scheme leveraging wire lifting, deliberate re-routing, and application of VLSI test principles. Magaña *et al.* [8] used routing blockages to lift wires and thereby to mitigate routing-centric attacks as those proposed in their study. Patnaik *et al.* [21] proposed a combined placement-and-routing perturbation scheme using selective gate-level placement randomization and wire lifting. Chen *et al.* [22] proposed a selection of BEOL nets based on a signal-priority factor that captures the nets' effect on the primary outputs. More recently, Sengupta *et al.* [23] proposed a new paradigm for split manufacturing wherein the FEOL layout is locked with additional key-gates, with the signals driving these key-gates implemented through the BEOL.

Besides those above studies addressing proximity attacks, the works of [7], [24], [25] focus on hardware Trojans in the context of split manufacturing while the work of [26] establishes IP protection using 3D split manufacturing. Furthermore, Patnaik *et al.* [27] pursued BEOL-centric and large-scale layout camouflaging; the authors note that their scheme is also promising in the context of split manufacturing.

Limitations of Existing Protection Schemes. The approach of Rajendran *et al.* [9] is only applicable to hierarchical designs. More importantly, pin swapping is limited in practice from a security standpoint; 87% correct connections were reported in [9]. In general, placement-centric schemes would ideally (re-)arrange gates randomly, thereby "dissolving" any hint of spatial proximity. As this induces excessive PPA overheads [10], [24], placement perturbation is typically applied more selectively [10], [11]. However, as we show in Sec. VII, overly restrictive placement perturbation limits the protection, primarily when layouts are split after higher layers.

TABLE I
PITCH DIMENSIONS FOR METAL LAYERS IN THE 45NM NODE [9]

Layer	**M1**	M2	**M3**	M4	M5	**M6**	M7	**M8**	M9	M10
Pitch (nm)	130	140	140	280	280	280	800	800	1600	1600

Practical split layers are denoted in blue. Splitting after any other layer is also possible, but we argue that any split should occur just below the next larger pitch such that the BEOL facility has to manufacture only those larger pitches.

Some routing-centric techniques such as [20], [28] protect only a small subset of the design (a few nets), to limit PPA overheads. These techniques are further subject to available routing resources, and re-routing may be restricted to short local detours, which can be easy to resolve for an advanced attacker. Besides, implicit re-routing by insertion of blockages [8] can fall short of protecting selected nets and controlling the routing of wires. The work by Chen *et al.* [22] is only applicable to hierarchical designs. Furthermore, the authors do not provide any results for proximity attacks.

III. ON THE TRADE-OFFS FOR COST VERSUS SECURITY

It is challenging to determine the most appropriate split layer as such a decision has a direct and typically opposing impact on security versus costs.

Recall that some prior art, motivated by the inherent security promises, suggested splitting only after lower metal layers; however, this comes at a high commercial cost for the trustworthy BEOL fab. In contrast, splitting after higher layers allows for large-pitch and low-end processing setups at the BEOL facility. For example, for the 45nm node (Table I), one may prefer to split after M3 (or M6, or even M8) over splitting after M1. However, splitting after higher layers can easily undermine security. This is because the higher the split layer, the fewer open pins arise in the FEOL. For an attacker operating at the FEOL, observing fewer open pins translates directly to a reduced solution space and may lower his/her efforts to recover the protected design.

To substantiate this discussion, we plot in Fig. 3 the attacker's success rate versus the normalized count of open pins for various split layers. One can see that there are strongly reciprocal correlations across the layers, confirming that layouts split after higher layers are easier to attack and an attacker can correctly decipher larger parts of the underlying design. This is also because more and more nets are routed completely within the FEOL once we split after higher layers—these FEOL-routed nets yield no open pins and, thereby, impose no efforts for an attacker, to begin with.

One way to enforce many open pins while splitting after higher layers is *wire lifting*, i.e., the deliberate routing of nets through the BEOL (Fig. 2(b)). There is a common concern of overly high PPA costs for large-scale wire lifting [6], [24]. We confirm this in Table II, where we tabulate the layout overheads for *naive lifting* of randomly selected nets on selected ITC-99 benchmarks. We implement naive lifting by placing one "elevating cell" next to the driver for each

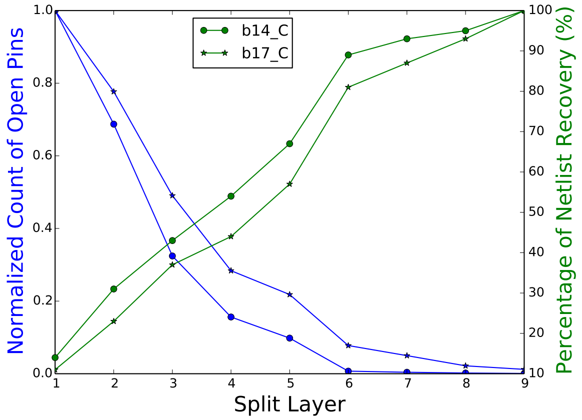

Fig. 3. Attacker's success rate, measured as percentage of netlist recovery (see also Sec. IV) versus normalized count of open pins, plotted as functions of the split layer. The original, unprotected layouts are split as is, and the attack is based on [11], [15].

TABLE II
POST-LAYOUT AREA (A), POWER (P), AND DELAY (D) COST FOR NAIVE WIRE LIFTING ON SELECTED ITC-99 BENCHMARKS

Benchmark	**10% Lifting**			**30% Lifting**			**50% Lifting**		
	A	**P**	**D**	**A**	**P**	**D**	**A**	**P**	**D**
b14_C	0	3.45	16.32	0	9.47	25.47	16.67	18.65	44.33
b17_C	7.69	14.34	20.47	27.27	23.49	37.89	55.55	31.34	58.74
b20_C	0	7.76	19.34	16.67	19.61	36.44	27.27	29.54	47.23
b22_C	0	17.44	22.67	27.27	29.98	37.11	40	36.87	56.53
Average	**1.92**	**10.75**	**19.76**	**17.8**	**20.64**	**34.22**	**34.86**	**29.1**	**51.71**

All cost are in percentage. We leverage the *NanGate* 45nm library [29] with synthesis runs constrained for iso-performance of 5ns (200 MHz).

randomly selected net.[3] Such naive lifting enforces routing at least to some degree above the split layer, thereby inducing more open pins and hampering an attacker's success rate. Note that all the metrics are impacted in Table II. This is because lifting more and more wires in an uncontrolled manner induces significant routing congestion, which can only be managed by re-routing the nets, increasing the wirelength, thus impacting both timing and power. Lifting more wires also consumes considerable routing resources, contributing to an increase in congestion and is typically mitigated by enlarging the die outlines, thus impacting the die area overhead.

Overall, there is a clear need for split manufacturing schemes ensuring resilience (e.g., through a large number of open pins), while splitting after higher layers, thereby enabling low commercial cost, all without inducing high PPA cost. We believe that such schemes are essential to expedite the broader acceptance of split manufacturing. In this work, we propose such a scheme through the notion of *concerted wire lifting*.

IV. SECURITY METRICS

A. Attack-Based Metrics

We discuss metrics which serve to gauge both (i) the effectiveness of proximity attacks and (ii) the resilience of layouts against particular attacks. Motivated by inherent shortcomings in those metrics, we also introduce a new attack-based metric.

The **Hamming distance (HD)** quantifies the average bit-level mismatch between the outputs of the original and the attacker's reconstructed design while under simulation [9]. Note that the HD reveals the degree of functional mismatch, but not structural mismatches, as any Boolean function can be represented using different gate-level implementations.

The **output error rate (OER)** indicates the probability for any output bit being incorrect while applying a (typically large) set of input patterns to an attacker's reconstructed netlist under simulation [11], [20]. Since this metric tends to approach 100% for any imperfect attack (or any reasonable defense), it does not reflect well on the degree and type of errors made by an attacker (or the efficiency of the defense), but rather whether any error (or resilience) was observed at all.

The **correct connection rate (CCR)** is the ratio of connections correctly inferred by an attacker over the number of protected nets. We note that Wang *et al.* [20] defined an *incorrect connection rate (ICR)*, which is simply the inverse of this metric. Unlike the HD or OER, this metric can quantify the attack efficiency (or the resilience of a layout) considering the netlist structure. For example, correctly recovering 20 out of 100 protected nets results in a CCR of 20%. This CCR would be equivalent to, e.g., correctly recovering 200 out of 1,000 nets, but the latter scenario would be more challenging for an attacker. This is because the complexity of the underlying pairwise mapping problem for correctly recovering nets based on open pins does not scale linearly.

The **percentage of netlist recovery (PNR)**—our newly proposed metric—quantifies the structural similarity between the original and the attacker's reconstructed netlist. To do so, it considers the ratio of correctly inferred connections over the total number of nets. Accordingly, we argue that PNR is more generic and comprehensive than CCR, as it accounts for the entire netlist, not only for protected nets.

For example, consider that an attacker correctly reconstructs 20 out of 100 protected nets in a design containing 10,000 nets. Now consider further that an attacker can readily recover all the nets routed completely in the FEOL. Assuming 2,000 nets are routed in the FEOL, the resulting PNR would be 20.2%, but for 6,000 nets routed in the FEOL, the PNR would already be 60.2%, indicating a much better outcome for the attacker in the second case. In contrast, CCR would remain at 20% for both cases, reiterating the CCR metric's shortcoming.

B. An Information-Theoretic Metric

While relevant in general and for quantitative studies of actual attacks in particular, any empirical evaluation (based on the metrics above or others) holds some notable limitations when assessing the security promises of defense schemes:

1) Explicitly running one or more attacks can be time-consuming and, thus, ineffective for evaluating the security of larger designs.
2) The security evaluation is specific to the employed attack(s) and, thereby, could easily fail to quantify the layout's resilience against other attacks.
3) Attacks can typically only be executed once the layouts are finalized, i.e., they are not applicable for early security evaluation during design time.

[3] It is important to note that elevating cells do not impact the FEOL device layer; they are solely designed to elevate/lift a given net. See Secs. V and VI for details on elevating cells and their usage.

Therefore, we introduce an information-theoretic metric, which helps to quantify the *information leakage* underlying a physical layout. The metric can be integrated within any physical-design flow to help evaluate security at design time.

Most, if not all, proximity attacks leverage the fact that distances between cells (more precisely, between their open pins) leak information about their connectivity. In other words, given that distances play an essential role within physical layouts for connectivity, they constitute a leakage model that an attacker may leverage. Without loss of generality, we define two variables, C and D, which capture the connectivity and distance between open pins as

$$C = \begin{cases} 1 & \text{if two open pins } u \text{ and } v \text{ are connected,} \\ 0 & \text{otherwise;} \end{cases} \tag{1}$$

$$D = \texttt{Manhattan\_distance}(x1, y1; x2, y2), \tag{2}$$

where, for two points P1 (x1, y1) and P2 (x2, y2) in the plane, the `Manhattan_distance` is $|x2 - x1| + |y2 - y1|$.

To quantify the amount of information revealed through distances about the connectivity between cells/open pins and, thus, to quantify the resilience of a layout against any proximity attack based on this principle, we determine the *mutual information (MI)* as:

$$MI = H[C] - H[C|D], \tag{3}$$

where

$$H[C] = -\sum_{c \in C} \Pr[C = c] \cdot \log \Pr[C = c] \tag{4}$$

$$H[C|D] = -\sum_{d \in D} \Pr[D = d] \cdot H[C|D = d]. \tag{5}$$

To obtain $H[C]$ and $H[C|D]$, we determine the distribution of C and D for a given layout, considering a given split layer, in a pairwise manner for all open pins. In turn, this allows for a straightforward and efficient computation of Eq. 3.

The mutual information, defined as above, quantifies the inherent resilience of a layout against proximity attacks; the lower the mutual information, the less the correlation between connectivity C and distance D and, thus, the better the resilience. In Fig. 4, we exemplify how mutual information can increase considerably for higher split layers. Therefore, utilizing higher split layers, renders layouts less resilient. This is expected—any net is to be fully routed at some point, and once reaching higher layers to do so, the related wire segments (which are giving rise to open pins after splitting) tend to become close and directed toward each other. Along with the fact that the number of open pins is reducing for higher split layers (also exemplified again in Fig. 4), this observation serves as a re-motivation for protection schemes incorporating concerted wire lifting, especially for higher split layers.

V. Concept and Strategies for Concerted Wire Lifting

As motivated in Sec. III and Sec. IV, the number of open pins in the FEOL should be as large as possible and their underlying correlation of distances and connectivity should be as little as possible, all while avoiding high cost (pertaining to commercial and PPA costs).

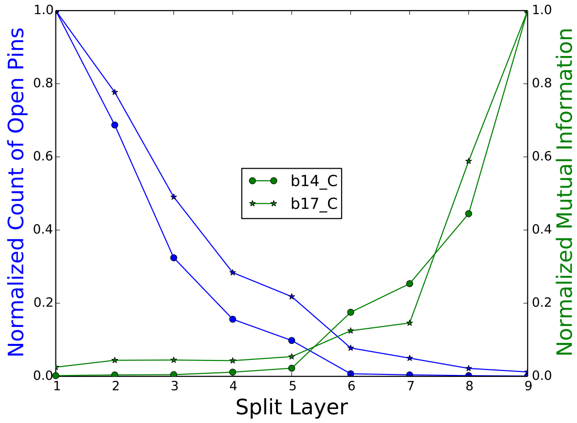

Fig. 4. Normalized mutual information and normalized count of open pins, plotted as functions of split layer, for two selected ITC-99 benchmarks. The original, unprotected layouts are split as is, and attack is based on [11], [15].

We tackle this problem with our custom *elevating cells* and *routing blockages*. Routing nets through elevating cells serves to establish pins and wire segments in a metal layer of choice (above the split layer), thereby inducing open pins for those nets by construction. Routing blockages serve to limit the usage of the lower metal layer(s) of choice, which forces the router to consider higher layers to a more considerable degree. Elevating cells and routing blockages have their respective benefits and limitations; hence, they are both considered in our methodology (Sec. VI). On the one hand, elevating cells guarantee that pins and wire segments are introduced in higher layers, whereas routing blockages in lower layers only indirectly establish a preference for the latter. Furthermore, elevating cells allows controlling the distances between open pins, thereby directly diminishing the underlying information leakage. On the other hand, routing blockages act more comprehensively, thus allowing to lift multiple nets and their wire segments at once.

Next, we introduce our strategies for concerted wire lifting. They are based on exploratory but comprehensive layout-level experiments. These strategies outperform naive lifting and recent prior art regarding security while inducing moderate PPA overhead, as we will show in detail in Sec. VII. See Sec. VI for implementation details of all strategies and for our methodology which enables the application of all strategies.

Strategy 1, Lifting of High-Fanout Nets (HiFONs). This is an important strategy for two reasons: (i) any wrong connection made by an attacker propagates the error to multiple locations in/outputs of the netlist, and (ii) lifting HiFONs helps introduce many open pins. In this work, we define nets with two or more sinks as HiFONs.

Consider Fig. 5 as an example. Here, a HiFON is initially connecting to four sinks. Depending on how and where the HiFON is lifted, the attacker has different scenarios to cope with. In (a), only one simple cut net arises, which is trivial to attack/resolve. In (b), two pairs of opens pins are to be tackled: (A,B) and (A,C). Given that an attacker cannot tell how many

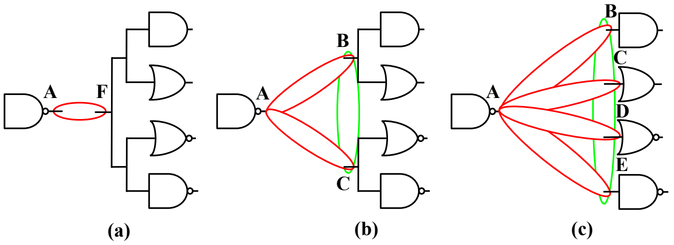

Fig. 5. Impact of lifting high-fanout nets on the number of open pins.

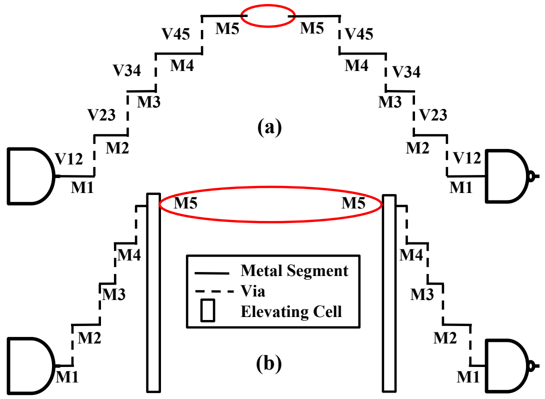

Fig. 6. Distances between open pins. In (a), only a short distance arises implicitly due to naive wire lifting, whereas in (b), while using two elevating cells, we can precisely control the distance.

sinks exactly to consider,[4] either one of the two options or even both at once could be representing the original net; the attacker has three options to consider. In (c), up to 14 possibilities arise; there are four pairs of open pins, namely (A,B), (A,C), (A,D), and (A,E), and in addition to considering them individually, ten possible combinations are arising for handling those pairs. Naturally, once other nets are lifted as well, the number of open pins scales up even further in a combinatorial manner.

We are leveraging routing blockages to realize this strategy; details for the implementation are provided in Sec. VI.

Strategy 2, Controlling the Distances between Open Pins. Besides increasing the number of open pins (e.g., through lifting of HiFONs), it is also beneficial to control the distances between open pins. For example, in Fig. 6(a), there exists a short open wire segment in M5, simplifying an attacker's efforts in reconnecting that particular cut net. Such scenarios can arise especially for naive wire lifting and other routing-centric defense techniques, where the FEOL metal layers to avoid are defined but the actual routing paths in the BEOL layers are not explicitly handled.

In our technique, we manage the distances between the open pins by controlling the placement of elevating cells (Fig. 6(b)). In simple terms, placing elevating cells close to the driver and the sink(s) enlarges the distance between open pins and, thus, increases an attacker's efforts.

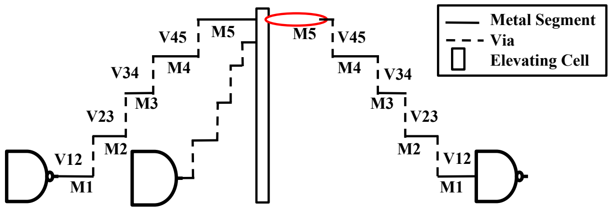

Fig. 7. Short nets are obfuscated by connecting the real and a dummy driver to an extended type of elevating cell. The dummy driver's wire connected with the elevating cell is left dangling beyond the split layer. Note that the dummy driver is a real driver for another net; the related wires are not illustrated here.

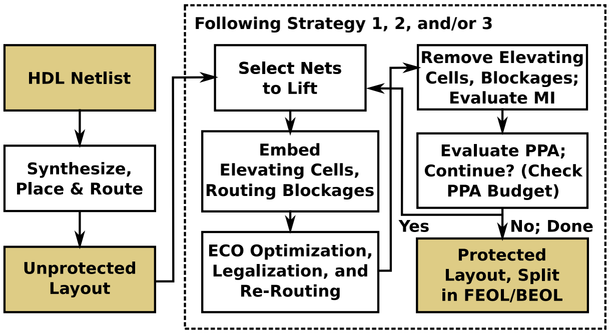
Following Strategy 1, 2, and/or 3
HDL Netlist
Synthesize, Place & Route
Unprotected Layout
Select Nets to Lift
Embed Elevating Cells, Routing Blockages
ECO Optimization, Legalization, and Re-Routing
Remove Elevating Cells, Blockages; Evaluate MI
Evaluate PPA; Continue? (Check PPA Budget)
Yes
No; Done
Protected Layout, Split in FEOL/BEOL

Fig. 8. The workflow of our methodology.

Strategy 3, Obfuscation of Short Nets. Above, we have discussed that controlling the distances between open pins is practical and useful. For short nets, however, enlarging those distances typically requires routing detours out of the net's bounding box, which adversely impacts the underlying design's power and timing. Furthermore, short nets may be easier for an attacker to identify and localize, to begin with, based on the low driving strength of the driver.

To tackle both issues, we design another, extended type of elevating cell. In short, as indicated in Fig. 7, this type serves the obfuscation of wiring by means of additional, dummy pins and drivers; more details are given in Sec. VI.

VI. Methodology

Our methodology serves to implement and evaluate wire lifting following the strategies introduced in Sec. V. Note that the security-enforcing designer has to decide which strategy/strategies and what scope of lifting to consider, as the prospects of each strategy depend largely on the design to be protected. For example, if a design consists of many HiFONs and/or long nets, Strategy 1 would be best suited; if a design consists largely of short nets, then a combination of Strategies 2 and 3 would be more preferable. In any case, leveraging all the strategies is also possible using our methodology.

The workflow of our methodology is illustrated in Fig. 8. It is integrated with *Cadence Innovus* using custom *TCL* procedures. Given a design in VHDL/Verilog format, we initially perform synthesis using *Synopsys Design Compiler* under appropriate timing constraints and then place and route the design using *Cadence Innovus*. The resulting original layout forms the baseline for security and layout analysis.

The original layout is protected as follows. First, nets to lift are selected, and the strategies outlined in Sec. V are applied

[4] While an attacker can readily infer any gate's driving strength by observing the FEOL; he/she cannot easily resolve their original use. Any high-strength driver may be either reconnected to many sinks nearby or a few sinks away. Once wire lifting is conducted across the whole layout, drivers and sinks of various nets will be "spatially intermixed," notably increasing an attacker's efforts to map drivers to sinks correctly.

as desired and appropriate. While doing so, elevating cells and routing blockages are inserted temporarily. Note that elevating cells do not impact the FEOL device layer; they are solely designed to elevate/lift a given net, as explained in more detail further below. Next, we perform *engineering change order (ECO)* optimization and legalization, leveraging customized scripts. Then, we re-route the design and thereafter remove the elevating cells and routing blockages. At this point, we also evaluate the mutual information to gauge the fundamental resilience of the design. Next, we extract the RC information, and report the PPA numbers. In case the PPA budget allows for additional wire lifting, we continue iteratively. Finally, a DEF file split into FEOL/BEOL is exported for security analysis against proximity attacks.

Implementation Details for Strategy 1, Lifting of HiFONs. Initially, using *Cadence Innovus*, we load the routing database for the original layout and set the attribute `skip routing = true` for all nets in the design. Next, we sort all nets by their fanout degree in descending manner. That is, all nets with the largest fanout degree overall come first, then all nets with the second-largest degree, etc.; for ties with nets of same fanout degree, these are sorted by names. Depending on a user-defined percentage range of nets to be lifted, nets are chosen from the above list; for those nets, we reset `skip routing = false`. Multiple routing blockages are created with user-defined X/Y offsets and widths in metal layers M3, M4, M5, and M6, respectively (assuming a split layer of M6). If the designer intends for a higher split layer, e.g., M8, then blockages would be correspondingly inserted in M4, M5, M6, and M7, respectively. Given the aforementioned user-defined parameters/ranges (i.e., percentage of nets to lift, X/Y offsets, and width of the routing blockages), we iterate through all combinations defined by those parameters/ranges, and we generate all the resulting, protected layouts. We also calculate the mutual information for all layouts and keep track of the number of open pins.

Next, we re-route the design of choice. Note that only the selected nets (those with `skip routing = false`) shall be re-routed, whereas the remaining nets will remain untouched even though such restrictions might cause design-rule check (DRC) violations. To fix any DRCs thereafter, we delete those routed nets inducing DRCs (using the command `editDeleteViolations`) and also remove the related routing blockage(s). These nets are then re-routed (using `ecoRoute`) to obtain a DRC-clean layout. We note that this last step of re-routing for DRC resolution might push some of the selected nets to the lower layers. However, the majority of previously lifted nets remains in higher layers, which we confirm/track by dumping metal-layer routing reports prior and post our lifting strategy.

Implementation Details for Strategy 2, Controlling the Distances between Open Pins. After loading the routing database for the original design, we initially obtain a list of single-fanout nets in order of decreasing lengths. Depending on a user-defined percentage of nets to be lifted, we choose a subset from that list. We invoke our custom procedures for wire lifting using elevating cells as follows.

Two elevating cells are inserted for each chosen net, one near the driver and one near the (single) sink. The notion for such usage of elevating cells is to ensure that the major portion of the net will be routed in the higher metal layers. The type of elevating cells used here was introduced in Fig. 6(b) and is now exemplified in Fig. 9(a).

While inserting the elevating cells, a random function is used,[5] coupled with a user-defined maximum threshold, so as to derive the placement of the elevating cells. For example, a maximum threshold of 10 μm would lead our procedure to choose a random location within a radius of 10 μm around both driver and sink for the respective elevating cells. Note that such a random procedure might as well reduce the distance between the open pins for driver and the associated sink, thereby potentially increasing again the vulnerability of the layout towards proximity attack. In order to resolve this, we leverage the principle of *obtuse angles*. That is, the randomly chosen locations are bound to form obtuse angles, thus ensuring the chosen location is always in the opposite direction of the driver sink connection. Thereafter, the placement of elevating cells is legalized, in order to arrange their pins properly on the routing tracks. Next, only these newly introduced nets concerning the elevating cells are routed; all other nets are skipped. Once the design is routed, the elevating cells are removed, and we perform `ecoRoute`, which handles the routing of nets that have DRC violations, while ensuring minimal changes to other parts of the layout.

Implementation Details for Strategy 3, Obfuscation of Short Nets. After loading the routing database for the original design, we initially obtain a list of short nets in order of increasing lengths. Depending on a user-defined percentage of nets to be lifted, we choose a subset from that list. Here we invoke wire lifting using an extended type of elevating cells as follows.

The extended type, introduced in Fig. 7 and now exemplified in Fig. 9(b), places two pins close to each other—one "true" pin that is connected to the short net's driver, and another "dummy" pin that is connected to a randomly but carefully selected gate, representing a dummy driver. An attacker cannot easily distinguish these two drivers due to the following: (i) the dummy driver is selected such that no combinatorial loops arise if the driver would be connected to the short net's sink(s), and (ii) we adapt both drivers' strength, also accounting for the routing detours, via ECO optimization. That is, both drivers' strengths are adapted through ECO optimization such that they could both plausibly connect to the sink of the obfuscated short net. While effectively obfuscating the true wiring, this strategy also helps to increase the overall number of open pins. Note that we insert only one elevating cell for short nets, specifically between their real and dummy driver. We refrain from inserting another elevating cell near the sink of short nets, as we observed that doing so contributes little for enhancing security but hampers routability which, in turn, ultimately degrades PPA.

[5] We use the *TCL* command `expr(rand())` in *Cadence Innovus* which generates a pseudo-random floating point number r, where $0.0 < r < 1.0$.

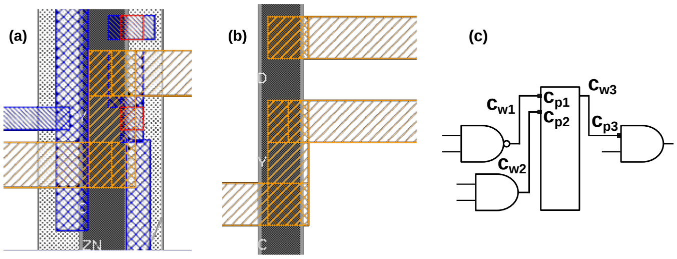

Fig. 9. Details of elevating cells. (a), (b): Post-processed snapshots of two types of elevating cells. Wires in M6 are in orange, wires/vias in M1 are in blue/red. (a) A regular elevating cell (dark grey) is seen overlapping an inverter (light grey, dotted). (b) The extended elevating cell has two input pins, C and D, as required for obfuscation of short nets. (c) Working principle of load annotation. The input-pin capacitances c_{p1}, c_{p2} are annotated such that they equal the wire capacitance c_{w3} and the sink's input-pin capacitance c_{p3}.

Design of Elevating Cells. As with any custom cell, our elevating cells are embedded in the technology library. Figure 9 exemplifies the two types of elevating cells we propose; important properties of those cells are discussed below.

1) All I/O pins are set up in one metal layer. Since the pins must reside above the split layer, to facilitate wire lifting, we implement different elevating cells as needed for various layers.
2) The pin dimensions and offsets are chosen such that the pins can be placed onto the respective metal layer's tracks. This helps in minimizing routing congestion.
3) Elevating cells may overlap with any other standard cell (e.g., Fig. 9(a)) without inducing any routing conflicts. This is because the standard cells have their pins exclusively in lower metal layers, whereas elevating cells neither impact those layers nor the FEOL device layer.
4) Custom legalization scripts serve to prevent the pins of different elevating cells from overlapping with each other.
5) Timing and power characteristics of `BUF_X2` (i.e., buffer of driving strength 2) are leveraged for the elevating cells; a detailed library characterization is not required. We use `BUF_X2` for two reasons. First, elevating cells are best represented by simple buffers, as elevating cells by themselves do not impose any other Boolean functionality. Second, the driving strength 2 is sufficient to model timing and power characteristics of elevating cells, as these are implementing only interconnects.
6) To enable proper ECO optimization, elevating cells are set up for load or capacitance annotation at design time. Such annotation is required to capture the capacitive load of (i) the wire from the elevating cell connecting to the sink(s) and (ii) the sink(s) itself/themselves (Fig. 9(c)). As outlined above, such annotation is also essential for obfuscating dummy drivers used in Strategy 3 as such.

VII. Experimental Investigation

In this section, we first describe the experimental setup in detail, along with the benchmarks under study. Next, we conduct a thorough security evaluation (concerning various metrics outlined in Sec. IV) and illustrate the efficacy of our proposed techniques against multiple attacks presented in the split manufacturing community. Finally, we present the impact of our proposed techniques on layout costs.

A. Setup

Layout Evaluation. Our techniques are implemented as custom *TCL* procedures for *Cadence Innovus 17.1* and impose negligible runtime overheads. We perform synthesis using *Synopsys Design Compiler* for the slow process corner. We leverage the *Nangate 45nm Open Cell Library* [29] comprising of ten metal layers and all library components. Our custom elevating cells are tailored for lifting wires to M6/M8 unless specified otherwise. PPA analysis has been carried out at 0.95V and 125°C for the slow process corner with a default switching activity of 0.2. We obtain power and timing results from *Cadence Innovus*.

The initial utilization rates for the original layouts are chosen such that the final congestion is 0%. For example, the results rates for all the ISCAS-85 benchmarks are 70%. This is essential for our work, namely to ensure that our wire lifting procedures act without any side effects and can be evaluated objectively. More specifically, congested original layouts would imply that a large share of routing would be handled in higher layers by nature of the design tools, thereby introducing some notion of wire lifting by itself, which would be difficult to separate from the contributions of our procedures.

Security Evaluation. We follow the conventional threat model for split manufacturing [9] where the FEOL foundry is untrustworthy but the BEOL foundry and end-users are considered trustworthy. We empower an attacker with the FEOL layout, a sound knowledge of the technology library and physical design files like LEF. The threshold rate for selecting nets to be lifted, across all the strategies, is 10% for the ITC-99 benchmarks, 25–30% for the ISCAS-85 benchmarks, and 3–5% for the industrial *IBM-superblue* benchmarks.

To evaluate our protected layouts, first we utilize the open-source network-flow attack by Wang *et al.* [11], [15]. The further advancements proposed in [28] have been incorporated in the attack binary we employ, as confirmed by the authors. We consider the ISCAS-85 and ITC-99 benchmarks here.

For attacking the industrial *IBM-superblue* benchmarks, which triggers scalability issues for that network-flow attack, we next employ the open-source routing-congestion-aware attack *crouting* by Magaña *et al.* [8], [16]. The authors found that this attack is the most effective among other notions of proximity [8]. Since the attack [8], [16] does not provide any actual netlists, the attack-based metrics introduced in Sec. IV cannot be used. Instead, we leverage the metrics defined in [8], namely *virtual pins (vpins)*, *average candidate list size E[LS]*, and *figure of merit (FOM)*. Note that *vpins* is equivalent to open pins. By definition, [8], [16] consider only *vpins* of two-pin nets, i.e., nets with single fanout. *E[LS]* is the average number of candidate *vpins* to match with other *vpins* (i.e., the number of possible pairs of open pins) over a specific region. *FOM* describes the number of possible pairs as normalized over the area of the specific region considered. Thus, when averaged across the whole layout, *FOM* serves to quantify the complexity of exploring all possible BEOL connections.

TABLE III
REDUCTION OF MUTUAL INFORMATION

Benchmark	Naive Lifting	Strategy 1	Strategy 2	Strategy 3
b14_C	-63.78	-96.84	-91.52	-94.84
b15_C	-16.07	-84.25	-74.91	-79.81
b17_C	-5.13	-35.97	-15.22	-39.62
b20_C	-34.78	-90.19	-83.94	-89.11
b21_C	-37.54	-90.75	-85.13	-89.97
b22_C	-31.16	-84.83	-76.42	-84.73
Average	**-31.41**	**-80.47**	**-71.19**	**-79.68**

All values are in percentage. Strategy 1 is lifting of HiFONs, Strategy 2 is controlling the distances between open pins, and Strategy 3 is obfuscation of short nets. The baseline are original layouts of the selected ITC-99 benchmarks split after M6.

TABLE IV
NUMBER OF OPEN PINS FOR SELECTED ITC-99 BENCHMARKS SPLIT AFTER M6

Benchmark	Original	Naive Lifting	Strategy 1	Strategy 2	Strategy 3
b14_C	47	644	1,253	292	269
b15_C	407	859	2,565	701	622
b17_C	2,789	2,749	3,296	2,947	2,873
b20_C	286	873	3,053	753	673
b21_C	289	912	3,602	730	714
b22_C	640	1,242	4,878	1,333	1,159
Median	**407**	**912**	**3,296**	**753**	**714**

Strategy 1 is lifting of HiFONs, Strategy 2 is controlling the distances between open pins, and Strategy 3 is obfuscation of short nets.

Finally, to demonstrate our protected layouts' resilience also against deep learning-based attacks, we further leverage the attack of [13], [14].

Functional equivalence of the attacker's reconstructed design is validated using *Synopsys Formality*. The OER and HD are calculated using *Synopsys VCS* by applying 1 million random input patterns, whereas PNR is calculated using *Cadence Conformal LEC*. All proximity attacks are executed on a high-performance computing (HPC) facility where each computational node has two 14-core Intel Broadwell processors (Xeon E5-2680), running at 2.4 GHz. Further, each node has 128 GB RAM in total, and 4 GB RAM are guaranteed (by the *Slurm* HPC scheduler) for each attack.

Benchmarks. We conduct experiments on combinational benchmarks from traditional suites (i.e., ISCAS-85 and ITC-99), but also sequential benchmarks from the large-scale *IBM-superblue* suite [30]. For the latter, we leverage scripts from [31] to generate *Verilog* files from the original *bookshelf* format. We employ custom scripts to convert the post-layout DEF files to *rt* and *out* format, as required when executing the *crouting* attack [8], [16].

Public Release. We provide our final protected layouts in [17]. The implementation of elevating cells, as LEF and liberty files, is also made available on request.

B. Security Analysis

Reduction of Mutual Information. As motivated in Sec. IV-B, the goal of a security-enforcing designer is to generate layouts such that information leakage is minimized. Hence, we first evaluate our strategies' effectiveness with regards to the reduction of mutual information when compared to original, unprotected layouts.

The reduction of mutual information for selected ITC-99 benchmarks is illustrated in Table III. We observe that all our strategies reduce the information leakage compared to layouts where naive wire lifting is applied. On average, naive wire lifting reduces the information leakage by 31.41% while our proposed strategies lead to a significant reduction of 80.47%, 71.19%, and 79.68%, respectively.

To further understand the impact on mutual information, we examine the distances of wiring segments. For example, in Fig. 10, we provide the distances across FEOL layers for the original, unprotected layout of ITC-99 benchmark b14_C for three split layers M2, M4, and M6, respectively. We observe that higher the metal layer (e.g., M6), the shorter the distances for the related metal segments, and fewer nets remain to be completed. In Fig. 11, we provide the distances of wiring segments after split layer of M6 for the protected layout of b14_C. Here we observe that the distances are considerably enlarged and more nets remain to be completed—both confirm the efficacy of our protection, and the enlarged distances also serve to explain the reduction of mutual information.

Increase of Open Pins/Vias. Recall that an increase in the total number of open pins (or up-ward vias in the split layer) renders any proximity attack more challenging, since the underlying solution search space is increased.

Our experiments demonstrate that the proposed strategies for wire lifting successfully increase the number of open pins over both original layouts and layouts where naive wire lifting is employed. We extract the numbers for open pins while executing the network-flow attack [11], [15] on selected ITC-99 benchmarks assuming a split layer of M6. The related results are presented in Table IV. We observe that while naive lifting increases the median number of open pins by 2.24× over original, unprotected layouts, our proposed strategies increase the median number of open pins by 8.1×, 1.85×, and 1.75×, respectively. While all our strategies increase the number of open pins compared to the original layouts, it is evident that Strategy 1, which lifts HiFONs, results in a significantly larger increase of open pins. Thus, this strategy might be particularly effective and preferable for protecting these selected ITC-99 benchmarks.

We also employ our techniques in the context of the million-gate *IBM-superblue* benchmarks derived from modern, industrial designs [30]. We tabulate the increases in vias and wirelengths in Table V and Table VI, respectively, for the protected layouts when compared with the original, unprotected layouts. The average increase in total vias and total wirelength is about 16.42% and 4.36%, respectively. While the total wirelength increases for protected layouts, there is a decrease in wirelength for metal layers below the split layer (i.e., M6). For example, considering the benchmark *superblue12*, which is the largest amongst all, we observe that while Δ_+M7 (i.e., the percentage increase in M7 wirelength pre- and post-lifting), Δ_+M8, and Δ_+M9 are 37.3%, 47.65%, and 27%, respectively, there is a drop in Δ_+M4 (-13.56%), Δ_+M5 (-32.73%), and Δ_+M6 (-18.59%). Thus, our wire lifting procedures succeed to increase the percentage of nets residing in M6, M7, and M8 (by 2.7%, 2.66%, and 1.88%, respectively).

Reduction in CCR. The reduction in mutual information (Table III), an increase of open pins (Table IV), and an increase in metal wirelengths (Table VI) imply that our protected

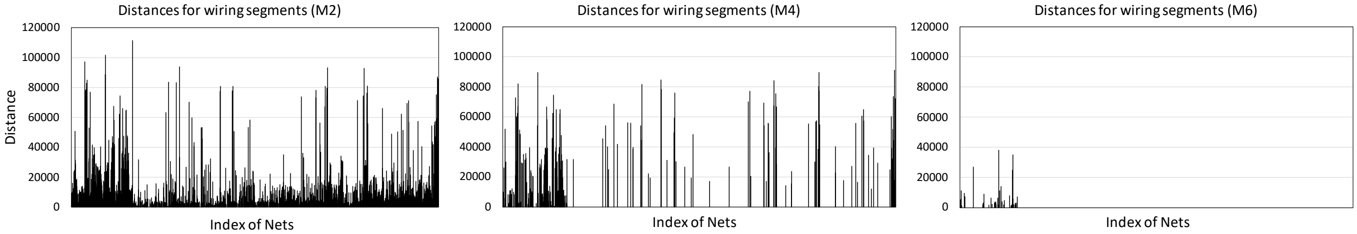

Fig. 10. Distances for wiring segments across selected FEOL metal layers for the original, unprotected layout for ITC-99 benchmark b14_C.

TABLE V
INCREASE IN VIAS FOR OUR PROTECTED LAYOUTS COMPARED TO ORIGINAL IBM-SUPERBLUE BENCHMARKS

Benchmark Statistics				Original			Proposed		
Name	Total Nets	Total Cells	Placement Util. (%)	V67 Before Lifting	V78 Before Lifting	V89 Before Lifting	V67 After Lifting	V78 After Lifting	V89 After Lifting
superblue1	859,958	851,638	69	72,541	50,388	28,808	161,478	96,184	47,952
superblue5	760,056	748,395	77	75,828	53,356	31,130	159,653	97,665	50,195
superblue10	1,122,059	1,111,605	75	106,768	74,379	43,880	236,521	144,117	74,916
superblue12	1,508,370	1,506,434	56	133,312	86,967	50,238	319,719	189,053	93,738
superblue18	672,401	668,480	67	49,263	35,133	16,873	137,824	89,444	41,403

TABLE VI
INCREASE IN WIRELENGTHS FOR OUR PROTECTED LAYOUTS COMPARED TO ORIGINAL IBM-SUPERBLUE BENCHMARKS

Benchmark	Original				Proposed			
	M7 Before Lifting	M8 Before Lifting	M9 Before Lifting	Total Wirelength Before Lifting	M7 After Lifting	M8 After Lifting	M9 After Lifting	Total Wirelength After Lifting (%)
superblue1	1,023,487.92	1,042,187.04	457,636.84	27,999,206.63	1,433,857.36	1,644,565.01	608,749.168	29,033,113.03 (3.69%)
superblue5	957,401.44	1,059,707.8	454,472.24	24,738,361.05	1,260,168.03	1,479,537.28	550,326.16	25,811,947.91 (4.34%)
superblue10	1,458,623.04	1,599,290.76	708,875.52	38,328,111.92	2,002,771.6	2,361,406.01	900,278.39	39,862,138.38 (4%)
superblue12	1,799,858.76	1,697,609.04	859,506.4	49,634,686.54	2,592,862.04	2,895,033.44	1,141,976.69	51,901,919.51 (4.57%)
superblue18	617,515.32	785,922.56	233,510.08	19,893,697.55	996,302.08	1,414,079.44	419,359.84	20,929,799.83 (5.21%)

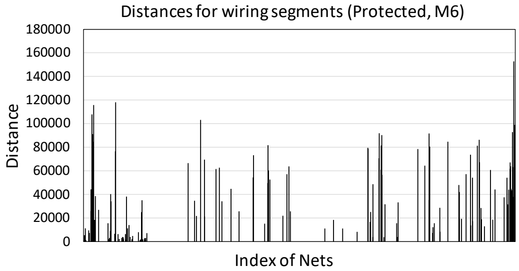

Fig. 11. Distances for wiring segments at the split layer M6 for the protected layout (using lifting of HiFONs) for ITC-99 benchmark b14_C. Note the different scale for the Y-axis when comparing to Fig. 10.

TABLE VII
CCR VALUES OBSERVED FOR NETWORK-FLOW ATTACK [15] ON SELECTED ITC-99 BENCHMARKS SPLIT AFTER M6

Benchmark	Original	Naive Lifting	Strategy 1	Strategy 2	Strategy 3
b14_C	51.06	34.74	10.22	17.47	12.64
b15_C	34.64	33.79	9.98	26.82	18.49
b17_C	24.24	23.89	16.14	20.02	15.49
b20_C	61.19	45.76	13.82	21.65	17.68
b21_C	56.41	50.35	12.11	23.84	22.55
b22_C	47.19	42.34	12.57	21.16	22.86
Average	**45.79**	**38.47**	**12.47**	**21.83**	**18.29**

All reported values are in percentage. Strategy 1 is lifting of HiFONs, Strategy 2 is controlling the distances between open pins, and Strategy 3 is obfuscation of short nets.

layouts should be challenging to attack compared to original and naively lifted layouts. We quantify the resilience for selected ITC-99 benchmarks by executing the network-flow attack [15]; those results are presented in Table VII and Fig. 13. Note that the attack [15] is not scalable for the *IBM-superblue* benchmarks; hence, such results cannot be provided.

In Table VII, concerning Strategy 1 (lifting of HiFONs) the CCR observed for the protected layouts when split after M6 is on average reduced by 33.32 *percentage points (pp)*, i.e., the arithmetic and absolute difference of two percentage values, or 3.67×. For Strategies 2 and 3 (controlling the distances between open pins and obfuscation of short nets), this reduction is 23.96 *pp* or 2.09×, and 27.5 *pp* or 2.5×, respectively. Figure 13 illustrates the CCR values for Strategy 1 for a large range of layouts and attack runs, also for higher split layers (M7 and M8). As expected, the higher the split layer, the higher the CCR. More important is the observation that most benchmarks, with a marginal exception for b17_C for split layer of M6, and b18_C, b19_C for split layer M8, became considerably more difficult to attack after applying the particularly promising Strategy 1—the average reduction in CCR is 49.6 *pp* compared to original layouts.

Reduction in PNR. Next, we provide an example to illustrate why PNR is considered as a better metric compared to CCR. While a CCR of, e.g., 34.64% for the original ITC-99 benchmark b15_C might seem low to begin with, the netlist recovered by the attacker will have just 266 wrongly inferred connections (over 4,779 nets), leading to PNR of 87%. As for a counterexample, with our Strategy 1 (lifting of

TABLE VIII
COMPARISON WITH PLACEMENT PERTURBATION TECHNIQUES

Benchmark	Original Layout [11]		BEOL+Physical [11]		Logic+Physical [11]		Logic+Logic [11]		Placement Perturbation [10]				Proposed		
	CCR	OER	CCR	OER	CCR	OER	CCR	OER	Random	G-Color	G-Type1	G-Type2	CCR	OER	HD
									CCR	CCR	CCR	CCR			
c880	100	N/A	86.11	16.70	82.53	95.09	81.75	93.89	56.12	84.32	81.47	78.54	32.28	99.99	45.57
c1355	69.12	60.11	46.81	92.74	56.14	95.29	62.25	79.23	N/A	N/A	N/A	N/A	27.15	99.99	46.34
c1908	87.79	50.28	60.74	97.86	46.07	98.11	61.75	95.54	70.83	83.94	81.95	79.95	30.99	99.99	48.74
c2670	87.79	75.02	69.02	78.74	75.12	81.09	87.79	74.77	52.84	66.67	66.92	56.54	29.11	99.99	43.87
c3540	51.37	95.72	49.61	98.98	28.59	97.58	38.71	99.99	44.83	40.31	41.72	42.44	10.73	99.99	47.51
c5315	100	N/A	92.39	61.26	92.91	32.35	91.89	56.14	49.53	54.12	50.17	56.29	17.47	99.99	43.87
c6288	90.56	39.51	41.93	76.84	68.77	80.81	77.65	98.07	N/A	N/A	N/A	N/A	11.36	99.99	48.74
c7552	92.91	19.77	90.63	23.76	84.56	24.62	63.54	28.01	56.92	48.94	53.38	48.57	15.16	99.99	47.72
Average	**84.94**	**56.73**	**67.15**	**68.36**	**66.84**	**75.62**	**70.67**	**78.21**	**55.18**	**63.05**	**62.60**	**60.39**	**21.78**	**99.99**	**46.55**

All values are in percentage. The values reported for our scheme are averaged across the three proposed strategies and for splitting after M3, M4, and M5, respectively. That is, multiple runs are conducted for each benchmark and the results are averaged thereafter. The values reported for prior art are quoted from the respective publications; N/A indicates not available. The attack leveraged is [15].

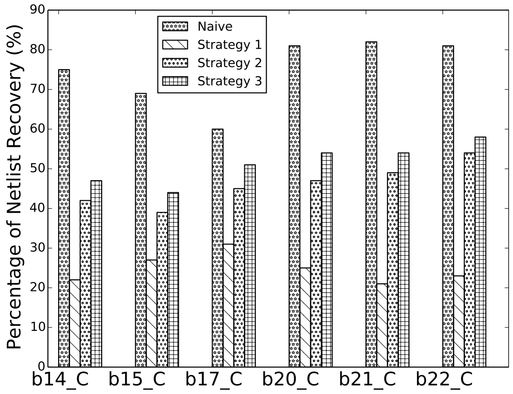

Fig. 12. Comparison between PNR for naive lifting and our proposed strategies for selected ITC-99 benchmarks split after M6.

HiFONs), we succeed in undermining the attack's effort. For the same benchmark b15_C, we now observe 2,309 incorrect connections for an attacker, leading to a PNR of 27%.

Figure 12 compares the PNR for (i) naive lifting and (ii) our wire lifting procedures in more detail, while splitting the layouts of selected ITC-99 benchmarks after M6. For a fair evaluation, we lift the same percentage of nets (i.e., 10%) for all benchmarks and strategies. Compared to naive lifting, we achieve an average reduction of PNR by 41.7 *pp*, 35 *pp*, 21 *pp*, 42.7 *pp*, 45.5 *pp*, and 41.5 *pp*, respectively, for the ITC-99 benchmarks in Fig. 12 seen from left to right. This affirms that the recovered netlists are considerably dissimilar in structure from the original designs.

Unfortunately, we cannot calculate PNR for the prior art, as we do not have access to the protected and attacker's reconstructed netlists, which are required by definition to compute PNR. Accordingly, the following comparative experiments are largely based on CCR (and other metrics as applicable).

Comparison With Placement Perturbation Techniques. Using CCR and OER as established metrics, we contrast in Table VIII the resilience offered by our techniques with seven placement-perturbation techniques proposed in [10], [11]. Intuitively, the original layouts are most vulnerable and, on average, the attack [15] can infer 85% of the missing connections correctly. Three protection techniques are proposed in [11]; on average, these techniques offer a CCR reduction of 17.79 *pp*, 18.1 *pp*, and 14.27 *pp*, respectively, when compared to original layouts. The strategies pursued by Sengupta *et al.* [10] offer a further reduction and accordingly more robust protection; their strategy of placement randomization, in particular, imposes a notable drop for CCR of 29.76 *pp*. Still, we argue that the protection provided by such placement perturbation (especially when applied restrictively to honor PPA budgets) is eventually offset by routing at higher layers. Hence, placement perturbation schemes are limited once a higher split layer is desired.

In contrast, we observe superior results while splitting the layouts after M6. On average, we observe a drop of 63.16 *pp* in CCR and an increase of 43.26 *pp* in OER.

Comparison With Routing Perturbation Techniques. Next, we compare our techniques with four routing-perturbation techniques [9], [20], [23], [28] in Table IX. On average, the techniques proposed by Wang *et al.* [20] help reduce the CCR to 30.86% and increase the HD between the attack's reconstructed design and the original design under simulation to 31.58%. The synergistic strategies for routing perturbations proposed by Feng *et al.* [28] show a more substantial impact on CCR (reduced to 22.82% on average) while the HD numbers are lower when compared to [20]. The technique proposed in [23] achieves a higher HD when compared to both [20], [28]. Our techniques offer the highest protection, by both means of CCR (average value of 21.78%) and HD (average value of 46.55%), over prior routing perturbation techniques. We also compare our scheme with the recent work of [23] utilizing the large-scale ITC-99 benchmarks (Table X). On average, our techniques decrease the CCR by 12.4 *pp* (or 1.71×) and increase HD by 18.18 *pp* (or 1.73×).

Next, we compare our technique with the protection scheme proposed by Magaña *et al.* [8]. Having no access to the protected *IBM-superblue* layouts by Magaña *et al.* [8], we can only compare on a qualitative level with their work. In Table XI, we contrast the counts of *additional vias*, assuming a split layer of M6. Please note that only total via counts across all layers before lifting and the layer-wise differences in via counts after lifting are provided in [8], but not the original via counts per layer. On average, our increases in via counts Δ_+V67, Δ_+V78, and Δ_+V89 are 106.34%, 54.56, and 48.95%, respectively, which indicates that our scheme would be accordingly more difficult to attack.

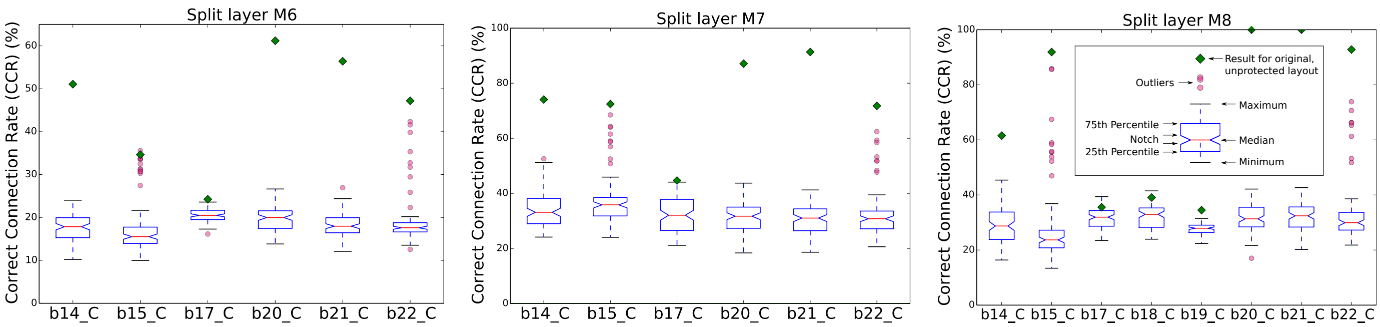

Fig. 13. CCR values for selected ITC-99 benchmarks, protected through lifting of HiFONs, for split layers M6, M7, and M8, respectively. For each benchmark, we have independently generated 100 layouts and execute the network-flow attack [11], [15]. The results are summarized in boxplots as follows: the lower and upper boundaries of each box span from the 25th to the 75th percentile for the respective data set, whiskers represent the minimum and maximum values, the red bar represents the median, red dots reflect outliers, and notches display the variability of the median (the height of a notch is computed so that boxes whose notches do not overlap have different medians at the 5% significance level). Besides, green diamonds represent the CCR for the respective original, unprotected layouts. Two of the largest benchmarks, b18_C and b19_C, were subject to attack failures for split layers M6 and M7 and are accordingly omitted.

TABLE IX
COMPARISON WITH ROUTING PERTURBATION TECHNIQUES

	Original Layout [28]			Pin Swapping [9]			Routing Perturbation [20]			Synergistic SM [28]			Unlock BEOL [23]			Proposed		
Benchmark	**CCR**	**OER**	**HD**	**CCR**	**OER**	**HD**	**CCR**	**OER**	**HD**	**CCR**	**OER**	**HD**	**CCR**	**OER**	**HD**	**CCR**	**OER**	**HD**
c880	N/A	N/A	N/A	85	N/A	25.39	N/A	N/A	N/A	N/A	N/A	N/A	***	100	35.7	32.28	99.99	45.57
c1355	N/A	N/A	N/A	86	N/A	40	N/A	N/A	N/A	N/A	N/A	N/A	29.38	100	32.3	27.15	99.99	46.34
c1908	N/A	N/A	N/A	86.2	N/A	26.11	N/A	N/A	N/A	N/A	N/A	N/A	***	100	34.4	30.99	99.99	48.74
c2670	62	N/A	11.9	N/A	N/A	N/A	31.4	N/A	20.3	33.3	N/A	20.5	N/A	N/A	N/A	29.11	99.99	43.87
c3540	77.6	N/A	13.5	83.5	N/A	50	28.8	N/A	38.5	11.5	N/A	35	32.03	100	37.8	10.73	99.99	47.51
c5315	58.8	N/A	13.3	92.5	N/A	41.22	42.1	N/A	24.1	14.9	N/A	23.6	28.29	100	45.2	17.47	99.99	43.87
c6288	100	N/A	0	N/A	N/A	N/A	27.7	N/A	44.3	33.1	N/A	40.6	N/A	N/A	N/A	11.36	99.99	48.74
c7552	54.8	N/A	15.5	91	N/A	47.79	24.3	N/A	30.7	21.3	N/A	24.7	28.57	100	28.3	15.16	99.99	47.72
Average	**70.64**	**N/A**	**10.84**	**88.1**	**N/A**	**38.42**	**30.86**	**N/A**	**31.58**	**22.82**	**N/A**	**28.88**	**29.57**	**100**	**35.62**	**21.78**	**99.99**	**46.55**

The setup is the same as in Table VIII. Besides, for a meaningful comparison with the work of [23], we execute the attack [15] ourselves and 1) calculate the CCR over all nets and 2) average this CCR for splitting after M3, M4, and M5, respectively. ***: These entries indicate network-flow attack failures by segmentation fault.

TABLE X
COMPARISON WITH [23] ON ITC-99 BENCHMARKS SPLIT AFTER M6

Benchmark	**Unlock BEOL [23]**			**Proposed**		
	CCR	**OER**	**HD**	**CCR**	**OER**	**HD**
b14_C	21.27	100	25	13.44	99.99	41.71
b15_C	37.48	100	20	18.43	99.99	43.59
b17_C	21.26	100	31	17.22	99.99	37.31
b20_C	29.62	100	19	17.72	99.99	45.34
b21_C	38.72	100	26	19.5	99.99	46.23
b22_C	31.23	100	27	18.86	99.99	44.87
Average	**29.93**	**100**	**25**	**17.53**	**99.99**	**43.18**

The setup is the same as in Table IX.

TABLE XI
COMPARISON WITH [8] ON SELECTED IBM-SUPERBLUE BENCHMARKS

	Wire Lifting [8]			**Proposed**		
Benchmark	**Δ_+V67**	**Δ_+V78**	**Δ_+V89**	**Δ_+V67**	**Δ_+V78**	**Δ_+V89**
superblue1	23.28	65.07	6.19	122.60	90.89	66.45
superblue5	12.74	24.01	6.45	110.55	83.04	61.24
superblue10	64.85	84.09	113.69	121.53	93.76	70.73
superblue12	16.99	35.59	21.73	139.83	117.38	86.59
superblue18	24.73	58.06	37.57	179.77	154.59	145.38
Average	**28.52**	**53.36**	**37.13**	**134.86**	**107.92**	**86.08**

All reported values are in percentage.

Results for Routing-Congestion-Aware Attack [8], [16]. Next, we execute the *crouting* attack put forth by Magaña *et al.* [8], [16], while considering ITC-99 benchmarks split after M6 (Table XII). We observe a significant increase for *vpins* using our scheme; on average, there are 11.85× more *vpins* when compared to unprotected layouts. The average candidate list size, or *E[LS]*, also increases notably for the protected layouts, attesting that the search space for the attack has increased. Accordingly, both increase in *vpins* and *E[LS]* impact the attack complexity, denoted by *FOM*. For example, *FOM* increases by 366.67% and 425%, respectively, for the largest ITC-99 benchmarks b18_C and b19_C.

TABLE XII
RESULTS FOR *crouting* PROXIMITY ATTACK [8], [16] ON SELECTED ITC-99 BENCHMARKS SPLIT AFTER M6

Benchmark	**Original**			**Proposed**		
	vpins	***E[LS]***	***FOM***	***vpins***	***E[LS]***	***FOM***
b15_C	66	1.36	0.028	327	2.86	0.059
b17_C	438	3.14	0.026	775	5.23	0.044
b18_C	332	2.4	0.006	1,399	9.54	0.028
b19_C	302	6.35	0.004	4,011	28.98	0.021
b20_C	26	0.15	0.003	494	2.71	0.057
b21_C	14	0.07	0.001	432	2.32	0.048
b22_C	74	0.45	0.007	656	3.24	0.051

The values reported for our scheme are averaged across the three proposed strategies. Recall Sec. II and Sec. VII-A for the basics and the metrics for this attack, or see [8] itself for further details. For a fair comparison across different benchmarks, we set up each attack run with a search region equal to $1/8$ of the half perimeter of the respective die outlines.

Results for Deep Neural Network-Based Attack [13], [14] Finally, we perform a comprehensive analysis leveraging the attack of [13], [14]. Towards this end, we consider two different training approaches. In the first approach, for each benchmark, we randomly select 40 out of 100 protected layouts to train the DNN model, cross-validate the model using

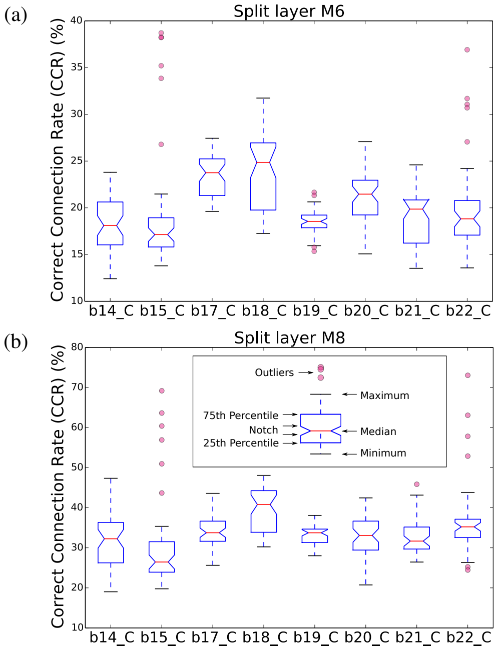

Fig. 14. CCR values obtained by the DNN-based attack [13], [14] on ITC-99 benchmarks, protected using lifting of HiFONs (Strategy 1). *Robust* training is used here. Details regarding the boxes are the same as in Fig. 13.

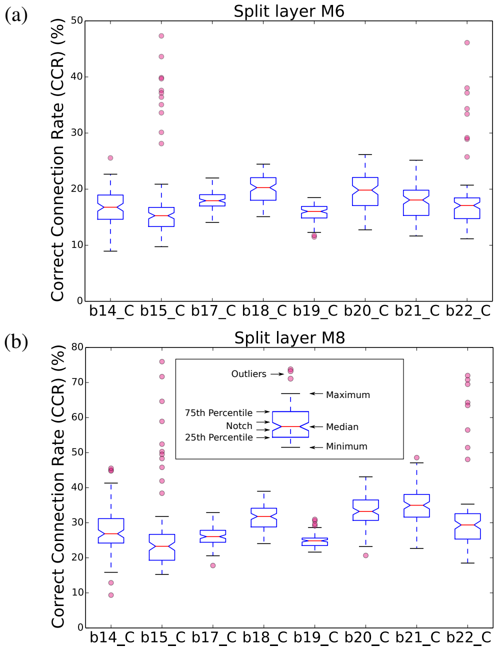

Fig. 15. CCR values obtained by the DNN-based attack [13], [14] on ITC-99 benchmarks, protected using lifting of HiFONs (Strategy 1). *Leave-one-out* training is used here. Details regarding the boxes are the same as in Fig. 13.

ten layouts, and attack the remaining 50 unseen layouts.[6] That is, we independently train separate and dedicated models for each benchmark under attack. We refer to this approach as *robust* as it represents the most stringent option for evaluating the strength of our defense, which can only be conducted by the security-enforcing designers, not actual attackers. In the second approach, which can also be taken by an attacker, we leverage the standard *leave-one-out* approach to separate training and testing data. Given that we consider eight ITC-99 benchmarks in total, we use seven designs (with ten layouts each) to train a model; we then attack the one remaining, unseen design. Note that separate models are created for each design under attack. As lifting of HiFONs (Strategy 1) provided, on average, the strongest security results, we study this strategy also against the DNN-based attack [13], [14]. We use the CCR metric to demonstrate the efficacy of our technique against the DNN-based attack.

First, we discuss the results for the *robust* learning approach. We perform the attack evaluation for one of our proposed technique (i.e., lifting of HiFONs, Strategy 1), while splitting after M6; the results are illustrated in Fig. 14a. The average CCR values observed for the protected layouts are 18.28%, 19.38%, 23.46%, 23.83%, 18.45%, 20.82%, 19%, and 19.66%, for the benchmarks b14_C, b15_C, b17_C, b18_C, b19_C, b20_C, b21_C, and b22_C, respectively. For example, when compared to the network-flow attack results on the original, unprotected layouts (Table VII), this manifests as an average reduction of 25.69 *pp*,[7] The best results, from the security-enforcing designer's perspective, show CCR values of 12.42%, 13.80%, 19.62%, 17.26%, 15.35%, 15.08%, 13.53%, and 13.58%, respectively. This implies two findings: the DNN-based attack is more effective than the network-flow attack, and the proposed Strategy 1, lifting of HiFONs, remains resilient even when under the DNN-based attack. While executing the DNN-based attack on the same set of benchmarks split after M8 (Fig. 14b), we observe the following. The average CCR values observed for the protected layouts are 32.02%, 30.5%, 33.87%, 39.84%, 33.29%, 32.92%, 32.96%, and 36.47%, respectively. This results in an average reduction of 35.43 *pp* when compared to original, unprotected layouts. The best results, from the security-enforcing designer's perspective, show CCR values of 19.03%, 19.77%, 25.63%, 30.24%, 28.02%, 20.74%, 26.45%, and 24.46%, respectively. Overall, these experiments illustrate (i) the strength of our proposed technique in the most stringent threat scenario of the powerful DNN-based attack [13], [14] and (ii) that splitting after higher metal layers is feasible without compromising security too much.

Next, we consider the *leave-one-out* training approach. The average CCR values observed for the layouts protected by Strategy 1 (lifting of HiFONs), split after M6, are 16.84%, 17.22%, 17.99%, 19.97%, 15.77%, 19.49%, 17.78%, and 17.78%, respectively, for the same set of ITC-99 benchmarks (Fig. 15a). When comparing these results as well to the network-flow attack runs on the original, unprotected designs this manifests as average reduction of 27.94 *pp*. The best results, again from the security-enforcing designer's perspective, show CCR values of 8.94%, 9.74%, 14.07%, 15.09%, 11.47%, 12.75%, 11.65%, and 11.15%, respectively. While executing the DNN-based attack on same set of benchmarks split after M8 (Fig. 15b), an average reduction of 39.85 *pp* is observed.

We conclude the following from the observations above.

[6] Note that the DNN model works on all the open pins within these layouts, not the layouts as one entity. Accordingly, the total number of data points is much larger than 40/10/50 (layouts), with the actual values depending on the design, split layer, strategy for lifting, placement and routing, etc.

[7] The network-flow attack [15] failed for the larger ITC-99 benchmarks b18_C, b19_C; thus, the corresponding results are excluded for this average.

The final version of record is available at http://dx.doi.org/10.1109/TCAD.2021.3056379

1) Our proposed wire lifting strategy considering HiFONs makes it difficult even for an advanced attacker leveraging a powerful DNN-based attack. Lifting of HiFONs provides reasonably strong protection even when layouts are split after higher layers (i.e., M6 and M8).
2) The proposed strategy is largely independent in its effect from the properties of the layout it is applied to. When comparing the results for the *robust* to the *leave-one-out* approach, the former is not significantly better, which implies that the knowledge of the actual design under attack does not benefit the DNN framework significantly.
3) The DNN-based attack [13], [14] is in general more effective than the network-flow attack [11]. Furthermore, while the network-flow attack fails for the larger ITC-99 benchmarks b18_C and b19_C, the DNN-based attack succeeds which demonstrates its scalability.

C. Layout Cost Evaluation

First, note that we cannot directly compare to prior works of, e.g., Wang *et al.* [11], [20], Magaña *et al.* [8], and Sengupta *et al.* [23]. This is because the technology node, utilization parameters, timing constraints, etc., all play a crucial role in layout evaluation, are commonly omitted in the prior art.

Second, for our scheme, when considering the *IBM-superblue* benchmarks, we observe marginal average costs, namely 2.97%, 1.31%, and 0% for power, delay, and die area, respectively. We also explore the power and delay overheads for selected ITC-99 benchmarks, investigating all our strategies individually, along with a zero die-area cost constraint; the results are given in Table XIII.

Next, we discuss the impact of our techniques on layout overheads in general. Recall that our custom elevating cells and routing blockages do not impact any standard-cell area; thus, we report area overhead in terms of die area. Wherever necessary to avoid DRC violations, we increase die outlines step-wise to make room for more routing resources. We ensure that final, protected layouts are devoid of any DRCs; most often, however, doing so does not require any increase of die outlines at all. As our techniques serve to move selected nets to higher metal layers, an increase of wirelength is expected (e.g., see Table VI). Accordingly, we observe some overheads for power and timing. Once the security-enforcing designer intends to lift a large percentage of nets, the positive effect of relatively low resistance in higher metal layers is offset by a steady increase in routing congestion. Typically, congestion is managed by re-routing, which tends to lengthen nets further, aggravating the overheads to some degree.

Overall, we maintain that our notion of *concerted wire lifting* is feasible and rarely inducing any die-are cost, i.e., the commercial cost for silicon is not impacted. We also conclude that re-routing (due to lifting in general and insertion of routing blockages and/or elevating cells in particular) imposes some power overheads by automatic insertion of additional buffer(s) and/or upsizing of standard cells, but these overheads can be well managed under the constraint of iso-performance.

TABLE XIII
POWER AND DELAY OVERHEADS ON SELECTED ITC-99 BENCHMARKS FOR ZERO DIE-AREA OVERHEAD

Benchmark	Strategy 1		Strategy 2		Strategy 3	
	Power	**Delay**	**Power**	**Delay**	**Power**	**Delay**
b14_C	5.14	16.24	4.74	15.58	3.95	12.07
b15_C	4.67	12.01	6.18	16.82	3.98	7.21
b17_C	1.93	3.61	2.91	21.72	3.74	15.42
b20_C	4.59	11.55	4.76	20.83	4.06	15.36
b21_C	4.82	17.59	5.23	18.07	4.33	15.14
b22_C	4.68	10.75	4.59	22.91	3.93	17.71
Average	**4.31**	**11.96**	**4.74**	**19.32**	**3.99**	**13.82**

VIII. Conclusion

We proposed a scheme for concerted wire lifting, advancing the prospects of split manufacturing. The objectives we addressed here are (i) to support splitting after higher metal layers, thereby reducing the impact of split manufacturing on manufacturing requirements and commercial cost, (ii) superior resilience, and (iii) reasonable and controllable PPA cost.

Through exploratory experiments, we have shown that splitting at higher layers can undermine the security promises of split manufacturing to a substantial degree. Prior art is widely susceptible to this concern, given that the security introduced by both placement- or routing-centric protection schemes is eventually offset through wiring segments in higher layers. We found that the notion of wire lifting is essential to maintain security, but also that naive lifting can introduce considerable PPA cost. Thus, we proclaimed the need for cost- and security-aware, concerted wire lifting schemes.

We proposed and studied three different strategies for concerted wire lifting: 1) lifting of high-fanout nets, 2) controlling the distances between open pins, and 3) obfuscation of short nets through dummy wires. For all three strategies, the key principles are to increase the number of open pins and to dissolve hints of physical proximity at higher split layers. For the implementation of these strategies, we devised custom "elevating cells" as well as procedures for routing blockages. Besides, we introduced two advanced metrics for security evaluation—mutual information and percentage of netlist recovery—which serve well to quantify the foundational resilience during design time and the netlist-scale resilience after design time, respectively. All strategies and techniques were implemented in a commercial-grade CAD workflow.

We conducted a thorough experimental evaluation which showed the superior strength of the proposed scheme—it succeeds to increase open pins, reduce the mutual information, and improve important attack-based metrics (CCR, PNR, OER, and HD), all to a substantial degree when compared to unprotected layouts as well as prior art. Finally, regarding layout costs, we observe that these can be well managed, in particular without any die-are or silicon overheads.

As a part of future work, we plan to extend and/or generalize the information-theoretic metric by incorporating additional metrics such as directionality of dangling open pins and hints related to load capacitance and overall timing of the design.

References

[1] S. Patnaik *et al.*, "Concerted wire lifting: Enabling secure and cost-effective split manufacturing," in *Proc. Asia South Pac. Des. Autom.*

Conf., 2018, pp. 251–258.

[2] C. McCants. (2016) Trusted integrated chips (TIC) program. [Online]. Available: https://www.ndia.org/-/media/sites/ndia/meetings-and-events/divisions/systems-engineering/past-events/trusted-micro/2016-august/mccants-carl.ashx

[3] B. Hill *et al.*, “A split-foundry asynchronous FPGA,” in *Proc. Cust. Integ. Circ. Conf.*, 2013, pp. 1–4.

[4] K. Vaidyanathan *et al.*, “Building trusted ICs using split fabrication,” in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2014, pp. 1–6.

[5] K. Xiao *et al.*, “Efficient and secure split manufacturing via obfuscated built-in self-authentication,” in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2015, pp. 14–19.

[6] Q. Shi *et al.*, “Securing split manufactured ICs with wire lifting obfuscated built-in self-authentication,” in *Proc. Great Lakes Symp. VLSI*, 2017, pp. 339–344.

[7] S. Patnaik *et al.*, “A modern approach to IP protection and Trojan prevention: Split manufacturing for 3D ICs and obfuscation of vertical interconnects,” *Trans. Emerg. Top. Comp.*, 2019.

[8] J. Magaña *et al.*, “Are proximity attacks a threat to the security of split manufacturing of integrated circuits?” *Trans. VLSI Syst.*, vol. 25, no. 12, 2017.

[9] J. Rajendran *et al.*, “Is split manufacturing secure?” in *Proc. Des. Autom. Test Europe*, 2013, pp. 1259–1264.

[10] A. Sengupta *et al.*, “Rethinking split manufacturing: An information-theoretic approach with secure layout techniques,” in *Proc. Int. Conf. Comp.-Aided Des.*, 2017, pp. 329–336.

[11] Y. Wang *et al.*, “The cat and mouse in split manufacturing,” *Trans. VLSI Syst.*, vol. 26, no. 5, pp. 805–817, 2018.

[12] B. Zhang *et al.*, “Analysis of security of split manufacturing using machine learning,” in *Proc. Des. Autom. Conf.*, 2018, pp. 141:1–141:6.

[13] H. Li *et al.*, “Attacking split manufacturing from a deep learning perspective,” in *Proc. Des. Autom. Conf.*, 2019, pp. 135:1–135:6.

[14] H. Li *et al.*, “Deep learning analysis for split manufactured layouts with routing perturbation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2020.

[15] L. Feng *et al.* (2018) The cat and mouse in split manufacturing. [Online]. Available: https://github.com/seth-tamu/network_flow_attack

[16] J. Magaña *et al.* (2018) Are proximity attacks a threat to the security of split manufacturing of integrated circuits? [Online]. Available: http://homepages.cae.wisc.edu/∼adavoodi/

[17] S. Patnaik. (2020) Layouts for Concerted WireLifting. [Online]. Available: https://github.com/DfX-NYUAD/Concerted_WL

[18] J. Knechtel *et al.*, “Protect your chip design intellectual property: An overview,” in *Proc. Conf. Omni-Layer Intell. Sys.*, 2019, pp. 211–216.

[19] T. D. Perez *et al.*, “A survey on split manufacturing: Attacks, defenses, and challenges,” *IEEE Access*, vol. 8, pp. 184 013–184 035, 2020.

[20] Y. Wang *et al.*, “Routing perturbation for enhanced security in split manufacturing,” in *Proc. Asia South Pac. Des. Autom. Conf.*, 2017, pp. 605–610.

[21] S. Patnaik *et al.*, “Raise your game for split manufacturing: Restoring the true functionality through BEOL,” in *Proc. Des. Autom. Conf.*, 2018, pp. 140:1–140:6.

[22] S. Chen *et al.*, “Improving the security of split manufacturing using a novel BEOL signal selection method,” in *Proc. Great Lakes Symp. VLSI*, 2018, pp. 135–140.

[23] A. Sengupta *et al.*, “A new paradigm in Split Manufacturing: Lock the FEOL, unlock at the BEOL,” in *Proc. Des. Autom. Test Europe*, 2019, pp. 414–419.

[24] F. Imeson *et al.*, “Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation,” in *Proc. USENIX Sec. Symp.*, 2013, pp. 495–510.

[25] M. Li *et al.*, “A practical split manufacturing framework for Trojan prevention via simultaneous wire lifting and cell insertion,” *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, vol. 38, no. 9, pp. 1585–1598, 2018.

[26] S. Patnaik *et al.*, “Best of both worlds: Integration of split manufacturing and camouflaging into a security-driven CAD flow for 3D ICs,” in *Proc. Int. Conf. Comp.-Aided Des.*, 2018, pp. 1–8.

[27] S. Patnaik *et al.*, “Obfuscating the interconnects: Low-cost and resilient full-chip layout camouflaging,” *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, vol. 39, no. 12, pp. 4466–4481, 2020.

[28] L. Feng *et al.*, “Making split fabrication synergistically secure and manufacturable,” in *Proc. Int. Conf. Comp.-Aided Des.*, 2017, pp. 313–320.

[29] (2011) NanGate FreePDK45 Open Cell Library. Nangate Inc. [Online]. Available: http://www.nangate.com/?page_id=2325

[30] N. Viswanathan *et al.*, “The ISPD-2011 routability-driven placement contest and benchmark suite,” in *Proc. Int. Symp. Phys. Des.*, 2011, pp. 141–146.

[31] A. B. Kahng *et al.*, “Horizontal benchmark extension for improved assessment of physical CAD research,” in *Proc. Great Lakes Symp. VLSI*, 2014, pp. 27–32. [Online]. Available: http://vlsicad.ucsd.edu/A2A/

Satwik Patnaik received the B.E. degree in electronics and telecommunications from the University of Pune, India, the M.Tech. degree in computer science and engineering with a specialization in VLSI design from the Indian Institute of Information Technology and Management, Gwalior, India, and the Ph.D. degree in Electrical engineering from Tandon School of Engineering, New York University, Brooklyn, NY, USA in September 2020.

He is currently a Post-Doctoral Researcher with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA. His current research interests include hardware security, trust and reliability issues for CMOS and emerging devices with particular focus on low-power VLSI Design. Dr. Patnaik received the Bronze Medal in the Graduate Category at the ACM/SIGDA Student Research Competition held at ICCAD 2018, and the Best Paper Award at the Applied Research Competition held in conjunction with Cyber Security Awareness Week, in 2017.

Mohammed Ashraf received the bachelor’s degree in electronics and telecommunication engineering from the College of Engineering Trivandrum, Thiruvananthapuram, India, in 2005.

He is a Senior Physical Design Engineer from India. He carries an experience of ten years in the VLSI industry. He has worked with various multinational companies like NVIDIA Graphics, Santa Clara, CA, USA, Advanced Micro Devices, Santa Clara, and Wipro Technologies, Bengaluru, India. He worked also with Dubai Circuit Design, Dubai Silicon Oasis, Dubai, United Arab Emirates. He is currently a Research Engineer with the Center for Cyber Security, New York University Abu Dhabi, United Arab Emirates. His work focus on the Physical Design/Implementation of the ARM Cortex M0 processor and its four secure variants.

Haocheng Li received the B.Eng. degree from the School of Information and Communications Engineering, Xi’an Jiaotong University, China, in 2016. He is currently pursuing a Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His research interests include electronic design automation, hardware security, and combinatorial optimization. He received the best paper award in ISPD 2017.

Johann Knechtel received the M.Sc. degree in Information Systems Engineering (Dipl.-Ing.) and the Ph.D. degree in Computer Engineering (Dr.-Ing., summa cum laude) from TU Dresden, Germany, in 2010 and 2014, respectively.

He is a Research Scientist with New York University Abu Dhabi, United Arab Emirates. From 2015 to 2016, he was a Postdoctoral Researcher with the Masdar Institute of Science and Technology, Abu Dhabi; from 2010 to 2014, he was a Ph.D. Scholar with the DFG Graduate School "Nano- and Biotechnologies for Packaging of Electronic Systems" hosted at TU Dresden; in 2012, he was a Research Assistant with the Chinese University of Hong Kong; and in 2010, he was a Visiting Research Student with the University of Michigan at Ann Arbor, MI, USA. His research interests cover VLSI physical design automation, with particular focus on emerging technologies and hardware security. He has (co-)authored around 50 publications.

Ozgur Sinanoglu is a professor of electrical and computer engineering at New York University Abu Dhabi. He obtained his Ph.D. in Computer Science and Engineering from University of California San Diego. He has industry experience at TI, IBM and Qualcomm, and has been with NYU Abu Dhabi since 2010. During his Ph.D. he won the IBM Ph.D. 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-for-security and design-for-trust for VLSI circuits, where he has more than 200 conference and journal papers, and 20 issued and pending US Patents. Prof. Sinanoglu is the director of the Center for CyberSecurity at NYU Abu Dhabi. His recent research in hardware security and trust is being funded by US National Science Foundation, US Department of Defense, Semiconductor Research Corporation, Intel Corp and Mubadala Technology.