Deep Learning Analysis for Split Manufactured Layouts with Routing Perturbation

Haocheng Li, Satwik Patnaik, *Member, IEEE*, Mohammed Ashraf, Haoyu Yang, Johann Knechtel, *Member, IEEE*, Bei Yu, *Member, IEEE*, Ozgur Sinanoglu, *Senior Member, IEEE*, and Evangeline F.Y. Young

***Abstract*—Split manufacturing of integrated circuits means to delegate the front-end-of-line (FEOL) and back-end-of-line (BEOL) parts to different foundries, in order to prevent overproduction, intellectual property (IP) piracy, or targeted insertion of hardware Trojans (i.e., threats arising from adversaries in the FEOL foundry). This paper challenges the security promise of split manufacturing by formulating various layout-level placement and routing hints as vector-based and image-based features that enable a sophisticated deep neural network (DNN), which can infer the missing BEOL connections with high accuracy. Compared with the network-flow attack [2], we achieve on average $1.21\times$ and $1.12\times$ of their correct connection rate (CCR; the higher, the better) when splitting after M1 and M3, respectively, with less than 1% of their runtime (across the same set of ISCAS-85 and ITC-99 benchmarks). Compared with [3], ours reduces the candidate list (the smaller, the better) by 47% with only 1% loss of accuracy, and we further achieve an average CCR of $2.2\times$ of that of [3]. Aside from these superior results, we propose a randomized, routing-blockage-centric defense strategy to escalate the resilience against our and other attacks. Our defense strategy, which can be integrated into any commercial design flow, leads on average to 22.78 *pp* (percentage points) degradation in CCR when compared with unprotected layouts, while inducing only 3.3% and 3.2% overheads on power and timing, respectively, within the same die outlines (i.e., zero area cost). The source code of our heterogeneous feature extraction is available at https://github.com/cuhk-eda/split-extract, and the source code of our DNN is available at https://github.com/cuhk-eda/split-attack.**

***Index Terms*—Split manufacturing, Hardware security, IP protection, Routing perturbation, Deep learning, Feature extraction, Very Large Scale Integration (VLSI)**

I. Introduction

HARDWARE becomes as vulnerable as software with the widespread globalization of design, synthesis, fabrication, and distribution of integrated circuits (ICs). Fabless design houses rely on offshore foundries for cost-effective access to advanced technology nodes, which enables various attack avenues on intellectual property (IP) for those outsourced foundries [4]. For example, malicious suppliers with complete knowledge of the exposed GDSII layouts can steal the designs and the underlying IP [5]. Attackers may also counterfeit defective ICs [6] or modify designs maliciously [7].

The IARPA agency advocated *split manufacturing* to safeguard chip designs from potentially malicious foundries [8]. An untrusted, high-end foundry fabricates the FEOL (i.e., the device layer and a few lower metal layers), whereas a trusted facility, which is low-end and possibly even in-house, integrates the BEOL (i.e., the higher metal layers) on top of the FEOL, all without noticeable impact on circuit performance [8], [9]. The untrusted foundries cannot get control of the full design while the fabless design houses can still benefit from access to the latest technology node.

However, splitting physical-design layouts as-is into FEOL and BEOL parts may fall short in terms of security. Traditional, security-oblivious design tools tend to place interconnected components close to each other in the FEOL layers and further wire them up through the BEOL layers using short paths [10], [11]. While delivering effective designs in terms of power, performance, and area, such an approach leads to some information leakage for the scenario of split manufacturing, where the structural information gathered from the FEOL layers can be utilized to infer the missing BEOL connections. This concept is known as *proximity attack* [12] and selected prior art for attacks and defense strategies is reviewed in Sec. II-A.

We believe (and demonstrate) that deep learning (DL) is a good match for attacking split manufacturing. Among other applications, DL has been used to increase the effectiveness and efficiency in gaming [13], object recognition [14], routability prediction [15], and design-for-manufacturability [16]. However, we caution that DL would have to handle a large variety of data for attacking split manufacturing. More specifically, vector-based data are ranging, e.g., from signed gate displacements to unsigned wirelengths and from integral pin counts to floating-point pin capacitances. Additionally, layout images can be used to encode routing segments and their directions as well as congestions. Such image-based data naturally constitute rich information that can be useful for an advanced attack. A limitation of current

This work is an extension of [1]. This work is supported in parts by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project No. CUHK14202218, and the Center for Cyber Security at New York University Abu Dhabi (NYUAD). Besides, this work was carried out in part on the HPC facility at NYUAD. The work of Satwik Patnaik was supported by the Global Ph.D. Fellowship at NYU/NYUAD. (*Corresponding authors: Haocheng Li and Satwik Patnaik.*)

Haocheng Li, Haoyu Yang, Bei Yu, and Evangeline F.Y. Young are 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; hyyang@cse.cuhk.edu.hk; byu@cse.cuhk.edu.hk; fyyoung@cse.cuhk.edu.hk).

Satwik Patnaik was with the Department of Electrical and Computer Engineering, Tandon School of Engineering, New York University (NYU), 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).

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).

neural network architectures is that they can only handle either vector- or image-based features, but not both types together. Accordingly, one challenge for our work is to combine vector-based and image-based features in a unified framework. Another limitation is that traditional two-class classifiers would only predict each possible BEOL connection's probability but not represent the physical-design reality that each sink pin is assigned to exactly one driver/source. Accordingly, when selecting the source with the largest predicted connection probability, traditional classifiers can be easily misled by outlying negative samples' predictions.

In this paper, we leverage DL to learn the characteristics of IC layouts thoroughly, exemplarily synthesized using the NanGate 45 nm Open Cell Library [17]. To the best of our knowledge, this is the first DL-based attack on split manufacturing that provides better results than the state-of-the-art, non-learning-based attacks [2]. Our methods also resolve the imbalance problem encountered by another learning-based attack [3], which has to use the same number of negative and positive samples. The primary contributions of our work are summarized as follows:

- We leverage deep learning (DL) for attacking split manufacturing. Using TensorFlow 2.1, we design and train a sophisticated deep neural network (DNN) architecture, which can predict the missing BEOL connections for an unknown FEOL layout with high accuracy.
- Our neural network makes use of vector-based and image-based layout features simultaneously. The feature structure is compatible with a wide range of designs while saving memory consumption and runtime.
- The proposed *softmax regression loss* allows our attack to directly and effectively select the most probable BEOL connection among the relevant candidates without suffering from an imbalance between positive and negative samples (as traditional classifiers would do).
- We further propose a randomized, routing-blockage-centric defense strategy which can be easily integrated into commercial design flows. The notion of this defense strategy is to prevent attackers from learning it, which we demonstrate, and we also demonstrate that it is effective against non-learning-based attacks.

The rest of the paper is organized as follows. Section II reviews selected prior art, outlines the threat model and provides the problem formulation. Section III describes our features for the DL attack. In Sec. IV, we illustrate the architecture and configuration of the proposed DNN, followed by the obfuscation strategy described in Sec. V. The effectiveness of both attack and defense are verified in Sec. VI. Section VII concludes the paper.

II. Preliminaries

A. Prior Art and Limitations

Rajendran *et al.* [12] demonstrated the first naïve proximity attack, where they leveraged the fact that interconnected modules are typically placed closed to each other and non-formation of combinational loops. The attack performed reasonably well for hierarchical designs with a few nets between the modules but showed limited success for flat designs and large layouts. Wang *et al.* [2] proposed an enhanced proximity attack based on a network-flow model. While constructing a flow graph, they set the weighted sum of the proximity on preferred and non-preferred routing directions as the edge cost and adopt the driver capacitance as the edge capacity. However, the network-flow formulation is relaxed to the naïve proximity attack once cell libraries have loose capacitance constraints. The attack also performs iterative edge removal when combinational loops occurred during recovery of the BEOL connections; this iterative work mode causes significant runtime. Zeng *et al.* [3] analyzed the security of split manufacturing on industrial designs with random-forest classifiers. However, their classifiers do not predict the BEOL connections directly, but generate only a list of candidates, which is often of considerable size. For instance, when attacking layouts split after metal layer 4 (M4 for short), their most successful classifier provides, on average several hundreds or even thousands of candidates for each broken connection; it can become practically impossible to retrieve all correct connections among those candidates. Additional details for prior work can be found in [4].

According to the various attack methods, several defense algorithms were proposed to escalate the security of split designs. Most defenses are focused on creating more candidates in the neighborhood of broken connections to complicate the proximity attack. Sengupta *et al.* [18] colored cells by connection or by type and placed cells with the same color into the same fence, seeking to decorrelate placement and connectivity. Magaña *et al.* [19] first added artificial routing blockages to the designated split layer after global routing and then performed global routing again, to make the routing tools elevate more wires over the split layer. However, their scheme can control the length/size of blockages only at the lower-left corner of each routing-grid bin, limiting the solution space for routing perturbation. Wang *et al.* [2] developed a security-driven placement-perturbation algorithm by obfuscating the cell placement based on the layer assignment after global routing. However, their placement perturbation caused large wirelength overheads.

B. Threat Model

Consistent with prior art [2], [3], we assume that the attacker has access to the full design information of the FEOL layers. Hence, the attacker can identify the gates and pins, the related FEOL routing, and the resulting but incomplete netlist. The attacker also knows the maximum load capacitances (from the cell library) and can estimate an upper bound for the delay. Further, consistent with most prior works, we assume that the attack occurs while chips are being fabricated. We acknowledge [20], where an oracle was leveraged to assist the attack on split manufacturing, but here we adopt the classical, stronger threat model where the chip is not available yet. Hence, oracle access is not available for the attacker. Finally, the attacker has a database of layouts generated similarly to the one under attack.

An attacker's objective in the untrusted FEOL foundry is to decipher the missing BEOL interconnects solely from

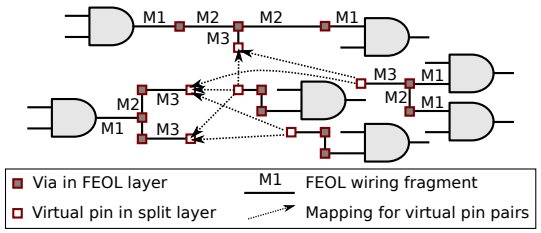


Fig. 1 Terms for learning on split manufacturing layouts. Examples of virtual pin pairs (VPPs) are shown by dashed arrows pointing from sink fragments to source fragments.

the available FEOL information. The corresponding goal is to reconstruct the design and ultimately pirate the chip IP, overproduce the chip, or insert targeted hardware Trojans.

C. Terminology and Problem Formulation

Split layer refers to the top-most FEOL layer, while *virtual pins* are vias manufactured to connect the FEOL with the BEOL [3]. During split manufacturing, *fragments* are connected parts of FEOL wires, holding at least one virtual pin in the split layer. There are two different types of fragments, as shown in Fig. 1:

- *Source fragment*: a driver/source along with fragments which are routed up until and within the split layer;
- *Sink fragment*: a fragment routed within the split layer and down towards sink pin(s). For multi-fanout nets, the sink pins may be routed together in the FEOL as one sink fragment or separately as several sink fragments.

Given are a set of m sink fragments, each of which has $c_1, c_2, \ldots, c_m$ sink pins, and a set of source fragments; all is easy to extract from the FEOL layout. *Virtual pin pairs* (VPPs) are mappings between virtual pins in sink fragments and virtual pins in source fragments. A VPP that is truly connected in the BEOL is called a *positive VPP*, whereas one that is not connected is called a *negative VPP*. The connection prediction problem is to select a VPP for each sink fragment maximizing the correct connection rate (CCR) which is the rate of sink pins that are successfully restored [2]:

$$CCR = \frac{\sum_{i=1}^{m} c_i x_i}{\sum_{i=1}^{m} c_i}, \tag{1}$$

where $x_i = 1$ (0) when a positive (negative) VPP is selected for the i-th sink fragment. Note that sink pins which do not belong to any sink fragment are excluded from consideration by definition, as this part of the design is already fully exposed in the FEOL. Accordingly, CCR serves well as a measure for attack effectiveness, but not so much for IP protection.

III. Feature Extraction

The BEOL part is only available at training time, where the true connectivity is extracted to label VPPs as positive or negative ones. The FEOL part is available for both training and testing/attacking phases. Hence, all features have to be extracted from the FEOL part. We propose two feature categories for our DL attack, namely vector-based and image-based features. The source code of our feature extraction is available at https://github.com/cuhk-eda/split-extract. We explain how to integrate these heterogeneous features into a unified DNN architecture in Sec. IV.

A. Vector-based Features

1) Distances for VPPs: These features are inspired by the working essence of design tools, where gates to be connected are typically placed closer to each other and related wires are typically routed along the shortest available path [10], [11]. Still, by the virtues of (a) being able to learn on various layouts and (b) the joint working of these and all other features proposed in this work, any deviation patterns from this essence can be captured as well. This is because all features have been devised to represent a physical layout in reasonable detail while remaining agnostic to particular design characteristics.

Following routing principles, the distances for VPPs arising along the preferred and non-preferred routing directions are considered separately. To mitigate scaling issues across different layouts used for the same model, instead of measuring distances by database unit, distances are normalized by the pitch of the metal tracks in the split layer. All distances are also duplicated and normalized separately by encoding in the ratios of the chip width or height, respectively. Therefore, designs based on different technology nodes and exhibiting different floorplan shapes and dimensions are made compatible with joint training as well as testing/attack.

2) Number of Sink Pins and Load Capacitance: These features track the number and total load of sink pins for each VPP. As we are handling split or incomplete layouts, the load capacitances can only be defined by two bounds, namely by

- an upper bound: maximum capacitance of the driver, as derived from the cell library (which is available to the attacker);
- a lower bound: capacitance of the sink pins connected within the sink fragment, plus wire capacitances of the two related source and sink fragments.

3) FEOL Layer Wirelengths and Vias: These features capture the wirelength contribution in each FEOL metal layer individually. Contributions are tracked separately for the two fragments of a VPP. Within each layer, all wire paths of a fragment are summed up. The number of vias in each FEOL cut layer is also considered.

4) Driver Delay: For each VPP, we track the driver delay based on the underlying timing paths. Note that timing paths obtained from split layouts can only provide lower bounds for delays, as the paths may be incomplete. Thus, this feature tends to become more meaningful for higher split layers when more of the paths are already completed in the FEOL.

B. Image-based Features

For each virtual pin, we represent the routing in the vicinity as gray-scale layout images. To be able to capture routing detours, we consider three different scales with the same image shape but different precisions, as shown in Fig. 2.

There are two properties of the routed wires which will be encoded in the layout images of a virtual pin: the nets they

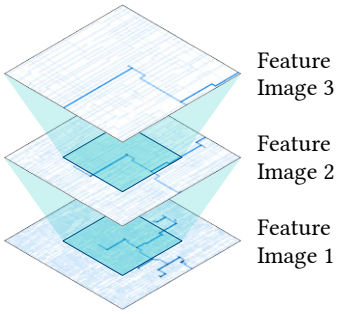


Fig. 2 Layout image scaling.

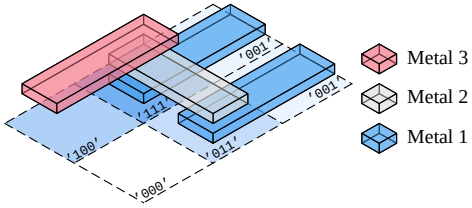


Fig. 3 Layout image representation.

belong to and the layers they are routed within. Let m be the number of metal layers in the FEOL. The total number of bits in a pixel to represent the layout information is $2m$, and we call these bits *layer bits*. The $2m$ layer bits are needed because wires of the same fragment as the virtual pin and wires from all other fragments are to be represented by different layer bits; the first m most-significant bits represent the routed wires of the virtual pin's fragment while the remaining m least-significant bits represent the wires of other fragments. Since wires closer to the BEOL carry more information about the missing connection, those in higher metal layers are encoded in more significant bits while those in lower metal layers are encoded in less significant bits. Vias connecting two layers are represented in both layer bits. More specifically, a '1' is assigned to the b-th bit with $b = m, \ldots, 2m-1$ in a pixel if the virtual pin's fragment is routed in metal layer $b-m+1$ in that region. Similarly, a '1' is assigned to the b-th bit with $b = 0, \ldots, m-1$ in a pixel if there is some wire or via arising from other fragments in metal layer $b+1$ of that region.

Fig. 3 shows an example with parts of the image data for a layout split after M3, i.e., wires in three different FEOL layers. Routed wires in the six consecutive regions bounded by the dashed lines are encoded into 2×3 pixels. Note that here we only show the values of the sixth, fifth, and fourth layer bits at the corner of each region, which together represent an exemplary virtual pin's fragment.

This image-based feature extraction for large designs with, e.g., more than a million fragments, will be time-consuming. Considering a fragment f, note that constructing a layout image of fragments other than f means to check which of all these other fragments hold wires in the nearby regions. Also note that the information carried by a layout image of fragment f and the image of fragments other than f is equivalent to the information carried by the layout image of f and the layout image of *all* fragments. Hence, to manage the computational efforts, we let the m least-significant bits represent the wires of *all* fragments instead of *other* fragments, i.e., a '1' is assigned to the b-th layer bit when the $b+m$-th bit is '1' where $b = 0, \ldots, m-1$. Thus, to construct the image-based features efficiently, we construct a large layout image for all nets covering the whole die area at the beginning. Thereafter, when generating a layout image centering any particular net, we only need to crop that large layout image to save most of the computational efforts incurred otherwise. Besides, for feature extraction using t threads, t large layout images can be constructed simultaneously and in parallel and then merged together.

IV. Deep Learning Framework

In this section, we first describe our strategy of VPP sample selection for data cleaning. We then elaborate on the DNN architecture and discuss our proposed SoftMax regression loss and its advantages. The source code of our DNN is available at https://github.com/cuhk-eda/split-attack.

A. Sample Selection

Due to an underlying tendency towards imbalanced datasets and long inference runtime, it is not practical to consider all possible VPPs, mainly because the correct connections are very few among all possible ones, which leads to a biased or inaccurate ML model. For N nets, even in the simplest scenario where (a) each FEOL wiring fragment holds only one virtual pin in the split layer and (b) each net is split into exactly one source and one sink fragment, the sampling size is already N^2, whereas only $\frac{1}{N}$ samples are true positives.

Thus, based on three criteria discussed next, we select the n most relevant candidate VPPs for each sink fragment, irrespective of the number of sink pins in the fragment.

The first is the direction criterion. We apply a looser criterion than [2] to avoid neglecting some positive VPPs, based on our observation that wires with non-preferred routing direction are relatively common in congested designs. For a VPP (p, q), where p is a virtual pin located at (x_p, y_p) and q is a virtual pin located at (x_q, y_q), if there is a wire segment between p and (x'_p, y'_p), and q satisfies

$$\begin{cases} (x_q - x_p)(x'_p - x_p) \leq 0, & y'_p = y_p, \\ (y_q - y_p)(y'_p - y_p) \leq 0, & x'_p = x_p, \end{cases} \tag{2}$$

we then say the virtual pin p does *not rule out* virtual pin q, meaning that the two related fragments might be connected in the BEOL. Our direction criterion is that a VPP is *not* considered as a candidate if and only if the above condition is *not met individually for both of the virtual pins.* In other words, a VPP is only disregarded if we find that neither the source fragment might be connected to the sink fragment nor vice versa. As indicated, this is a rather loose criterion, and particularly helpful to avoid neglecting some positive VPPs where parts of the related fragments are routed along non-preferred directions. Note that in case multiple virtual pins are present within a fragment, the condition is to be evaluated separately for each virtual pin. Also note that the final outcome of the direction criterion is independent of the order between virtual pins p and q; the criterion is symmetric.

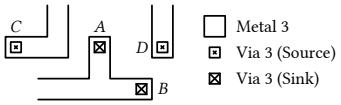


Fig. 4 Examples for the direction criterion. Except VPP (B, C), all other VPPs are considered as candidates.

TABLE I Direction Criterion for VPPs in Fig. 4

Virtual Pin p / Virtual Pin q	A / C	A / D	B / C	B / D
p does not rule out q	✓	✓	✗	✓
q does not rule out p	✗	✓	✗	✓
Direction Criterion	✓	✓	✗	✓

For illustration of this criterion, the exemplary VPPs in Fig. 4 are evaluated in TABLE I. For example, the wire of the source fragment connecting to the virtual pin C is pointing from right to left, while the virtual pin A of the sink fragment resides further to the right of C, so the condition in Equation (2) is not met and we cannot say that the source fragment might be connected to the sink fragment. For the counterpart evaluation, required to decide on the criterion, note that the wire of the sink fragmenting connecting to A is pointing upward, while the virtual pin C is on the same height as A (i.e., C is just not below A), and the condition is met. Therefore, the direction criterion is fulfilled, and VPP (A, C) is still considered as a candidate.

The second criterion is relevance. If the sink and source fragments have multiple virtual pins, only the VPP(s) with the shortest distance apart in the routing direction orthogonal to the preferred direction of the split layer is (are) considered as candidate(s). This is because metal stacks exhibit an alternating order for routing preferences and net wirelengths are restricted to meet timing closure. For example, consider the preferred routing direction in the split layer is horizontal, then the preferred direction for the next layer above the split layer—which is the first layer of the BEOL—is vertical. We assume that the first layer of the BEOL plays a significant role for the remaining wiring and, thus, the shortest distances in its preferred direction are leveraged in this criterion.

The third criterion is distance itself. If the number of VPPs remaining after considering the relevance criterion is still greater than n, the VPPs with shorter distance in the preferred direction of the first BEOL layer have a higher priority to be selected. Furthermore, if multiple VPPs are tied, the distance in the non-preferred routing direction is considered as a tie-breaker for the selection.

B. Model Architecture

For a batch of n VPPs selected for a sink fragment, the input data for the neural network include the vector-based features of n selected VPPs, the image-based features of n source fragments in the related VPPs, and the image-based features of the sink fragment itself. The output data are scores for every VPPs in the batch. To handle vector- and image-based features in the same network, the proposed neural network illustrated in Fig. 5 first extracts underlying

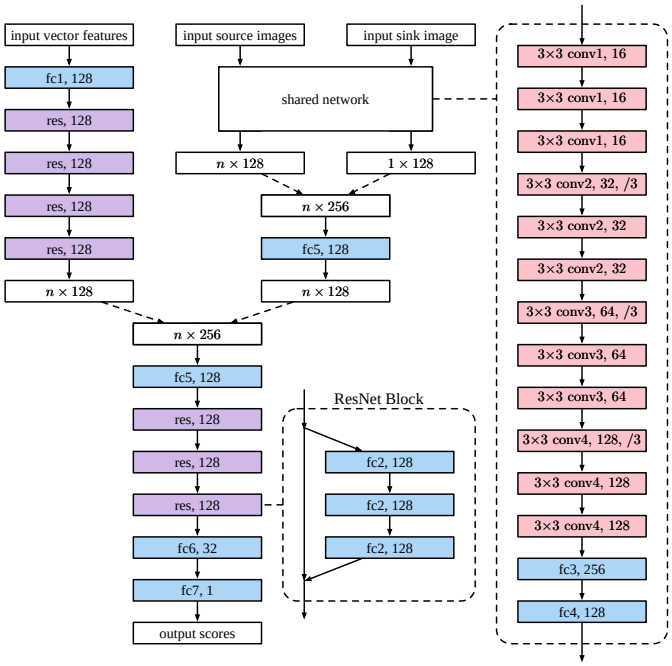


Fig. 5 Neural network architecture.

TABLE II Neural Network Configuration

Part	Layer	Parameter	Output
Vector part	fc1	27×128	$n \times 128$
	fc2	$[128 \times 128] \times 12$	$n \times 128$
Image part	conv1	$[3 \times 3, 16] \times 3$	$(n+1) \times 99 \times 99 \times 16$
	conv2	$[3 \times 3, 32] \times 3$	$(n+1) \times 33 \times 33 \times 32$
	conv3	$[3 \times 3, 64] \times 3$	$(n+1) \times 11 \times 11 \times 64$
	conv4	$[3 \times 3, 128] \times 3$	$(n+1) \times 4 \times 4 \times 128$
	fc3	128×256	$(n+1) \times 256$
	fc4	256×128	$(n+1) \times 128$
	fc5	256×128	$n \times 128$
Merged part	fc5	256×128	$n \times 128$
	fc2	$[128 \times 128] \times 9$	$n \times 128$
	fc6	128×32	$n \times 32$
	fc7	32×1	$n \times 1$

features from heterogeneous input by processing vector-based features (shown in the upper left) and image-based features (shown in the upper middle) individually, and then processing them together (shown in the lower left) after concatenating the output of the vector and image part together.

For the image part of the network, note that the image-based features of the sink fragment are the same in the batch, so we only process them once, to save runtime, and its output is distributed to the output of every source images. Besides, all the image-based features go through the same shared network because the same set of information is needed to be extracted. Thus, each image-based feature is first processed individually through a shared convolutional neural network to reduce runtime. Processing image-based features from source fragments and sink fragments through the same network can also make better use of all layout images. The shared network contains twelve convolution layers (red colored, labeled as `conv`) and two fully connected layers (blue colored, labeled as `fc`). The output from the sink image is then concatenated

with every output from the source images and the combination passes through one more 128-way fully connected layer. For the vector part of the network, vector-based features are first transformed by a 128-way fully connected layer. Then, there are four residual networks (ResNet) blocks (purple colored, labeled as `res`) which can resolve the gradient vanishing problem while training very deep neural networks [21]. The output of a ResNet block is the sum of its input and the output of three fully connected layers as shown in the middle sub-figure of Fig. 5. After that, the output from the image part is concatenated with the output from the vector-based features. There is one 128-way fully connected layer to down-size the combination. The network ends with three ResNet blocks and two more fully connected layers. The filter and parameter configuration of the neural network is listed in TABLE II. Both fully connected layers and convolutional layers are followed by a leaky rectified linear unit (LReLU) with $y = \max(0.01x, x)$ as activation, where x is the input and y is the output [22].

C. SoftMax Regression Loss

Given a query of a batch of n VPPs with at most one positive VPP, the network predicts the connection probability $s_1, s_2, \ldots, s_n$ for each VPP. The task for connection prediction is to determine the index of the correct VPP to be connected:

$$\arg\max_i s_i, \tag{3}$$

as there can only be one source in a net.

While prior work handles similar problems as multi-class classification or two-class classification, e.g., see [3], we note that conventional multi-class classification approaches are in lack of two important properties for our work. In fact, exponential effort would be required to conduct data augmentation if we were to use conventional multi-class classification methods. Firstly, the classification result for prior approaches depends on the order of classes, whereas for this work, the connection prediction should be independent of the order. Secondly, and more importantly, none of the prior methods can handle a variable number of classes, which is natural for the VPP connection prediction in our work, as this prediction is subject to a variable number of candidates.

Simply modeling the VPP connection problem as a two-class classification problem is not appropriate, either. The main difference between our problem and classical regression problems is that we only care about the relative predicted probability between the only one positive VPP and the remaining negative ones, instead of their absolute values. Consequently, only the VPP with the largest predicted probability matters in the result. Moreover, an outlying negative VPP prediction would easily mislead the matching. Assuming a traditional two-class classification formulation, where the input of the neural network contains n VPPs with the same sink fragment, the loss of the two-class classification is

$$l_r = -\frac{1}{n}\left(\log\frac{e^{s_t^+}}{e^{s_t^-}+e^{s_t^+}} + \sum_{j\neq t}\log\frac{e^{s_j^-}}{e^{s_j^-}+e^{s_j^+}}\right), \tag{4}$$

whose partial derivative with respect to each score of either class is

$$\frac{\partial l_r}{\partial s_j^+} = -\frac{\partial l_r}{\partial s_j^-} = \begin{cases} -\frac{e^{s_j^-}}{n\left(e^{s_j^-}+e^{s_j^+}\right)} & \text{if } j = t, \\ \frac{e^{s_j^+}}{n\left(e^{s_j^-}+e^{s_j^+}\right)} & \text{otherwise,} \end{cases} \tag{5}$$

where s_j^+ and s_j^- are the scores of connection and non-connection for the j-th source fragment with $1 \leq j \leq n$ and t is the index of the true connection. The partial derivative with respect to the i-th weight of either neuron in the last fully connected layer is

$$\frac{\partial l_r}{\partial w_i^+} = -\frac{\partial l_r}{\partial w_i^-} = \frac{1}{n}\left(\sum_{j=1}^{n}\frac{e^{s_j^+}x_{i,j}}{e^{s_j^-}+e^{s_j^+}} - x_{i,t}\right), \tag{6}$$

where $x_{i,j}$ is the i-th input value of the last fully connected layer for the j-th source fragment. Therefore, the score of each source fragment acts independently on the gradient. The coefficient of the positive part of the gradient, which is due to the negative samples, is limited to 1 so that the VPP with even the largest connection probability will not dominate the gradient. As a result, misprediction of one VPP, which would significantly influence our desired output as in Equation (3), barely affects the average loss. Additionally, the numbers of positive and negative VPPs are imbalanced as most of the VPPs are negative samples. The negative part of the gradient, which is due to the only positive sample, is divided by the number of VPPs in the batch. Therefore, such a two-class classification model has a serious imbalance problem as it can easily gain a high accuracy by simply classifying all VPPs as negative, which is meaningless.

To resolve these problems, we consider only one score s_j for the j-th source fragment with $1 \leq j \leq n$. We propose the following *SoftMax regression loss*

$$l_c = -\log\frac{e^{s_t}}{\sum_{j=1}^{n}e^{s_j}}, \tag{7}$$

whose partial derivative with respect to each score of connection is

$$\frac{\partial l_c}{\partial s_j} = \begin{cases} \frac{e^{s_j}}{\sum_{j=1}^{n}e^{s_j}} - 1 & \text{if } j = t, \\ \frac{e^{s_j}}{\sum_{j=1}^{n}e^{s_j}} & \text{otherwise.} \end{cases} \tag{8}$$

The partial derivative of our proposed loss with respect to the i-th weight of the only neuron in the last fully connected layer is

$$\frac{\partial l_c}{\partial w_i} = \frac{\sum_{j=1}^{n}e^{s_j}x_{i,j}}{\sum_{j=1}^{n}e^{s_j}} - x_{i,t}, \tag{9}$$

in which the shortcomings of conventional two-class and multi-class classification models are resolved as follows. Firstly, the source fragment with higher score contributes more significantly in the gradient with an exponential factor. Let j_{max} be the index of the largest s_j. As the positive part of the loss is dominated by $x_{i,j_{max}}$, we have $\frac{\partial l_c}{\partial w_i} \approx x_{i,j_{max}} - x_{i,t}$. Secondly, the summation of the coefficients in the positive part equals to that of the negative part, so there is no imbalance issue. Thirdly, given any permutation

of source fragments, the most probable source fragment is consistently selected. Fourthly, the network can handle any number of source fragments as input.

With these four advantages considered, the proposed Soft-Max regression loss better reflects our way of computing the output as in Equation (3), which is also supported by the empirical results.

V. Defense against Deep Learning Attack

Routing perturbations represent an effective means for security-aware physical design to protect split-manufactured layouts from proximity attacks [19]. In contrast, placement perturbations can incur large overheads and the perturbations are eventually offset by routing, rendering designs vulnerable, especially when split after higher layers [23], [24].

In this work, we seek to defend split-manufactured layouts by randomly inserting routing blockages within the FEOL metal layers. Since commercial tools from leading vendors employ deterministic physical-design algorithms, a DNN which is trained on a sufficiently large database of physical layouts can help capturing the essence of the behavior of those tools. Therefore, to ensure that advanced DL-based attacks (or any other attack) cannot easily circumvent the security promises offered by our defense, we shall introduce sufficient randomness during the layout generation. Given the same inputs and constraints, multiple design runs should provide sufficiently different solutions, to prevent attackers from learning the defense strategy. Still, all solutions have to remain fully compliant with design and manufacturing rules, which is achieved by employing commercial-grade tools.

It is understood that randomized routing-level perturbations will have an impact on the power, performance, and area (PPA) of the design and, hence, the degree of randomness should also remain controllable. Therefore, during the first step of our defense strategy, the designer has to provide the percentage of *g-cells* which shall be blocked at various layers. For example, assuming a split layer of M6, the designer should insert blockages throughout any layer(s) of choice below M6. Next, we identify the die and core boundary of the design and the size of a g-cell. The total number of g-cells is derived accordingly for all the layers where the designer seeks to insert blockages. Then, an iterative process is conducted as follows: a random layout location (x, y, z), snapped to the nearest g-cell location, is chosen, and a routing blockage of the same size as the g-cell is introduced into the design. This process is repeated until the blockage requirements specified by the designer are fulfilled. Note that we keep track of the number of blockages already added across the metal layers, also accounting for the preferred routing directions of those layers. We do so to guide the iterative process such that no bias is introduced (by random chance) toward a particular metal layer and/or a specific routing direction.

Once all routing blockages have been introduced into the design, the global router is invoked again, re-routing the blocked parts of all affected nets. Note that we freeze the placement, to support a fair PPA comparison and a fair security evaluation. Next, we perform a design rule check (DRC) for the re-routed solution and, once the design is devoid of any DRC violations, the routing blockages are removed again and the Design Exchange Format (DEF) is generated and streamed out for attack analysis. In case DRC violations are reported, which is expressed by an overflow of routing resources introduced by some particular blockages, we select among those violating blockages and iteratively remove some of them until a DRC-clean layout can be obtained. Exemplary layout snapshots for randomly inserted routing blockages and the re-routed, DRC-clean layout for the ITC-99 benchmark `b22_C` are shown in Fig. 6.

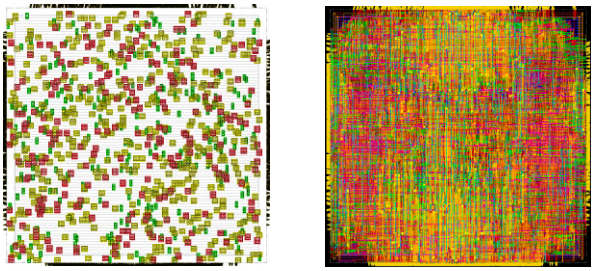


Fig. 6 Example of routing blockages inserted in ITC-99 benchmark `b22_C`. (Left) Routing blockages in green are randomly inserted for M3, in yellow for M4, and in red for M5, respectively. Note that colors for the background and for pins at the core boundary are value-inverted for better visibility. (Right) Re-routed layout after blockage insertion.

VI. Experimental Investigation

We conduct six sets of experiments as follows. In the first set, we evaluate the effectiveness of our proposed DL attack and compare it with the state-of-the-art network-flow attack [2] and the machine-learning attack [3]. In the second set, we compare the performance of our attack against the network-flow attack for a particular congested design. In the third set, we illustrate the impact of randomized insertion of routing blockages on the aforementioned attacks [2], [3]. In the fourth set, we evaluate the impact of blockages on timing-critical and congested designs. In the fifth and sixth sets of experiments, we analyze the layout cost as induced by routing blockages on regular ITC-99 benchmarks with timing-critical and congested versions, respectively.

We implement our feature extraction with C++ and train the model with Python and TensorFlow [25]. Without loss of generality, we select 31 VPPs for each sink fragment as the input of our DL attack based on the proposed criteria in Section IV-A. The learning rate is set as 0.001 and decayed to 60% for every 10 epochs. We execute all DL experiments on a 64-bit Linux machine with Intel Xeon 2.2 GHz CPUs and an NVIDIA Titan V GPU. We set the maximum runtime as 100,000 seconds (more than 24 hours) for all attacks and report CCR (Equation (1)) as the primary metric. Recall that CCR serves well as a measure for attack effectiveness, but not so much for IP protection. Besides, all the network-flow 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.

A. Evaluation and Comparison with State-of-the-Art Attacks

*1) **Setup**:* In the first set of experiments, we derive a total of nine training and five validation designs (all combinational ones) from the ISCAS-85 [26], MCNC [27], and ITC-99 benchmark suite [28]. Concerning testing layouts, we use the same benchmarks as mentioned in [2] to ensure a fair comparison. We guarantee that the training, validation, and testing layouts are derived from different designs.

We use the academic NanGate 45 nm Open Cell Library [17] with ten metal layers. *Synopsys Design Compiler M-2016.12-SP2* is used for synthesis; *Cadence Innovus 17.1* is used for placement and routing. All training, validation, and testing layouts are devoid of any DRC violations. Once a layout is generated, we export the DEF file and split the layout after M1 or M3, respectively, providing two sets to evaluate the attacks for different split layers. We also use this setup for all other sets of experiments, unless specified otherwise.

*2) **Results**:* We list the CCR for our proposed attack and the state-of-the-art attack [2] in TABLE III, where the results vary across the different designs. In general, the fewer nets are split, the fewer candidates are to consider for each fragment, and the higher the CCR tends to be. Besides, we note that design rules, timing constraints, and core utilization may also affect CCR, but for ours, interpreting the working of the DL attack in more detail would not be straightforward, as is the case with most DL models. We evaluate the success of the network-flow attack ourselves using the binary released in [29]. We note that the runtime of [29] exceeds the limit on several large designs (due to repetitive trials for removal of combinational loops). Our DL attack outperforms the state-of-the-art attack by $1.21\times$ and $1.12\times$ CCR when splitting after M1 and M3, respectively. Our inference time (including feature extraction) is significantly shorter, namely only $0.001\times$.

We further verify the effectiveness of our proposed softmax regression loss and image-based features. For these experiments, the baseline is using only the vector-based features with the loss Equation (4) for simple two-class classification. With the softmax regression loss in Equation (7), the average CCR is $1.07\times$ that of the baseline. When additionally employing the image-based features, the average CCR further improves to $1.09\times$. We note that using the softmax regression loss also marginally improves the runtime. Thanks to the efficient layout encoding and network structure, the runtime for further using the image-based features remains comparable to that of only using the vector-based features.

We also compare our method with another machine-learning attack [3]. Originally, the attack [3] provides only a list of candidates (LoC) for every fragment, no matter whether it is a source or sink fragment. We modify the code provided by [3] to only report the LoCs for sink fragments since an attacker can readily distinguish sink fragments from source fragments; this is relevant as an attacker needs to select a source for each sink fragment. We consider the three metrics proposed in [3]: (1) $|LoC|$ designates the average size of the identified list of candidates for each testing benchmark, (2) classification *accuracy* measures the number of times that the actual match of a fragment is included in its LoC, and (3) *success rate of proximity attack*, which is identical to CCR. We introduce a fourth metric, called *precision*, which is the fraction of actual matching among LoC, calculated as accuracy over $|LoC|$. TABLE IV provides the results for [3] and for our proposed attack. For ours, note that we select every VPP into the LoC whose score is higher than a reference value s_0; $s_0 = -8$ across all benchmarks. While achieving very similar accuracy, our $|LoC|$ is on average just $0.53\times$, meaning that we can correctly infer the actual match using much smaller LoCs. On average, we achieve $1.24\times$ the precision and even $2.20\times$ the CCR compared to [3].

Besides, we have also synthesized the designs considered in this section using an advanced technology node. We have conducted the *first-ever* attack on split manufacturing in the context of the 7 nm node, using the ASAP7 library [30]. Here we like to caution that it is not meaningful to directly compare the final CCR results across two nodes; the related technologies are quite different in many ways, including the cell types, numbers of metal layers, resistance and capacitance for each layer, design rules, etc., resulting in considerably different physical layouts. For example, the layouts obtained using the ASAP7 library exhibit, on average, around $5\times$ the number of source pins and $4\times$ the number of sink pins, when compared to the layouts obtained using the NanGate 45 nm Open Cell Library [17]. Accordingly, we observe that CCR efficacy tends to be more limited. Still, for larger designs such as the ITC-99 benchmark `b18_C`, our results obtained for the advanced node [30] even outperform those obtained for the mature node [17], which indicates that our DL framework is capable of handling large-scale, advanced, and more complex layouts. More results on attacking split-manufactured layouts of advanced nodes will be presented in future work.

B. Evaluation on Congested Design

*1) **Setup**:* In the second set of experiments, we execute our DL attack on the Low-Density Parity Check (LDPC) benchmark from [31], which is an inherently wire-dominated design and thus suitable for exploring the impact of congestion on our attack. We synthesize with a timing constraint of 5 ns (200 MHz) and place and route with the utilization of 15%.

While performing initial experiments, we noticed that the LDPC benchmark was unroutable, with around 17k DRC violations and many congestion hotspots forming only after the detailed placement stage. Upon investigation, we could attribute this to large counts of AOI22 cells, which are characterized by high pin densities, thereby not only inducing congestion but even hindering routing in the vicinity of many instances. Thus, we next employed a setup change as follows: AOI22 cells are disabled during synthesis, but no such restriction is imposed on *Cadence Innovus* during place and route. Doing so restored routability, resulted in DRC-clean layouts, all while allowing for some AOI22 instances to be introduced by layout optimization. Importantly for this set of experiments, the design remained congested, as confirmed per the congestion maps examined after placement.

TABLE III Comparison With [2] on Selected ISCAS-85 and ITC-99 Benchmarks

Design	Split Layer: Metal 1						Split Layer: Metal 3					
	# Sink Pins	# Source Pins	CCR (%) [2]	CCR (%) Ours	Runtime (s) [2]	Runtime (s) Ours	# Sink Pins	# Source Pins	CCR (%) [2]	CCR (%) Ours	Runtime (s) [2]	Runtime (s) Ours
b07_C	520	235	8.43	10.19	326.13	8.55	115	51	55.65	84.35	0.67	3.62
b11_C	738	296	9.05	10.03	1719.46	11.06	213	57	66.67	66.67	0.94	4.20
b13_C	430	215	10.42	17.91	130.82	7.53	88	52	42.05	70.45	0.44	3.55
b14_C	6338	2864	N/A	8.57	> 100000	77.62	2117	583	30.33	30.42	2576.42	16.08
b15_C	10176	3847	N/A	5.79	> 100000	130.30	4910	1235	26.42	24.24	38292.53	33.50
b17_C	32385	12479	N/A	4.08	> 100000	599.47	16190	4590	N/A	19.03	> 100000	157.61
b18_C	84292	33703	N/A	4.59	> 100000	2861.27	32719	9359	N/A	23.74	> 100000	453.66
c1355	403	226	9.90	12.41	151.22	7.65	77	32	89.61	97.40	0.50	3.53
c1908	432	213	8.49	11.11	260.50	7.45	54	27	94.44	87.04	0.47	3.34
c2670	803	428	6.32	9.46	2251.82	11.70	206	120	54.85	58.74	1.48	4.64
c3540	1354	512	6.41	8.49	39187.25	17.55	452	124	54.87	51.11	7.39	5.42
c432	231	121	11.26	8.23	15.62	5.29	43	21	76.74	86.05	0.37	3.35
c5315	1919	847	7.50	9.33	94281.90	23.59	590	248	52.20	62.03	26.11	6.81
c6288	4124	2160	N/A	14.52	> 100000	49.64	551	78	63.16	61.52	7.13	4.22
c7552	2008	1108	12.10	11.11	48656.51	22.82	296	175	50.34	72.30	7.64	3.72
c880	460	234	11.09	13.91	568.99	6.31	77	37	71.43	76.62	0.74	2.34
Average*			9.18%	11.11%	13889.37 s	10.67 s			59.20%	66.35%	2923.06 s	7.02 s
Ratio			1.00×	**1.21**×	1.000×	**0.001**×			1.00	**1.12**×	1.000×	**0.002**×

* For fairness, designs on which [2] times out are excluded for the calculation of average values.

TABLE IV Comparison With [3] on Selected ISCAS-85 and ITC-99 Benchmarks Split after Metal 3

Design	Accuracy (%) [3]	Accuracy (%) Ours	$\|LoC\|$ [3]	$\|LoC\|$ Ours	Precision (%) [3]	Precision (%) Ours	CCR (%) [3]	CCR (%) Ours
b07_C	94.12	96.08	15.96	12.27	5.90	7.83	35.29	52.94
b11_C	82.46	85.96	23.60	20.60	3.49	4.17	21.05	64.91
b13_C	100.00	98.08	18.44	14.42	5.42	6.80	28.85	44.23
b14_C	85.93	76.67	103.11	47.39	0.83	1.62	15.61	43.74
b15_C	83.56	78.06	178.85	86.15	0.47	0.91	8.18	33.52
b17_C	61.39	58.98	273.02	127.05	0.22	0.46	4.81	22.33
b18_C	54.91	50.25	209.84	74.45	0.26	0.67	4.01	23.29
c1355	100.00	100.00	11.88	10.22	8.42	9.79	40.63	78.13
c1908	100.00	100.00	12.15	10.30	8.23	9.71	29.63	81.48
c2670	95.83	97.50	31.16	29.95	3.08	3.26	31.67	53.33
c3540	91.13	89.52	42.18	29.39	2.16	3.05	11.29	64.52
c432	95.24	95.24	7.10	11.10	13.42	8.58	52.38	85.71
c5315	95.56	96.77	65.77	47.25	1.45	2.05	21.37	52.42
c6288	82.05	82.05	29.77	8.72	2.76	9.41	35.90	78.21
c7552	98.86	99.43	41.51	28.34	2.38	3.51	32.57	53.14
c880	100.00	100.00	14.22	10.49	7.03	9.54	32.43	62.16
Average	88.82%	87.79%	67.41	35.51	4.10%	5.08%	25.35%	55.88%
Ratio	**1.00**×	0.99×	1.00×	**0.53**×	1.00×	**1.24**×	1.00×	**2.20**×

We further perform iterative synthesis runs to generate ten different netlists, which are functionally equivalent but exhibit different gate-level implementations. We perform placement and routing for all eleven layouts, and arrange the layouts into ten for training and cross-validation, and one for testing.

*2) **Results**:* We execute the network-flow attack [2] on the LDPC benchmark considering M8 as the split layer; this particular layer had to be considered since the attack [2] already ran into time-out for the split layer of M6. The CCR obtained for the network-flow attack is 28.92%. Recall that, to handle heavily congested layouts, our image-based features are specifically devised to capture routing detours. Accordingly, our attack achieves a CCR of 39.63%, which is a notable improvement over the network-flow attack.

C. Routing Perturbation as Defense

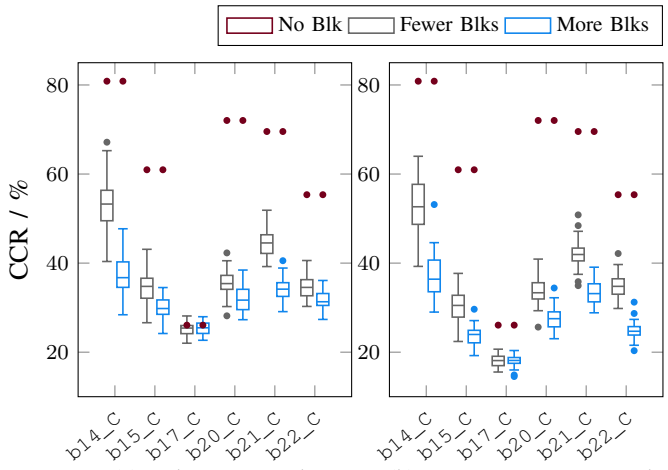
*1) **Setup**:* In the third set of experiments, we derive six combinational designs from the ITC-99 benchmark suite [28]. The essence for layout generation and the DL setup is described in Sec. VI-A1. Furthermore, the procedures for routing perturbation (Sec. V) are implemented as custom *TCL* scripts working with *Cadence Innovus 17.1*. We consider splitting after M6 and, hence, insert routing blockages in M3, M4, and M5, respectively. In case a different split layer is chosen by the designer, blockages can be added accordingly. Since we divide the number of blockages across three metal layers, out of which M3 and M5 are horizontal metal layers, and M4 is a vertical layer, more blockages are assigned to M4, while the remaining blockages are distributed uniformly across the horizontal layers. For the first batch of experiments, labeled as *Fewer Blks*, we block 12%, 22%, and 12% of the g-cells in M3, M4, and M5, respectively, while in the second batch (*More Blks*) we block 17%, 25%, and 17% g-cells for the same layers. In general, we add blockages such that timing overheads do not exceed 5% much, and all layouts are clocked at iso-performance of 5ns. For each design in these two batches, we generate 100 layouts with routing blockages inserted randomly following our proposed defense strategy. As indicated, we ensure that the final layouts remain routable after our perturbation procedures and are devoid of any DRC violations. We release our protected layouts in [32].

We perform a comparative analysis on the randomized insertion of routing blockages leveraging the proposed DL attack and the network-flow attack [2]. For our DL attack, we consider two different training approaches as follows. In the first approach, we pick 40 of the 100 layouts to train the DL model, cross-validate the model using ten other layouts, and attack the remaining 50 unseen layouts; all layouts arising from one design. For each benchmark under attack, a corresponding model is trained individually and that particular model is used only for attacking its respective benchmark. We refer to this as the *robust approach*; it represents the most rigorous approach for evaluating the strength of the defense, which can only be conducted by the

TABLE V Comparison With [2] on Selected ITC-99 Benchmarks Split after M6

Benchmark	No Blk		Fewer Blks			More Blks		
	[2]	Ours	# Blks	[2]	Ours	# Blks	[2]	Ours
b14_C	51.06	80.85	605	55.58	53.61	800	40.10	37.26
b15_C	56.16	60.96	916	31.88	34.50	1216	25.59	29.84
b17_C	24.24	26.07	2377	20.11	25.12	2687	20.90	25.40
b20_C	61.19	72.03	1271	36.12	35.67	1685	29.78	32.00
b21_C	56.40	69.55	1269	41.94	44.53	1682	32.98	34.15
b22_C	47.64	55.36	1874	32.92	34.73	2486	27.36	31.62
Average	49.45	**60.80**	–	36.42	**38.03**	–	29.45	**31.71**

The CCR results for our DL attack are obtained using the *robust approach*.



(a) Robust approach. (b) Leave-one-out approach.

Fig. 7 CCR results for our DL attack when splitting after M6, considering selected ITC-99 benchmarks, which are protected with randomly inserted routing blockages. For each benchmark and for each configuration/scenario, we have independently conducted 50 runs; all the results are summarized in boxplots. The upper and lower boundaries of each box span from the 5th to the 95th percentile for the respective data set, while the whiskers represent the minimal and maximal values, the bars inside the boxes represent the median, and the grey dots reflect outliers; all concerning the 50 runs for the respective configuration. Besides, red dots represent the attack results for the respective original, unprotected layouts.

security-enforcing designers itself, not by an actual attacker. In our second approach, we leverage a *leave-one-out* scheme as follows. Given six designs in total, we use five designs (with ten layouts each) to train a model that is then used to attack the one remaining, unseen design. Accordingly, a model is created for each design under attack, with 50 layouts available for learning. Note that an attacker can take such an approach.

*2) **Results**:* For both the network-flow attack [2] and our DL attack, we present the CCR results for layouts split after M6 in TABLE V. In the presence of the defense, our attack outperforms the network-flow attack in all cases. Next, we describe the findings in more detail.

First, we discuss the results for the *robust* learning approach. Again, we are considering this approach to thoroughly evaluate the strength of our routing-perturbation defense scheme. The corresponding CCR results are illustrated in TABLE V and Fig. 7 (a). The average CCR results for the layouts with less blocked g-cells ("Fewer Blks," grey bars) are 53.61%, 34.5%, 25.12%, 35.67%, 44.53%, and 34.73%, respectively. This corresponds to an average reduction of CCR by 22.78 percentage points (i.e., the arithmetic difference of percentage values, *pp* for short) across all benchmarks when compared to the original, unprotected designs. Once we block even more g-cells ("More Blks," blue bars), the CCR accuracy drops further: average CCR values are 37.26%, 29.84%, 25.4%, 32%, 34.15%, and 31.62%, respectively. This corresponds to an average reduction of CCR by 29.09 *pp* across all benchmarks, indicating the strength of the proposed defense even for this robust evaluation mode.

Next, we consider the regular *leave-one-out* learning approach where we assume that the design to be attacked is not available for training. The results for both batches of g-cell blockages are illustrated in Fig. 7 (b). The average CCR results for "Fewer Blks" (grey bars) are 52.24%, 30.58%, 17.91%, 33.45%, 41.55%, and 34.67%, respectively. This corresponds to a CCR reduction of 25.74 *pp* on average across all benchmarks. Increasing the number of blockages has a noticeable impact; the average CCR results for "More Blks" (blue bars) are 36.13%, 23.92%, 17.68%, 27.81%, 33.52%, and 24.55%, respectively, which corresponds to a CCR reduction of 33.53 *pp* across all benchmarks on average.

Overall, we note that the CCR can be reduced significantly by our randomized routing-perturbation defense. We also note that there is no significant difference for CCR results between the *robust* and the *leave-one-out* learning approach, which confirms the generality and efficacy of the models learned across different designs. For larger designs like b17_C, however, which are more difficult to attack in general (give the many fragments to be considered), we note that more blockages are more challenging to attack under the leave-one-out approach in particular. This demonstrates the effectiveness of our defense for large designs under the realistic attack/learning model.

We also perform similar experiments for the network-flow attack [2]. The corresponding CCR results are illustrated in Fig. 8. For the "Fewer Blks" batch (grey bars), the average CCR values are 55.58%, 31.88%, 20.11%, 36.12%, 41.94%, and 32.92%, respectively. Comparing these with the CCR values observed for original, unprotected layouts, we observe reductions by 13.02 *pp* on average across all benchmarks. Note that there is no significant reduction for benchmark b14_C; this is because the CCR result for the unprotected layout came out lower than expected, to begin with, i.e., when considering expectations arising from our DL attack (Fig. 7). For the "More Blks" batch (blue bars), as expected, the CCR numbers are further reduced: average CCR results are 40.1%, 25.59%, 20.91%, 29.78%, 32.98%, and 27.36%, respectively. Compared to the original layouts, this setup of blocking more g-cells provides better protection by enforcing lower CCR by 20.00 *pp* on average across all benchmarks. When comparing the network-flow attack with our proposed DL attack, our method outperforms in all cases, as also shown in TABLE V.

TABLE VI Increase in Via Counts for Our Proposed Defense on Selected ITC-99 Benchmarks

Benchmark				Fewer Blks			More Blks		
Name	Total Nets	Placement Util. (%)	Total Vias Before Lifting	Δ_+V67† After Lifting	Δ_+V78† After Lifting	Total Vias After Lifting	Δ_+V67† After Lifting	Δ_+V78† After Lifting	Total Vias After Lifting
b14_C	3009	70	17810	235	50	19478	370	129	20302
b15_C	4306	60	31347	623	313	36481	869	438	38376
b17_C	15477	70	113187	2166	1087	137754	2489	1210	142073
b20_C	7425	70	41455	799	359	47373	1107	556	49658
b21_C	7407	70	41377	684	254	45855	998	447	47907
b22_C	11439	65	62883	1145	497	71614	1603	783	75027

† This denotes the increase in the number of vias when compared to original, unprotected layouts, as averaged over 100 protected layouts.

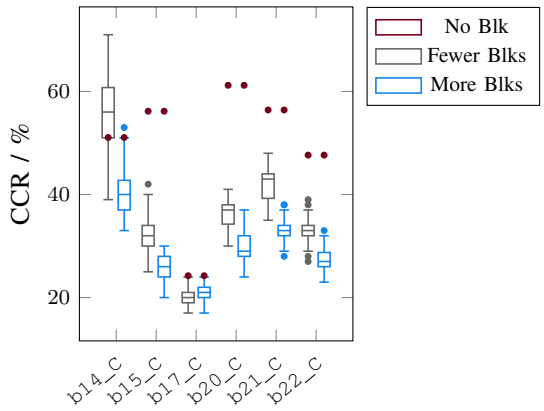


Fig. 8 CCR results for the network-flow attack [2] when splitting after M6, considering selected ITC-99 benchmarks, which are protected with randomly inserted routing blockages. Each scenario for each benchmark considered covers 50 runs. The interpretation of the boxplot is the same as Fig. 7.

D. Routing Perturbation as Defense on Congested and Timing-Critical Designs

*1) **Setup**:* In the fourth set of experiments, we evaluate the impact of randomized insertion of routing blockages on the security of timing-critical and congested designs, respectively. To mimic timing-critical designs, we synthesize the selected ITC-99 benchmarks for a 4 ns constraint (250 MHz frequency); we note that constraining at even faster timing (e.g., for 3 ns) violated some paths during synthesis. Concerning the congested design, we leverage the LDPC design following the same setup explained in Sec. VI-B1. We consider splitting after M6 for timing-critical ITC-99 benchmarks and insert routing blockages in M3, M4, and M5. In contrast, we consider splitting after M8 for LDPC and accordingly insert blockages in M4, M5, M6, and M7. We block 12%, 22%, and 12% of the g-cells in M3, M4, and M5, respectively, for timing-critical ITC-99 benchmarks, whereas we block 5%, 6%, 6%, and 5% g-cells for M4, M5, M6, and M7, respectively, for the congested LDPC benchmark. These numbers of blockages are without loss of generality but chosen carefully after multiple runs to ensure DRC-clean.

*2) **Results**:* The baseline CCR for unprotected layouts is 71.64%, 65.21%, 63.16%, and 56.35% for b14_C, b15_C, b20_C, and b22_C, respectively. Upon inserting the randomized routing blockages, we observe an average reduction of 30.37 *pp*, 35.6 *pp*, 25.69 *pp*, and 26.63 *pp*, respectively, when compared to these CCR baselines. The CCR for the unprotected LDPC benchmark is 28.92%, and invoking our defense strategy reduces the CCR to 25.42%. Although the drop in CCR is only at 3.50 *pp*, our defense helps to increase the absolute number of wrongly inferred connections significantly, and thereby increases the scale of IP protection. For example, splitting the original LDPC benchmark after M8 results in 2,743 cut nets, while for our defense technique, we note 3,629 cut nets, i.e., an increase of 32.3%.

E. Layout Costs Induced by Routing-Perturbation Defense

*1) **Setup**:* As demonstrated, the proposed defense is effective in hindering both the network-flow attack [2] and the our DL attack. In this fifth set of experiments, we investigate the timing and power costs incurred by this defense. Recall that we do not incur any overheads for die area. The analysis is carried out for the slow process corner, using a supply voltage of 0.95V. To ensure fairness for this layout evaluation (and the above security evaluation), we did "freeze" the placement of all the designs and introduced randomized routing blockages only to affect the routing of the layouts. Also, we add blockages such that the timing overheads do not exceed 5% much and such that no DRC violations occur. All layouts are clocked at iso-performance of 5ns.

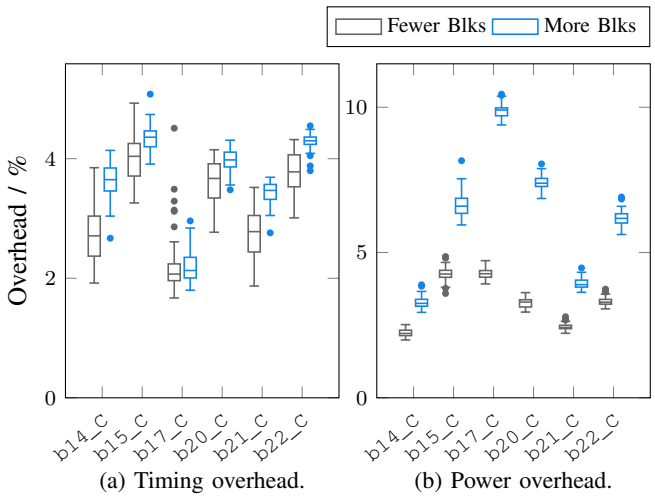
*2) **Results**:* The timing and power overheads for selected ITC-99 benchmarks are shown in Fig. 9. For the "Fewer Blks" batch, the power overheads are on average 2.24%, 4.21%, 4.29%, 3.27%, 2.43%, and 3.35%, respectively for b14_C, b15_C, b17_C, b20_C, b21_C, and b22_C over original, unprotected layouts. The average timing overheads for the same batch and same set of benchmarks are 2.71%, 3.97%, 2.16%, 3.61%, 2.7%, and 3.76%, respectively. Upon increasing the number of blockages ("More Blks"), we observe a steady increase in power: the average overheads are now 3.32%, 6.78%, 9.91%, 7.46%, 3.95%, and 6.15%, respectively. This increase is, as expected, particularly pronounced for larger designs like b17_C. The timing overheads, however, increase only marginally, to on average 3.67%, 4.32%, 2.21%, 3.97%, 3.42%, and 4.26%.

Since the insertion of routing blockages forces the router to lift the nets above the split layer and/or detour nets through regions where there are no blockages, an increase in the total count of vias (due to lifting of nets) and wirelength (due to detouring of nets) is expected. We confirm this by contrasting

TABLE VII Increase in Metal Wirelength for Our Proposed Defense on Selected ITC-99 Benchmarks

Benchmark		Fewer Blks				More Blks			
Name	Total Wirelength Before Lifting (μm)	Δ_+M6‡ After Lifting	Δ_+M7‡ After Lifting	Δ_+M8‡ After Lifting	Total Wirelength After Lifting	Δ_+M6‡ After Lifting	Δ_+M7‡ After Lifting	Δ_+M8‡ After Lifting	Total Wirelength After Lifting
b14_C	30540	2309	1537	362	33885	2710	1777	824	35273
b15_C	62470	3780	2676	2031	70193	5256	3415	2930	75012
b17_C	261088	14937	7100	4142	304556	15878	7387	4333	307821
b20_C	79397	5384	5225	2852	94243	7600	6139	4500	101268
b21_C	76678	5336	4418	1810	85760	6734	5340	2769	90437
b22_C	130983	9717	7640	4342	145396	13134	8841	6737	155888

‡ This denotes the increase in the respective metal layer wirelength when compared to original, unprotected layouts, as averaged over 100 protected layouts.



(a) Timing overhead. (b) Power overhead.

Fig. 9 Timing and power overheads for selected ITC-99 benchmarks for zero die-area overhead. Each scenario for each benchmark considered covers 50 runs. The interpretation of the boxplots is the same as Fig. 7.

these metrics for the unprotected layouts without blockages versus our protected layouts with blockages.

In TABLE VI we note an increase in vias for selected ITC-99 benchmarks. For the "Fewer Blks" batch, the increase in total vias are on average 9.37%, 16.38%, 21.7%, 14.28%, 10.82%, and 13.88%, respectively, for b14_C, b15_C, b17_C, b20_C, b21_C, and b22_C over original, unprotected layouts. Upon increasing the number of blockages ("More Blks"), the total increase in total vias for the same set of benchmarks are 14%, 22.42%, 25.52%, 19.79%, 15.78%, and 19.83%, respectively. These numbers attest to the fact that more nets have been lifted above the split layer for both configurations of the proposed routing-perturbation scheme.

Furthermore, the increases in wirelength are shown in TABLE VII. With "Fewer Blks," the average increase in total wirelength for the same set of benchmarks are 10.95%, 12.36%, 16.65%, 18.69%, 11.84%, and 11%, respectively. As the number of blockages are increased with "More Blks," the average increase rises to 15.5%, 20.07%, 17.9%, 27.55%, 17.94%, and 19.01%, respectively, which shows that larger parts of the nets reside in the higher layers.

F. Layout Costs Induced by Routing-Perturbation Defense on Congested and Timing-Critical Designs

*1) **Setup**:* In the sixth set of experiments, we evaluate layout costs for the ITC-99 benchmarks (where timing closure is performed at 4 ns to mimic timing-critical designs) and for the congested LDPC benchmark. For the ITC-99 benchmarks, all designs are generated considering an initial utilization of 70%. For the LDPC benchmark, setup details are the same as in Sec. VI-B1. For both benchmarks, blockages are iteratively added such that timing overheads do not exceed 5% much and such that no DRC violations occur.

*2) **Results**:* First, we discuss the timing and power overheads for ITC-99 benchmarks considering the aggressive timing closure. The power overheads are on average 2.82%, 5.39%, 8.25%, 2.59%, 2.48%, and 3.42%, respectively, for b14_C, b15_C, b17_C, b20_C, b21_C, and b22_C over original, unprotected layouts. The average timing overheads for the same batch and same set of benchmarks are 1.96%, 3.72%, 0.46%, 2.31%, 2.32%, and 2.77%, respectively. Second, we discuss the overheads incurred for the congested LDPC benchmark. We observe an instance increase of 2.07%, which translates to an increase in standard-cell area of 4.5%; note that this additional standard-cell area does not impact the die area. This increase in instance count and the increase in wirelength (4.49%) leads to a power overhead of 7.1%, albeit at a minimal timing overhead of 0.72%.

Thus, we conclude that our proposed routing-perturbation technique is feasible even for timing-critical and congested designs, and that the re-routing required after insertion of routing blockages is naturally imposing power overheads, by virtue of using additional buffer(s) and/or upsizing of standard cells, under the traditional objective of maintaining timing.

VII. Conclusion

In this work, we presented an effective and efficient attack on split manufacturing using deep learning. Firstly, we proposed suitable vector-based and image-based features as well as a neural network architecture that simultaneously processes these heterogeneous features. We further proposed a SoftMax regression loss that directly reflects on the accuracy for the virtual pin pair matching problem of split manufacturing and eliminates imbalance issues found in the prior art. Compared with the state-of-the-art network-flow attack [2], the correct connection rate (CCR) is improved by 21% and 12% when splitting after metal layers M1 and M3, respectively. Moreover, our attack's runtime is significantly better, namely less

than 1% when compared to [2]. We extended our comparison to [3], considering the size and accuracy for their notion of lists of candidates; for ours, the average size is significantly reduced, namely by 47%, while the accuracy is sacrificed only marginally, namely by 1%. Furthermore, our CCR is on average 2.2× that of [3]. For the first time, we also studied the prospects of attacking layouts split for an advanced node, where we found that our proposed attack performed relatively well for larger designs. Taking this motivating finding further will be the scope for future work. Finally, we proposed a randomized strategy for routing-blockage insertion, which degrades the effectiveness (expressed by CCR) for the network-flow attack and our deep-learning attack, with an acceptable impact on power and timing and zero overhead for die area.

REFERENCES

[1] H. Li *et al.*, "Attacking split manufacturing from a deep learning perspective," in *Proc. DAC*. IEEE, 2019, pp. 1–6.

[2] Y. Wang *et al.*, "The cat and mouse in split manufacturing," *IEEE TVLSI*, vol. 26, no. 5, pp. 805–817, 2018.

[3] W. Zeng *et al.*, "Analysis of security of split manufacturing using machine learning," *IEEE TVLSI*, vol. 27, no. 12, pp. 2767–2780, 2019.

[4] J. Knechtel *et al.*, "Protect your chip design intellectual property: An overview," in *Proceedings of the International Conference on Omni-Layer Intelligent Systems*, 2019, pp. 211–216.

[5] S. Patnaik *et al.*, "Obfuscating the interconnects: Low-cost and resilient full-chip layout camouflaging," *IEEE TCAD*, pp. 1–1, 2020.

[6] G. L. Zhang *et al.*, "TimingCamouflage+: Netlist security enhancement with unconventional timing," *IEEE TCAD*, pp. 1–1, 2020.

[7] Y. Jin and D. Pan, "Quantitative metric and automated toolset for obfuscated logic security evaluation," University of Florida Gainesville United States, 2020.

[8] 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

[9] K. Vaidyanathan *et al.*, "Building trusted ICs using split fabrication," in *Proc. HOST*, 2014, pp. 1–6.

[10] H. Li *et al.*, "Routability-driven and fence-aware legalization for mixed-cell-height circuits," in *Proc. DAC*, 2018, pp. 1–6.

[11] ——, "Dr. CU 2.0: A scalable detailed routing framework with correct-by-construction design rule satisfaction," in *Proc. ICCAD*, 2019, pp. 1–7.

[12] J. Rajendran *et al.*, "Is split manufacturing secure?" in *Proc. DATE*, 2013, pp. 1259–1264.

[13] D. Silver *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[14] B. Yang *et al.*, "Learning object bounding boxes for 3D instance segmentation on point clouds," in *Advances in Neural Information Processing Systems (NIPS)*, 2019, pp. 6737–6746.

[15] J. Chen *et al.*, "PROS: A plug-in for routability optimization applied in the state-of-the-art commercial EDA tool using deep learning," in *Proc. ICCAD*, 2020.

[16] B. Jiang *et al.*, "Neural-ILT: Migrating ILT to neural networks for mask printability and complexity co-optimization," in *Proc. ICCAD*, 2020, pp. 1–9.

[17] J. Knudsen, "Nangate 45nm open cell library," *CDNLive, EMEA*, 2008.

[18] A. Sengupta *et al.*, "Rethinking split manufacturing: An information-theoretic approach with secure layout techniques," in *Proc. ICCAD*, 2017, pp. 329–326.

[19] J. Magaña *et al.*, "Are proximity attacks a threat to the security of split manufacturing of integrated circuits?" *IEEE TVLSI*, vol. 25, no. 12, pp. 3406–3419, 2017.

[20] S. Chen and R. Vemuri, "On the effectiveness of the satisfiability attack on split manufactured circuits," in *Proc. VLSI-SOC*, 2018, pp. 83–88.

[21] K. He *et al.*, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[22] A. L. Maas *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, no. 1, 2013, p. 3.

[23] S. Patnaik *et al.*, "Concerted wire lifting: Enabling secure and cost-effective split manufacturing," in *Proc. ASP-DAC*, 2018, pp. 251–258.

[24] ——, "Raise your game for split manufacturing: Restoring the true functionality through BEOL," in *Proc. DAC*. IEEE, 2018, pp. 140:1–140:6.

[25] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. OSDI*, vol. 16, 2016.

[26] M. C. Hansen *et al.*, "Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering," *IEEE MDTC*, vol. 16, no. 3, pp. 72–80, 1999.

[27] (1991) MCNC benchmarks. [Online]. Available: http://vlsicad.cs.binghamton.edu/benchmarks.html

[28] F. Corno *et al.*, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE MDTC*, vol. 17, no. 3, pp. 44–53, 2000.

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

[30] V. Vashishtha and L. T. Clark, "A FinFET-based framework for VLSI design at the 7 nm node," *Energy Efficient Computing & Electronics: Devices to Systems*, p. 1, 2019.

[31] (2020) Reference community for free and open source IP cores. Opencores. [Online]. Available: https://opencores.org

[32] S. Patnaik. (2020) Layouts resilient against ML-based attacks. [Online]. Available: https://github.com/DfX-NYUAD/Randomized_routing_perturbation

Haocheng Li received the B.Eng. degree from the School of Information and Communications Engineering, Xian 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.

Satwik Patnaik (Member, IEEE) 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 bachelors 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.

Haoyu Yang received his B.E. degree from Qiushi Honors College, Tianjin University in 2015, and Ph.D. Degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong in 2020. He is currently working as a post-doctoral fellow in the same department at CUHK. His research interests include machine learning in VLSI design for manufacturability, high performance VLSI physical design with parallel computing and machine learning security.

Johann Knechtel (Member, IEEE) 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.

Bei Yu (Member, IEEE) received the Ph.D. degree from The University of Texas at Austin in 2014. He is currently an Assistant Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He has served as TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He is Editor of IEEE TCCPS Newsletter. He received six Best Paper Awards from ICTAI 2019, Integration, the VLSI Journal in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, ICCAD 2013, ASPDAC 2012, and five ICCAD/ISPD contest awards.

Ozgur Sinanoglu (Senior Member, IEEE) is a professor of electrical and computer engineering at New York University Abu Dhabi. He obtained his PhD 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 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. Sinanoglus 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.

Evangeline F.Y. Young received her B.Sc. degree in Computer Science from The Chinese University of Hong Kong (CUHK). She received her Ph.D. degree from The University of Texas at Austin in 1999. She is currently a professor in the Department of Computer Science and Engineering in CUHK. Her research interests include EDA, Optimization, algorithms and AI. Her research focuses on floorplanning, placement, routing, DFM and EDA on Physical Design in general. Dr. Young has served on the organization committees of ICCAD, ISPD, ARC and FPT and on the program committees of conferences including DAC, ICCAD, ISPD, ASP-DAC, SLIP, DATE and GLSVLSI. She also served on the editorial boards of IEEE TCAD, ACM TODAES and Integration, the VLSI Journal. Besides, her research group has won best paper awards from ICCAD 2017, ISPD 2017, SLIP 2017 and FCCM 2018, and several championships and prizes in renown EDA contests, including the 2018-20, 2015-16, 2012-13 CAD Contests at ICCAD, DAC 2012, and ISPD 2015-20 and 2010-11.