How Secure Is Split Manufacturing in Preventing Hardware Trojan?

YAJUN YANG, ZHANG CHEN, and YUAN LIU, ShanghaiTech University
TSUNG-YI HO, National Tsinghua University
YIER JIN, University of Florida
PINGQIANG ZHOU, ShanghaiTech University

With the trend of outsourcing fabrication, split manufacturing is regarded as a promising way to both acquire the high-end nodes in untrusted external foundries and protect the design from potential attackers. However, in this article, we show that split manufacturing is not inherently secure, that a hardware Trojan attacker can still recover necessary information with a proximity-based or a simulated-annealing-based mapping approach together with a probability-based or net-based pruning method at the placement level. We further propose a defense approach by moving the insecure gates away from their easily attacked candidate locations. Results on benchmark circuits show the effectiveness of our proposed methods.

CCS Concepts: • **Security and privacy** → **Hardware attacks and countermeasures**; • **Hardware** → *Design for manufacturability*;

Additional Key Words and Phrases: Split manufacturing, hardware trojan, proximity-based attak, simulated annealing attack

ACM Reference format:
Yajun Yang, Zhang Chen, Yuan Liu, Tsung-Yi Ho, Yier Jin, and Pingqiang Zhou. 2020. How Secure Is Split Manufacturing in Preventing Hardware Trojan? *ACM Trans. Des. Autom. Electron. Syst.* 25, 2, Article 20 (March 2020), 23 pages.
https://doi.org/10.1145/3378163

1 INTRODUCTION

The integrated circuit (IC) production flow has become globalized due to the high cost of owning a state-of-the-art manufacturing foundry and the complexity of the IC design. Although the separation of design and fabrication brings economic benefits to the design companies, it also raises significant concern on the security of the fabricated circuits, since potential threats come from all stages of the supply chain in the form of hardware attacks. As one of the most common attacks on ICs, Hardware Trojan (HT) [20] tries to insert additional malicious circuits into the original design to change the functionality of the circuit or steal the vital information such as secret key and user account. HTs may not only cause huge economic losses but also bring tremendous harm to the military, governmental, and public safety [2, 14]. The threats of HTs mainly come from the

Authors' addresses: Y. Yang, Z. Chen, Y. Liu, and P. Zhou, ShanghaiTech University, Shanghai; email: {yangyj, chenzhang, liuyuan1, zhoupq}@shanghaitech.edu.cn; T.-Y. Ho, National Tsinghua University, Hsinchu; email: tyho@cs.nthu.edu.tw; Y. Jin, University of Florida, Orlando; email: yier.jin@ece.ufl.edu.

1084-4309/2020/03-ART20 $15.00
https://doi.org/10.1145/3378163

following parties who engaged in the IC design and fabrication flow: IC design house, H/W IP vendor, CAD tool vendor, and foundry.

To counter these attacks, all stages of the design and fabrication flow need to take security into account [12] and the defense methods can be classified into two categories: HT detection and design for security (DFS). HT detection attempts to detect the potential HTs inserted into the circuit at either the post-silicon or the pre-silicon stage. Reverse engineering (RE) [24] is one of the post-silicon methods that de-packages an IC to obtain the microscopic images of each layer. The drawback of RE is that it is destructive, since the chip becomes ineffective after RE and the analysis result is often not applicable to other circuits. Therefore, pre-silicon methods such as logic testing and side-channel analysis [5, 9] have been proposed and such methods have shown to be more effective in detecting HTs than RE without destroying the chip. In contrast, the DFS tries to make the insertion of HTs difficult or increasing the insertion cost by intervening one or more of the whole production stages. For instance, the obfuscation-based methods protect the circuit by either obscuring its function from the designer's perspective or changing its structure from the CAD tool's side [21, 31]. Another prevailing method is split manufacturing [17]. Several chips have been fabricated successfully using this method [26, 27]. In split manufacturing, the whole design is split into two parts: (1) the Front End of Line (FEOL) consisting of transistors and lower metal layers, and (2) the Back End of Line (BEOL) consisting of top metal layers. Only FEOL layers need to be fabricated in an untrusted high-end foundry, while BEOL layers can be fabricated in a trusted low-end foundry. In other words, with split manufacturing, only the transistors and the limited number of connections in lower metal layers are exposed to potential threats, while the connections in top metal layers are hidden from the attackers.

Recently, References [16, 19, 29, 30] analyzed the security of a group of ISCAS-85, ITC-99 and ISPD 2011 benchmark circuits, which are assumed to be fabricated using split manufacturing method and an attacker is attempting to retrieve the missing BEOL connections from the FEOL connections. In Reference [19], the authors first assumed that the split layer is Metal 4 (M4), and then proposed an attack method named proximity attack to recover the hidden connections between the circuit partitions in the BEOL layers, by leveraging the local physical information revealed by the physical design tools. Their results show that they can retrieve 96% of the missing connections correctly. Wang et al. [29, 30] studied more general flattened designs and proposed the network-flow attack method to retrieve the missing connections. Results show that on average their method can retrieve up to 67% of the connections when varying the split layer. Reference [16] analyzed the effectiveness of different proximity-based techniques on identifying the candidate pins to be connected for each cutting net in the FEOL layers. Their results show that when split layer is M2, up to 82% of the candidate pin lists contain the correct pins to be connected.

To effectively retrieve most of the hidden BEOL connections, the aforementioned works need enough information of the FEOL connections as input, implying that the lowest split layer is M2. However the splitting layer has great impacts on the effectiveness of attack. For instance, in Reference [30] the correctly retrieved rate of BEOL connections for benchmark c880 is close to 100% when the split layer is M5 or higher, but it drops to 27% when the split layer changes to M3. This indicates that a lower split layer can obfuscate the attacker more efficiently. Vaidyanathan et al. [27] propose to split at M1, in which case all the FEOL connections are hidden, just leaving the attacker a sea of gates, thus could make the attacker difficult to recover the missing wires. But they only justify the security of splitting at M1 theoretically; however, our work shows that splitting after M1 still carries the threat of HT insertion.

In this work, we study the security of a circuit under HT attack when split layer is M1 and assume that the attacker can implant HT at multiple spots to guarantee success. We then come up with two metrics to quantitatively measure the security of a circuit under HT attack. For the

attack, we propose two approaches—one is adapted from proximity attack [19], by leveraging the heuristic that the connected gates tend to be close to each other. The other is based on simulated annealing (SA), which is widely used in placement stage in the back end design of chips to minimize the total wirelength [23]. These two approaches incorporate either local heuristic (such as logic connections) or global information (such as total wirelength) and leverage the common knowledge that minimizing total wirelength is a fundamental objective during placement. Furthermore, to counter our proposed attack methods, we provide a defense method that moves gates out of their easily attacked candidate locations.

The contributions of our work are as follows:

- Two metrics, namely, EMSR (Effective Mapped Set Ratio) and AMSPR (Average Mapped Set Pruning Ratio) are proposed to evaluate the security level of a circuit under HT attacks.
- Two attack methods are proposed in our work. The SA-based attack method that integrates global heuristic is presented in our conference paper [6], and in this article, we propose an improved proximity attack method that uses the local heuristic. The experimental results show that improved proximity-based attack method is effective in terms of finding the correct mapped set for each gate (i.e., the candidate gate set to implant HT), and improved proximity attack is faster than the SA-based attack.
- To further reduce the cost of the attacker and the risk of implanted HT being detected, we propose two pruning methods, i.e., probability-based pruning (see our conference paper [6]) and net-based pruning, to prune the mapped sets obtained after attacks.
- Finally, we propose a corresponding gate-swapping-based defense approach. Our results show that it can significantly improve the security of a circuit compared with the state-of-the-art methods.

The remainder of this article is organized as follows. In Section 2, we review the related work. In Section 3.2, we discuss our threat model and the problem formulation. We describe our security metrics and attack approaches in Section 4, and then present our defense method in Section 5. We show our experimental results in Section 6 and finally make conclusions in Section 7.

2 RELATED WORK

An example of targeted HT attack is shown in Reference [22], where the state of hardware registers is modified to maliciously raise privilege level. To achieve successful attack, the gate and wire that corresponds to the privilege bit need to be determined in the circuit layout. Thus, to support such targeted HT attack and reduce the attack cost, one needs both the complete gate-level netlist and the mapping of the gates in netlist to their physical locations in the layout [27].

Split manufacturing was first proposed to improve chip yield in early 2000s and more recently, it was used to enhance circuit security [29] by hiding the BEOL connections in layout from the attackers. Recently a few research works have analyzed the security of a circuit using split manufacturing [16, 19, 29, 30]. Proximity heuristic [19] has been proposed to retrieve the hidden connections in BEOL by exploiting the fact that connected pins should be placed close to each other. On top of proximity heuristic, Reference [29] further proposed a network-flow-based attack framework, which considers more information such as timing and load capacitance constraints. The results of their works show that for general split manufacturing with enough layers of metal connections included in FEOL, the attacker can reconstruct a large portion of the connections in BEOL, but the security can be enhanced by lowering the split layer, because less information is exposed to the attacker. In the extreme case, Reference [27] proposed to lower the split layer to be M1 so that the attacker can only see a sea-of-gates with no inter-cell connections. As for defense, pin swapping technique [19] is adopted to reduce the correctness of the reconstructed

BEOL connections, while placement perturbation-based defense is proposed in Reference [29] to make its network-flow-based attack less effective.

In our work, we study the security of a split manufactured circuit under HT attack, where the split layer is M1. Because there is no connection information available in the FEOL, we cannot directly use the methods in References [16, 19, 29, 30]. Since the attacker aims to implant HT in the circuit, to fulfill this task, he/she needs to do his/her best to find the possible locations of the target gates in the layout. In our work, we first adapt the proximity heuristic [19] together with the logical connection and size of gate in the netlist to reverse engineer the possible netlist-layout mapping in Section 4.3 to find the target gate, instead of using it to recover the hidden connections. Considering that proximity heuristic does not take global information such as total wirelength into account, we further propose a simulated-annealing-based approach for the attack. To quantitatively measure the security of a circuit under HT attack, where the attacker can implant HT at multiple spots to guarantee success, we come up two metrics, namely, EMSR and AMSPR. We also propose a corresponding gate-swapping-based defense approach.

Our work is mostly related to Reference [10], where a graph isomorphism method is proposed to obtain the netlist-layout mapping. Two connection graphs are, respectively, constructed—one for the netlist and the other for the physical layout of FEOL layers. Then, the mapping is obtained by graph isomorphism between the two connection graphs, which is unfortunately a NP-hard problem. Also the authors assume that the attacker can only exploits the proximity information of two connected pins. To evaluate the effectiveness of the attack, k-security metric is proposed in [10]: If a gate in the logical netlist can be mapped to k candidate locations in the layout, then this gate is k-secure, meaning that it cannot be distinguished from the other $k - 1$ gates. A wire lifting technique is then demonstrated to uplift the security of the circuit by lifting part of the wires to the trusted BEOL, with wirelength and area overhead of up to 2×. In contrast, in our work, all the wires are manufactured in the BEOL, and there is no wire available to be lifted. Thus, our gate-swapped-based defend method could increase the security of the circuit without any chip area overhead.

In the literature, in addition to k-security, there are many other metrics proposed to measure the effectiveness of both the attack and defense method under different threat models. Percentage of gates correctly extracted from a layout [24] and number of signals correctly matched [15] are leveraged to estimate the utility of reverse engineering. Reference [29] exploits the correct connection rate and output error rate to evaluate its attack and defense techniques. Reference [13] uses detection rate to measure the correctness of detecting HT. For the sake of IP piracy and IC overbuilding attack, hamming distance [16, 19, 32], number of brute force and number of input patterns [3] are used for measuring obfuscation. Reference [11] proposes four metrics, namely, neighbor connectedness, standard-cell composition bias, cell-level obfuscation and entropy, to evaluate the effectiveness of its obfuscation techniques. Although the above metrics are effective for their corresponding attack and defense techniques, they cannot be used directly for our work. For example, in our attack model described in Section 3.1, k-security [10] can only describe how hard for the attacker to insert HT for general gates, while we need metrics that can characterize how accuracy of inserting HT in a target gate and the associated cost as well.

3 PROBLEM FORMULATION

In this section, we first discuss our threat model, then present our problem formulation on HT attack.

3.1 Threat Model

As discussed in Section 2, the attacker wants to carry out a targeted HT attack, which requires getting the mapping between the gates in the netlist and their locations in the layout. While the

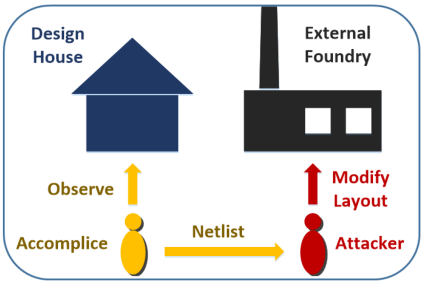

Fig. 1. Threat model.

attacker can implant HT at all possible locations, such strategy increases the workload of the attacker as well as the risk of implanted HT being detected. Therefore, the attacker should reduce the number of implanted HTs as much as possible.

In our threat model shown in Figure 1, the attacker can get help from two roles in two stages: a rogue element in the untrusted foundry who can modify the FEOL layout during fabrication and a malicious observer in the design stage who cannot do malicious changes on the design but has access to the precise gate-level netlist of the entire circuit. Our threat model aligns with the one used in Reference [10].

The reason that the attacker is assumed to be able to obtain gate-level netlist is that unlike software attackers, organizations that intend hardware attack are resourceful and are willing to pay for the related cost, because successfully implanted HT is capable of executing valuable attacks [7].

Moreover, to show the vulnerabilities of split-fabricated circuits to HT attacks, we assume that the circuit is split after M1, which gives the least information to the attacker. We also assume that primary inputs and outputs in the layout can be uniquely identified based on the specification of the design and can be correctly mapped to their counterparts in netlist without further efforts.

Finally, we assume that the attacker will not have access to the fully integrated IC thereby preventing any reverse engineering [27]. Under this assumption, the attacker could then insert malicious hardware in future batches of the IC as they are fabricated in the foundry [10].

3.2 Problem Formulation

With the physical layout of the FEOL layers and the gate-level netlist of the entire circuit, the goal of the attacker is to map the gates in netlist to the physical locations in layout to implant HT, which is similar to References [10, 33]. We use Figure 2 to illustrate our problem definition. Figure 2(a) is the graph corresponding to the gate-level netlist of a circuit. Figure 2(b) is the complete physical layout of this circuit, which shows the correct mapping between the gates in netlist (as shown in Figure 2(a)) and the physical gates in layout. Figure 2(c) is the layout that the attacker sees, with no inter-cell connections and all physical gates labelled differently.

Let $V_n = \{1, 2, 3, 4\}$ be the set of the gates in the netlist and $V_l = \{a, b, c, d\}$ be the set of the gates in the layout that the attacker sees, then the mapping problem is to find a mapping $\phi : V_n \rightarrow V_l$ that is close to the correct mapping $\phi_c : V_n \rightarrow V_l$. From Figures 2(b) and 2(c), the correct mappings are all bijective: $\phi_c(1) = a$, $\phi_c(2) = c$, $\phi_c(3) = b$, $\phi_c(4) = d$.

For the attacker, since he can get information from the untrusted foundry, he can reverse engineer [25] all the components in FEOL [30], after which he knows the gate type of all the physical gates in the layout. Consequently, the initial mapping for the attacker is $\phi_{ini}(x) = \{a, b, c\}$ for $x = 1, 2, 3$ and $\phi_{ini}(4) = d$. Based on the initial mapping, if his target is gate 3, then he has to im-

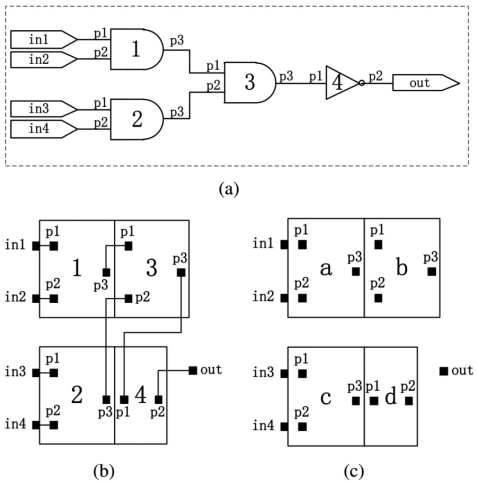

Fig. 2. (a) Logical connection corresponding to the gate-level netlist. (b) The complete physical layout with all gates correctly mapped to their counterparts in netlist. (c) The physical layout that the attacker sees.

plant HT at all three AND gates. Note that ϕ is not restricted to be bijective. The reason for this is obvious: the main goal of the attacker is to successfully implant HT, so if the location for the target cannot be uniquely identified, then multiple implantations are acceptable. We call $\phi(V_n(i))$ the *mapped set* of the ith gate in netlist, which contains all the possible physical gates for $V_n(i)$ to map to. $|\phi(V_n(i))|$ is the size of the mapped set of ith gate. For example, in Figure 2(a), the mapped set of gate "1" could be represented as $\phi(1) = \{a, b, c\}$, and $|\phi(1)| = 3$ means the number of gates it can be mapped to in the layout is 3. Finally, it is obvious that $|\phi(V_n(i))|$ may not always be 1, but the attacker can try to prune $\phi(V_n(i))$ to reduce the cost and risk of the attack.

Generally, let m be the number of gate types in a circuit, the mapping problem becomes finding m mappings $\phi_1, \phi_2, \phi_3, \ldots, \phi_m$ such that $\phi_j : V_n^j \rightarrow V_l^j$ is the mapping between the gates of type j in netlist and the physical gates of the same type in the layout. For any gate $V_n^j(i)$ of type j, its initial mapped set is the set of all the physical gates of type j. For example, in Figure 2(a), $m = 2$, since there are two types of gates, i.e., AND and NOT gate. And for AND gate "1," its initial mapped set is $\phi(1) = \{a, b, c\}$, which includes all the AND gates in the layout. Then the problem is to prune $\phi(1)$ in an appropriate way. We will present our solutions to the mapping and pruning problems in Section 4. After that, we will present our defense method in Section 5.

4 ATTACK METRICS AND FLOW

In this section, we first propose two metrics that are specific to our HT attack model and then present the overall attack flow.

4.1 Metrics

As discussed in Section 3.1, to achieve successful HT attack, the attacker should do his best to ensure that the mapped set for a target gate contains the correct target gate to implant HTs. Meanwhile, he also wants to reduce the size of the mapped set to be as small as possible to decrease the cost and risk of HT implantation. Obviously, the metrics discussed in Section 2 are not feasible for our attack model; thus, we propose the following two metrics to quantitatively measure the effectiveness of our attack approaches:

4.1.1 Effective Mapped Set Ratio (EMSR). A mapped set of a gate is effective only when it contains the correct location (i.e., candidate gate in layout) of the gate. If a mapped set does not contain the correct location, then it will mislead the attacker to miss the targeted spot and harms the effectiveness of the attack. EMSR is defined as the percentage of effective mapped sets among all the mapped sets. Formally,

$$\text{EMSR} = \frac{\sum_{i=1}^{|V_n|} |\phi(V_n(i)) \cap \phi_c(V_n(i))|}{|V_n|}, \tag{1}$$

where $\phi(V_n(i))$ is the mapped set of gate $V_n(i)$, $\phi_c(V_n(i))$ is the correct physical gate for $V_n(i)$. $|V_n|$ is the total number of gates in the circuit.

4.1.2 Average Mapped Set Pruning Ratio (AMSPR). AMSPR evaluates the average reduction ratio of a mapped set in a circuit after attack, which is calculated as the ratio between the reduced size of the set after the attack and the size of the original set. Formally,

$$\text{AMSPR} = \frac{1}{|V_n|} \cdot \sum_{i=1}^{|V_n|} \left(1 - \frac{|\phi(V_n(i))|}{|\phi_{ini}(V_n(i))|}\right) \cdot |\phi(V_n(i)) \cap \phi_c(V_n(i))|, \tag{2}$$

where $\phi_{ini}(V_n(i))$ is the initial mapped set for $V_n(i)$ that contains all the physical gates with the same gate type as $V_n(i)$. Note that if a mapped set $\phi(V_n(i))$ becomes ineffective during pruning, i.e., $\phi(V_n(i)) \cap \phi_c(V_n(i))$ is empty, then this mapped set would only mislead the attacker no matter how much it is pruned. Thus, if a mapped set becomes ineffective, then we set the pruning ratio of this mapped set to be 0, meaning that the pruning is fruitless.

Take the circuit shown in Figure 2 as an example, $V_n = \{1, 2, 3, 4\}$; $\phi_c(1) = a, \phi_c(2) = c, \phi_c(3) = b$, $\phi_c(4) = d$; $\phi_{ini}(1) = \{a, b, c\}$, $\phi_{ini}(2) = \{a, b, c\}$, $\phi_{ini}(3) = \{a, b, c\}$, $\phi_{ini}(4) = d$. Initially, the mapped set for each gate ϕ is equal to its initial mapping ϕ_{ini}; thus, according to Equation (1),

$$\text{EMSR} = \frac{1 + 1 + 1 + 1}{4} = 1.$$

Now, suppose after attack, the mapped sets reduce to $\phi(1) = \{a, b\}, \phi(2) = \{c\}, \phi(3) = \{a, c\}, \phi(4) = \{d\}$, then according to Equations (1) and (2),

$$\text{EMSR} = \frac{1 + 1 + 0 + 1}{4} = 0.75,$$

and

$$\text{AMSPR} = \frac{(1 - \frac{2}{3}) \cdot 1 + (1 - \frac{1}{3}) \cdot 1 + (1 - \frac{2}{3}) \cdot 0 + (1 - \frac{1}{1}) \cdot 1}{4} = 0.25.$$

Note that the EMSR reduces from 1 to 0.75 after attack because the correct mapping "b" was pruned out from the initial mapping set $\phi_{ini}(3) = \{a, b, c\}$ for gate "3." From this example, we can also see that pruning the mapped set is also a very effective way to reduce the attack cost.

In general, the attacker seeks to find the mapped sets that have both large EMSR and AMSPR—the former implies high attack correctness while the later indicates low attack cost.

4.2 Attack Flow

Figure 3 shows our attack flow, which mainly comprises three steps:

- **Analyzing:** First, since the attacker has the gate-level netlist and FEOL layout information, he/she will first obtain the initial mapped set for each gate in the netlist, that is, all gates of the same type in the FEOL layout. In our experiments, we show that the size of an initial

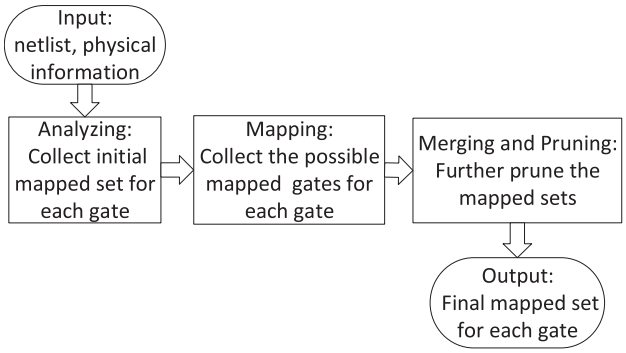

Fig. 3. Our attack flow.

mapped set can be larger than one thousand, which motivates us to propose novel techniques to prune the initial mapped sets. In our work, we prune the initial mapped sets in the following two steps.

- **Mapping:** This step aims to find enough number N of the possible netlist-layout mappings, by leveraging either the local physical heuristics in the layout or the common knowledge that minimizing total wirelength is a fundamental objective during placement. We will present two approaches for this step in Section 4.3. By merging all the found N mappings, we can get a reduced mapped set for each gate.
- **Pruning:** Finally, our experimental results show that the mapping step alone is not enough to reduce the size of the mapped set, so we propose two approaches (see Section 4.4) to further reduce the mapped sets at this step.

4.3 Mapping

Mapping aims to obtain a certain number of possible netlist-layout mappings. In our work, we have proposed both simulated annealing-based method and proximity-based method by leveraging the physical heuristics at different granularities. Simulated annealing method is widely leveraged in placement algorithms [4, 23]. Thus, we propose a SA-based mapping method that exploits the global heuristic that placement tools generally minimize the total wirelength of a circuit. For the details of this approach, the readers are referred to our conference paper [6]. In this section, we introduce the proximity-based method.

Before introducing the specific process of our proximity-based mapping method *PM*, some preliminaries need to be clarified:

- In physical layout, the gates of the same type (or size) have the same number of pins, and the offset location of each pin to the center of the gate is determined. For example, there are two groups whose gate type is the same in Figure 4(b): $\{a, b\}$ is the AND type and $\{c, d, e\}$ is the OR type. Each of AND type gates has three pins and the locations of these pins are determined, i.e., all these gates have pins $\{p1, p2, p3\}$ and the location of each pin is fixed.
- Since all the terminals have only one pin and their size can be neglected compared to the standard cells. Let terminalname$_n$ and terminalname$_l$ to represent terminal in netlist and layout, respectively. For example, $in1_n$ means terminal *in1* in Figure 4(a), $in1_l$ stands for terminal *in1* in Figure 4(b). Names of other terminals are similar. For other nonterminal gates, denote the ith pin of gate j as $pin_{i,j}$. For example, $pin_{1,1}$ and $pin_{1,a}$ mean the pin *p1* of gate 1 and pin *p1* of gate a, respectively, as shown in Figure 4. The pins having the same offset locations and the same associated gate types are called same-type pins. For example,

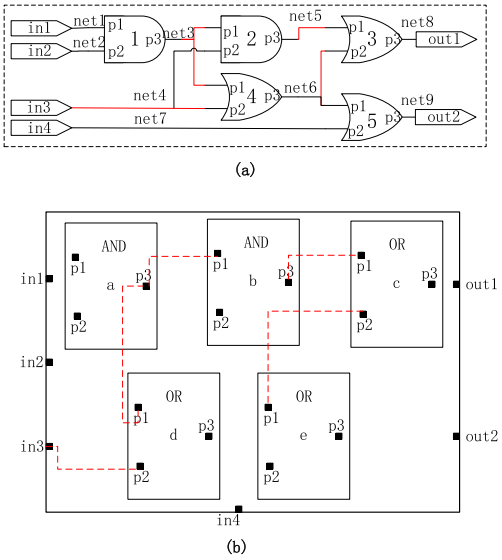

Fig. 4. Example of recovering netlist-layout mapping by proximity: (a) Logical connection corresponding to the gate-level netlist. (b) The complete physical layout that the attacker sees.

$\{pin_{1,a}, pin_{1,b}\}$ are the same-type pins, while neither $\{pin_{1,a}, pin_{2,b}\}$ nor $\{pin_{1,a}, pin_{1,c}\}$ is the same, since their gate type and pin location are not the same, respectively.

- Once a pin is mapped, the gate it belongs to is also mapped, and vice versa. For example, in Figure 4, if $pin_{1,1}$ is mapped to $pin_{1,a}$, then gate 1 is also mapped to gate a with all their pins are mapped to each other, respectively, i.e., $\{pin_{1,1}, pin_{2,1}, pin_{3,1}\}$ is mapped to $\{pin_{1,a}, pin_{2,a}, pin_{3,a}\}$, respectively, indicating that map a pin is equivalent to map a gate.
- All the terminals are already correctly mapped as explained in Section 3.2.A. For example, $\{in1_n, in2_n, in3_n, in4_n, out1_n, out2_n\}$ are mapped to $\{in1_l, in2_l, in3_l, in4_l, out1_l, out2_l\}$, respectively.

Proximity attack is first proposed in Reference [19] to recover the hidden connections of the circuit in the BEOL layers, it tends to connect the two closest pins as the missing wire. Instead of using proximity to recover the hidden connections, in our work, we propose to use the proximity idea to find the possible netlist-layout mappings. Except for leveraging the hints including input-output relationship and proximity mentioned in Reference [19], the logical connection information in the netlist and the type/size information of gates are also considered in the proposed mapping process. *PM* leverages these hints to find the pin's mapped pin to find gate's mapped gate: For example, considering net3 in Figure 4(a). This net has three relevant pins $\{pin_{3,1}, pin_{1,2}, pin_{1,4}\}$, assume gate "1" is already mapped to gate "a," which means pin $pin_{3,1}$is mapped to $pin_{3,a}$, consider the next pin to be mapped is $pin_{1,4}$ to find the mapped gate of gate "4." The proximity attack considers $pin_{1,b}$ be the most possible mapped pin of $pin_{1,4}$, since it is the pin nearest to pin $pin_{3,a}$ among all the other unmapped pin, thus map gate "4" to gate "b." While in the proposed PM process, the gate type of $pin_{1,4}$ is the OR type observed from the netlist, thus only the unmapped same-type pins of $pin_{1.4}$ in layout need to be considered, others can be directly pruned, i.e., choose a nearest pin from $\{pin_{1,d}, pin_{1,e}\}$ as the mapped pin of $pin_{1,4}$, finally PM take $pin_{1,d}$ as the mapped pin of $pin_{1,4}$ and map gate "4" to gate "d." Thus, in proximity-based mapping, a pin in netlist can only be mapped to the nearest same-type pins in layout. *PM* chooses an unmapped pin each time

to find its possible mapped pin in layout until all pins or gates are mapped. Once a pin is mapped, the gate it belongs to is also mapped, and vice versa. For example, in Figure 4, if $pin_{1,1}$ is mapped to $pin_{1,a}$, then gate "1" is also mapped to gate "a" with all their pins are mapped to each other, respectively, i.e., $\{pin_{1,1}, pin_{2,1}, pin_{3,1}\}$ is mapped to $\{pin_{1,a}, pin_{2,a}, pin_{3,a}\}$, respectively.

Here, we use the circuit in Figure 4 to illustrate the process of *PM* summarized in Algorithm 1:

ALGORITHM 1: Proximity-based Mapping

1: **Input**: Netlist, layout positions
2: **Output**: Mapping result of each gate
3: Initially, map all the terminal gates correctly as $G_{terminal}$;
4: $G_{mapped} \leftarrow \emptyset$, $g_{mapped} \leftarrow \emptyset$;
5: $p_{mapped} \leftarrow \emptyset$, $p_{next} \leftarrow \emptyset$;
6: **while** there exits unmapped gates or unmapped pins **do**
7: **if** $G_{mapped} = \emptyset$ **then**
8: Randomly choose a gate from $G_{terminals}$ as g_{mapped} and p_{mapped} whose belong to net has unmapped pins;
9: **else if** $G_{mapped} \neq \emptyset$ && all nets connected to g_{mapped} has no unmapped pins **then**
10: Randomly choose a gate from G_{mapped} as g_{mapped}, at least one of whose nets has unmapped pins, and ramdomly choose a pin from g_{mapped} as p_{mapped};
11: **end if**
12: $net_i \leftarrow$ RandomChoose(g_{mapped}'s nets who have unmapped pins);
13: $p_{next} \leftarrow$ RandomChoose(net_i's unmapped pins);
14: $p_{layout} \leftarrow$ FindNearestUnmappedSametypePin(p_{mapped});
15: map p_{layout} to p_{next};
16: map p_{next}'s gate to p_{layout}'s gate;
17: G_{mapped}.push_back(p_{next}'s gate);
18: $p_{mapped} \leftarrow p_{next}$;
19: $g_{mapped} \leftarrow p_{layout}$'s gate;
20: **end while**
21: // all gates mapped, get a possible mapping solution;
22: **return** Mapping result of each gate;

- First, map all the terminals correctly. $G_{terminal}$ stores the terminal gates, G_{mapped} stores all the previously mapped gates except for the terminal gates, while g_{mapped} stores the recent mapped gate, p_{mapped} is one pin of g_{mapped}'s pins, and p_{next} is the next pin to be mapped, which corresponds to lines 3–5. For example, in Figure 4, $\{in1_n, in2_n, in3_n, in4_n, out1_n, out2_n\}$ are mapped to $\{in1_l, in2_l, in3_l, in4_l, out1_l, out2_l\}$, respectively.
- Second, randomly choose one net as net_i from g_{mapped}'s nets who have unmapped pins, then one pin is randomly chosen from net_i's unmapped pins as p_{next}. Obviously, at the beginning, the g_{mapped} is one of the terminals, since only terminal gates are mapped corresponding to lines 7 and 8. Otherwise, the g_{mapped} is the recently mapped gate. However, if all nets of g_{mapped} have no unmapped pin, then a new gate should be chosen randomly as the g_{mapped} from previous mapped gates(i.e., G_{mapped}) and a pin is also randomly chosen from g_{mapped}, which alias to lines 9–11. For instance, in Figure 4, assume initially we choose $p_{mapped} = in3_n$ whose belonging to net is net_4 with its unmapped pins being $\{pin_{2,2}, pin_{2,4}\}$. According to lines 12 and 13, assume $p_{next} = pin_{2,4}$ after randomly choosing.
- Third, p_{next}'s mapped pin in the layout could be found, which is its unmapped same type pin nearest to p_{mapped}'s mapped pin, i.e., *PM* takes $pin_{2,d}$ as $pin_{2,4}$'s mapped pin, since it

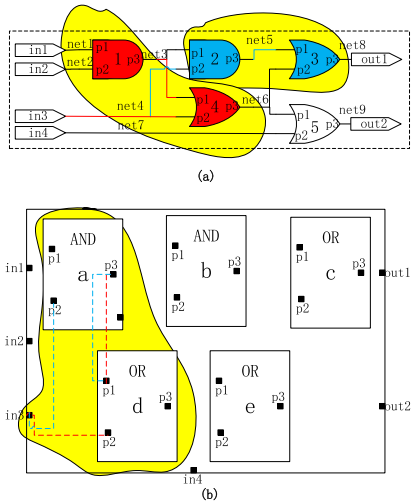

Fig. 5. Example of different mapping results obtained by PM.

is the nearest to $in3_l$ among its unmapped same type pins $\{pin_{2,c}, pin_{2,d}, pin_{2,e}\}$. Then map gate "4" to gate "d" and the remaining pins, i.e., $\{pin_{1,4}, pin_{3,4}\}$ is mapped to $\{pin_{1,d}, pin_{3,d}\}$, respectively, corresponding to lines 15 and 16.

- Finally, start from the recently mapped pin and gate according to lines 17–19, the above steps repeat until all the gates or pins are mapped.

The red line in Figure 4 shows the example mapping process of *PM*. Start from $in3_n$ (mapped to $in3_l$), then use the connection $\{in3_n, pin_{2,4}\}$ to map $pin_{2,4}$ (mapped to $pin_{2,d}$, and gate 4 is mapped to gate d), the remaining used connections are: $\{pin_{1,4}, pin_{3,1}\}$ (map $pin_{3,1}$ to $pin_{3,a}$, and gate "1" to gate "a"), $\{pin_{3,1}, pin_{1,2}\}$, $\{pin_{3,2}, pin_{1,3}\}$, $\{pin_{2,3}, pin_{1,5}\}$. Finally, the mapping result is that gates $\{1, 2, 3, 4, 5\}$ are mapped to gates $\{a, b, c, d, e\}$, respectively.

It should be pointed out that the mapping results obtained by *PM* are strongly dependent on the mapping order. For example, start from $in3_n$ in Figure 5, if the subsequent *next_pin* is $\{pin_{2,2}, pin_{1,3}\}$ alias to the the blue partition, then it will map gate $\{2, 3\}$ to gate $\{a, d\}$, respectively. While for the mapping order $\{pin_{2,4}, pin_{3,1}\}$ alias to the red partition, gate $\{4, 1\}$ is mapped to gate $\{d, a\}$, respectively. Thus, different mapping solutions can be obtained if PM process is run for multiple times.

4.4 Pruning

After the mapping step, the size of the mapped set for some gates may still be too large for the attacker. Take Figure 4 as an example, assume the $N = 5$ mapping solutions for $V_n = \{1, 2, 3, 4, 5\}$, $V_l = \{a, b, c, d, e\}$ is

$$G^{N\times|V_n|} = \begin{bmatrix} a & b & e & d & c \\ a & b & d & c & e \\ a & b & d & c & e \\ a & b & c & d & e \\ b & a & e & c & d \end{bmatrix}. \tag{3}$$

With $G_{i,j}$ being the mapped gate of gate $V_n(j)$ in ith mapping solution. After merging $G_{:,j}$, the mapped set of each gate is $\phi(1) = \{a, b\}, \phi(2) = \{a, b\}, \phi(3) = \{c, d, e\}, \phi(4) = \{c, d\}, \phi(5) = \{c, d, e\}$. We can see that only the size of mapped set of gate "4" is reduced. If the attacker would like to

insert HT at gate "3" successfully, then he/she still needs to insert the HT at all the gates in $\phi(3)$. Therefore, we propose two methods, probability-based pruning and net-based pruning, to further prune the mapped sets. Probability-based pruning is a statistical method, which assumes that a mapped gate is less possible if it occurs few times in the mapping solutions, we have described it in our conference paper [6]. Therefore, in this section, we only present the net-based pruning.

The probability-based pruning method considers only the occurrence probability of the elements in the mapped set; it may prune out the correct mapped gates, which occur a few times. For example, $\phi(3)$ excludes its correct gate $G_{4,3} = c$ after probability-based pruning, since $P_3(c) < P_b(3)$. To make the pruning more effective, net-based pruning also considers the fact that placement tools tend to make the wirelength of each net small. The main idea of net-based pruning is to only prune those mapped gates whose net wirelength is long and occur less, while other mapped gates are kept directly. We use the mapping solution of G in Equation (3) to show how net-based pruning works: There are total $K = 9$ nets in Figure 4(a). Here, we first consider $net5$, its connected gates in netlist are $V_n(net5) = \{2, 3\}$. The N mapped gates of each gate in $V_n(net5)$ is

$$G_{n5} = [G_{:,2}, G_{:,3}]. \tag{4}$$

Then HPWL of $net5$ of the N mapping solutions could be calculated according to the physical locations of the mapped gates:

$$\text{HPWLs}^T = [2 \quad 2.5 \quad 2.5 \quad 1 \quad 3]. \tag{5}$$

Finally, consider the corresponding mapping solutions, denoted as solution_A, whose HPWL satisfies the pruning condition:

$$\text{HPWL\_}i \quad > \text{Min(HPWLs)} + \beta * (\text{Max(HPWLs)} - \text{Min(HPWLs)}), \tag{6}$$

where $\beta \in [0, 1]$ is the parameter of net-based pruning. In terms of the example, we set $\beta = 0$, then when $i = \{1, 2, 3, 5\}$, HPWLs(i) satisfies equation(6), thus

$$\text{solution\_A} = \begin{bmatrix} G_{1,2} & G_{1,3} \\ G_{2,2} & G_{2,3} \\ G_{3,2} & G_{3,3} \\ G_{5,2} & G_{5,3} \end{bmatrix}. \tag{7}$$

Those mapped gates of $V_n(net(i))$ among solution_A are thought to be less possible, since the HPWL of each net tends to be longer. And they need to be further pruned by probability-based method. While those do not satisfy are directly kept in the mapped set: namely, $G_{4,2}$ and $G_{4,3}$ are kept in $\phi(2)$ and $\phi(3)$, respectively. Thus, reserving the correct mapped gate of "3" in $\phi(3)$. For each net, the above process executes until all the nets are processed. Finally, merge all the remaining mapped gates, and the final pruned mapped sets for each gate are obtained. In this way, net-based pruning method prunes out the impossible mapped gates by using both probability and net-HPWL heuristic. It is obvious that β dominates the percentage of mapped gate to be pruned, and the larger β is, the less solutions are included in solution_A. The proper value of β can be determined in the experiment.

The difference between pruning and the previous mapping process discussed in Section 4.3 lies in their different goals: Mapping aims to find all possible locations for a gate so that the attacker will not miss the real location, while pruning aims to further prune the possible locations to reduce the cost and risk of attacks. We refer to the possible locations after pruning as the candidate locations for a gate. The attacker may implant HT either at all or stochastically at any subset of these locations, based on his attacking capability.

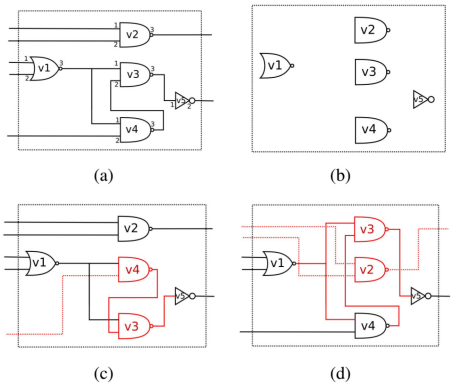

Fig. 6. Example circuit for defense: (a) original circuit, (b) FEOL for split after M1, (c) circuit with v3, v4 swapped, and (d) circuit with v2 and v3 swapped.

5 DEFENSE

To fulfill the security provided by split manufacturing, physical design techniques need to be deployed in the placement stage to reduce the information revealed by design heuristics. Local movement of gates [30] is usually applied to disrupt the proximity of connected gates. While it performs well against greedy proximity attack, its performances are not naturally generalized to global information-based attack. To see this, we use the example circuit in Figure 6. There is one NOR2, one INV, and three NAND2 gates in this circuit. Figure 6(a) is the original layout. We assume that the fabrication is split after M1 layer, so the attacker cannot see any inter-cell connections just as in Figure 6(b). Despite the absence of connections, the attacker still can correctly reconstruct most connections even without gate-level netlist, because the circuit is well organized in terms of pin positions. Now, if we only consider disrupting pin proximity and swap v3 with v4 as in Figure 6(c), then the attack effectiveness is mitigated. But for an HT attacker who has netlist, he can get around the intentionally misleading pin positions by using the global wirelength as the heuristic: When swapping v3 with v4, the global wirelength of the circuit is only affected smally, so the attacker will treat v3 and v4 as mutually interchangeable gates and implants HT at both locations. Although the cost of the attack increases, the effort in defense is much more nullified. However, if the defender swaps v2 with v3 as shown in Figure 6(d), then the global wirelength is significantly changed and the attacker will not regard this placement as a possible one, which prevents v3 from being correctly attacked. Therefore, although it is undesirable to have increased wirelength, for the sake of security in critical chips, there is a tradeoff for us to explore sometimes.

To counter the threat from a more powerful HT attacker, we propose a defense method that incorporates global wirelength information. Figure 7 shows our proposed defense workflow. The goal of this defense is to hide gates from their candidate locations, which means to decrease EMSR. As the defender is required to know the candidate locations obtained by the attack, he first needs to go through the entire attack process to collect candidate locations for each gate. Then a greedy gate-swapping-based defense algorithm is used, which is shown in Algorithm 2. The goal is to swap the locations of gates so that they are not in one of their candidate locations obtained by the attack.

We start from the gates with the fewest possible candidates, because they are the most insecure [10]. For each gate g, we find its out-of-candidate gates, which is defined as the union set of each gate g_o that is with the same gate type as g but not in the candidate locations of g. Then, the security elevation for the location swapping between g and any g_o is measured using Algorithm 3. The reason we only consider swappings between same-type gates is that in this case, no matter

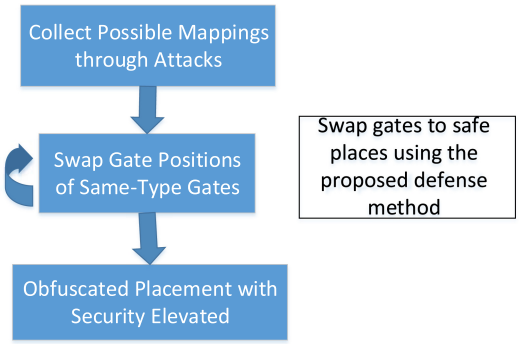

Fig. 7. Defense flow.

ALGORITHM 2: Greedy Gate-swapping-based Defense

Input:
The candidate locations for each gate ϕ, the original placement
Output:
The placement with improved security
$G \leftarrow V_n$
Ascendingly sort all the gates in G based on the number of their candidate locations
while $G \neq \emptyset$ **do**
  $ToSwap \leftarrow \emptyset$
  Pop the first gate $V_n(f)$ from G and add it to ToSwap
  Find all the gates in G whose number of candidate locations equals $|\phi(V_n(f))|$, pop them from G and add into ToSwap.
  while $ToSwap \neq \emptyset$ **do**
    for each gate g in ToSwap **do**
      if $g \notin \phi(g)$ or $|\phi(g)| = S(g)$ **then**
        Pop g from ToSwap
      else
        for each gate g_o such that $g_o \in S(g)$and $g_o \notin \phi(g)$ **do**
          Get the security elevation and wirelength increase if g swaps its location with g_o
        end for
      end if
    end for
    Get the pair of g and corresponding g_o that gives highest nonzero security elevation. If multiple pairs have the same security elevation, then get the one with least wirelength increase
    Swap the locations of g and g_o
    Pop g from ToSwap
  end while
end while
return The placement with improved security;

how swappings are made, the candidate locations for each gate remain unchanged. So if a gate is swapped to an out-of-candidate location, its security elevation will not be nullified by swappings of other gates. The calculation of security elevation is based on the goal that we want to move as many gates to their out-of-candidate locations as possible. For a gate g, if it is not in an out-of-candidate location, then its probability of being correctly mapped by random guessing is $\frac{1}{\phi(g)}$. After being swapped to an out-of-candidate location, its probability of being correctly mapped becomes

ALGORITHM 3: Security Elevation Calculation

Input:
Two gates g and g_o
Output:
The amount of security elevation if the locations for g and g_o are swapped
if $g \notin \phi(g_o)$ and $g_o \notin \phi(g)$ **then**
SecurityElevation = $2 + \frac{1}{|\phi(g)|} + \frac{1}{|\phi(g_o)|}$
else if $g_o \notin \phi(g_o)$ and $g \in \phi(g_o)$ **then**
SecurityElevation = $\frac{1}{|\phi(g)|} - \frac{1}{|\phi(g_o)|}$
else
SecurityElevation = $1 + \frac{1}{|\phi(g)|}$
end if
return SecurityElevation;

0. So its security elevation is computed as $\frac{1}{\phi(g)}$. However, we want to enforce more gates to out-of-candidate locations, so we add security elevation by 1 if a gate is moved from an in-candidate location to an out-of-candidate location.

Note that the gate-swapping defense is executed after the placement process having been completed. By swapping the same type of gates, the chip area will remain the same, since the two swapped gates in the defense process is the same kind of standard cells, which means the area cost is zero when uplifting the security of the circuit. However, the wirelength overhead and the security level could be balanced demonstrated in Section 6.

6 EXPERIMENTAL RESULTS

6.1 Experimental Setup

The proposed techniques are evaluated on eight circuits from ISCAS-85 benchmarks [8] and two larger scale circuits from ITC-99 benchmarks [18]. OSU technology library [1] is used for synthesis and placement is performed by wirelength-driven placer FastPlace3 [28] for the ISCAS-85 benchmarks. The algorithms run on a Linux machine with 8 Intel i7-3770 CPU cores with 3.4 GHz frequency and 24 GB memory.

6.2 Proper Number of Mappings N

As stated in Section 4.2, we run mapping algorithms to find N netlist-layout mappings to eliminate improbable locations while letting the correct locations to be included in the mapped sets. If N is too small, then correct locations are likely to be excluded from mapped sets. However, if N is too large, then it would be a waste of time after the correct locations are already included. Thus, the proper number of mappings, N, needs to be figured out first.

In our experiments, we found that N is related to the maximum number of same-type instances in the circuit, because they have the most possible locations to choose from. Table 1 shows the physical information of all the benchmarks and the value of N in our experiments. A suitable choice for N is two times of $\#max\ same-type$, since this can guarantee the most of these benchmarks to reach >95% intial EMSR.

6.3 Effectiveness of Attack

6.3.1 Effectiveness of Two Mapping Methods: PM and SA. After obtaining N netlist-layout mappings by either proximity-based mapping or SA-based mapping method, the mapped set for each gate of the circuit can be obtained by merging all these N mapping solutions, respectively. Then,

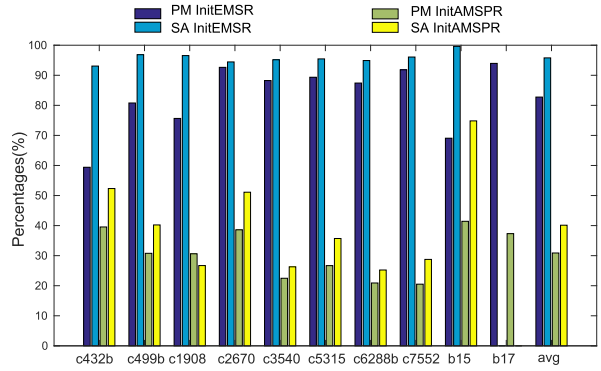

Fig. 8. EMSR and AMSPR after applying two mapping methods: PM and SA.

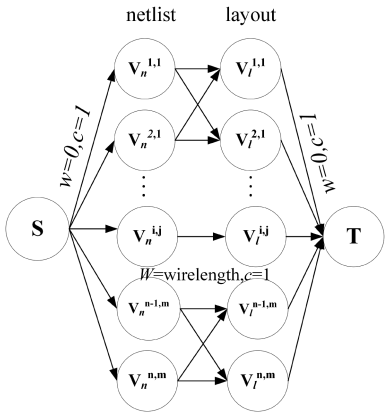

Fig. 9. Network flow model to get the netlist-layout mapping.

we can use two proposed metrics EMSR and AMSPR to evaluate the solutions. Figure 8 shows the two metrics, namely, EMSR and AMSPR, of each benchmark under PM and SA. Without further pruning, the average EMSR and AMSPR of PM is 82.83% and 30.89%, respectively, while for SA they are 95.79% and 40.13%. It can be seen that comparing with PM, 13% more mapped sets found by SA contain correct mapped gates, meanwhile SA can prune the initial mapped sets better. This tells us that global metric (total wirelength) is better than local metric (proximity) in finding the correct mappings for the benchmarks. Even though PM is less effective than SA, however, it is less time consuming and the running time is shown in Table 1. It can be concluded that PA is more suitable for the larger size benchmarks. For example, more than 48 hours are needed to get the mapped sets of benchmark b17 under SA, while that of PM is only 1.6 hour, speeding up the attacking process by more than 200×. However, though the EMSR is smaller than SA, the attacker could accept the slight difference if time is the prior consideration.

6.3.2 Comparison with Network Flow-based Attack in Reference [30]. Since the network flow-based attack method proposed in Reference [30] aims to obtain the connections that has the minimum total wirelength and satisfies the other constraints at the same time, it cannot be used in our threat model directly. Instead, we adapt the idea of network flow to formulate a min cost bipartite mapping problem to get the netlist-layout mapping while at the same time minimizing the total wirelength. Figure 9 shows the network flow model used to get the netlist-layout mapping.

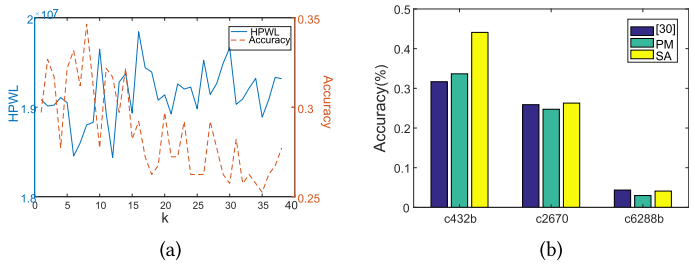

Fig. 10. (a) HPWL and mapping accuracy versus K. (b) Comparison of Reference [30], PM, and SA.

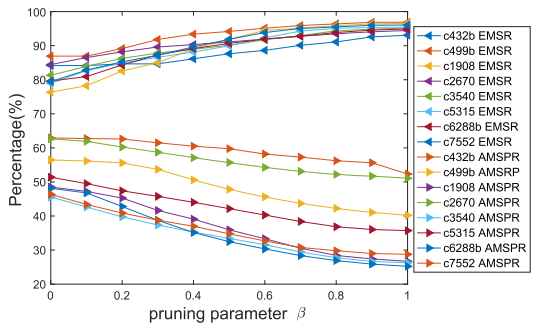

Fig. 11. EMSR and AMSPR of net-based pruning with respect to the pruning parameter β.

The left vertices correspond to all the gates in netlist, while the right are the gates in layout. $V_n^{i,j}$ is the ith cell of type j in netlist and it could only be mapped to the same type gate of it, corresponding to edges to all the j type of gates (vertices) in the right. Finally, weight w and capacity c of source and sink edges set to 0 and 1, respectively, the capacity of the other connected edges is 1 and weight is the wirelength relevant to vertice $V_n^{i,j}$ when map it to $V_l^{i,j}$. Since the initial netlist-layout mapping is unknown, we set the location of unmapped gates in netlist to be the average position of their k-nearest same type gate, thus the edge weight could be calculated and later be minimized. Figure 10(a) shows how k impacts the mapping results of benchmark c432b under network flow-based attack. For each benchmark, we choose the best k under which both HPWL and mapping accuracy are optimal. Finally, different from the proposed attack methods, network flow-based attack can only have one mapping result no matter how many times the process is executed, thus we compare mapping accuracy of one mapping result of them. Figure 10(b) shows the mapping accuracy under three attack methods for several benchmarks. It shows that SA is the most effective when the circuit is small size, but as the circuit goes to larger size, the mapping accuracy tends to be close to each other and both has low accuracy. For a network flow-based attack, this will mislead the attacker the wrong location to insert HTs. While the proposed mapping methods could make the HT insertion more likely to be successful, since multiple possible netlist-layout mappings are generated.

6.3.3 Effectiveness of Net-based Pruning. Similar to α in probability-based pruning, different value of β also impacts the pruning results, since it dominates the percent of gates to be pruned. From Section 4.4, we know that when $\beta = 1$, there are no gates needed to be pruned and the results of which is equivalent to that of without further pruning. Here, we just experiment on the results obtained by SA, since it is better than PM. Figure 11 shows how EMSR and AMSPR of each benchmark change with different value of β. It shows that when β decreases, the AMSPR increases

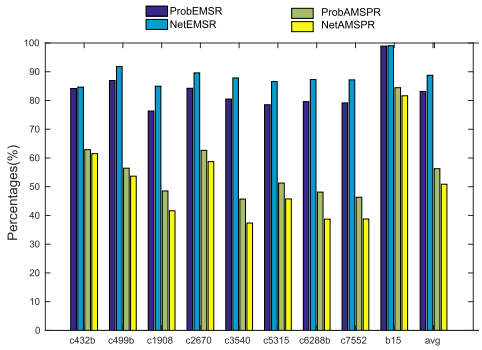

Fig. 12. Results of two pruning methods under SA attack.

Table 1. Benchmark Size and Running Time of Attacks

Benchmark	c432b	c499b	c1908	c2670	c3540	c5315	c6288b	c7552	b15	b17
#Gate Types	12	14	8	8	8	8	15	8	30	39
#PI&PO	43	73	58	373	72	301	64	315	934	2,897
#Gate Number	159	562	521	1,176	1,646	2,844	2,956	3,733	5,533	17,161
#Total Gate Number	202	635	579	1,549	1,718	3,145	3,020	4,048	6,467	20,058
#max same-type	38	190	208	459	547	1,016	623	1,172	1,272	3,535
N	50	200	500	700	1,000	1,400	900	2,400	2,400	3,500
#Running Time of One SA Run	0.4s	1.3s	1s	2.5s	2.9s	5.8s	7s	7.7s	101.51s	359.84s
#Running Time of One PM Run	0.0004s	0.0002s	0.003s	0.015s	0.027s	0.074s	0.060s	0.079s	0.186s	1.644s
#Total Running Time of SA	20s	4.33m	8.33m	29.17m	48.33m	2.26h	1.75h	7.7s	>48h	>48h
#Total Running Time of PM	0.02s	0.4s	1.5s	10.5s	27s	1.73m	54s	3.16m	7.44m	1.60h

while EMSR decreases, and it is obvious that the variation of EMSR and AMSPR is smaller than probability-based pruning. The results also show that $\beta = 0.3$ gives a good tradeoff between EMSR and AMSPR. We then compare the net-based pruning with the probability-based pruning. Figure 12 shows the results of two different pruning methods under SA with $\alpha = 0.9$, $\beta = 0.3$. It indicates that on average the net-based pruning improves the EMSR by 5.59% compared to probability-based pruning, with slightly lower AMSPR. The reason that net-based pruning attains a higher EMSR is that net-based pruning considers both the net-HPWL heuristic and the occurrence probability, thus more mapped sets reserve their correct mapped gate during pruning.

6.3.4 Discussion. In this part, we try to explore the final results after the whole attack process including mapping and pruning. Since SA-based mapping and net-based pruning method have been proved to be better than their counterparts, here we only analyze the final mapped sets obtained by them. Table 1 shows the physical information of all benchmarks and the running time of attack techniques. First, note that the effectiveness of SA is not affected much by circuit scale. For example, c6288b has more than five times the number of gates than c1908 but the attack is more effective on c6288b in terms of both metrics showed in Figure 12. The underlying impact on the effectiveness of the attack comes from the diversity of a circuit. Benchmark c1908 has only 8

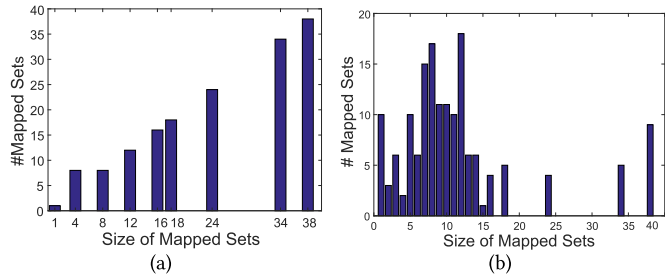

Fig. 13. Distribution of the size of mapped sets of c432: (a) before attack, (b) after attack.

Table 2. The Number of Mapped Sets of Different Sizes: *B* Means Before Attack, *A* Means After Attack

Circuit	Size of mapped set									
	[0,10)		[10,30)		[30,100)		[100,200)		[200,+∞)	
	B	A	B	A	B	A	B	A	B	A
c432b	17	80	70	65	72	14	0	0	0	0
c499b	0	24	116	269	256	248	190	21	0	0
c1908	1	1	0	82	162	219	150	177	208	42
c2670	0	11	0	45	126	399	271	394	779	327
c3540	0	0	0	11	169	200	112	254	1,365	1,181
c5315	0	2	0	7	0	141	168	640	2,676	2,054
c6288b	0	5	47	61	204	315	294	449	2,411	2,126
c7552	0	2	0	13	0	77	0	395	3,733	3,246
b15	28	118	52	382	193	1650	440	2,353	4,820	1,030
b17	-	-	-	-	-	-	-	-	-	-

gate types including primary input/output and almost half of the gates belong to a same gate type. In comparison, c6288b has 15 gate types, with the maximum number of same-type instances being only $\frac{1}{5}$ of the total number of instances. We also compare the running time between probability-based pruning and net-based pruning in Table 3. The upper part is under proximate attack and lower part is under SA attack. It's obviously that probability-based pruning is faster than net-based pruning, since net-based pruning takes time to calculate the wire length.

However, SA-based attack approach not only performs well on global metrics such as EMSR and AMSPR but also prunes many mapped sets to a small size without turning the mapped sets into ineffective. Figure 13 shows the changes in the distribution of the sizes of mapped sets of c432b. Figure 13(a) is the distribution before attack while Figure 13(b) is the distribution after attack. For ineffective mapped sets, their set sizes are restored to their initial sizes, i.e., the number of gates with corresponding types. In Figure 13(b), most mapped sets reside in the region of $size < 15$, which largely reduces the cost and risk of the attacker. For each benchmark circuit, Table 2 shows the number of mapped sets of different sizes before and after SA attack, where many mapped sets have their sizes reduced to be under 30 after attack.

6.4 Effectiveness of Defense

Since defense tends to protect the circuit from being attacked or increasing the cost of the attacker to insert the HTs, we consider the effectiveness of defense on the attack results obtained by

Table 3. Running Time(s) of Probability-based Pruning and Net-based Pruning in Different Benchmarks Under Different Attacks

benchmarks	c432b	c499b	c1908	c2670	c3540	c5315	c6288b	c7552
prob-based	0.0479	0.1388	0.2881	0.8620	1.4220	3.4245	2.1940	7.2333
net-based	0.4278	4.6809	11.5906	39.2878	72.1106	186.8645	119.8071	487.4236
prob-based	0.0227	0.1372	0.2823	0.8248	1.4007	3.3295	2.1736	7.0445
net-based	0.3695	4.6261	11.2752	38.1949	69.0310	179.7954	112.2774	466.8182

The upper part is under proximity attack and the lower part is under simulated annealing attack.

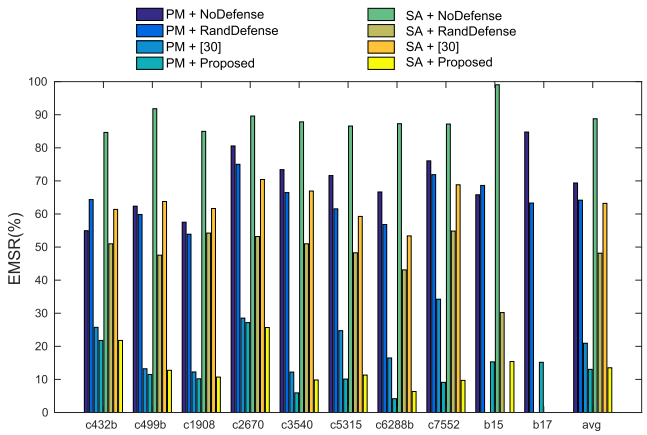

Fig. 14. EMSR of PM and SA attack under different defense techniques.

net-based pruning method with $\beta = 0.3$, which performs better than probability-based pruning method during attack.

6.4.1 Effectiveness of the Defense Method. The effectiveness of defense is directly measured by EMSR, since the goal of the defender is to hide the correct mapped gates away from their mapped sets. We first experiment it under all the proposed attack methods: PM + no defense, PM + defense, SA + no defense, SA + defense. The maximum reduction of EMSR under different attacks is demonstrated in Figure 14. It shows that EMSR of all benchmarks under SA with defense goes below 30% compared to that of without defense, and the proposed defense method is effective to proximity-based attack as well. This is because the defense targets to obfuscate the global heuristic, i.e., total wirelength, which indirectly impacts the local heuristic, i.e., local connection of pins. However, the average EMSR of SA after leveraging the proposed defense approach is up to 14%, with average reduction being about 74%. It means most of the mapped sets contains no correct gates and resulting only a small number of effective mapped sets left, showing the effectiveness of the proposed defense method. The reason is that the defense approach moves most gates to locations that are immune to the global wirelength-based attack. Besides, since we assign higher priority to more insecure gates, i.e., the gates with smaller mapped sets, the number of small effective mapped sets will decrease, significantly elevating the cost and risk of the attacker.

6.4.2 Comparison with State-of-the-Art. We also compare our defense method with the state-of-the-art methods proposed in References [19, 30]. First, since the pin-swapped defense method in Reference [19] aims to swap pins of partitions of a hierarchical design, while our benchmarks are flatted designs and the split layer is M1, thus swap pin is equivalent to swap gate in our model.

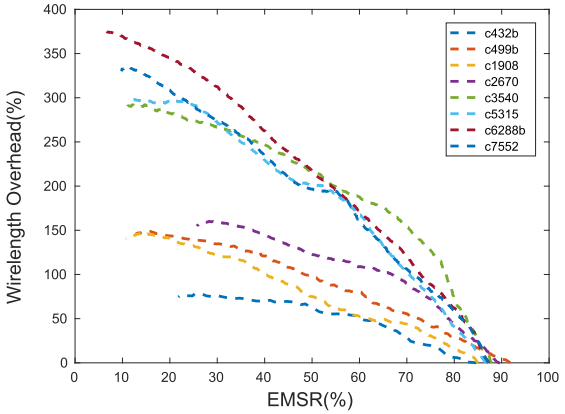

Fig. 15. Security vs. Wirelength overhead under SA attack.

A random gate-swapped defense technique is then implemented to be compared with ours. Obviously the number of gates swapped could impact the proximity hint, and in our experiments, all the gates are randomly swapped to thwart the proximity hints the most. The results in Figure 14 show that random gate-swapped method has little effect on PM attack but could weaken the effectiveness of SA attack. However the average EMSR under "SA + RandDefense" could still be nearly 50%, while that of "SA + Proposed" is less than 14%. Second, a placement perturbation-based defense method in Reference [30] is implemented and compared. It is worth mentioning that since Reference [30] assumes several metal layers are manufactured in the BEOL, only a few portion of gates are extracted to be perturbed. But in our work, since the split layer is M1, all the gates should be extracted according to the idea of its defense method. This will cause exponential computation cost, because the worst computation complexity of the defense technique in Reference [30] is $O(K^V)$ with V the number of extracted tree vertices and K the number of space locations in the layout, for example, due to large gate and space number in the layout, perturbation to benchmark b15 and b17 have not been completed yet due to exponential computation. To speed up the perturbation, the vertice number of extracted tree is set to be less than 5 in our experiment. Figure 14 shows the average EMSR reduction of "SA + [30]" is 25.6%, while that of "SA + Proposed" is 75.3%. However, the effectiveness of [30] on PM attack largely correlates with the size of the benchmark. For example, EMSR of c432b under "PM + [30]" is 25.74%, which is close to that of "PM + Proposed," but EMSR of "PM + [30]" is less than that of "PM + Proposed" by 25.12%, indicating the proposed defense method is more effective.

6.4.3 Security–Wirelength Overhead Tradeoff. Since our defense method can move gates to places that wirelength-driven placement does not choose, the security against attacks that utilize global wirelength heuristic is guaranteed. Also, the gates in out-of-candidate locations are secure in a solid way, which means the best chance for the attacker is to randomly guess their locations. Although the experimental results show the effectiveness of our proposed defense method, it is under the worst wirelength cost circumstance. It deserves considering the security level versus wirelength overhead specifically. Here, we just consider the case with SA attack method.

Figure 15 shows the tradeoff between EMSR and wirelength overhead for ISCAS-85 benchmarks. As EMSR decreases during gate swapping, the wirelength overhead increases approximately linearly with EMSR especially for large scale benchmarks. The reason is that we want as many as possible gates whose mapped sets exclude their correct mapping of the circuit, resulting in a large number of gates to be swapped, thus enlarge the final total wirelength. However, for a targeted

HT insertion, the attacker generally only concerns certain gate types, leaving the other type gates secure intrinsically. So there is no need for the defender to swap all the gates, thus the wirelength overhead could be controlled. Finally, the defender can also set a wirelength overhead budget and only allow defense within this budget to protect all the gates.

7 CONCLUSIONS

In this work, we have shown that split manufacturing is not secure intrinsically even when split at M1 level, which is considered as the most secure scenario. The attacker can still get necessary information through our proposed mapping approaches at the placement level, by leveraging the global or local physical heuristics, to decrease the cost of inserting hardware Trojans. The experimental results show that the proposed SA attack method can decrease the average mapped set ratio (AMSPR) to above 50% while maintaining the average effective mapped set ratio (EMSR) high enough (about 88.6% after net-based pruning), while the PM approach could speed up the mapping process by about 200×. Compared to the state-of-the-art attack method in Reference [30], our method is more effective to find the mapped sets. We further propose an effective defense approach from the placement level to defend against such attacks. Experimental results show that the proposed defense method is effective under both the PM and SA attack methods compared to previous proposed defense methods [19, 30]. With defense, EMSR of benchmarks under SA and PM all decrease to below 30%, which means most of the mapped sets of each benchmark contain no correct gates, resulting in higher cost for the attacker to insert hardware Trojans. Note that even the defense can attain a maximum EMSR of 14%, which greatly increases the cost of the attacker; however, it is under the worst wirelength cost circumstance. Practically, the defender could only defend partial gates of the circuit or just set a wirelength budget to defend the circuit within it.

REFERENCES

[1] Oklahoma State University. 2015. System on chip (SoC) design flows. Retrieved from http://vlsiarch.ecen.okstate.edu/flow/.

[2] Intelligence Advanced Research Projects Activity. 2011. Trusted integrated circuits program. Retrieved from https://www.fbo.gov/utils/view?id=b8be3d5d5babbffc6975c370247a6.

[3] R. S. Chakraborty and S. Bhunia. 2009. HARPOON: An obfuscation-based SoC design methodology for hardware protection. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 28, 10 (Oct. 2009), 1493–1502. DOI: https://doi.org/10.1109/TCAD.2009.2028166

[4] Abhijit Chatterjee and Richard Hartley. 1991. A new simultaneous circuit partitioning and chip placement approach based on simulated annealing. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*. ACM, 36–39.

[5] X. Chen, L. Wang, Y. Wang, Y. Liu, and H. Yang. 2017. A general framework for hardware trojan detection in digital circuits by statistical learning algorithms. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 36, 10 (Oct. 2017), 1633–1646. DOI: https://doi.org/10.1109/TCAD.2016.2638442

[6] Zhang Chen, Pingqiang Zhou, Tsung-Yi Ho, and Yier Jin. 2016. How secure is split manufacturing in preventing hardware trojan? In *Proceedings of the IEEE Asian Hardware-Oriented Security and Trust (AsianHOST'16)*. IEEE, 1–6.

[7] J. Francq and F. Frick. 2015. Introduction to hardware Trojan detection methods. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition*. 770–775. DOI: https://doi.org/10.7873/DATE.2015.1101

[8] M. C. Hansen, H. Yalcin, and J. P. Hayes. 1999. Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering. *IEEE Design Test Comput.* 16, 3 (July 1999), 72–80. DOI: https://doi.org/10.1109/54.785838

[9] J. He, Y. Zhao, X. Guo, and Y. Jin. 2017. Hardware trojan detection through chip-free electromagnetic side-channel statistical analysis. *IEEE Trans. Very Large Scale Integr. Syst.* 25, 10 (Oct. 2017), 2939–2948. DOI: https://doi.org/10.1109/TVLSI.2017.2727985

[10] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara. 2013. Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation. In *Proceedings of the USENIX Conference*. 495–510.

[11] M. Jagasivamani, P. Gadfort, M. Sika, M. Bajura, and M. Fritze. 2014. Split-fabrication obfuscation: Metrics and techniques. In *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust*. 7–12. DOI: https://doi.org/10.1109/HST.2014.6855560

[12] Yier Jin. 2015. Introduction to hardware security. *Electronics* 4 (2015) 763–784.

[13] Yier Jin and Y. Makris. 2008. Hardware trojan detection using path delay fingerprint. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust*. 51–57. DOI: https://doi.org/10.1109/HST.2008.4559049

[14] H. Li, Q. Liu, J. Zhang, and Y. Lyu. 2015. A survey of hardware trojan detection, diagnosis and prevention. In *Proceedings of the International Conference on Computer-Aided Design and Computer Graphics*. 173–180. DOI: https://doi.org/10.1109/CADGRAPHICS.2015.41

[15] W. Li, Z. Wasson, and S. A. Seshia. 2012. Reverse engineering circuits using behavioral pattern mining. In *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust*. 83–88. DOI: https://doi.org/10.1109/HST.2012.6224325

[16] J. Magaa, D. Shi, J. Melchert, and A. Davoodi. 2017. Are proximity attacks a threat to the security of split manufacturing of integrated circuits?*IEEE Trans. Very Large Scale Integr. Syst.* 25, 12 (Dec. 2017), 3406–3419. DOI: https://doi.org/10.1109/TVLSI.2017.2748018

[17] U.S. Patent. 2004. Split manufacturing method for advanced semiconductor circuits.

[18] J. Rajendran, E. Gavas, J. Jimenez, V. Padman, and R. Karri. 2010. Towards a comprehensive and systematic classification of hardware Trojans. In *Proceedings of IEEE International Symposium on Circuits and Systems*. IEEE, Paris, France, 1871–1874.

[19] J. Rajendran, O. Sinanoglu, and R. Karri. 2013. Is split manufacturing secure?. In *Design, Automation Test in Europe Conference Exhibition*. 1259–1264. DOI: https://doi.org/10.7873/DATE.2013.261

[20] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri. 2013. Hardware security: Threat models and metrics. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 819–823. DOI: https://doi.org/10.1109/ICCAD.2013.6691207

[21] J. A. Roy, F. Koushanfar, and I. L. Markov. 2010. Ending piracy of integrated circuits. *Computer* 43, 10 (Oct 2010), 30–38. DOI: https://doi.org/10.1109/MC.2010.284

[22] A. Cozzie C. Grier W. Jiang S. T. King, J. Tucek and Y. Zhou. 2008. Designing and implementing malicious hardware. In *Proceedings of the USENIX Conference*. 51–58.

[23] Carl Sechen. 1988. Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing. In *Proceedings of the 25th ACM/IEEE Design Automation Conference*. IEEE Computer Society Press, 73–80.

[24] R. Torrance and D. James. 2007. Reverse engineering in the semiconductor industry. In *Proceedings of the IEEE Custom Integrated Circuits Conference*. 429–436. DOI: https://doi.org/10.1109/CICC.2007.4405767

[25] Randy Torrance and Dick James. 2011. The state-of-the-art in semiconductor reverse engineering. In *Proceedings of the 48th ACM/EDAC/IEEE Design Automation Conference (DAC'11)*. IEEE, 333–338.

[26] K. Vaidyanathan, B. P. Das, and L. Pileggi. 2014. Detecting reliability attacks during split fabrication using test-only BEOL stack. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. 1–6. DOI: https://doi.org/10.1145/2593069.2593123

[27] K. Vaidyanathan, B. P. Das, E. Sumbul, R. Liu, and L. Pileggi. 2014. Building trusted ICs using split fabrication. In *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust*. 1–6. DOI: https://doi.org/10.1109/HST.2014.6855559

[28] N. Viswanathan, M. Pan, and C. Chu. 2007. FastPlace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. In *Proceedings of the Asia and South Pacific Design Automation Conference*. 135–140. DOI: https://doi.org/10.1109/ASPDAC.2007.357975

[29] Y. Wang, P. Chen, J. Hu, G. Li, and J. Rajendran. 2018. The cat and mouse in split manufacturing. *IEEE Trans. Very Large Scale Integr. Syst.* 26, 5 (May 2018), 805–817. DOI: https://doi.org/10.1109/TVLSI.2017.2787754

[30] Y. Wang, P. Chen, J. Hu, and J. J. V. Rajendran. 2016. The cat and mouse in split manufacturing. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. 1–6. DOI: https://doi.org/10.1145/2897937.2898104

[31] K. Xiao, D. Forte, and M. M. Tehranipoor. 2015. Efficient and secure split manufacturing via obfuscated built-in self-authentication. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust*. 14–19. DOI: https://doi.org/10.1109/HST.2015.7140229

[32] Y. Xie, C. Bao, and A. Srivastava. 2017. Security-aware 2.5D integrated circuit design flow against hardware IP piracy. *Computer* 50, 5 (May 2017), 62–71. DOI: https://doi.org/10.1109/MC.2017.121

[33] Wenbin Xu, Lang Feng, Jeyavijayan J. V. Rajendran, and Jiang Hu. 2019. Layout recognition attacks on split manufacturing. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. ACM, 45–50.

Received September 2019; revised December 2019; accepted January 2020