A Practical Split Manufacturing Framework for Trojan Prevention via Simultaneous Wire Lifting and Cell Insertion

Meng Li *Student Member, IEEE*, Bei Yu *Member, IEEE*, Yibo Lin, *Student Member, IEEE*, Xiaoqing Xu *Member, IEEE*, Wuxi Li,David Z. Pan *Fellow, IEEE*

***Abstract*— Trojans and backdoors inserted by untrusted foundries have become serious threats to hardware security. Split manufacturing is proposed to hide important circuit structures and prevent Trojan insertion by fabricating partial interconnections in trusted foundries. Existing split manufacturing frameworks, however, usually lack security guarantee and suffer from poor scalability. It is observed that inserting dummy cells and wires can have a high potential for overcoming the security and scalability problems of existing methods, but it is not compatible with current security definition. In this paper, we focus on answering the questions on how to define the notion of security and how to realize the required security level effectively and efficiently when the insertion of dummy cells and wires is considered. We first generalize existing security criterion by modeling the split manufacturing process as a graph problem. Then, a sufficient condition is derived for the proposed security criterion to avoid the computationally intensive operations in traditional methods. To further enhance the scalability of the framework, we propose a secure-by-construction split manufacturing flow. For the first time, a novel mixed-integer linear programming (MILP) formulation is proposed to simultaneously consider cell and wire insertion together with wire lifting. A Lagrangian relaxation algorithm with a minimum-cost flow transformation technique is employed to solve the MILP formulation efficiently. With extensive experiments, our framework demonstrates significantly better efficiency, overhead reduction, and security guarantee compared with the previous state-of-the-art.**

***Index Terms*—Split Manufacturing, Trojan Prevention, k-isomorphism, Simultaneous Cell and Wire Insertion, MILP Formulation, Lagrangian Relaxation**

I. Introduction

WITH the globalization of the integrated circuit (IC) supply chain, the design complexity and cost of design houses have been reduced significantly. However, many emerging security vulnerabilities have come along as well, including hardware Trojans [1]–[4], reverse engineering [5], [6] and so on, resulting in economic losses in the order of billions of dollars annually. Hardware Trojans inserted by untrusted foundries are extremely harmful to the system security, while the detection of such hardware Trojans remains to be very difficult. Therefore, how to prevent the Trojan insertion by untrusted foundries is becoming a very critical issue.

To prevent Trojan insertion proactively, split manufacturing is proposed. [7]–[17], In the split manufacturing process, the circuit layout is split into front-end-of-line (FEOL) layers, which consist of all the cells and interconnections in lower metal layers, and back-end-of-line (BEOL) layers, which consist of all the interconnections in higher metal layers. Because the fabrication of BEOL layers usually requires less advanced technologies, it is affordable to maintain such trusted foundries for the BEOL layer fabrication, by which important circuit information can be hidden to prevent Trojan insertions by untrusted foundries.

In recent years, different split manufacturing frameworks have been proposed. The first formal security criterion for split manufacturing against Trojan insertion, named as k-security, is proposed in [13]. A circuit is defined to be k-secure if for each cell in the original netlist, there exist k cells in the FEOL layers that can be its actual physical implementation and are indistinguishable to the attackers. The security definition is formalized based on graph isomorphism [18], as will be discussed in Section III. To realize k-security, a greedy algorithm is also proposed to determine the wires to be lifted from the FEOL layers to the BEOL layers. In [19]–[24], techniques in physical synthesis stage, including fault-analysis based pin swapping, placement perturbation, and so on, are proposed to prevent the untrusted foundries from reverse engineering the hardware intellectual property. These methods are proposed under another orthogonal attack model, the main target of which is to prevent reverse engineering by untrusted foundries. It is currently not clear how these proposed methods can be leveraged for hardware Trojan prevention.

Despite the extensive researches on split manufacturing, existing approaches still suffer from an insufficient security guarantee, poor computational efficiency, and large performance overhead as will be detailed in Section II. In this paper, besides wire lifting, the insertion of dummy cells and wires are considered simultaneously to address the security and practicality issues of existing methods. Considering existing security criterion cannot model the situation where FEOL layers contain cells and wires that do not exist in the original netlist, we propose a new criterion that is fully compatible with the insertion of dummy nodes and wires. Our security criterion can also balance the trade-off between security and overhead by allowing the flexibility of protecting any arbitrary subset of circuit nodes. We further derive a sufficient condition for the security criterion to avoid the computationally intensive graph isomorphism checking and enable an efficient security realization. To realize the security criterion while minimizing the introduced overhead, we propose a holistic framework. Our framework consists of a novel mixed-integer

M. Li, Y. Lin, W. Li, and D.Z. Pan are with the Department of Electrical and Computer Engineering, University of Texas, Austin TX, USA (email: meng_li@utexas.edu).

B. Yu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, NT, Hong Kong.

X. Xu is with the ARM Inc.

linear programming (MILP) formulation for the FEOL layer generation and a Lagrangian relaxation (LR) algorithm [25], [26] to significantly speed up the generation process. A layout refinement technique is also proposed to guarantee security in the physical synthesis stage. We summarize our contributions as follows:

- A new security criterion fully compatible with cell and wire insertion is proposed with its sufficient condition derived to enable an efficient split manufacturing process.
- An MILP-based formulation is proposed to generate the FEOL layers considering dummy cell and wire insertion with wire lifting simultaneously and further accelerated with an LR-based algorithm.
- A layout refinement technique is proposed to guarantee security in the physical synthesis stage.
- The proposed flow is validated by extensive experimental results and demonstrates good efficiency and practicality.

The rest of the paper is organized as follows. Section II defines the attack model and describes an example to illustrate the motivation of the paper and the state-of-the-art split manufacturing flow in detail. Section III formally formulates the split manufacturing problem and defines our new security criterion. Section IV proposes a sufficient condition to achieve the proposed criterion. Section V describes our split manufacturing framework. Section VI demonstrates the performance of the framework, followed by conclusion in Section VII.

II. Preliminary

In this section, the attack model of untrusted foundries is first reviewed. A motivating example is analyzed to explain the insufficiency when only wire lifting is considered in the split manufacturing flow. We also describe the state-of-the-art FEOL generation flow proposed in [13] in detail.

A. Attack Model of Untrusted Foundries

We consider attackers from untrusted foundries that target at inserting malicious hardware Trojans into the design. We assume the following attack model as described in [13]:

- The attacker has the gate-level netlist of the design.
- The attacker has full knowledge of the FEOL layers, including the cells and wires in lower metal layers as well as their physical information.
- The attacker knows the algorithms of generating the FEOL layers but does not know the specific mapping between the cells in the FEOL layers and the original netlist.

The assumption on the knowledge of the gate-level netlist is pretty strong but indeed possible. The main reason is that the attackers who intend for such Trojan insertion can potentially be resourceful enough to have malicious observers in the design stage [13]. Meanwhile, the profit of a successful Trojan insertion can also be pretty large, especially for military applications [27]. Given the gate-level netlist, the attackers can first determine the target gates in the design for the Trojan insertion. Then, the attackers will try to identify the physical implementation of the target gates based on the information of the FEOL layers and insert the Trojan.

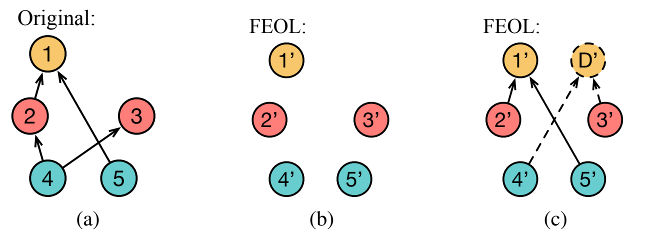


Fig. 1: A motivating example of split manufacturing process and the insufficiency of state-of-the-art framework: (a) the original netlist; (b) FEOL layers generated by the original flow; (c) FEOL layers generated by our new framework (nodes with the same colors have the same functionalities and the dotted lines indicate the inserted dummy edges).

B. Motivating Example

As described in Section II-A, given the information on the original circuit netlist and the FEOL layers, the attackers can try to locate the actual implementation for the target gates identified in the original netlist. According to [13], the attack process can be formulated as searching for a bijective mapping of gates in the FEOL layers to the gates in the original netlist. Consider the circuit netlist as shown in Fig. 1(a) and the FEOL layers shown in Fig. 1(b). There exist 4 distinct bijective mappings between the FEOL layers and the original netlist, i.e. $f_1 : \{1, 2, 3, 4, 5\} \rightarrow \{1', 2', 3', 4', 5'\}$, $f_2 : \{1, 2, 3, 4, 5\} \rightarrow \{1', 3', 2', 4', 5'\}$, $f_3 : \{1, 2, 3, 4, 5\} \rightarrow \{1', 2', 3', 5', 4'\}$, $f_4 : \{1, 2, 3, 4, 5\} \rightarrow \{1', 3', 2', 5', 4'\}$. Following the current mapping relations, both Gates $2'$ and $3'$ in the FEOL layers can be mapped to Gate 2. From the attacker's perspective, both Gates $2'$ and $3'$ can implement Gate 2 in the original netlist. Therefore, if the attacker targets at Gate 2 for the Trojan insertion, his capability to accurately insert the Trojan is significantly weakened.

However, there are at least two problems with the FEOL layers in Fig. 1(b). On one hand, for Gate 1, only $1'$ in the FEOL layers share the same functionality, which indicates the attacker can always determine its identity. In fact, because the other gates in the original netlist all have different functionalities compared to Gate 1, simply by lifting wires to the BEOL layers can never help enhance the security of Gate 1. On the other hand, in Fig. 1(b), all the wires are lifted to the BEOL layers. Because there are usually much fewer routing resources in higher metal layers, design houses are forced to either increase the number of layers fabricated in trusted foundries or reduce the area utilization to mitigate the routing congestion in higher metal layers, both of which increase the overhead of split manufacturing significantly.

In this paper, we propose a new framework that considers the insertion of dummy gates and wires simultaneously with the wire lifting. Consider the FEOL layers shown in Fig. 1(c). A dummy gate D' of the same gate type as 1 is inserted. Two dummy wires $(3', D')$ and $(4', D')$ are inserted to the FEOL layers as well. In this way, for any gate targeted by the attacker in the original circuit, there are two gates that cannot be distinguished in the FEOL layers. Meanwhile, only two wires, i.e. $(4', 2')$ and $(4', 3')$, are lifted to the BEOL layers

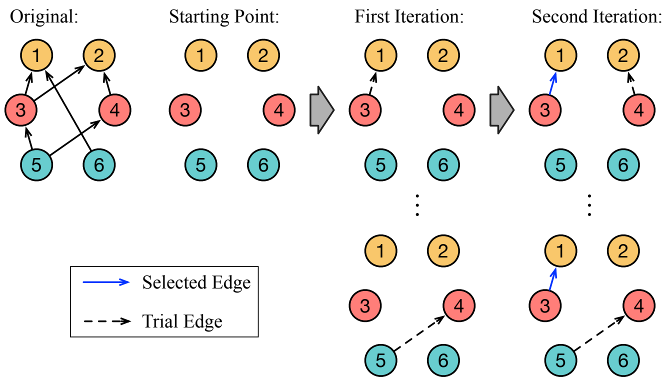


Fig. 2: Traditional split manufacturing flow.

while the number of wires in the FEOL layers remain the same as that in the original netlist. Thereby, the two drawbacks of the original framework [13] can be well solved.

However, it should be noted that due to the insertion of dummy gates and wires, the bijective mappings between the original netlist and the FEOL layers do not hold anymore, which means the original formalization of the attack process and the definition of security criterion cannot be applied anymore. In Section III, we will propose our new formulation for the split manufacturing protection and the Trojan insertion attack.

C. State-of-the-art Split Manufacturing Flow

In Section II-B, we use a motivating example to compare the FEOL layers generated by the original flow and our framework. In this section, we will review the process of the FEOL layer generation proposed in [13]. Consider the original netlist shown in Fig. 2. To determine the wires to be lifted to the BEOL layers, the proposed framework starts by lifting all the wires to the BEOL layers first. Then, it adds the wires back to the FEOL layers iteratively following a greedy selection strategy. In each iteration, it tries to add each wire back to the FEOL layers, and then determine the security level for the current FEOL layers. The wire that provides the best security level will be selected and added back to the FEOL layers. The procedure continues until the security level can no longer be satisfied.

The state-of-the-art split manufacturing flow suffers from scalability issue. As described above, in each iteration, to determine the security level when a wire is added back, repetitive checking is carried out to search for the bijective mappings between the whole circuit and the FEOL layers. Although it can be elegantly formulated as a satisfiability (SAT) problem, the computation cost makes the method intractable quickly even for small benchmark circuits. In this paper, we target at solving all the above-mentioned problems of the existing method to provide better security guarantee, reduce the introduced overhead, and enhance the scalability of the split manufacturing flow.

III. Split Manufacturing Security Analysis

In this section, we will formulate the split manufacturing problem as a graph problem. To accommodate the insertion

TABLE I: Notations used for security definition and analysis.

$\phi(v)$	Corresponding node that implements v in the FEOL layers
$\ell(v)$	Label of v representing its cell type
$\omega(v)$	Weight of v indicating whether it is selected for protection
$\mathcal{C}(v)$	Set of candidate nodes in the FEOL layers that can implement v from the attacker's point of view
$\mathcal{S}_v(v')$	Set of spanning subgraph isomorphism relations that map v' in FEOL layers to v in the original netlist
$\mathcal{P}_v(v')$	Probability of candidacy for $v' \in \mathcal{C}(v)$

of dummy cells and wires, we will formally define the split manufacturing process and the attack process and propose a new security criterion. For convenience, some notations used in this paper are summarized in TABLE I, which will be defined and explained in detail in this section.

A circuit can be regarded as a graph $G = \langle V, E, \ell, \omega \rangle$. V is the set of vertices, with each vertex corresponding to one circuit node. E is the set of directed edges corresponding to the wires in the circuit. Label function $\ell : V \rightarrow [t]$ maps each vertex to a cell type, where $[t] = \{1, \dots, t\}$ denotes the set of all possible cell types in the circuit. $\omega : V \rightarrow \{0, 1\}$ assigns a binary weight to each vertex with $\omega(v) = 1$ indicating that the vertex v is selected for protection. ω is defined to make the framework flexible to protect a subset of circuit nodes[1] due to overhead constraints and balance the trade-off between security and the introduced overhead.

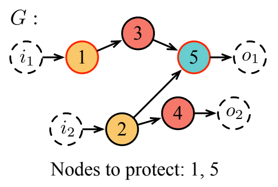
The original netlist and the generated FEOL layers can be represented as two graphs. For the graph representation of the original circuit, denoted as G, V_G, E_G, and ℓ_G are straightforward to define. ω_G is determined by the designer considering the circuit functionality, overhead constraints and so on. To determine these parameters for the graph representation of the FEOL layer, denoted as H, we need to consider its generation process. To generate H, for each $v \in V_G$, we add v' to V_H such that $\ell_H(v') = \ell_G(v)$ and $\omega_H(v') = \omega_G(v)$. We denote $v' = \phi(v)$ as the corresponding node for v, which represents the actual cell in FEOL that implements v in the netlist. Meanwhile, for each $(v, u) \in E_G$, we add $(\phi(v), \phi(u))$ to E_H. Then, we consider the three operations for the generation of H:

1) **wire lifting**: if $(u', v') \in E_H$ is lifted to BEOL, then, $E_H = E_H \setminus \{(u', v')\}$ with V_H, ℓ_H and ω_H unchanged;
2) **dummy node insertion**: if u' with with the cell type $\ell_{u'}$ is inserted, then, $V_H = V_H \cup \{u'\}$ with $\ell_H(u') = \ell_{u'}$, $\omega_H(u') = 0$ and E_H is unchanged;
3) **dummy wire insertion**: if (u', v') is inserted, then, $E_H = E_H \cup \{(u', v')\}$ with V_H, ℓ_H and ω_H unchanged.

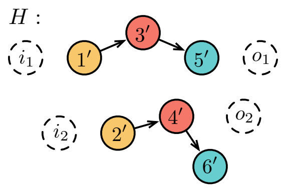
It should be noted that to guarantee the circuit functionality is not changed and to get rid of floating input pins, we only allow inserting wires pointing to the dummy nodes. Based on the description of the allowed operations, V_H, E_H, ℓ_H and ω_H can be acquired accordingly.

Example 1. *Consider an example of G and H in Fig. 3. In G, we have nodes 1 and 2 with the same cell type, i.e. $\ell_G(1) = \ell_G(2)$. Assume that we select nodes 1 and 5 for protection, then, $\omega_G(1) = \omega_G(5) = 1$. To generate H, we first*

[1]In the paper, nodes and cells are the same and used interchangeably.



(a)



(b)

Fig. 3: Example of (a) Original graph G (nodes with red stroke have non-zero weights), (b) FEOL graph H. i_1 and i_2 are input pins while o_1 and o_2 are output pins.

add the corresponding nodes to H for each node in G, i.e. $1', 2', 3', 4', 5'$. Then, we add node $6'$ and wire $(4', 6')$ to H and lift wire $(2', 5')$. Therefore, we have $\omega_H(1') = \omega_G(1) = 1$ and $\omega_H(5') = \omega_G(5) = 1$. For the other nodes in H, we have $\omega_H(2') = \omega_H(3') = \omega_H(4') = \omega_H(6') = 0$.

As described in Section II, to insert a Trojan, the attacker will first select $v \in V_G$ based on the analysis of the design and then, try to locate its corresponding node $\phi(v)$ in H. To formalize the process of locating $\phi(v)$, state-of-the-art method [13] leverages the concept of graph isomorphism.

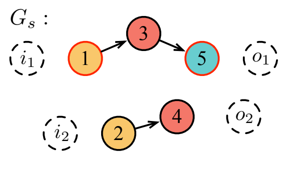
Definition 1 (Graph Isomorphism)**.** *Two graphs G_1 and G_2 are isomorphic if there exists a bijective mapping $f : V_{G_1} \to V_{G_2}$ such that $(u, v) \in E_{G_1}$ if and only if $(f(u), f(v)) \in E_{G_2}$ and $\ell_{G_1}(u) = \ell_{G_2}(f(u))$, $\ell_{G_1}(v) = \ell_{G_2}(f(v))$.*

Because only wire lifting is considered in existing methods, we must have $V_H = V_G$ and $E_H \subseteq E_G$. Therefore, there must be a subgraph of G that is isomorphic to H, based on which for each $v \in V_G$, a set of nodes can be identified that may implement v in FEOL. This enables the previous work [13] to formally define the security criterion.

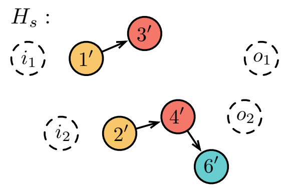
However, *when the insertion of dummy wires and cells are considered, the original isomorphic relation is not satisfied anymore*. This is because H contains nodes and edges that do not present in G so that $V_G \neq V_H$ and $E_H \nsubseteq E_G$. To formalize the relation between G and H, we first have the following observations on the relations between H and G that always hold:

- $\forall v \in V_G$, $\exists v' \in V_H$ s.t. $v' = \phi(v)$.
- $\forall v', u' \in V_H$, if $\exists u \in V_G$ s.t. $u' = \phi(u)$, then, if $\forall (v', u') \in E_H$, then, there must exist $v \in V_G$ s.t. $v' = \phi(v)$ and $(v, u) \in E_G$.

The first observation indicates that for each circuit node in G, there must be one node in H that implements it. The second observation indicates that if $u' \in V_H$ is the corresponding node of u in the netlist, then, for all the edges that point to u', e.g. $(v', u') \in E_H$, there must be $v \in V_G$ with v' as the corresponding node and v is connected to u in G. This is because we are not allowed to add dummy edges pointing to the corresponding node of $u \in V_G$. For example, in Fig. 3, suppose $5' = \phi(5)$, since we are not allowed to add any dummy edges pointing to $5'$, we must be able to find $3 \in V_G$ such that $3' = \phi(3)$ and $(3, 5) \in E_G$. To formalize the relations described above, we leverage the concept of spanning subgraph and induced subgraph [28].



(a)



(b)

Fig. 4: Example of (a) Spanning subgraph G_s of G in Fig. 3(a), and (b) induced subgraph H_s of H in Fig. 3(b).

Definition 2 (Spanning Subgraph)**.** *A subgraph G_s of G is referred to as a spanning subgraph if $V_{G_s} = V_G$.*

Definition 3 (Induced Subgraph)**.** *A subgraph G_s of G is referred to as an induced subgraph if $\forall (u, v) \in E_G$ with $u, v \in V_G$, $(u, v) \in E_{G_s}$ if and only if $u, v \in V_{G_s}$.*

Example 2. *Consider an example shown in Fig. 4. G_s is a spanning subgraph of G in Fig. 3(a) since $V_{G_s} = V_G$. H_s in Fig. 4(b) is an induced subgraph of H in Fig. 3(b) because for any pair of nodes in H_s, if there exists an edge between them in H, the edge also exists in H_s. For example, nodes $1'$ and $3'$ exist in H_s. Because $(1', 3') \in E_H$, for H_s to be an induced subgraph, we must have $(1', 3') \in E_{H_s}$.*

Then, considering the spanning subgraph of G and the induced subgraph of H, we define the relation of spanning subgraph isomorphism as below.

Definition 4 (Spanning Subgraph Isomorphism)**.** *Given two graphs G and H, we say that G is spanning subgraph isomorphic to H if there exists a spanning subgraph of G that is isomorphic to an induced subgraph of H.*

Spanning subgraph isomorphism defines the criterion for the attackers to identify the corresponding node $\phi(v)$ in FEOL for a target node v in the netlist. For example, in Fig. 4, since G_s and H_s are isomorphic, G is spanning subgraph isomorphic to H with $1, 2, 3, 4, 5$ being matched to $2', 1', 4', 3', 6'$, respectively. Therefore, $2'$ is possible to implement node 1 in the final layout from the attacker's point of view. We denote $2'$ as the candidate node for 1.

For the spanning subgraph isomorphism relation, there is one additional constraint to consider. Because inserting dummy wires pointing to the corresponding nodes in the FEOL layers is not allowed, it is possible for some spanning subgraph isomorphism relation to be invalid. For example, consider G and H as shown in Fig. 5. There exists a spanning subgraph isomorphism relation that maps $1, 2, 3, 4$ in G to $5', 2', 3', 4'$ in H, respectively. Following the current mapping, node $1'$ becomes dummy. However, because $2'$ is the corresponding node of 2 in the current mapping and we are not allowed to insert dummy edges pointing to the corresponding node, $(1', 2')$ must be an edge that exists in the original netlist, which is contradictory to the conjecture that node $1'$ is dummy. We define the spanning subgraph isomorphism relations that satisfy the constraints on wire insertion as valid isomorphism relations. Only the valid isomorphism relations can enhance the security against hardware Trojan insertion.

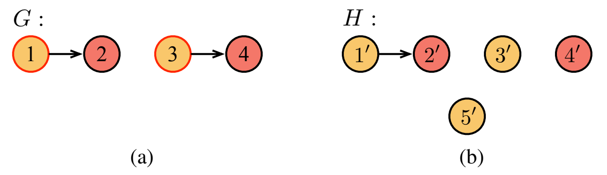


Fig. 5: Example on the weight and probability difference for different candidate nodes.

The proposed spanning subgraph isomorphism relation is more general compared with the graph isomorphism relation. *When only wire lifting is considered, it reduces to the graph isomorphism. It can also capture the situations where $V_G \neq V_H$ and $E_H \nsubseteq E_G$, which enables us to consider cell and wire insertion in the split manufacturing process.*

Because multiple spanning subgraph isomorphism relations may exist between G and H, for $v \in V_G$, a set of candidate nodes can be identified, denoted as the candidate set $\mathcal{C}(v)$. For the nodes in the candidate set, the number of spanning subgraph isomorphism relations that can map them to the original node is different. For example, as shown in Fig. 5, $1'$, $3'$ and $5'$ are the candidate nodes for 3. For $1'$, there are two different isomorphism relations mapping it to 1, i.e. $f_1 : \{1,2,3,4\} \rightarrow \{1',2',3',4'\}$ and $f_2 : \{1,2,3,4\} \rightarrow \{1',2',5',4'\}$. For $3'$ and $5'$, there is only one isomorphism relation mapping each of them to 1, i.e. $f_3 : \{1,2,3,4\} \rightarrow \{3',4',1',2'\}$ and $f_4 : \{1,2,3,4\} \rightarrow \{5',4',1',2'\}$. The nodes with a larger number of spanning subgraph isomorphism relations are more likely to be recognized and selected by the attackers. Therefore, for $v \in V_G$, we define the probability of candidacy for $v' \in \mathcal{C}(v)$ as

$$\mathcal{P}_v(v') = \frac{|\mathcal{S}_v(v')|}{\sum_{u' \in \mathcal{C}(v)} |\mathcal{S}_v(u')|}, \tag{1}$$

where $\mathcal{S}_v(v')$ denotes the set of valid spanning subgraph isomorphism relations that maps v' to v and $|\cdot|$ calculates the cardinality of the set.

Besides the difference in the probability of candidacy, the weights of the candidate nodes are also different. As shown in Fig. 5, $1'$, $3'$ and $5'$ are the candidate nodes for 3. Because $1'$ and $3'$ are the corresponding nodes of 1 and 3, they have non-zero weights while for $5'$, the weight is zero since it is dummy.

Now we propose our security criterion for a cell as follows to capture the spanning subgraph isomorphism relation and the observations above.

Definition 5 (k-Secure Cell)**.** *Given original graph G and FEOL graph H, we say that $v \in V_G$ is k-secure with respect to G and H if*

$$\sum_{u' \in \mathcal{C}(v)} \mathcal{P}_v(u')\omega_H(u') \leq \frac{1}{k}.$$

Following the definition above, for each $v \in V_G$ with k-security, the probability to pick a candidate node with a non-zero weight from $\mathcal{C}(v)$ is limited within $1/k$. In this way, the difference in weight and the probability of candidacy are enforced in the security criterion. Now we define the security criterion for the circuit netlist.

Definition 6 (k-Security)**.** *Given G and H, we say that $\langle G, H \rangle$ is k-secure if $\forall v \in V_G$ with $\omega_G(v) = 1$, v is k-secure with respect to G and H.*

By the above security criterion, we can guarantee that for any node that the attackers may target at, the probability to insert the Trojan into a node with a non-zero weight is always no greater than $1/k$. In this way, by making k large enough, we can guarantee much higher cost and risk for the Trojan insertion.

IV. k-Security Realization

To determine the spanning subgraph isomorphism relation, isomorphism checkings between the subgraphs of G and H are usually required, which can be very computation intensive. To avoid direct graph comparison, we adopt recent progress in privacy-preserving network publishing [18] to derive a sufficient condition for k-security. Our heuristic solution relies on the following concept denoted as k-isomorphism [18].

Definition 7 (k-Isomorphism)**.** *A graph is k-isomorphic if it consists of k disjoint isomorphic subgraphs.*

For example, the graph H of FEOL in Fig. 3(b) is 2-isomorphic with $V_{H_{s,0}} = \{1',3',5'\}$ and $V_{H_{s,1}} = \{2',4',6'\}$. Specifically, we call nodes $1'$ and $2'$ in the same position of $H_{s,0}$ and $H_{s,1}$. For $1'$ and $2'$, if $1' \in \mathcal{C}(1)$, then, $2' \in \mathcal{C}(1)$. Moreover, we must have $\mathcal{P}_1(1') = \mathcal{P}_1(2')$. Assume $1' = \phi(1)$, then, if $\omega(2') = 0$, 1 is 2-secure with respect to G and H in Fig. 3. Based on the observation, we have the following lemma for a k-isomorphic graph.

Lemma 1. *Given G and $H = \{H_{s,0}, \ldots, H_{s,k-1}\}$, which is k-isomorphic. $\forall v \in V_G$ with $\omega_G(v) = 1$ and $\phi(v) \in V_{H_{s,i}}$, where $i \in \{0, \ldots, k-1\}$, if each $u' \in V_{H_{s,j}} (j \neq i)$, where u' and $\phi(v)$ are in the same position of $H_{s,j}$ and $H_{s,i}$, respectively, satisfies $\omega_H(u') = 0$, then, v is k-secure with respect to G and H.*

We prove Lemma 1 in Appendix A. Lemma 1 formalizes the condition for $v \in V_G$ to be k-secure. Because we only require the nodes with non-zero weight to be k-secure, we have the following theorem for k-security.

Theorem 1. *Given G and H, assume $H = \{H_{s,0}, \ldots, H_{s,k}\}$, where $\{H_{s,0}, \ldots, H_{s,k-1}\}$ are k-isomorphic. G is k-secure with respect to H if $\forall v \in V_G$ with $\omega_G(v) = 1$, the following conditions are satisfied:*

1) *$\phi(v) \in V_{H_{s,i}}$ where $i \in \{0, \ldots, k-1\}$.*
2) *$\omega_H(u') = 0$, $\forall u' \in V_{H_{s,j}} (j \in \{0, \ldots, k-1\}, j \neq i)$, where u' and $\phi(v)$ are in the same position of $H_{s,j}$ and $H_{s,i}$, respectively.*

Example 3. *Consider the example shown in Fig. 6. H is composed of 3 subgraphs with $H_{s,0}$ and $H_{s,1}$ being isomorphic to each other. Nodes with non-zero weights like $3', 5', 6', 9'$ are either in $H_{s,0}$ or in $H_{s,1}$, while the weights of the nodes in the same position as them, i.e. $D, 2', 7', 8'$ are zero. Therefore,*

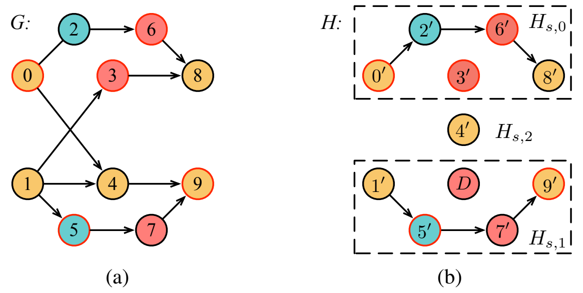


Fig. 6: Example for Theorem 1: G is 2-secure with respect to H.

they are 2-secure with respect to G and H according to Lemma 1. Node 4 remains unprotected since its weight is zero. Therefore, $\langle G, H\rangle$ is 2-secure. By introducing weights for each node and $H_{s,k}$, our framework is flexible to protect an arbitrary subset of circuit nodes to balance the trade-off between security and the introduced overhead.

Theorem 1 works as a sufficient condition for the proposed security criterion. It is not only fully compatible with the insertion of dummy cells and wires but also eliminates the requirements and computation overhead of determining the security level through graph isomorphism checkings in the FEOL generation process. The remaining question is how to effectively and efficiently achieve the requirements posed in Theorem 1. In the next section, we will describe our split manufacturing flow for the FEOL layer generation.

V. Practical Framework for Trojan Prevention

In this section, we propose our framework to generate the FEOL and BEOL layers. The inputs to the framework include the original circuit netlist and the selected nodes for protection. An MILP-based formulation, which considers the insertion of dummy wires and gates with wire lifting simultaneously, is first proposed to generate the k-secure FEOL layers. We further propose a novel LR-based algorithm and a minimum-cost-flow [26], [29], [30] transformation to enhance the scalability of the framework. In the second step, we propose a layout refinement technique, which enables us to leverage commercial tools for physical synthesis while guaranteeing the security in the placement stage.

A. MILP-based FEOL Generation

Following the sufficient condition proposed in Theorem 1, to achieve k-security, we need to generate $H = \{H_{s,0}, \dots, H_{s,k}\}$ from G so that all the nodes with non-zero weights are added to the first k subgraphs. Because the insertion of dummy wires and nodes is allowed, one trivial solution to generate H is to copy G for $k-1$ times. *This indicates that k-security can always be achieved when the insertion of dummy cells and wires is considered.* However, such a naive solution usually suffers from large overhead.

To reduce the introduced overhead, in this section, we propose a novel FEOL generation algorithm, whose pseudocode is shown in Algorithm 1. Our algorithm anonymizes all the

Algorithm 1 Iterative FEOL Generation

```
1: // $V_r$: the set of nodes that have not been inserted
2: $V_{crit} \leftarrow \{v \in V_G : \omega_G(v) = 1\}$, $V_r \leftarrow V_G$;
3: while $V_{crit} \neq \varnothing$ do
4:     $V_{min} \leftarrow \varnothing, c_{min} \leftarrow +\infty$;
5:     // $[t]$: the set of cell types
6:     for $i \in [t]$ do
7:         $V_i \leftarrow \{v \in V_r : \ell(v) = i\}$;
8:         $V_{sel}, c_{sel} \leftarrow$ NodeSelect$(k, V_i)$;
9:         if $c_{min} > c_{sel}$ then
10:            $V_{min} \leftarrow V_{sel}, c_{min} \leftarrow c_{sel}$;
11:        end if
12:    end for
13:    InsertToFEOL$(V_{min}, H_{s,0}, \dots, H_{s,k-1})$;
14:    $V_{crit} \leftarrow V_{crit} \setminus V_{min}$, $V_r \leftarrow V_r \setminus V_{min}$;
15: end while
16: $H_{s,k} \leftarrow V_r$;
```

TABLE II: Notations used in the MILP formulation.

x_i	$x_i = 1$ if the ith node is selected
x_{ij}	$x_{ij} = 1$ if the ith node is inserted to $H_{s,j}$
ω_i	weight of the ith node
d_j	$d_j = 1$ if a dummy node is inserted to $H_{s,j}$
y_l	$y_l = 1$ if an edge can be added from lth location to current location in $H_{s,0}, \dots, H_{s,k-1}$
y_{lj}	$y_{lj} = 1$ if an edge can be added from lth location to current location in $H_{s,j}$
z_l	$z_l = 1$ if an edge can be added from current location to lth location in $H_{s,0}, \dots, H_{s,k-1}$
z_{lj}	$z_{lj} = 1$ if an edge can be added from current location to lth location in $H_{s,j}$
IN_{ij}	set of starting locations of edges pointing to current location that can be added if ith node is added to $H_{s,j}$
OUT_{ij}	set of ending locations of edges pointing from current location that can be added if ith node is added to $H_{s,j}$
RES_i	set of edges connected to ith node from unadded node

selected nodes iteratively until all the nodes with non-zero weights are added to $H_{s,0}, \dots, H_{s,k-1}$. In each iteration, we select k nodes of the same label and make sure that exactly one node has a non-zero weight to satisfy Theorem 1 (lines 4–10). To select the nodes, we first cluster all the remaining nodes by their labels and then, select k nodes from each cluster with the minimized cost through an MILP-based formulation. The k nodes with the minimized cost among all the clusters are selected and inserted to $H_{s,0}, \dots, H_{s,k-1}$ (line 11). The iterative algorithm continues until all the nodes with non-zero weights are added to $H_{s,0}, \dots, H_{s,k-1}$.

The core part of the FEOL generation algorithm is the MILP-based node selection, i.e. NodeSelect. Before we introduce our MILP formulation, we list the notations used in the formulation in TABLE II and use the following example to illustrate the iterative strategy and the problem that we will solve in each iteration.

Example 4. *Consider the original graph G in Fig. 6(a). Assume nodes $0, 3, 5, 6, 9$ are selected for protection and the required security level is 2. To generate H from G, our strategy is to iteratively anonymize the selected nodes with non-zero weights by adding them to $H_{s,0}$ and $H_{s,1}$. As shown*

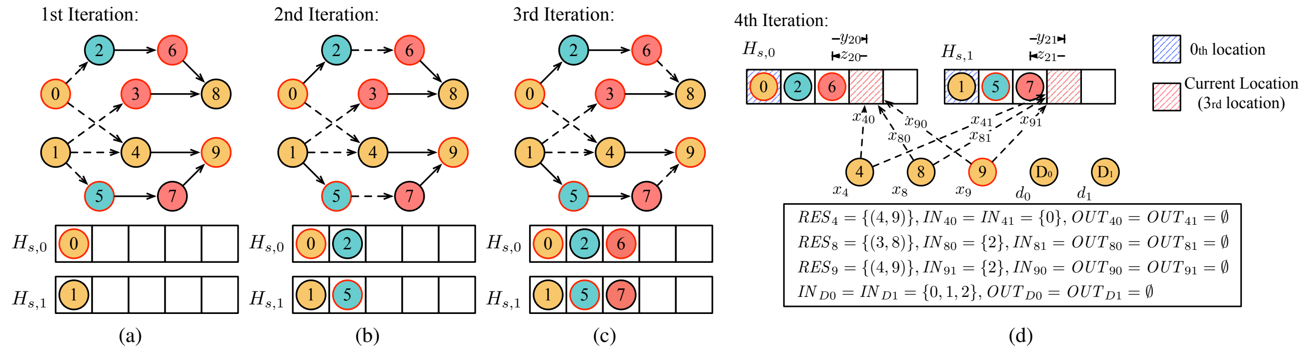

1st Iteration:
2nd Iteration:
3rd Iteration:
4th Iteration:
$H_{s,0}$
$H_{s,1}$
y_{20}
z_{20}
y_{21}
z_{21}
x_{40}
x_{90}
x_{80}
x_{41}
x_{81}
x_{91}
x_4
x_8
x_9
d_0
d_1
D0
D1
0th location
Current Location (3rd location)
$RES_4 = \{(4,9)\}, IN_{40} = IN_{41} = \{0\}, OUT_{40} = OUT_{41} = \emptyset$
$RES_8 = \{(3,8)\}, IN_{80} = \{2\}, IN_{81} = OUT_{80} = OUT_{81} = \emptyset$
$RES_9 = \{(4,9)\}, IN_{91} = \{2\}, IN_{90} = OUT_{90} = OUT_{91} = \emptyset$
$IN_{D0} = IN_{D1} = \{0,1,2\}, OUT_{D0} = OUT_{D1} = \emptyset$
(a)
(b)
(c)
(d)

Fig. 7: Example of the iterative strategy and the formulation in each iteration: (a) (b) (c) the first three iterations (dotted lines are the wires to be lifted to BEOL layers); (d) parameters and formulation for the fourth iteration.

in Figs. 7(a), 7(b), and 7(c), in the first three iterations, nodes $0, 2, 6$ and nodes $1, 5, 7$ are added to $H_{s,0}$ and $H_{s,1}$, respectively. For the nodes in the same position in $H_{s,0}$ and $H_{s,1}$, e.g. 2 and 5 in the first location, only one of them has a non-zero weight, which follows the requirement in Theorem 1. In each iteration, to select the nodes to insert into $H_{s,0}$ and $H_{s,1}$, we propose an MILP-based formulation to select a pair of nodes that share the same label and achieve the smallest insertion cost. We use Fig. 7(c) to explain the MILP formulation. Consider node 9 that has a non-zero weight. To anonymize it, we can find nodes 8 and 4 of the same cell type as node 9 and also allow the insertion of dummy nodes d_0 and d_1. If we add node 4 to $H_{s,0}$, because edge $(0,4)$ exists in G, we have $IN_{40} = \{0\}$, which indicates that there is one edge, i.e. $(0,4)$, pointing from the 0th location in $H_{s,0}$ to the current location that can be added to $H_{s,0}$ if node 4 is inserted. Similarly, if we add node 9 to $H_{s,1}$, because edge $(7,9)$ exists in G, we have $IN_{91} = \{3\}$. For the dummy nodes D_0, we have $IN_{D0} = \{0,1,2\}$ and $OUT_{D0} = \emptyset$. This is because to retain the correct circuit functionality, we allow inserting dummy edges connecting to the input of the dummy nodes but forbid using the dummy nodes to drive other nodes. Hence, we can determine IN and OUT for each node following the rule, which is listed in Fig. 7(d). Meanwhile, because $(4,9)$ is the only edge connecting node 4 to unadded nodes, we have $RES_4 = \{(4,9)\}$. When node 4 is added, all the edges in RES_4 will need to be lifted to BEOL.

Now, we introduce our MILP formulation for the node selection. We split the formulation into different parts to enable an easy explanation. The objective function is to minimize the cost of node selection,

$$\min_{x,d} \quad \alpha \sum_i |RES_i| x_i - \beta k \sum_l (y_l + z_l) + \gamma A \sum_j d_j. \tag{2}$$

The cost function mainly consists of three parts: the number of edges to be lifted to BEOL layers, i.e. $\sum_i |RES_i| * x_i$, the number of edges that can be added back to the FEOL layers, i.e. $k \sum_l (y_l + z_l)$, and the area of the inserted dummy nodes $A \sum_j d_j$. α, β, and γ are coefficients used to control the trade-off between dummy node insertion and wire lifting. In our framework, to achieve better efficiency, a linear function is used as the optimization objective. By using a linear function, we implicitly assume that the introduced overhead is linearly dependent on the number of lifted wires and the area of the dummy nodes. Meanwhile, the cost of the lifted wires and the cost of the dummy nodes are assumed to be independent. To capture the dependency between the lift wires and the dummy nodes, a more complex nonlinear function is required, which may not be convex and can be extremely computation-intensive to optimize. We empirically find that with such a linear objective function, our framework can already significantly reduce the introduced overhead compared with the existing method. We leave in-depth research on the possibility and advantage of using more complicated non-linear functions for the optimization objective as one of our future research directions.

Now we explain the constraints. For a node i, it can at most be inserted into one subgraph, which is enforced by Eq. (3a). Meanwhile, for the jth subgraph, we require exactly one node to be inserted as enforced by Eq. (3b). We further pose the constraint in Eq. (3c) to ensure that exactly one node has a non-zero weight to satisfy Theorem 1.

$$\sum_{j=0}^{k-1} x_{ij} = x_i, \qquad \forall i; \tag{3a}$$

$$\sum_i x_{ij} + d_j = 1, \qquad \forall j \in \{0, \ldots, k-1\}; \tag{3b}$$

$$\sum_i x_i w_i = 1. \tag{3c}$$

Next, we need to determine the conditions for an edge to be inserted back to the FEOL layers. Consider the edge pointing from the lth position to the current position in $H_{s,0}, \ldots, H_{s,k-1}$. In the jth subgraph $H_{s,j}$, an edge pointing from lth position can be added back under two conditions: 1) a dummy cell is inserted to the current position; 2) node i with $l \in IN_{ij}$ is inserted. Furthermore, to satisfy the requirement on subgraph isomorphism, the edge pointing from the lth position can be added back only when it can be added back in all the k subgraphs. These two requirements can be formalized with constraints Eq. (4a). Note $1_{l \in IN_{ij}}$ is the indicator function that equals to 1 when $l \in IN_{ij}$ and equals to 0, otherwise. Similarly, for the edge pointing from the current position to the lth position, we have almost the same constraints as shown in Eq. (4b) except that the insertion

Algorithm 2 LR-based Node Selection

```
Require: $k$: security level, $V$: the set of vertices to select.
Ensure: $V_{sel}$: selected vertices, $c_{sel}$: cost of vertex selection.
1: function NodeSelect($k, V$)
2:     $\lambda_{jl} \leftarrow 0$, $\mu_{jl} \leftarrow 0$, $it \leftarrow 0$;
3:     while $it \le it_{max}$ do
4:         // See Section V-B1
5:         $V_{sel}, c_{sel} \leftarrow$ LagRelaxationSolve($V, \lambda_{jl}^{it}, \mu_{jl}^{it}$);
6:         // See Section V-B2
7:         $\lambda_{jl}^{it+1}, \mu_{jl}^{it+1} \leftarrow$ UpdateCoeff($\lambda_{jl}^{it}, \mu_{jl}^{it}$);
8:     end while
9: end function
```

of dummy edges pointing to the corresponding nodes is no longer allowed.

$$
\begin{aligned}
&y_l \le y_{lj}, \quad y_{lj} \le \sum_i x_{ij} \cdot 1_{l \in IN_{ij}} + d_j, && \forall j, l;\\
&z_l \le z_{lj}, \quad z_{lj} \le \sum_i x_{ij} \cdot 1_{l \in OUT_{ij}}, && \forall j, l.
\end{aligned}
$$

The constraints can be further simplified by substituting y_{lj} and z_{lj}, we have

$$
\begin{aligned}
&y_l \le \sum_i x_{ij} \cdot 1_{l \in IN_{ij}} + d_j, && \forall j, l; && (4a)\\
&z_l \le \sum_i x_{ij} \cdot 1_{l \in OUT_{ij}}, && \forall j, l. && (4b)
\end{aligned}
$$

Based on the explanation above, we have the following ILP formulation for the node selection and insertion.

$$
\begin{aligned}
\min_{x,d} \quad & \text{Eq. (2)}\\
\text{s.t.} \quad & \text{Eqs. (3a)} - \text{(3c)}, \text{(4a)} - \text{(4b)}.
\end{aligned}
$$

While all the variables in the formulation, including x_{ij}, d_j, y_l, and z_l, should be integer variables, we can relax y_l and z_l to be continuous without changing the optimal solution and achieve a better efficiency. By continuing the process iteratively, we can insert all the nodes with non-zero weights into the first k subgraphs while keeping the k subgraphs isomorphic at the same time. Then, we add all the remaining nodes into $H_{s,k}$.

B. Lagrangian Relaxation Algorithm

The MILP-based formulation enables us to select and insert k nodes to $H_{s,0}, \ldots, H_{s,k-1}$ with a minimum cost for each iteration. However, it is still computationally expensive and suffers from unaffordable runtime for large benchmarks. We observe that two constraints that are hard to solve are Constraints (4a) and (4b). Therefore, to accelerate the framework, we apply LR to relax the last two constraints and modify the objective function as in (5).

$$
\begin{aligned}
&\alpha \sum_{i,j} |RES_i| x_{ij} - \beta k \sum_l (y_l + z_l) + \gamma A \sum_j d_j\\
&\quad + \sum_{j,l} \lambda_{jl} (-\sum_i x_{ij} \cdot 1_{l \in IN_{ij}} - d_j + y_l)\\
&\quad + \sum_{j,l} \mu_{jl} (-\sum_i x_{ij} \cdot 1_{l \in OUT_{ij}} + z_l)\\
=& \sum_{i,j} (\alpha |RES_i| - \sum_l \lambda_{jl} \cdot 1_{l \in IN_{ij}} - \sum_l \mu_{jl} \cdot 1_{l \in OUT_{ij}}) x_{ij}\\
&\quad + \sum_j (\gamma A - \sum_l \lambda_{jl}) d_j - \sum_l (\beta k - \sum_j \lambda_{jl}) y_l\\
&\quad - \sum_l (\beta k - \sum_j \mu_{jl}) z_l. \qquad (5)
\end{aligned}
$$

Here $\mu_{jl} \ge 0$ and $\lambda_{jl} \ge 0$ are the Lagrangian multipliers. The constraints now only consist of Constraints (3a), (3b), and (3c). Compared with the original formulation, we remove the hard constraints, i.e. Constraints (4a) and (4b), and penalize the constraint violations in the objective function by updating λ_{jl} and μ_{jl}. By repeating the process of solving and updating the new formulation, the node selection algorithm will progressively converge to a legal solution to the original formulation. The proposed algorithm is summarized in Algorithm 2.

1) Minimum-Cost Flow Transformation: For the new formulation, given fixed Lagrangian multipliers λ_{jl} and μ_{jl}, one important observation is that x_{ij} and d_j become independent with y_l and z_l. Therefore, we can decompose the new formulation into two independent subproblems. The first subproblem is defined as below:

$$
\begin{aligned}
\min_{x,d} \quad & \text{Eq. (5)}\\
\text{s.t.} \quad & \text{Eqs. (3a)} - \text{(3c)},
\end{aligned}
$$

where x_{ij}, x_i and d_j are all binary variables. The second subproblem is defined as below:

$$
\min_{x,d} \; -\sum_l (\beta k - \sum_j \lambda_{jl}) y_l - \sum_l (\beta k - \sum_j \mu_{jl}) z_l \qquad (6)
$$

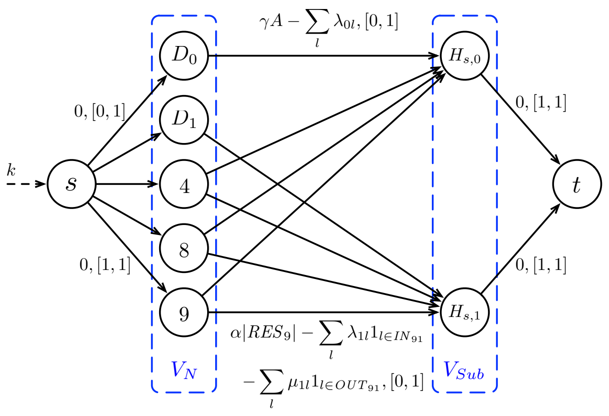
where y_l and z_l are binary variables.

The solution to the second subproblem can be acquired easily as below since the objective function is monotone with y_l and z_l while y_l and z_l are independent given fixed λ_{jl} and μ_{jl} for different l in each iteration.

$$
y_l = \begin{cases} 0, & \beta k - \sum_j \lambda_{jl} < 0;\\ 1, & \text{otherwise}, \end{cases}
$$

$$
z_l = \begin{cases} 0, & \beta k - \sum_j \mu_{jl} < 0;\\ 1, & \text{otherwise}. \end{cases}
$$

For the first subproblem, one notable merit is that it can be transformed into a minimum-cost flow problem. Fig. 8 shows an example of the constructed graph for the minimum-cost flow problem. The variables, constraints, and objectives for the first subproblem can be transformed to the concepts in the



$\gamma A - \sum_l \lambda_{0l}, [0,1]$
$0, [0,1]$
$0, [1,1]$
$0, [1,1]$
$0, [1,1]$
k
s
t
D_0
D_1
4
8
9
$H_{s,0}$
$H_{s,1}$
$\alpha|RES_9| - \sum_l \lambda_{1l} 1_{l \in IN_{91}}$
$-\sum_l \mu_{1l} 1_{l \in OUT_{91}}, [0,1]$
V_N
V_{Sub}

Fig. 8: Example of the minimum-cost flow formulation for node selection (k=2 in the example).

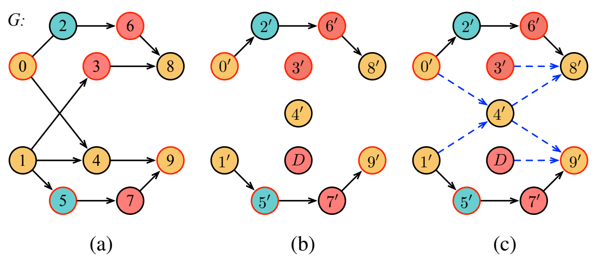


Fig. 9: Comparison of the existing placement strategy and our layout refinement strategy: (a) original circuit graph; (b) existing strategy only considers the FEOL layers in the placement stage; (c) our strategy adds virtual nets to force a cell to be placed close to its neighbors (dotted blue lines are the virtual nets).

flow problem. As shown in Fig. 8, V_N represents the set of vertices corresponding to the cells to be inserted, including the remaining nodes, i.e. nodes $4, 8$ and 9, and the dummy nodes, i.e. D_0 and D_1. V_{sub} denotes the set of vertices corresponding to the subgraphs, i.e. $H_{s,0}$ and $H_{s,1}$. Edges correspond to the variables in the formulation. For example, $(s, 4)$ corresponds to x_4 while $(4, H_{s,0})$ corresponds to x_{40}. Each edge is marked with the cost as well as the upper and lower bound of the capacity in Fig. 8. While the capacity is determined by the range of the variables, the cost for each edge is determined following the coefficients in the objective function. It should be noted that for some edges, both the upper and lower bound of the capacity is 1, which means we require a non-zero flow for the edge in the final solution. This indeed corresponds to the constants in the constraints. For example, for $(9, H_{s,1})$, which corresponds to x_{91} in the original formulation, the cost becomes $\alpha|RES_9| - \sum_l \lambda_{1l} \cdot 1_{l\in IN_{91}} - \sum_l \mu_{1l} \cdot 1_{l\in OUT_{91}}$, which equals to α based on Fig. 7(d). Based on the transformation above, we can easily verify all the constraints and the objective in the original formulation can be realized in the minimum-cost flow problem.

The minimum-cost flow transformation enables us to leverage efficient graph algorithms [30] to solve the originally MILP problem. As we will show in Section VI, significant runtime improvement can be achieved through the transformation.

2) *Lagrangian Multiplier Update:* One key step within the current node selection framework shown in Algorithm 2 is how to update the Lagrangian multiplier λ_{jl} and μ_{jl} after each iteration. Various updating strategies may have different convergence issues. Following [25], the most widely used updating strategy for λ_{jl} and μ_{jl} is

$$\lambda_{jl}^{it+1} = \max(0, \lambda_{jl}^{it} + t_{it}(y_l - \sum_i x_{ij} \cdot 1_{IN_{ij}} - d_j)),$$
$$\mu_{jl}^{it+1} = \max(0, \mu_{jl}^{it} + t_{it}(z_l - \sum_i x_{ij} \cdot 1_{OUT_{ij}})),$$

where $t_{it} = 1/it^{\eta}$ is the step size chosen for the update [25] and η is a constant.

Ideally, by iteratively updating λ and μ, the number of violations of the relaxed constraints can be reduced and the objective function in Eq. (5) gradually converges. However, while the number of violations indeed reduces significantly in the first several iterations, we observe severe oscillation for the objective function afterwards. To overcome the convergence problem, after the first several iterations, we modify the original updating strategy as below,

$$\lambda_{jl}^{it+1} = \lambda_{jl}^{it} + \max(0, t_{it}(y_l - \sum_i x_{ij} \cdot 1_{IN_{ij}} - d_j)),$$
$$\mu_{jl}^{it+1} = \mu_{jl}^{it} + \max(0, t_{it}(z_l - \sum_i x_{ij} \cdot 1_{OUT_{ij}})).$$

Our updating strategy increases λ and μ monotonically to force the value of y_l and z_l towards 0 in order to resolve the constrain violations and guarantee the convergence of the node selection algorithm. By controlling the maximum iteration, i.e. it_{max}, and the step size, i.e. η, we can control the trade-off between the solution quality and the runtime of the program.

C. *k-Secure Layout Refinement*

After solving the MILP-based formulation, cells and connections in the FEOL layers H can be determined such that $\langle G, H\rangle$ is k-secure. The next step is to do physical synthesis to generate the layouts for the FEOL and BEOL layers. In the placement stage, existing commercial tools usually target at minimizing the total wirelength, and thus, tend to place the cells with actual connections close to each other. This, however, makes it possible for the attackers to recover the connections in the BEOL layers based on the physical proximity information [14], [19].

To guarantee the security while leveraging existing physical synthesis tools, the previous method [13] chooses to ignore the lifted wires in the BEOL layers in the placement stage. For example, consider the original circuit graph shown in Fig. 9(a), following [13], only the FEOL graph shown in Fig. 9(b) is considered in the placement stage. This helps to avoid the impact of connections in the BEOL layers, and thus, forbids the attacker from determining the identity of the nodes by physical proximity information. Though secure, this method can suffer from large overhead. This is because when the wire connections in the BEOL layers are ignored, many cells in the FEOL layers are left floating, e.g. nodes $3', 4', D$

in Fig. 9(b). Therefore, the distances between the cells that are actually connected in the BEOL layers, e.g. nodes $4'$ and $9'$, become highly unoptimized.

To reduce the introduced wirelength overhead, we propose a novel layout refinement technique in the placement stage. As shown in Fig. 9(c), the basic idea of the refinement technique is to insert virtual nets between the circuit nodes that may or may not be connected in the original netlist, so that both the physical proximity between originally connected nodes and the indistinguishability among candidate nodes can be preserved.

More specifically, consider $v_i, u_j \in V_G$ with $(v_i, u_j) \in E_G$ and $i, j \in \{0, \ldots, k\}$. Their corresponding nodes v'_i, u'_j locate in the ith and jth subgraph of H, respectively, i.e. $v'_i \in V_{H_{s,i}}$, $u'_j \in V_{H_{s,j}}$. Then, depending on i and j, there are following situations:

- When $i = j = k$, (v'_i, u'_j) must exist in the FEOL layers and thus, no virtual nets need to be added.
- When $i = k$ and $j \neq i$, (v'_i, u'_j) is lifted to the BEOL layers. $\forall u'_{j'} \in V_{H_{s,i'}}$ with $u'_{j'}$ in the same position as u'_j and $j' \in \{0, \ldots, k-1\}$, we insert a virtual net $(v'_i, u'_{j'})$.
- When $j = k$ and $j \neq i$, (v'_i, u'_j) is lifted to the BEOL layers. $\forall v'_{i'} \in V_{H_{s,i'}}$ with $v'_{i'}$ in the same position as v'_i and $i' \in \{0, \ldots, k-1\}$, we insert a virtual net $(v'_{i'}, u'_j)$.
- When $i \neq k$, $j \neq k$, and $i = j$, then, $\forall v'_{i'}, u'_{i'} \in V_{H_{s,i'}}$ with $v'_{i'}$ and $u'_{i'}$ in the same positions as v'_i and u'_j, respectively, and $i' \in \{0, \ldots, k-1\}$, we insert a virtual net $(v'_{i'}, u'_{i'})$.
- When $i \neq k$, $j \neq k$, and $i \neq j$, we do not insert any virtual nets.

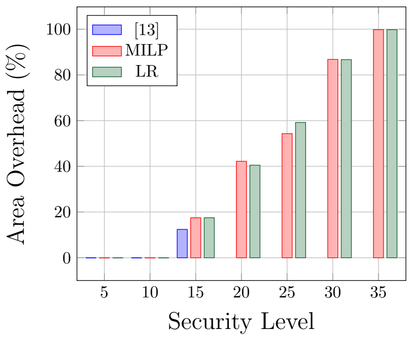
Example 5. *Consider the original graph and the FEOL graph in Figs. 9(a) and 9(c).* $\{0', \ldots, 9'\}$ *are the corresponding nodes for* $\{0, \ldots, 9\}$*, respectively. For* $(0, 4) \in E_G$*, we have* $0' \in H_{s,0}$ *and* $4' \in H_{s,2}$*. Therefore, following the insertion rule above, we insert two virtual nets, i.e.* $(0', 4')$ *and* $(1', 4')$*, in the placement stage. Similarly, for* $(4, 9) \in E_G$*, we also insert two virtual nets, i.e.* $(4', 8')$ *and* $(4', 9')$*. For* $(3, 8) \in E_G$*, because both* $3'$ *and* $8'$ *locate in* $H_{s,0}$*, we insert virtual nets* $(3', 8')$ *and* $(D, 9')$ *to* $H_{s,0}$ *and* $H_{s,1}$ *For* $(1, 3) \in E_G$*, because* $3' \in V_{H_{s,0}}$ *and* $1' \in V_{H_{s,1}}$*, we do not insert any virtual nets in this case.*

By inserting the virtual nets, we not only guarantee the security but also make sure a node is still placed close to its neighbors. As we will show in Section VI, our layout refinement technique allows for 49.6% overhead reduction compared with the existing method [13]. In the placement stage, because existing methods usually target at minimizing the total wirelength, cells with actual connections tend to be placed close to each other.

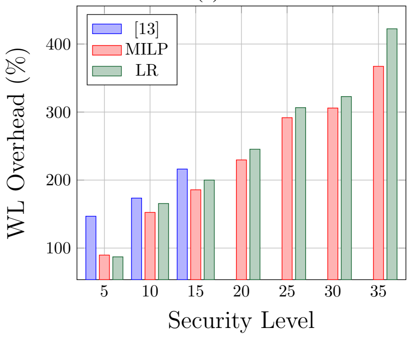
VI. Experimental Results

A. Experimental Setup

In this section, we report on our experiments to demonstrate the effectiveness of the proposed split manufacturing framework. The input to our framework is a gate-level netlist and the nodes to protect. In our experiments, to select the nodes for protection, we follow the Trojan insertion methods used by TrustHub [31]. Given the netlist, we first calculate the signal probability, logic switching probability and observability for each circuit node, and then, select the nodes with rare circuit events by comparing with a certain threshold. We modify the threshold to change the portion of nodes for protection. Our benchmarks are selected from the ISCAS benchmark suite [32] as well as the functional units (`shifter`, `alu`, and `div`) from the OpenSPARC T1 processor, the detailed statistics of which are shown in TABLE III. In our split manufacturing scheme, following [22], FEOL layers consist of all the cells and lower metal layers up to metal 3, while BEOL layers consist of metal 4 and above. We implement our framework in C++ and use GUROBI [33] and LEMON [34] packages to solve the MILP problem and the minimum-cost flow problem, respectively. We conduct physical synthesis using Cadence Encounter [35]. All the experiments are carried out on an eight-core 3.40 GHz Linux server with 32 GB RAM. We set the runtime limit for all the algorithms to 1.5×10^5 seconds.



(a)



(b)

Fig. 10: Comparison with [13] on the (a) wirelength and (b) area overhead for different security levels.

B. FEOL Generation Strategy Comparison

We compare the proposed MILP-based and LR-based algorithm with the previous method [13]. We set the required security level to be 10 and protect 5% of all the circuit nodes. We also set $\alpha = 0.5$, $\beta = 2.0$, and $\gamma = 0.6$. The number of LR iterations is 10 in the LR-based algorithm. We will demonstrate the impact of α, β, and γ.

We first compare the efficiency of the three algorithms. In TABLE III, "RT" denotes the runtime, while "ΔArea" and "ΔWL" denote the area and wirelength overhead compared

TABLE III: Runtime and Overhead comparison between the MILP-based and the LR-based algorithms.

Benchmark	# PI	# PO	# Protect	# Nodes	[13]			MILP			LR-based		
					RT (s)	ΔArea (%)	ΔWL (%)	RT (s)	ΔArea (%)	ΔWL (%)	RT (s)	ΔArea (%)	ΔWL (%)
`c432`	36	7	11	214	414.07	0.0	175.4	1.029	1.05	152.4	0.134	1.03	165.5
`c880`	60	25	35	450	32442	54.1	246.5	3.675	45.1	59.9	0.621	44.8	56.4
`c1908`	33	25	26	519	N/A	N/A	N/A	7.731	37.2	47.8	0.936	38.0	51.6
`c3540`	50	17	51	1012	N/A	N/A	N/A	40.10	37.9	96.3	4.080	37.7	104.5
`c5315`	178	101	94	1864	N/A	N/A	N/A	177.3	42.3	82.6	16.07	41.8	68.2
`c6288`	32	32	129	2568	N/A	N/A	N/A	276.1	38.1	206.9	32.68	38.4	220.9
`shifter`	79	64	130	2580	N/A	N/A	N/A	538.8	40.9	214.9	33.55	40.8	195.0
`alu`	400	121	153	3357	N/A	N/A	N/A	436.3	24.0	162.7	47.89	24.0	149.9
`div`	919	663	287	5720	N/A	N/A	N/A	2645.4	47.5	43.4	206.8	47.7	58.9

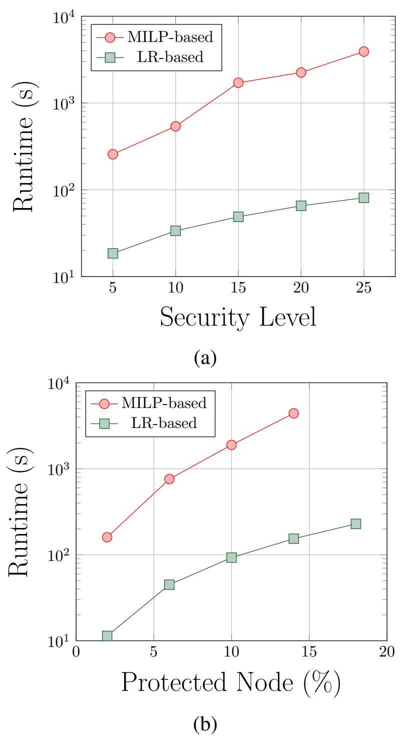


Fig. 11: Runtime dependency on (a) the required security level and (b) the number of protected nodes.

with the original circuit. As shown in TABLE III, on small benchmarks, compared with [13], the LR-based algorithm achieves 27000× speedup. For large benchmarks, while [13] cannot finish within the pre-defined time threshold, our LR-based algorithm can finish within 210s. Compared with the MILP-based algorithm, as shown in TABLE III, the LR-based algorithm can achieve on average 9.90× speedup.

We also explore the runtime dependency of the MILP-based and LR-based algorithms on the required security level k and the portion of the protected nodes. We choose the benchmark `shifter` for the study. As shown in Figs. 11(a) and 11(b), the LR-based algorithm achieves better scalability compared with the MILP-based algorithm. In Fig. 11(b), when the portion of protected nodes exceeds 18%, the MILP-based algorithm cannot be finished within the pre-defined time threshold, while it only takes 230s for the LR-based algorithm to finish.

We then compare the overhead introduced by the three algorithms on different benchmarks as shown in TABLE III. For the two small benchmarks, our MILP-based algorithm introduces on average 104% less wirelength overhead compared with the previous method with on average 3.97% area overhead reduction. The area and wirelength overhead introduced by the MILP-based and ILP-based algorithms are very similar.

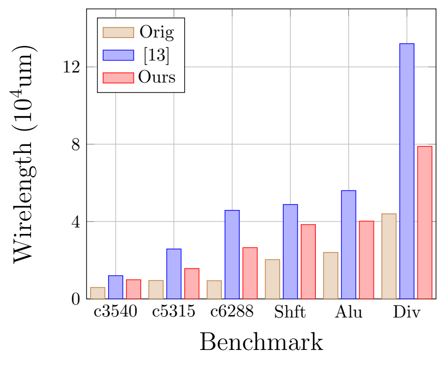
We then compare the overhead increase with the change of the required security level k. We use the benchmark `c432` as an example due to the runtime limit of the previous method. As shown in Fig. 10, with the increase of k, the introduced area and wirelength overhead of all the three methods increases significantly. Specifically, when k is small, e.g., k equals to 5, 10, or 15, our MILP-based method achieves much better wirelength overhead reduction with a slightly larger area overhead. When k is larger than 15, the previous method cannot generate the FEOL layers for the required security level, while our MILP-based method can guarantee to achieve all the required security level. Meanwhile, we also observe that with the increase of k, the difference on the introduced overhead by the MILP-based and LR-based algorithms becomes larger.

C. Physical Synthesis Comparison

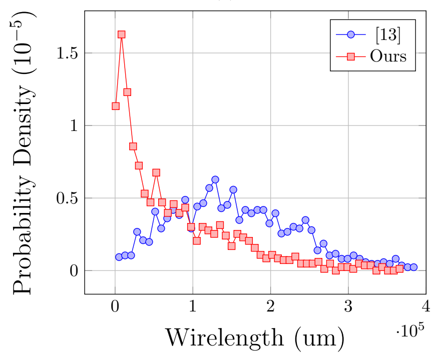
We then compare our placement refinement strategy based on the virtual net insertion with the original method proposed in [13]. The FEOL layers are generated with our MILP-based algorithm following the settings in Section VI-B. In Fig. 12(a), we show the routed wirelength for three different strategies, including direct placement without considering k-security ("Orig"), our placement refinement method ("Ours") and the previous method [13]. Compared with the previous method, our placement refinement strategy provides on average 97.5% wirelength overhead reduction. The overhead introduced by the three algorithms are the same. For the large benchmark `div`, our method achieves around 120% wirelength overhead reduction. To understand the origin of the large wirelength reduction, we plot the wirelength distribution for different nets in benchmark `div` in Fig. 12(b). As we can see, by inserting the virtual nets, the wirelengths between the neighboring cells are reduced significantly.

D. Physical Proximity Examination

We then carry out physical proximity checking to examine the security of the layout of the FEOL layers. In our framework, all the nodes in $H_{s,k}$ are unprotected and their identity



(a)



(b)

Fig. 12: Overhead comparison between our layout refinement technique and [13]: (a) wirelength comparison, and (b) wirelength distribution for `div`.

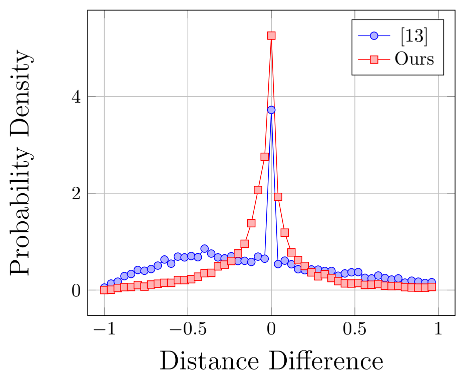


Fig. 13: Distance difference comparison with [13].

can be determined exactly by the attackers. For example in Fig. 9, node $4'$ can be identified as the corresponding node for node 4. To guarantee security, we need to prevent the attackers from identifying the protected nodes based on the identified unprotected nodes. For instance, while node $9'$ is connected to node $4'$ in the BEOL layers in Fig. 9, we hope that the distance between node $8'$ and $4'$ to be close to the distance between node $9'$ and $4'$. We select the benchmark `div` and set the security level to be 10. We then compare the selected nodes and their candidate nodes on the physical proximity to their neighbors. The distribution of the distance difference is shown in Fig. 13. As we can see, the distance difference is distributed symmetrically around 0, which indicates similar distance is achieved for the protected nodes and their candidates. This distance similarity makes the identification of the protected nodes to be nearly impossible. If we simply selected the nodes that are closest to the identified nodes, then, in all the benchmarks listed in TABLE III, we find the number of nodes that can be correctly identified is 0. The results indicate the requirement posed by k-security is much higher than that of the proximity attacks, which is also the origin of the large overhead introduced to achieve k-security.

E. Relation between Overhead and Framework Parameters

At last, we study the change of overhead as the increase of the security level k, the number of protected nodes and the coefficients γ in the MILP formulation. We use `shifter` benchmark as an example. In Fig. 14(a), to achieve 10-security, we show the increase of the overhead with the increase of the protected nodes. In Fig. 14(b), we show the relation between overhead and the required security level in order to protect 5% of nodes. In Fig. 14(c), we fix $\alpha = 0.5$, $\beta = 2.0$ in the MILP formulation and change γ from 0.6 to 1.4. By changing γ, cell insertion and wire lifting are balanced to help provide better usage of the routing resources and the chip space for different designs.

VII. Conclusion

In this paper, we propose a framework to enhance the security and practicality of split manufacturing. A new security criterion is proposed and its sufficient condition is obtained to enable more efficient realization. To realize the sufficient condition, wire lifting, dummy cell and wire insertion are considered simultaneously through a novel MILP formulation for the first time. Layout refinement that is fully compatible with existing physical design flow is also proposed. The proposed framework achieves much better efficiency, overhead reduction, and security guarantee compared with existing methods.

References

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[2] S. Bhasin and F. Regazzoni, "A survey on hardware trojan detection techniques," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, pp. 2021–2024.

[3] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 18–37.

[4] C. Krieg, C. Wolf, and A. Jantsch, "Malicious LUT: a stealthy FPGA trojan injected and triggered by the design flow," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 43:1–43:8.

[5] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *ACM/IEEE Design Automation Conference (DAC)*, 2011, pp. 333–338.

[6] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2017, pp. 95–100.

[7] Y. Xie, C. Bao, and A. Srivastava, "Security-aware design flow for 2.5D IC technology," in *International Workshop on Trustworthy Embedded Devices (TrustED)*, 2015, pp. 31–38.

[8] K. Vaidyanathan, R. Liu, E. Sumbul, Q. Zhu, F. Franchetti, and L. Pileggi, "Efficient and secure intellectual property (IP) design with split fabrication," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2014, pp. 13–18.

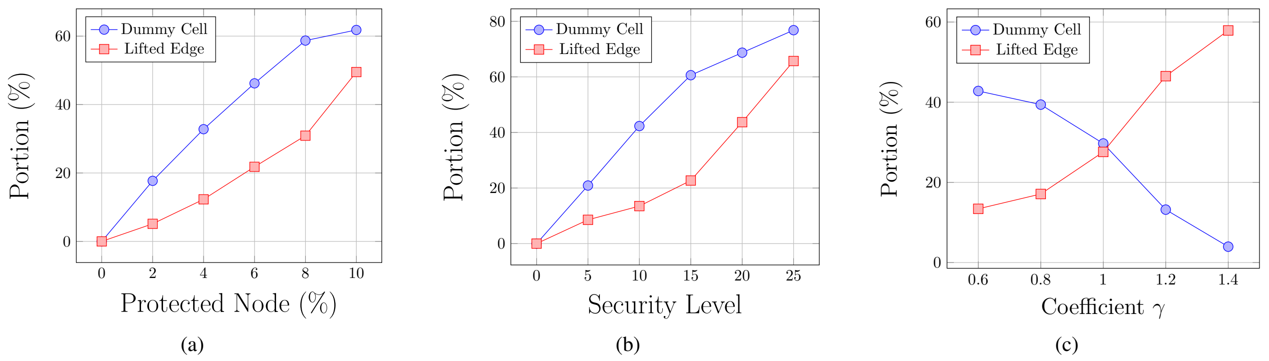


Fig. 14: The relation between overhead and (a) portion of protected nodes, (b) security level, and (c) MILP coefficients.

[9] B. Hill, R. Karmazin, C. T. O. Otero, J. Tse, and R. Manohar, "A split-foundry asynchronous FPGA," in *IEEE Custom Integrated Circuits Conference (CICC)*, 2013, pp. 1–4.

[10] J. Valamehr, T. Sherwood, R. Kastner, D. Marangoni-Simonsen, T. Huffmire, C. Irvine, and T. Levin, "A 3-D split manufacturing approach to trustworthy system development," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 4, pp. 611–615, 2013.

[11] K. Vaidyanathan, B. P. Das, and L. Pileggi, "Detecting reliability attacks during split fabrication using test-only BEOL stack," in *ACM/IEEE Design Automation Conference (DAC)*, 2014, pp. 156:1–156:6.

[12] K. Xiao, D. Forte, and M. M. Tehranipoor, "Efficient and secure split manufacturing via obfuscated built-in self-authentication," in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, 2015, pp. 14–19.

[13] F. Imeson, A. Emtenan, S. Garg, and M. V. Tripunitara, "Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation," in *USENIX Security Symposium*, 2013, pp. 495–510.

[14] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *IEEE/ACM Proceedings Design, Automation and Test in Eurpoe (DATE)*, 2013, pp. 1259–1264.

[15] Q. Shi, K. Xiao, D. Forte, and M. M. Tehranipoor, "Securing split manufactured ICs with wire lifting obfuscated built-in self-authentication," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2017, pp. 339–344.

[16] S. Garg and J. J. Rajendran, "Split manufacturing," in *Hardware Protection through Obfuscation*. Springer, 2017, pp. 243–262.

[17] Y. Xie, C. Bao, and A. Srivastava, "3D/2.5 D IC-based obfuscation," in *Hardware Protection through Obfuscation*. Springer, 2017, pp. 291–314.

[18] J. Cheng, A. W.-C. Fu, and J. Liu, "K-isomorphism: privacy preserving network publication against structural attacks," in *ACM Conference on Management of Data (SIGMOD)*, 2010, pp. 459–470.

[19] Y. Wang, P. Chen, J. Hu, and J. Rajendran, "The cat and mouse in split manufacturing," in *ACM/IEEE Design Automation Conference (DAC)*, 2016, pp. 165:1–165:6.

[20] J. Magaña, D. Shi, and A. Davoodi, "Are proximity attacks a threat to the security of split manufacturing of integrated circuits?" in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 90:1–90:7.

[21] Y. Wang, P. Chen, J. Hu, and J. J. Rajendran, "Routing perturbation for enhanced security in split manufacturing," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2017, pp. 605–510.

[22] Y. Wang, T. Cao, J. Hu, and J. Rajendran, "Front-end of line attacks in split manufacturing," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017.

[23] L. Feng, Y. Wang, W.-K. Mak, J. Rajendran, and J. Hu, "Making split fabrication synergistically secure and manufacturable," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017.

[24] A. Sengupta, S. Patnaik, J. Knechtel, M. Ashraf, S. Garg, and O. Sinanoglu, "Rethinking split manufacturing: An information-theoretic approach with secure layout techniques," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2017.

[25] M. M. Ozdal, "Detailed-routing algorithms for dense pin clusters in integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 28, no. 3, pp. 340–349, 2009.

[26] Y. Lin, B. Yu, X. Xu, J.-R. Gao, N. Viswanathan, W.-H. Liu, Z. Li, C. J. Alpert, and D. Z. Pan, "MrDP: Multiple-row detailed placement of heterogeneous-sized cells for advanced nodes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2017.

[27] S. Skorobogatov and C. Woods, "Breakthrough silicon scanning discovers backdoor in military chip," in *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*. Springer, 2012, pp. 23–40.

[28] D. B. West, *Introduction to Graph Theory*. Prentice Hall, 2000.

[29] A. V. Goldberg, "An efficient implementation of a scaling minimum-cost flow algorithm," *Journal of Algorithms*, vol. 22, no. 1, pp. 1–29, 1997.

[30] J. Kleinberg and E. Tardos, "Network Flow," in *Algorithm Design*. Pearson Education, 2005.

[31] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *IEEE International Conference on Computer Design (ICCD)*, 2013, pp. 471–474.

[32] F. Brglez, D. Bryan, and K. Koźmiński, "Combinational profiles of sequential benchmark circuits," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 1989, pp. 1929–1934.

[33] Gurobi Optimization Inc., "Gurobi optimizer reference manual," http://www.gurobi.com, 2016.

[34] "LEMON," http://lemon.cs.elte.hu/trac/lemon.

[35] "Cadence SOC Encounter," http://www.cadence.com.

Appendix A
Proof of Lemma 1

Proof. Consider $v \in V_G$ with $\omega_G(v) = 1$ and $H = \{H_{s,0}, \ldots, H_{s,k-1}\}$, which is k-isomorphic. Recall $\mathcal{C}(v)$ denotes the candidate set of v and for each $v' \in \mathcal{C}(v)$, the probability of candidacy, i.e. $\mathcal{P}_v(v')$, is defined in Eq. (1). For $v' \in \mathcal{C}(v)$, without loss of generality, we assume $v' \in V_{H_{s,0}}$. Then, in $H_{s,1}, \ldots, H_{s,k-1}$, there must be $k-1$ other nodes in the same position as v' that are also in $\mathcal{C}(v)$ and have the same probability of candidacy. Let $\mathcal{L}_i(v)$ be the set of positions of the nodes in $H_{s,i}$ that are in $\mathcal{C}(v)$, for $i \in \{0, \ldots, k-1\}$. Then, we have $\mathcal{L}_0(v) = \ldots = \mathcal{L}_{k-1}(v)$.

Let $V_{H_{s,i}}(j)$ be the node in the jth position of $H_{s,i}$, then, from the definition of the probability of candidacy, we have

$$\begin{aligned}\sum_{v' \in \mathcal{C}(v)} \mathcal{P}_v(v') &= \sum_{i=0}^{k} \sum_{j \in \mathcal{L}_i(v)} \mathcal{P}_v(V_{H_{s,i}}(j)) \\ &= k \sum_{j \in \mathcal{L}_0(v)} \mathcal{P}_v(V_{H_{s,0}}(j)) \\ &= 1.\end{aligned}$$

Therefore,

$$\sum_{j\in\mathcal{L}_0(v)} \mathcal{P}_v(V_{H_{s,0}}(j)) = \frac{1}{k}.$$

Meanwhile,

$$\begin{aligned}\sum_{v'\in\mathcal{C}(v)} \mathcal{P}_v(v')\omega_H(v') &= \sum_{i=0}^{k-1}\sum_{j\in\mathcal{L}_i(v)} \mathcal{P}_v(V_{H_{s,i}}(j))\omega_H(V_{H_{s,i}}(j)) \\ &= \sum_{j\in\mathcal{L}_0(v)}\sum_{i=0}^{k-1} \mathcal{P}_v(V_{H_{s,i}}(j))\omega_H(V_{H_{s,i}}(j)) \\ &= \sum_{j\in\mathcal{L}_0(v)} \mathcal{P}_v(V_{H_{s,0}}(j))\sum_{i=0}^{k-1}\omega_H(V_{H_{s,i}}(j)) \\ &\le \sum_{j\in\mathcal{L}_0(v)} \mathcal{P}_v(V_{H_{s,0}}(j)) \\ &= \frac{1}{k}.\end{aligned}$$

Note the inequality holds because following Theorem 1, for jth position in all the k subgraphs, there are at most 1 node with non-zero weight, i.e. $\sum_{i=0}^{k-1}\omega_H(V_{H_{s,i}}(j)) \le 1$.

Therefore, v is a k-secure cell. Because the property holds for all the nodes with non-zero weights, $\langle G, H\rangle$ must be k-secure. Hence proved. $\square$

Meng Li (S'15) received his B.S. degree in Microelectronics from Peking University, Beijing, China in 2013. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering, the University of Texas at Austin, under the supervision of Prof. David Z. Pan. His research interests include hardware-oriented security, reliability, power grid simulation acceleration and deep learning.

Meng Li received the first place in the grand final of ACM student research competition in 2018, best poster (Presentation) award in ASPDAC Ph.D. forum in 2018, gold medal in ACM ICCAD student research competition in 2017, and university graduate fellowship from UT Austin in 2013. He also received the best paper award in HOST'17 and best paper candidate in GLSVLSI'18.

Bei Yu (S'11–M'14) 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 in the editorial boards of Integration, the VLSI Journal and IET Cyber-Physical Systems: Theory & Applications. He received five Best Paper Awards from Integration, the VLSI Journal in 2018, International Symposium on Physical Design 2017, SPIE Advanced Lithography Conference 2016, International Conference on Computer Aided Design 2013, and Asia and South Pacific Design Automation Conference 2012, and four ICCAD/ISPD contest awards.

Yibo Lin (S'16–M'18) received the B.S. degree in microelectronics from Shanghai Jiaotong University, Shanghai, China, in 2013. He is currently pursuing the Ph.D. degree at the Department of Electrical and Computer Engineering, University of Texas at Austin. His research interests include physical design and design for manufacturability. He has received Franco Cerrina Memorial Best Student Paper Award at SPIE Advanced Lithography Conference 2016, University Graduate Continuing Fellowship in 2017, and National Scholarship at Shanghai Jiaotong University in 2012. He has interned at Toshiba, IMEC, Cadence, and Oracle.

Xiaoqing Xu (S'15-M'17) received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2012 and the M.S.E. and Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, in 2015 and 2017, respectively. He is now with ARM Research, Austin, TX, as a Senior Research Engineer. His research interests include robust standard cell design, design for manufacturability and physical design. His research has been recognized with numerous awards including Golden Medal at ACM Student Research Competition at ICCAD 2016, University Graduate Continuing Fellowship in 2016, SPIE BACUS Fellowship in 2016, Best in Session Award at SRC TECHCON 2015, William J. McCalla Best Paper Award at ICCAD 2013 and CAD Contest Award at ICCAD 2013.

Wuxi Li received the B.S. degree in microelectronics from Shanghai Jiao Tong University, Shanghai, China, in 2013. He is currently pursuing the Ph.D. degree at the Department of Electrical and Computer Engineering, University of Texas at Austin. His research interests include physical design automation for FPGAs. He was a recipient of the 1st-place awards in the FPGA placement contests of ISPD 2016 and 2017.

David Z. Pan (S'97–M'00-SM'06-F'14) received his BS degree from Peking University and MS/PhD degrees from UCLA. He is currently Engineering Foundation Professor at the Department of Electrical and Computer Engineering, The University of Texas at Austin. His research interests include cross-layer IC design for manufacturing, reliability, security, machine learning in EDA, design/CAD for analog/mixed signal designs and emerging technologies. He has published over 300 refereed journal/conference papers and 8 US patents. He has served in many journal editorial boards and conference committees, including various leadership roles. He has received many awards, including SRC Technical Excellence Award, 16 Best Paper Awards, DAC Top 10 Author Award in Fifth Decade, ASP-DAC Frequently Cited Author Award, Communications of ACM Research Highlights, ACM/SIGDA Outstanding New Faculty Award, NSF CAREER Award, IBM Faculty Award (4 times), UCLA Engineering Distinguished Young Alumnus Award, UT Austin RAISE Faculty Excellence Award, etc. He is a Fellow of IEEE and SPIE.