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

Meng Li[1], Bei Yu[2], Yibo Lin[1], Xiaoqing Xu[1], Wuxi Li[1], and David Z. Pan[1]

[1]ECE Department, University of Texas at Austin, Austin, TX, USA
[2]CSE Department, The Chinese University of Hong Kong, NT, Hong Kong

Abstract

Trojans and backdoors inserted by untrusted foundries have become serious threats to hardware security. Split manufacturing is proposed to prevent Trojan insertion proactively. Existing methods depend on wire lifting to hide partial circuit interconnections, which usually suffer from large overhead and lack of security guarantee. In this paper, we propose a novel split manufacturing framework that not only guarantees to achieve the required security level but also allows for a drastic reduction of the introduced overhead. In our framework, insertion of dummy circuit cells and wires is considered simultaneously with wire lifting. To support cell and wire insertion, we propose a new security criterion, and further derive its sufficient condition to avoid computation intensive operations in traditional methods. Then, for the first time, a novel mixed integer linear programming formulation is proposed to simultaneously consider cell and wire insertion together with wire lifting, which significantly enlarges the design space to guarantee the realization of the sufficient condition under the security requirements and overhead constraints. With extensive experimental results, our framework demonstrates much better efficiency, overhead reduction, and security guarantee compared with existing methods.

I Introduction

With the globalization of integrated circuit (IC) supply chains, 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, which result in economic losses on 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 becomes a very critical issue.

Design-for-security techniques, including IC camouflaging [7–9] and split manufacturing [10–17], are proposed to prevent the security vulnerabilities proactively. Split manufacturing directly targets at preventing the Trojan insertion from untrusted foundries. In the split manufacturing process, the circuit layout is split into front-end-of-line (FEOL) layers, which consist of all the transistors 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 BEOL layer fabrication, by which important circuit information can be hidden to prevent Trojan insertions by untrusted foundries.

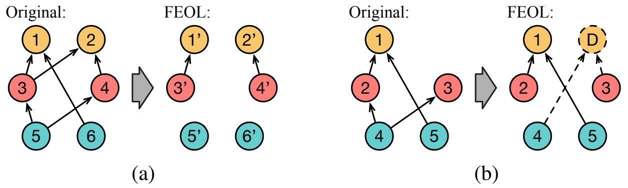


Fig. 1. (a) Realize 2-security by lifting wires to BEOL layers: both nodes $1'$ and $2'$ may implement 1 and are indistinguishable; (b) 2-security cannot be realized simply with wire lifting (different colors represent different cell types).

In recent papers [16–19], different split manufacturing frameworks have been proposed. The first formal security criterion for split manufacturing, named as k-security, is proposed in [16]. A circuit is defined to be k-secure if for each cell in the netlist, there exist k cells in FEOL layers, all of which may be its actual physical implementation and are indistinguishable for the attackers. Fig. 1(a) gives an example of a 2-secure circuit since each cell in original netlist may be implemented by 2 cells in BEOL layer. For instance, cell 1 can be implemented by cells $1'$ and $2'$. The security definition is formalized based on graph isomorphism [20], as will be discussed later. To realize k-security, a greedy algorithm is also proposed to determine the wires to be lifted to BEOL layers [16]. In [17–19], 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, for which Trojan prevention is not the target.

Despite the extensive researches on split manufacturing, there are three fundamental problems that have not been properly solved. Firstly, the required security level, i.e. k-security, is not guaranteed to achieve with existing split manufacturing strategy [16]. Consider the original netlist shown in Fig. 1(b), because cell 1 has different types compared with all other cells, it can always be identified. Therefore, 2-security can never be achieved by lifting wires to BEOL layers. Secondly, all the state-of-the-art methods suffer from scalability issue. As shown in [16], to determine the wires to be lifted, the isomorphism between large graphs needs to check repetitively. Though it can be formulated as a satisfiability (SAT) problem, the computation cost makes the method intractable quickly even for small circuits. Thirdly, existing methods also suffer from large overhead. According to [16], more than 2× wirelength overhead is introduced just to realize 2-security for a small benchmark circuit. To make split manufacturing practical, the above problems on security guarantee, efficiency and performance overhead must be solved.

In this paper, we propose a novel split manufacturing framework to address the security and practicality issues of existing methods. We observe that inserting dummy cells and wires in FEOL layers can enhance security and reduce overhead effectively. For example, consider the netlist in Fig. 1(b), by inserting dummy cell D and wires $(4, D), (3, D)$, 2-security can be realized. However, insertion of dummy nodes and wires can lead to situations when FEOL layers contain cells and wires that do not exist in original netlist, which cannot be modeled with existing framework [16]. We propose a new security criterion that is fully compatible with node and wire insertion and further derive its sufficient condition to enable an efficient realization. Our security criterion can also balance the trade-off between security and overhead by allowing the flexibility of protecting arbitrary subset of circuit nodes. A framework is further proposed to realize the security criterion while minimizing the introduced overhead. Our framework consists of a novel mixed integer linear programming (MILP) based formulation to enable simultaneous wire lifting and cell insertion, and a layout refinement technique to guarantee security in physical synthesis stage. We summarize our contributions as follows:

- A new security criterion that is fully compatible with cell and wire insertion is proposed.
- A sufficient condition to guarantee security is derived to enable more efficient split manufacturing process.
- For the first time, an MILP-based framework is proposed to simultaneously consider dummy cell and wire insertion with wire lifting.
- The proposed flow is evaluated with extensive experimental results and demonstrates good efficiency and practicality.

The rest of the paper is organized as follows. Section II defines the split manufacturing problem and our new security criterion. Section III proposes a sufficient condition to achieve the proposed criterion. Section IV describes our split manufacturing framework. Section V demonstrates the performance of the framework, followed by conclusion in Section VI.

II Problem Formulation

A Attack Model of Untrusted Foundries

In this paper, 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 [16]: 1) the attacker has the gate-level netlist of the design; 2) the attacker has full knowledge of the FEOL layers, including the cells and wires in lower metal layers as well as their physical information; 3) the attacker knows the algorithms for FEOL generation but does not know the specific mapping between the cells in FEOL and 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 [16]. Meanwhile, the profit of a successful Trojan insertion can also be pretty large, especially for military applications [21], Given the gate-level netlist, the attackers can determine the target nodes in the design and try to identify its physical implementation in FEOL layers for Trojan insertion. To protect the circuit nodes from being identified by the attackers, in the following sections, we will propose our split manufacturing framework.

B Split Manufacturing Security Analysis

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 $\ell : V \to [t]$ maps each vertex to a cell type, where $[t] = \{1, \ldots, t\}$ denotes the set of all possible cell types in the circuit. $\omega : V \to \{0, 1\}$ 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 balance the trade-off between security and introduced overhead.

The original netlist and the FEOL layers can be represented as two graphs, denoted as G and H. For the original circuit, 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 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 operation 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 $\ell_{u'}$ 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. We only allow inserting wires pointing to the dummy nodes to guarantee the circuit functionality is not changed. Based on the description of the allowed operations, V_H, E_H, ℓ_H and ω_H can be acquired accordingly.

Take an example of G and H in Fig. 2. In G, we have 1 and 2 with the same functionality, i.e. $\ell_G(1) = \ell_G(2)$. Assume that we select node 1 and 5 for protection, then, $\omega_G(1) = \omega_G(5) = 1$. To generate H, we first add all the correspond 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 $\omega_H(1') = \omega_H(5') = 1$ and the other nodes have zero weight.

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 [16] leverages the concept of graph isomorphism. Two graphs G_1 and G_2 are isomorphic if there exists a bijective mapping $f : V_1 \to V_2$ such that $(u, v) \in E_1$ if and only if $(f(u), f(v)) \in E_2$ and $\ell_1(u) = \ell_2(f(u))$, $\ell_1(v) = \ell_2(f(v))$. Because only wire lifting is considered in existing methods, we can also find a subgraph of G that is isomorphic to H. However, *when the insertion of dummy wires and cells are considered, the*

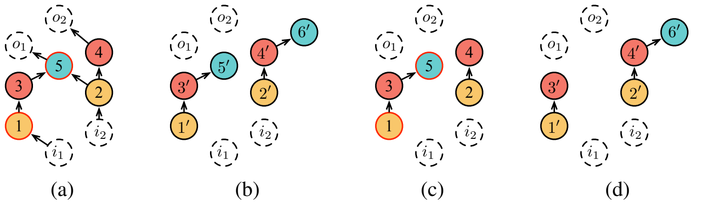


Fig. 2. Example of (a) Original graph G (i_1 and i_2 are input pins while o_1 and o_2 are output pins), (b) FEOL graph H, (c) Spanning subgraph G_s of (a), and induced subgraph H_s of H.

original isomorphic relation is not satisfied. This is because H contains nodes and edges that do not present in G. We observe the following relations for G and H 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, $\forall (v', u') \in E_H, \exists v \in V_G$ s.t. $v' = \phi(v)$ and $(v, u) \in E_G$.

The second relation holds because we cannot add dummy edges pointing to the corresponding node of $u \in V_G$. For example, in Fig. 2, suppose $5' = \phi(5)$, since we are not allowed to add any dummy edges pointing to $5'$, we must have $3' = \phi(3)$ and $(3, 5) \in E_G$. To formalize the relations described above, we leverage the concept of spanning subgraph [22] and induced subgraph [22].

Definition 1 (Spanning Subgraph)**.** *A subgraph G_s of G is referred to as a spanning subgraph if $V_{G_s} = V_G$.*

Definition 2 (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 $u, v \in V_{G_s}$.*

For example in Fig. 2(a), G_s is a spanning subgraph of G in Fig. 2(a) since $V_{G_s} = V_G$. H_s in Fig. 2(b) is an induced subgraph of H in Fig. 2(b) since 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, node $1'$ and $3'$ exist in H_s. Because $(1', 3') \in E_H$, for H_s to be an induced subgraph, we must have $(1', 3') \in E_{H_s}$. Consider the spanning subgraph of G and the induced subgraph of H, we define the relation of spanning subgraph isomorphism as below.

Definition 3 (Spanning Subgraph Isomorphism)**.** *Given two graphs G_1 and G_2, we say that G_1 is spanning subgraph isomorphic to G_2 if there exists a spanning subgraph of G_1 that is isomorphic to an induced subgraph of G_2.*

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. In Fig. 2(a), 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. Due to the proposed isomorphic relation, $2'$ is possible to implement node 1 in the final layout for the attacker. We denote node $2'$ as the candidate node for node 1.

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 when H and G have different number of vertices and when H contains edges that do not correspond to any edges in G. This enables us to consider cell and wire insertion in split manufacturing process.*

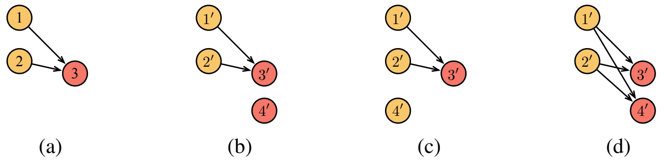


Fig. 3. Probability difference for different candidate nodes.

From the spanning subgraph isomorphism checkings between H and G, for $v \in V_G$, a set of candidate nodes can be identified in V_H, denoted as a candidate set. We observe that the nodes in the candidate set can differ from each other by their weight and their probability to be the actual corresponding node. The difference on weight is easy to understand according to the definition. The difference in probability can come from the randomness in the split manufacturing algorithm. For example in Fig. 3(a) and Fig. 3(b), if the probability for wire insertion is low in the algorithm, then, $3'$ is more likely to be the corresponding node for 3 since to form Fig. 3(b), only one dummy node needs to be added.

The difference can also come from the rule that inserting wires to the corresponding nodes is not allowed. For example in Fig. 3(c), $1', 2', 3'$ are all in the candidate set of 1. If $3' = \phi(3)$, $(1', 3')$ and $(2', 3')$ must exist in the original graph, which indicates $4'$ is dummy. We show an example in Fig. 3(d) that $3'$ and $4'$ are equally likely to be the corresponding node for 3.

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

Definition 4 (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 the probability for the attacker to pick a node with non-zero weight from its candidate set is no greater than $1/k$.*

In the definition, $\forall v \in V_G$, by enforcing the probability to pick a node with non-zero weight, both the difference on weight and the probability for nodes in the candidate set of v can be captured. Now we define the security criterion for split manufacturing.

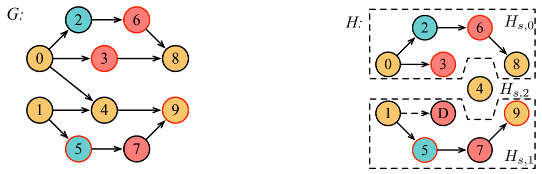
Definition 5 (k-Security)**.** *Given G and H, we say that $\langle G, H\rangle$ is k-secure if $\forall v \in V_G$ with $\omega_G(v) = 1$, v is k-secure with respect to G and H.*

By the above security criterion, we can guarantee that for any node that the attackers may target at, the probability of a successful Trojan insertion is always no greater than $1/k$. In this way, by making k large enough, we can guarantee much higher expense and risk of the Trojan insertion.

III k-Security Realization

To determine spanning subgraph isomorphism directly can be computation intensive due to graph isomorphism checkings. We adopt recent progress in privacy preserving network publishing [20] to derive a sufficient condition for k-security to avoid direct graph comparison. Our heuristic solution relies on the following concept denoted as k-isomorphism [20].

Definition 6 (k-Isomorphism [20])**.** *A graph is k-isomorphic if it consists of k disjoint isomorphic subgraphs.*



(a) (b)

Fig. 4. Example for Theorem 1: G is 2-secure with respect to H.

For example, the graph H of FEOL in Fig. 2(b) is 2-isomorphic with $V_{H_{s,1}} = \{1', 3', 5'\}$ and $V_{H_{s,2}} = \{2', 4', 6'\}$. Specifically, we call nodes $1'$ and $2'$ in the same position of $H_{s,1}$ and $H_{s,2}$. If node $1'$ is the candidate node for node 1, then, we find that $2'$ can also be a candidate node for 1. Moreover, their probability to be the actual corresponding node is also the same. Assume $1' = \phi(1)$, then, if $\omega(2') = 0$, 1 is 2-secure with respect to G and H in Fig. 2. 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.*

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.*

We skip the formal proof due to the limit of space and use the example in Fig. 4 to illustrate. In Fig. 4(b), 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, they are 2-secure with respect to G and H according to Lemma 1. Node 4 remains unprotected since its weight is zero. Therefore, G 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.

IV Practical Framework for Trojan Prevention

A MILP-based FEOL Generation

To generate$H = \{H_{s,0}, \ldots, H_{s,k}\}$ from G, because dummy wire and node insertion is allowed, one trivial solution of H is to copy G for $k-1$ times. *This indicates that k-security can always be achieved when insertion of dummy cells and wires is considered.*

TABLE I

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
r_m	$r_m = 1$ if the mth edge needs to be lifted to BEOL
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 $H_{s,0}, \ldots, H_{s,k-1}$
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 $H_{s,0}, \ldots, H_{s,k-1}$
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

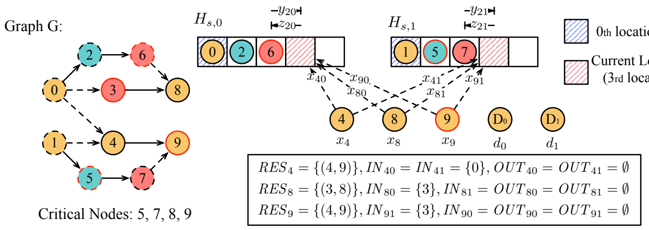

Graph G:
0
1
2
3
4
5
6
7
8
9
Critical Nodes: 5, 7, 8, 9
$H_{s,0}$
$H_{s,1}$
y_{20}
z_{20}
y_{21}
z_{21}
x_{40}
x_{90}
x_{80}
x_{41}
x_{81}
x_{91}
x_4
x_8
x_9
d_0
d_1
D0
D1
0th location
Current Location (3rd location)
$RES_4 = \{(4,9)\}, IN_{40} = IN_{41} = \{0\}, OUT_{40} = OUT_{41} = \emptyset$
$RES_8 = \{(3,8)\}, IN_{80} = \{3\}, IN_{81} = OUT_{80} = OUT_{81} = \emptyset$
$RES_9 = \{(4,9)\}, IN_{91} = \{3\}, IN_{90} = OUT_{90} = OUT_{91} = \emptyset$

Fig. 5. Example of parameters in the MILP formulation.

To reduce the introduced overhead, we propose a novel MILP-based framework. The inputs to the framework includes the original circuit netlist and the selected nodes for protection. Based on Theorem 1, we anonymize all the selected nodes by adding them to $H_{s,0}, \ldots, H_{s,k-1}$ iteratively. In each iteration, we select k nodes of the same label and add them to $H_{s,0}, \ldots, H_{s,k-1}$. We make sure that exactly one of k nodes has a non-zero weight to satisfy Theorem 1. We list the notations of the MILP formulation in TABLE I and use the following example to illustrate the problem.

Example 1. *Consider the original graph G in Fig. 5. Assume nodes $0, 2, 6$ and nodes $1, 5, 7$ are already added to $H_{s,0}$ and $H_{s,1}$ respectively in the first three iterations. All the dotted lines in the figure denote the edges that need to be lifted if only these nodes are added. To choose a pair of nodes of same label with minimum cost to insert into $H_{s,0}$ and $H_{s,1}$, we calculate the insertion cost for the remaining nodes with non-zero weight, i.e. nodes 3 and 9, by solving the MILP for each of them and picking the one with the smallest cost. For node 9, we get nodes 8 and 4 of the same cell type and also consider insertion of dummy node d_0 and d_1 to help anonymization. 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, we can determine IN and OUT for each node. 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 problem as shown in Formula (1). The objective in Formula (1) consists of the number of edges to be removed, the number of edges that can be added back and the area of the inserted dummy nodes, where A denotes the cell area. α, β and γ are user-defined parameters to balance the cell insertion with wire lifting. Constraint (1a) guarantees that one node can at most be inserted into one of $H_{s,0}, \ldots, H_{s,k-1}$. Constraint (1b) requires exactly one of the inserted nodes has weight 1. Constraints (1c) and (1d) indicate the condition when one edge can be added back to all subgraphs. The reason that d_j only exists in constraint (1c) is that we only allow inserting wires pointing to dummy nodes. To guarantee isomorphism for all the subgraphs, y_l equals to 1 only when the edges can be added in each subgraph, i.e. $y_{lj} = 1, \forall j \in \{0, \ldots, k-1\}$. Constraint (1e) indicates the condition when one edge needs to be lifted. While all the variables should be integers, by constraining x_{ij} and d_j to be integers, we find relaxing the rest variables to be continuous does not change the optimal solution, and also achieves much better efficiency.

$$\min_{x,d} \; \alpha \sum_l r_l - \beta k \sum_l (y_l + z_l) + \gamma A \sum_j d_j \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{k} x_{ij} = x_i, \qquad \forall i; \tag{1a}$$

$$\sum_i x_i w_i = 1; \tag{1b}$$

$$y_l \le y_{lj}, \qquad y_{lj} \le \sum_{i,l \in IN_{ij}} x_{ij} + d_j, \qquad \forall j, l; \tag{1c}$$

$$z_l \le z_{lj}, \qquad z_{lj} \le \sum_{i,l \in OUT_{ij}} x_{ij}, \qquad \forall j, l; \tag{1d}$$

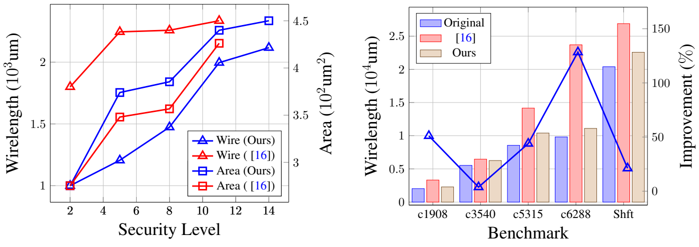
$$r_l \ge x_i, \qquad \text{if } r_l \in RES_i, \qquad \forall i, l. \tag{1e}$$

Besides the constraints above, we also set a hard constraint for the number of lifted wires. This is because the total number of vias through a certain layer is usually fixed, which limits the number of wires to be lifted. In the iterative process of generating FEOL, when we find such constraint is violated, we will increase the cost of wire lifting in the MILP formulation by increasing α and β, and run the iterative process again.

B *k-Secure Layout Refinement*

After the first two steps, cells and connections in FEOL layers H are determined such that $\langle G, H \rangle$ is k-secure. To guarantee security while reduce overhead in the physical synthesis stage, we propose a k-secure layout refinement technique.

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. This makes it possible for the attackers to recover the connections in BEOL layers based on the physical proximity information [17]. Existing method [16] chooses to ignore the lifted wires in placement stage to guarantee security but suffers from large overhead. This is because many cells are left floating in FEOL layers after wire lifting, which makes the distance between the cells that are connected in BEOL layers highly unoptimized. We propose to insert virtual nets to connect a node v that is selected for protection and its candidate nodes with all the adjacent nodes of v. The inserted virtual nets not only guarantee security, but also make sure a node is still placed close to its neighbors. As we will show in Section V, the layout refinement technique allows for 49.6% overhead reduction compared with [16].



(a)

(b)

Fig. 6. Comparison with [16]: (a) overall framework (dotted line indicates unrealizable); (b) layout refinement method.

V Experimental Results

A *Experimental Setup*

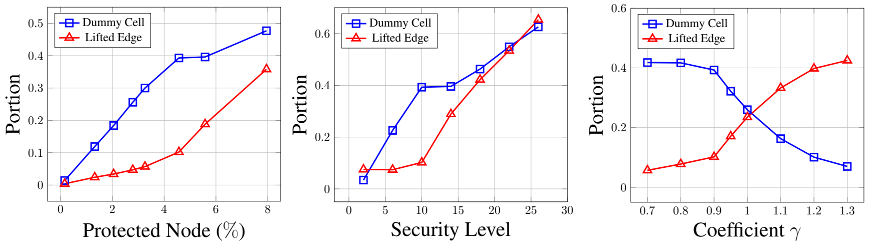
In this section, we report on our experiments to demonstrate the effectiveness of the proposed split manufacturing framework. The input to our framework is a gate-level netlist and the nodes to protect. To select the nodes for protection, we follow the Trojan insertion methods used by TrustHub [23]. 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. The benchmarks consist of six test cases from ISCAS benchmarks [24] and two functional units (`Shifter` and `Multiplier`) from OpenSPARC T1 processor. FEOL layers consist of all the cells and lower metal layers up to metal 3, and BEOL layers consist of metal 4 and above. We implement our framework in C++ and conduct physical synthesis using Cadence Encounter [25]. We run the experiments on an eight-core 3.40 GHz Linux server with 32 GB RAM.

B *Comparison with State-of-the-Art*

We first compare the proposed framework with [16] that only considers wire lifting. The original program is modified for our security criterion. We compare the change of overhead with the increase of the required security level. Small benchmark `c432` with 20% selected nodes is chosen due to runtime limit of the previous algorithm. As we show in Fig. 6(a), 12-security cannot be realized with the previous method, while our method guarantees to achieve all security levels. Meanwhile, to achieve 10-security, our methods can achieve on average 59.1% wirelength reduction with less than 4% area increase. Then, as in TABLE II, we compare the efficiency of the two frameworks. To achieve 10-security, our methods achieve on average more than 290× speedup in small circuits. For large circuits, the previous method cannot finish within 10^5s while our framework can finish within 300s. The performance of our framework can be further boosted by circuit graph partitioning. We leverage widely adopted partition algorithm Metis [26] to minimize the wires across different partitions. For benchmark `Shifter`, by partitioning the circuit graph into 2 subgraphs and applying our framework to each subgraph, the runtime can be

TABLE II
Comparison on Runtime

Bench	# Protect	# Nodes	[16] (s)	Ours (s)
c432	23	214	140.8	0.5
c880	19	355	979.6	3.2
c1908	24	519	>100000	8.1
c3540	49	1012	>100000	37.0
c5315	73	1864	>100000	135.0
c6288	90	2568	>100000	297.9
Shifter	84	2579	>100000	273.9



(a) (b) (c)

Fig. 7. The relation between overhead and (a) portion of protected nodes, (b) security level and (c) MILP coefficients.

reduced to 162.6s. For larger benchmark Multiplier with 23855 nodes and 695 nodes to protect, by partitioning the circuit graph into 10 subgraphs, our framework can finish within 300s.

We then compare our placement refinement method based on virtual net insertion with the proposed placement method in [16]. Large ISCAS benchmarks and Shifter unit are used to provide a more practical comparison. As shown in Fig. 6(b), our placement refinement method provides on average 49.6% wirelength improvement compared with [16]. Especially for benchmark c6288, since a large number of cells are floating in FEOL layers, our methods achieve more than 130% overhead reduction. We also compare the selected nodes and their candidate nodes on the physical proximity to their neighbors. The average of the distance difference is less than $1/40$ of the standard deviation, which leaves negligible probability for the attacker to recover the original connection based on proximity information.

C 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. III(a), to achieve 10-security, we show the increase of the overhead with the increase of the protected nodes. In Fig. III(b), we show the relation between overhead and the required security level in order to protect 5% of nodes. In Fig. III(c), we fix $\alpha = 0.5$, $\beta = 1$ in the MILP formulation and change γ from 0.7 to 1.3. By changing γ, cell insertion and wire lifting are balanced to help provide better usage of the routing resources and chip space for different designs.

VI 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 MDTC*, vol. 27, no. 1, pp. 10–25, 2010.

[2] S. Bhasin and F. Regazzoni, “A survey on hardware trojan detection techniques,” in *Proc. ISCAS*, 2015.

[3] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, “A2: Analog malicious hardware,” in *Proc. SP*, 2016.

[4] C. Krieg, C. Wolf, and A. Jantsch, “Malicious LUT: a stealthy FPGA trojan injected and triggered by the design flow,” in *Proc. ICCAD*, 2016.

[5] R. Torrance and D. James, “The state-of-the-art in semiconductor reverse engineering,” in *Proc. 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 *Proc. HOST*, 2017.

[7] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, “Security analysis of integrated circuit camouflaging,” in *Proc. CCS*, 2013.

[8] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, “Provably secure camouflaging strategy for IC protection,” in *Proc. ICCAD*, 2016.

[9] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, “Cyclic obfuscation for creating sat-unresolvable circuits,” in *Proc. GLSVLSI*, 2017.

[10] Y. Xie, C. Bao, and A. Srivastava, “Security-aware design flow for 2.5D IC technology,” in *Proc. TrustED*, 2015.

[11] K. Vaidyanathan, R. Liu, E. Sumbul, Q. Zhu, F. Franchetti, and L. Pileggi, “Efficient and secure intellectual property (IP) design with split fabrication,” in *Proc. HOST*, 2014.

[12] B. Hill, R. Karmazin, C. T. O. Otero, J. Tse, and R. Manohar, “A split-foundry asynchronous FPGA,” in *Proc. CICC*, 2013.

[13] 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 TCAD*, vol. 32, no. 4, pp. 611–615, 2013.

[14] K. Vaidyanathan, B. P. Das, and L. Pileggi, “Detecting reliability attacks during split fabrication using test-only BEOL stack,” in *Proc. DAC*, 2014.

[15] K. Xiao, D. Forte, and M. M. Tehranipoor, “Efficient and secure split manufacturing via obfuscated built-in self-authentication,” in *Proc. HOST*, 2015.

[16] 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 *Proc. USENIX Security Symposium*, 2013.

[17] J. Rajendran, O. Sinanoglu, and R. Karri, “Is split manufacturing secure?” in *Proc. DATE*, 2013.

[18] Y. Wang, P. Chen, J. Hu, and J. Rajendran, “The cat and mouse in split manufacturing,” in *Proc. DAC*, 2016.

[19] J. Magaña, D. Shi, and A. Davoodi, “Are proximity attacks a threat to the security of split manufacturing of integrated circuits?” in *Proc. ICCAD*, 2016.

[20] J. Cheng, A. W.-C. Fu, and J. Liu, “K-isomorphism: privacy preserving network publication against structural attacks,” in *Proc. SIGMOD*, 2010.

[21] S. Skorobogatov and C. Woods, “Breakthrough silicon scanning discovers backdoor in military chip,” in *Proc. CHES*, 2012.

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

[23] H. Salmani, M. Tehranipoor, and R. Karri, “On design vulnerability analysis and trust benchmarks development,” in *Proc. ICCD*, 2013.

[24] F. Brglez, D. Bryan, and K. Koźmiński, “Combinational profiles of sequential benchmark circuits,” in *Proc. ISCAS*, 1989.

[25] “Cadence SOC Encounter,” http://www.cadence.com.

[26] G. Karypis and V. Kumar, “Analysis of multilevel graph partitioning,” in *Proc. Supercomputing*. ACM, 1995.