This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2877012, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Obfuscated Built-In Self-Authentication with Secure and Efficient Wire-Lifting

Qihang Shi, Mark M. Tehranipoor, and Domenic Forte

ECE Department, University of Florida, qihang.shi@ufl.edu,{tehranipoor, dforte}@ece.ufl.edu

***Abstract*— Hardware Trojan insertion and intellectual property (IP) theft are two major concerns when dealing with untrusted foundries. Most existing mitigation techniques are limited in protecting against both vulnerabilities. Split manufacturing is designed to stop IP piracy and IC cloning, but it fails at preventing untargeted hardware Trojan insertion and incurs significant overheads when high level of security is demanded. Built-in self-authentication (BISA) is a low cost technique for preventing and detecting hardware Trojan insertion, but is vulnerable to IP piracy, IC cloning or redesign attacks, especially on original circuitry. In this paper, we propose an obfuscated built-in self-authentication (OBISA) technique that combines and optimizes both techniques so that they complement and improve security against both vulnerabilities, while at the same time minimizing design overheads to the extent that the proposed method does not incur prohibitive cost for designs of industrial-level sophistication. Our evaluation on AES and DES cores shows that the proposed technique can reach security levels more than two times higher, satisfy all existing layout-based security metrics, while reducing overheads from hundreds of percents to less than 13% in power, less than 5% in delay, and zero percent in area, as compared to best reported performance in existing techniques.**

I. Introduction

Changing economic trends have resulted in a globalized integrated circuit (IC) supply chain. It is no longer economically feasible for most IC producers to own foundries and fabricate ICs in-house. For the majority of the industry, fabrication is now being performed by contracted foundries and outside the control of original intellectual property (IP) owners. IP owners enjoy reduced cost and state-of-art fabrication technologies in off-shore fabrication, at the cost of reduced control and therefore reduced trust in the manufacturing process. This has raised serious concerns on whether trust between an IP owner and such fabs can be established [1]. An untrusted foundry with malicious intent could conduct a number of attacks including IP piracy [2], IC cloning and overproduction [3], [4], and hardware Trojan insertion [5]. For off-shore fabrication to stay secure, capability of the IP owner to prevent such attacks must be thoroughly substantiated.

Split manufacturing has been proposed [6] to address the threat of IP piracy, cloning, and overproduction. This technique proposes that an untrusted foundry manufactures the front-end-of-line (FEOL) part of the IC, and then ships it to a trusted foundry to deposit back-end-of-line (BEOL) part onto it (see example in Figure 1). By this arrangement, the untrusted foundry is denied complete information of the layout, and therefore prevented from stealing IP information, or committing attacks that require knowledge of the complete design.

Techniques against hardware Trojan insertion exist in two categories characterized by how they address the issue: The first category focuses on detecting Trojans, either by functional verification, side-channel signal analysis, or by new front-end design techniques such as design-for-trust [7]–[15]. Techniques in this category detect existence of hardware Trojans by generating a signature of the circuit under test (CUT), then classifying the CUT with this signature. To perform classification, they require a golden model, i.e. signature of a copy of the same circuit that is known to be free from hardware Trojans. Unfortunately, it remains doubtful whether golden models can be acquired for real world applications. The second category, Trojan prevention techniques focuses on preventing hardware Trojans from being inserted into a design, and do not have to deal with process variation and need for golden ICs. Built-in self-authentication (BISA) is the first proposed technique to prevent hardware Trojan insertion in circuit layout and mask [16], [17]. By occupying all available spaces for Trojan insertion and detecting malicious removal through built-in self test, BISA is able to deter hardware Trojan insertion without the requirement of golden models and free from classification errors introduced by process variation. Both split manufacturing and BISA are effective in addressing the threat they are designed to counter. However, problems remain with both techniques. To begin with, techniques against IP piracy do not usually consider the threat of hardware Trojan insertion, and neither do techniques against hardware Trojan insertion (such as BISA) consider IP theft, despite both attacks sharing the same adversary. In all likelihood, untrusted foundries will likely try all attacks in their arsenal, and IP owners will desire an overall solution that will secure his design against all of them. Therefore, a complete security solution to address the threat of untrusted foundry needs to consider all possible attacks, and both split manufacturing and BISA are limited on their own.

In this paper, we propose a new approach to an earlier proposed technique called Obfuscated Built-In Self-Authentication (OBISA) that combines both techniques [18]. The proposed

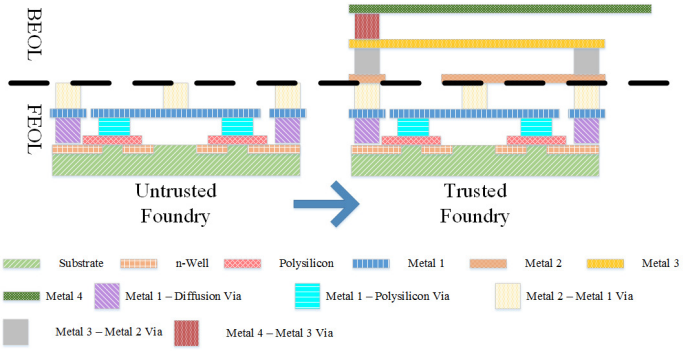


Fig. 1: A sample split manufacturing arrangement. It assumes split is made between Metal-2 and Metal-1 layers.

* This work is supported by Cisco Systems, Inc. and under NSF under grant CNS 1651701.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2877012, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

technique not only

- prevents weakness from either kind of attacks,
- is secure against attacks specific to BISA,
- is more secure in terms of proposed metrics of split manufacturing security,

but also

- has drastically reduced time complexity at generating wire lifting solutions,
- has drastically reduced design overheads of the implemented design,
- provides partitioning techniques to accommodate large designs,

so that the presented technique can be expected to be implemented on industrial-size designs, while keeping overheads in both design process and design itself manageable.

The rest of the paper is organized as follows. Section II provides a survey of existing research related to split manufacturing security and BISA, elaborates on weaknesses of known techniques and proposes ways to address these weaknesses using OBISA, before providing a detailed list of contributions claimed by the OBISA technique. Section III presents the proposed OBISA technique in terms of its application flow, how it integrates with existing back-end design flow, and how each claimed contribution is realized. Section IV presents experimental evaluation of the proposed OBISA technique in terms of its security and overheads. Finally, Section V concludes this paper.

II. Background

A. Threat model

The proposed technique is intended to address threats posed by an untrusted foundry against split manufacturing. These threats include malicious inclusion (e.g., hardware Trojans), as well as all possible attacks if the untrusted foundry succeeds in compromises security of split manufacturing by reverse engineering BEOL connections denied to him. In other words, the proposed technique is intended to be secure against Trojan insertion and as a split manufacturing technique. Since the proposed technique intends to achieve this goal by combining BISA [16], [17] with wire lifting [19], it also inherits assumptions of both techniques, e.g. the untrusted foundry is assumed to be able to access functional netlist as was assumed in [19][1]. Note that this quality of remaining secure even when the netlist becomes compromised does not make the technique insecure when the netlist becomes the goal of the attack.

B. Built-In Self-Authentication (BISA)

Built-In Self-Authentication (BISA) prevents hardware Trojan insertion by exhausting one resource essential to it: *white spaces*. Normally, during the placement step of the back-end design of the circuit, gates in the circuit are placed at optimized locations based on density and routability [20]. This leaves spaces in the layout that are not filled with standard cells. If they are replaced by Trojan gates, no performance loss significant enough to raise suspicion will likely be incurred, since Trojan gates are rarely triggered as well.

1) BISA architecture: All inserted BISA cells are connected into tree-like structures to form a built-in self test (BIST) circuitry, so that they could be tested to verify no BISA cell has been removed. Removal of its member cells will lead to a BIST failure, so that no attempt to make room for hardware Trojans will evade detection. As shown in Figure 2, BISA consists of three parts: the BISA circuit under test, the test pattern generator (TPG), and the output response analyzer (ORA). In order to increase its stuck-at fault test coverage, the BISA circuit is divided into a number of smaller combinational logic blocks, called BISA blocks shown in Figure 2. The TPG generates test vectors that are shared by all BISA blocks. The ORA will process the outputs of all BISA blocks and generate a signature. TPG has been implemented with Linear Feedback Shift Register (LFSR) while ORA has been implemented with Multiple Input Signature Register (MISR) in prior work [17].

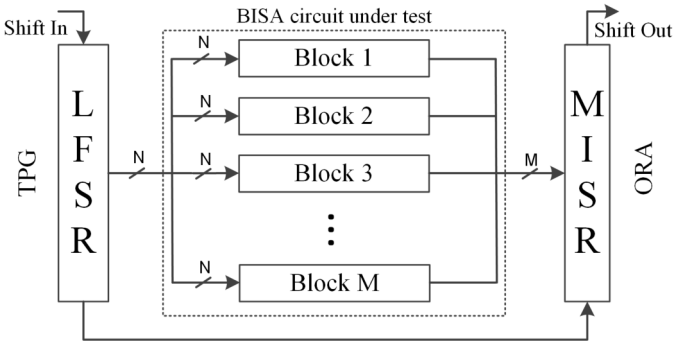


Fig. 2: Structure of BISA

The main advantage of BISA over other techniques with similar objectives is that it has no golden chip requirement. Since BISA relies on logic testing, process variation is not a factor either, as compared to Trojan detection techniques based on side-channel analysis. As an additional advantage, impact of BISA on the original design in terms of area and power is also negligible [16], [17]. This is due to the fact that BISA only occupy spaces originally occupied by decoupling filler cells, and do not become activated through out life of the IC except once after fabrication by the IP owner to verify the IC is free from malicious insertion by the untrusted foundry.

2) Attacks on BISA: The attack most likely to succeed against BISA is the so-called redesign attack. This attack replaces original circuitry with smaller functionally equivalent circuitry to make room for Trojan insertion. Prevention of such an attack would require anticipation of all possible custom cell designs that are functionally equivalent to any combination of BISA cells. That is not likely feasible except for very small BISA circuitry. Due to the existence of resizing attack, all BISA cells have to be of the smallest variant in area among standard cells of the same function, which might make it easier for the attacker to identify them.

C. Split Manufacturing Security and Limitations

1) Prior works on split manufacturing security: The prior work in this field [19], [21], [22] is motivated by one major objective: to establish a sound metric of security for designs fabricated using various split manufacturing methods. Most researchers attack the problem from a layout point of view. Publications in this category often examine irregularities in the layout and theorize how they can be used by a hypothetical attacker. Authors in [21] proposed *proximity attack*, which simulates an attacker who makes educated guesses on BEOL connections of open input/output pins in FEOL. This idea

[1]This however does NOT assume the untrusted foundry to also have access to BISA circuitry, as latter is only known well into the layout design, therefore knowledge of BISA netlist would also mean knowledge of the layout.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2877012, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

of proximity attack has received further development in later publications. [23] proposed to also consider load capacitance limitations as well as the direction of dangling wires, while [24] discussed more definitions of proximity based on known router behaviors. Another study [22] also uses layout information, but instead of performing hypothetical attacks, it seeks to define objective measures of the layout that might become useful to exploit. This leads to the following metrics being proposed:

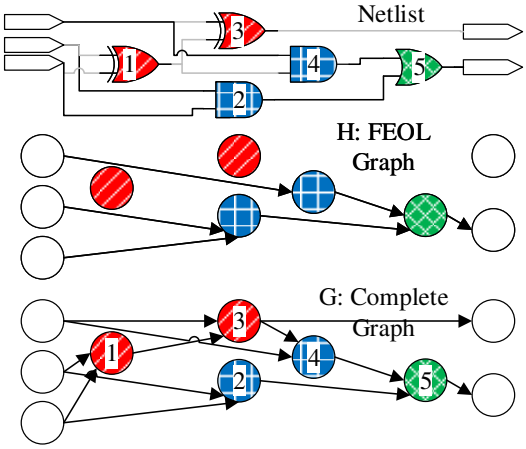
1) *Neighborhood Connectedness (NC)* is defined as function of count of connections per neighbor cell within range R, i.e. $C(R) = \frac{\sum \text{connections to neighbors in } R}{\sum \text{neighbors in } R}$;
2) *Standard-cell composition bias* is designed to reflect imbalance in number of instantiations of specific cells considered representative of design function[2];
3) *Cell-level obfuscation* measures percentage of standard cells that has functionality hidden with obfuscation;
4) *Entropy* measures disorder in FEOL information, and is given by $E = -\sum_{i=1}^{N} p_i \log(p_i)$, where N is total number of cell types and p_i is the percentage of each cell type among all cells; and
5) Overheads in area and delay are used to measure the cost of the obfuscation technique.

Security metrics based on layout information are available to designers through layout editor tools, and have seen recent application [25], [26]. Unfortunately, problems remain. First, since no attack has been reported to have successfully reconstructed BEOL connection from FEOL clues, no hypothetical attacks and objective metrics is more convincing than the other. Further, proposed metrics themselves don't scale linearly with difficulty in any conceivable attack, which makes it difficult for designers to estimate how much protection is enough. And finally, when multiple metric values are measured, it is very difficult to find a way to estimate the relative importance among them.

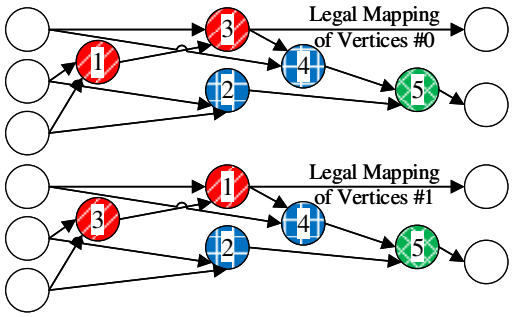
Another research [19] evaluates split manufacturing security based on graph connectivity of FEOL layout and seeks to define security with dimensions of the solution space from which the attacker must pick one correct solution. The proposed metric operates on Directional Acyclic Graphs (DAG) abstracted from both the complete layout (G) and FEOL layout (H) (see Figure 3a for an example). In the resulting DAG, gates are represented with *vertices*, i.e. colored circles in Figure 3a, whose color represents models of each gate; and nets are abstracted into sets of directional edges, each edge corresponds to a driving vertex and driven vertex pair (represented with arrows in Figure 3a). Then it computes the number of legal mappings k that maps each gate u_i in complete layout graph G to a distinct gate v_j in FEOL layout graph H. This number k is defined as the security of that gate, and the security of the complete layout is defined as the lowest k of all gates. In the example shown in Figure 3, XOR gates have a security $k = 2$ but all other gates have $k = 1$, therefore the overall security of the circuit remains at $k = 1$. This security metric is often referred to using its letter of choice k as *k-security*. A greedy algorithm is then presented to find a minimal subset of wires in the layout to uplift to BEOL while satisfying minimum security k, an optimization of split manufacturing security also known as *wire lifting*.

The introduction of k-security has two advantages. One, it is quantifiable, therefore an optimization algorithm can be designed with k-security as its objective function to improve the absolute security of the BEOL connections, instead of simply preventing every known loopholes. Two, its definition is not dependent on specific layout, which makes it compatible with most layout based approaches, and secure even when netlist of the design is compromised [19]. It is also effective against a wide spectrum of threats, owing to the fact that few attacks are possible without making sense of what function gates in the layout serves.

[2]e.g. XOR gates for their correlation with cryptographic circuits, flip-flops for their correlation with state machines, adders for their correlation with arithmetic cores, etc.



(a) A split manufactured full adder, its FEOL graph H, and its complete graph G.



(b) Both mappings of vertices in FEOL graph H to vertices in complete graph G are legal.

Fig. 3: Principle of k-security using a full adder as example.

However, it is not without weaknesses. Its first disadvantage is its difficulty to compute. k-security definition checks whether any given FEOL netlist has a security level of k by checking a property called subgraph isomorphism, which makes computation of the security level of any wire lifting solution NP-hard [19]. The proposed wire lifting algorithm in [19] functions by procedurally checking if lifting another wire lowers the security level, which makes it exponentially more complex. This makes it extremely computationally costly, and even less scalable than typical NP-hard algorithms. In addition, the fact that k-security is defined using graph connectivity also results in lack of consideration on any leakage of information from FEOL layout. This understandably deteriorated the performance of the design in every possible way, as it relies on layout *not* being optimized for performance to keep it from leaking security information. Further, need to boost k makes it necessary to reduce rare gate models, further restricting design performance.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2877012, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

2) Limitation of split manufacturing: Split manufacturing prevents all attacks that require complete knowledge of the whole layout, which also includes attacks against BISA such as identification of BISA cells. However, not all attacks require complete knowledge of the whole layout. One example is the untargeted Trojan insertion [18], a threat discussed in details in [27]. Untargeted hardware Trojans are capable of degrading the performance and/or reliability of manufactured ICs, or trigger a denial-of-service (DoS) in critical control systems [28].

D. Motivation: Implementing BISA with Split Manufacturing

In light of respective limitations of BISA and split manufacturing techniques, it makes sense to improve both with the relative advantage of the other. We henceforth term the combined technique obfuscated BISA (OBISA). The most apparent advantage of the resulting technique would be the security against untargeted hardware Trojan insertion, as well as security against IP piracy and IC cloning, both of which are the primary strengths of BISA and split manufacturing. Enhanced with split manufacturing, this new technique can also become secure against redesign attack that BISA was not able to fully prevent, since the attacker must first identify which existing cells are connected together before designing a functionally equivalent circuit to replace these existing cells. This will be much harder if the designer lifts the wires that connects them to BEOL, so that BISA structure becomes indistinguishable from original circuitry. Security against redesign attacks also reduces the necessity of using detection based anti-Trojan techniques, and relaxes prior necessity of only using minimum sized standard cell variant.

Split manufacturing security in OBISA could also benefit from BISA insertion. For example, additional cells and FEOL interconnects introduced to the layout by BISA insertion makes reaching higher k-security easier, and no longer restricted by rarely instantiated standard cell models, which has been leading to dilemma between either eschewing them or suffering from restricted security level. This makes wire lifting a particularly suitable candidate. Additional cells and interconnects introduced by BISA circuitry can help to homogenize distribution of FEOL features, and proximity based attacks could also be foiled by occupying white spaces and compensating spatial distribution of gate types with BISA cells, which makes OBISA secure when evaluated with layout-based security metrics for split manufacturing as well. To summarize, a combined OBISA technique improves from both split manufacturing and BISA in terms of their respective security metrics.

A few options exist to implementing OBISA, depending on how it implements split manufacturing. In a previous work [18], an approach with minimal computational cost was investigated. In that previous work, obfuscation connections were added between OBISA circuits and functional circuits, and between OBISA tree-like structures, while optimization on wire lifting was kept to a minimum; In this paper, we propose to investigate the opposite scenario, where level of security is desired, while keeping it viable for industrial level of integration. As explained earlier in Section II-C1, we find optimizing for split manufacturing security possible with wire lifting optimization for k-security. We also find it worth investigating because combining BISA with wire lifting more difficult than simply implementing both techniques, as additional cells introduced by BISA makes wire lifting exponentially more complicated as much as they make it more secure; in other words, combining them requires novel technique to overcome this non-trivial problem. Therefore, in this work we present an OBISA implementation with wire lifting using k-security definition, and investigations on its performance as well as overheads.

E. Contributions

In addition to theoretical advantages from combining BISA with wire lifting as was discussed in Section II-D, the presented technique also claims the following contributions from evaluations with implementations of the technique:

1) *A more efficient wire lifting algorithm*: By proposing a new set of solution constraints that are stronger than subgraph isomorphic [19], we were able to convert the wire lifting problem into a binary programming problem. In doing so, we developed a faster algorithm to find provably optimal[3] wire lifting solutions. Experiments on Circuit432 benchmark circuits yielded 75% to 155% of edges kept at $1.74 \times 10^5 X$ to $1.06 \times 10^6 X$ speed improvements over previous wire lifting algorithm.
2) *A comprehensive application framework on partitioning design into manageable layout*: Existing wire lifting algorithm is limited in size of layout it can process due to weakness in speed. The proposed fast wire lifting algorithm increased the size of layout it can realistically process by one to two orders of magnitude. In order to ensure applicability to industrial level of applications, we have investigated ways to partition designs, and proposed two approaches - one based on logic hierarchy, the other using simple geometry - that complement each other to cover all possible scenarios. Implementation with said partitioning techniques proved to be successful on designs up to $385,001$ gates large.
3) *Pin-based definition of edges*: An edge in [19] was defined based on its driving and driven vertices, which may not always be unique. With a wire lifting algorithm with greatly reduced time complexity, we are able to define edges using their driving and driven *pins* that eliminates this problem.
4) *Cell model compensation to further improve security level*: Unlike BISA, the proposed OBISA allows all standard cell models to be used. This allows us to compensate rarely instantiated gate models so that wire lifting restriction on rare standard gate models can be relaxed, meanwhile improving maximum achievable security level k. In doing so, we simultaneously improve security and reduce overhead.
5) *Secure in terms of almost all known layout-based security metrics*: Instead of scrambling layout at the cost of prohibitively high performance overheads as was opted in [19], in this presented approach of OBISA we perform normal performance-oriented optimizations typical of conventional design flows, and then show the resulting layout meets most known layout-based security metrics[4].

III. Obfuscated Built-In Self-Authentication via Wire Lifting

The proposed approach to implement OBISA is characterized by a major departure from its predecessor in [18]: it uses *wire*

[3]Optimum defined the same as the one used in [19], i.e. minimizing number of edges lifted.

[4]With the sole exception of *cell-level obfuscation*, which is beyond the scope of wire lifting or BISA, and can be addressed by combining dedicated obfuscation techniques with the presented technique.

lifting as its principal strategy to ensure split manufacturing security. A most rudimentary implementation is to simply insert BISA cells, and then solve for a wire lifting solution. This would allow most benefits by simply combining BISA with split manufacturing as described in Section II-D. However, it is possible to further improve OBISA security by modifying BISA insertion. This is illustrated in an example shown in Figure 4.

Consider the full adder as shown in Figure 3a, and graph for its FEOL layout. It is apparently impossible to distinguish the two XOR gates (represented by vertices shaded in red slash) in FEOL layout. XOR gates in this full adder have a $k = 2$ security. If the same can be said for all other gate models, the FEOL layout would have $k = 2$ security. Unfortunately, it is impossible for the full adder to reach $k = 2$ simply because it has only one OR gate.

There are a few ways to address this issue. In [19], only 3 to 7 gate models are allowed during design synthesis, in order to prevent rare gate models from restricting wire lifting optimization. From a designer's point of view, however, this approach seriously impacts the performance of the original circuitry in area and power. For example, restricting Circuit432 to 3 gate models almost triples total cell count (from 115 to 282) and doubles total power and area (1.7725×10^{-4} Watt, 1205.45 μm^2 to 3.4955×10^{-4} Watt, 2506.75 μm^2).

An OBISA technique that performs wire lifting does not have to submit to this restriction, because the number of instances of rare gate models can be compensated by inserting OBISA cells. This is illustrated in Figure 4. In this example, OBISA cells and interconnects are added, shown in dashed lines. We can see from the example how the bottleneck in the previous example - the single OR gate - is compensated with OBISA cells. In the shown wire lifting example, $k = 2$ security is reached; If we consider a more extreme solution, e.g. lifting all wires to BEOL, at maximum the layout could reach $k = 4$ security rating.

OBISA insertion does not impact timing, because they are not connected electrically to the functional circuit. It does not impact dynamic power because it is not going to be active during the design's functional mode. However, if the functional circuit is very small and number of rarely-instantiated standard cell models is large, white spaces in the layout might not be sufficient. In our experiment, compensating unrestricted Circuit432 layout using OBISA insertion to at least 10 cells per model required us to lower layout utilization to 0.42, which in turn increased area to 1205.45 μm^2 and power to 1.9549×10^{-4} Watt. This becomes less of a problem when applied on larger layouts. For example, in *one_round* submodule of 256 bit AES core, utilization ratio at 0.6 would allow us to compensate to at least 209 cells per model.

A. Time Complexity of Wire Lifting

Above discussion highlights an obstacle of simply combining BISA with wire lifting: The computational complexity of wire lifting algorithm, which will suffer exponentially if additional OBISA gates are added to its consideration.

Determining the security of any wire lifting solution has time complexity NP-hard. There are solutions with trivial difficulty to verify: for example, lifting all wires. However, most of these easy solutions demand a very high percentage of wires lifted to BEOL, which can cause overhead in timing and loss in fabrication yield due to increased difficulty of matching more vias. Therefore, a satisfactory wire lifting algorithm needs

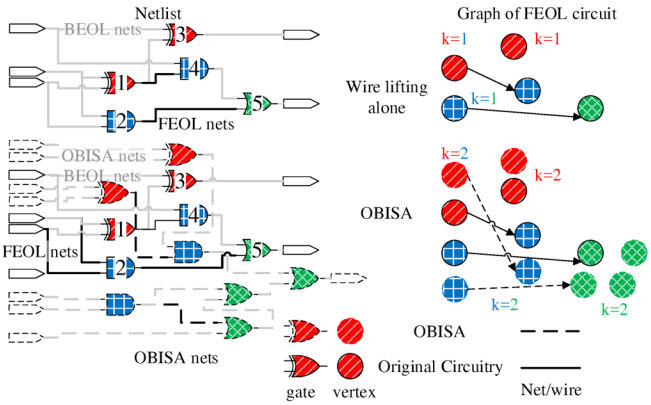


Fig. 4: Example: Due to OBISA insertion, wire lifting optimization on the same full adder can achieve higher k-security rating..

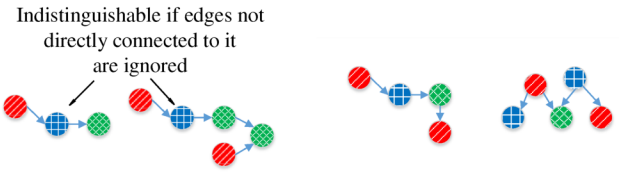
to minimize the number of wires lifted as its optimization objective.

So far, only one wire lifting algorithm based on this definition has been proposed [19], and it is based on greedy algorithm. Simply put, the algorithm (shown in Algorithm 1, henceforth referred to as "greedy wire lifting") starts from a wire lifting solution E' where all edges are assumed to have been lifted (i.e. E' equals to all edges in complete graph $E[G]$), then iteratively chooses each edge e among current E' to add back to FEOL and checks the resulting security σ of lifting solution E'. If the maximum resulting security $s = \max\{\sigma(E'\}) \geq k$ the algorithm adds its corresponding edge e_b to current solution E' and continue searching; otherwise it concludes with current solution E'. There are two problems with this approach: it is

```
   Input: All edges in graph G: E[G]
          Requested security level: k
   Output: Edges to lift: E'
1  begin
2      E' ← E[G]
3      while |E'| > 0 do
4          s ← 0
5          foreach e ∈ E' do
6              E' ← E' − {e}
7              if σ(E') > s then
8                  s ← σ(E')
9                  e_b ← e
10             end
11             E' ← E' ∪ {e}
12         end
13         if s < k then
14             return E'
15         end
16         E' ← E' − {e_b}
17     end
18     return E'
19 end
```

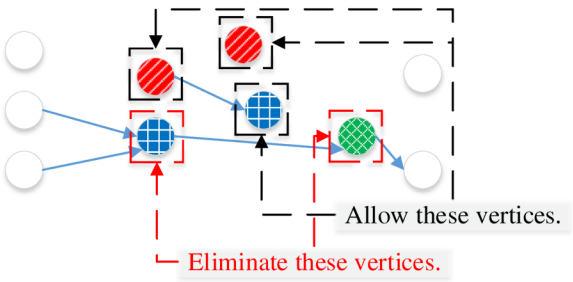
Algorithm 1: Greedy wire lifting algorithm [19].

not efficient, and it is not optimal. It is not optimal because the adding one wire back to FEOL will very likely preclude at least one other wire to be added back, and therefore limiting solutions the algorithm will be able to reach. Hence, the wire lifting solution available when choosing each wire to add back will be increasingly more reliant on choices made in earlier steps. If we choose to investigate all possible branches of the problem, the time cost will also be exponentially amplified. It is also not efficient because for each wire to be added back, security impact of adding each wire back to FEOL wires need



(a) Uniquely identifying two marked vertices in two subgraphs will need complete information of all vertices and edges in each subgraph.

(b) Two among many possible subgraphs with more than two interconnected vertices if vertices are allowed to connect to more than one edge.



(c) Constraint: only allow vertices connected to at most one edge.

Fig. 5: Edge types and vertex types: How to constrain the wire lifting problem to make it easier to solve.

to be determined. If we assume the number of wires kept to be a fraction of the total number of wires - a relationship usually holds in experiments - we see the complexity of the greedy wire lifting algorithm to be exponential with regard to the total number of wires in the design.

Unless a more efficient wire lifting algorithm is found, OBISA insertion will exponentially complicate the problem of wire lifting for exactly the same reason it aids the process.

B. Fast Wire Lifting

We have established two facts: One, that we know with mathematical certainty that even verifying k-security of any given wire lifting solution is NP-hard; Two, that we know with certainty that a wire lifting OBISA will need a more efficient wire lifting algorithm. In this subsection, we demonstrate that our proposed wire lifting OBISA technique can be efficient while satisfying those two seemingly prohibitive requirements, by providing an alternative approach to finding solutions to the wire lifting problem.

1) Binary Programming (BP)-based wire lifting algorithm: Since the problem of verifying k-security of *any given* wire lifting solution cannot be efficient, an efficient solution must not consist of it. It is easy to see that any wire lifting solutions can be represented with a n_e-bit binary vector, where n_e is the number of edges in the complete graph. If we can find a set of constraints so that all wire lifting solutions that satisfies said constraints also satisfy level k security, the problem of finding optimized wire lifting solutions becomes a Binary Programming (BP) problem, i.e.:

$$\begin{array}{ll} \text{maximize} & \sum_1^{n_e} x_i \\ \text{subject to} & \boldsymbol{Ax} \leq \boldsymbol{B}, \\ \text{where} & \boldsymbol{x} = (x_1, x_2, \ldots, x_{n_e})^\top, \\ & \forall i \in \{1, \ldots, n_e\}, x_i \in \{0, 1\}, \end{array} \tag{1}$$

$\boldsymbol{Ax} \leq \boldsymbol{B}$ is the set of constraints we have to find.

Now the problem becomes how to find such a set of constraints. k-security is about how many different vertices in complete graph can be mapped to the same vertex in FEOL graph. An apparent special case that satisfies this definition is that if s vertices of the same color (i.e., gates of the same model) in FEOL graph are indistinguishable from each other, we may say that $k = s$ for those s vertices; or more generally, for each group of vertices in FEOL graph that have a common identifiable feature that makes them distinct from other vertices and indistinguishable from among themselves, the security k of each vertex in this group equals to the number of vertices in this group s.

Vertices in a DAG have only three identifiable characteristics: its own color, the edges connected to it and vertices connected to these edges, and edges as well as vertices further connected. It is easy to see the third characteristic is most likely computationally the most complex: a vertex can be connected to a large number of vertices. This is obviously computationally complex and needs to be excluded. Satisfying constraints have to function with only information of the lifting decision on the edge to be decided only. This forces us to restrict each vertex to keep at most one edge (shown in Figure 5c), since as shown in Figure 5b, any combination of more than one edge per vertex will allow existence of subgraphs with more than two interconnected vertices.

All vertices in a wire lifting solution that satisfies this constraint will either be completely isolated (i.e., all edges lifted) or form a pair with its driving/driven vertex. The two-vertices-pair scenario has a very useful property for our purpose: that it is uniquely identified by the edge that connects both vertices, and the edge can be uniquely identified with only three pieces of information: the color of the driver vertex, the color of the driven vertex, and the direction of the edge. Based on this property, the set of constraints we need $\boldsymbol{Ax} \leq \boldsymbol{B}$ can be written as:

$$\begin{aligned} \boldsymbol{A}_{n_{t,a} \times n_e} \boldsymbol{x} \geq k, \quad a_{i,j} &= \begin{cases} 1 & \text{edge } e_j \text{ is of type } t_i \\ 0 & \text{otherwise} \end{cases} \\ \boldsymbol{B}_{n_{t,d} \times n_e} \boldsymbol{x} \leq 0, \quad b_{i,j} &= \begin{cases} 1 & \text{edge } e_j \text{ is of type } t_i \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

$$\boldsymbol{C}_{n_v \times n_e} \boldsymbol{x} \leq 1, \quad c_{i,j} = \begin{cases} 1 & \text{edge } e_j \text{ is connected to vertex } v_i \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$n_c - \boldsymbol{D}_{n_r \times n_e} \boldsymbol{x} \geq k \quad d_{i,j} \text{ is the number of vertices of reference } i \text{ that edge } e_j \text{ is connected to.} \tag{4}$$

Where k is the requested security level; t_i is a distinct type of edge defined using its driver vertex color, its driven vertex color, and its direction; $n_{t,a}$ is the total number of allowable edge types (i.e., standard cell models of both driving and driven cells of an edge); $n_{t,d}$ is the total number of edge types to be eliminated; n_v is the total number of vertices; n_c is the total number of colors of vertices (i.e., number of standard cell models that exist in the layout). Equation 2 constrains edges so that each distinct type of edge either has at least k indistinguishable instances, or are completely lifted to BEOL. The latter kind of types of edges can be determined by tallying total number of edges by types in the complete graph and banning types with fewer than k instances. Equation 3 constrains the lifting solution to leave at most one edge per vertex. Equation 4 constrains vertex colors to have at

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2877012, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

least k isolated vertices (i.e., vertices with all edges lifted).

One final piece in this puzzle is for the constraints to fit all possible scenarios. Constraints as shown in Equation 2 and 4 means at least one solution exists that satisfies all named edge types have at least k indistinguishable instances, and each color of vertices to have at least k vertices with all edges lifted. In reality, desirable solutions do not need to satisfy all these requirements. Some edge types that have more than k instances in the complete graph may have to be all lifted to ensure all others having at least k instances, or some colors of vertices may all keep an edge, leaving no need to lift all edges of at least k vertices of each of these colors.

To adapt our constraints to these different possible scenarios, we convert affected constraints into so-called either-or constraints by introduce a few extra variables $\boldsymbol{y}$ and $\boldsymbol{z}$ to choose between the alternatives. The complete description of the BP problem therefore becomes Equation 5. This is the complete set of constraints for the BP-based approach of fast wire lifting.

$$
\begin{aligned}
&\text{maximize} && \textstyle\sum_1^{n_e} x_i \\
&\text{subject to} && \\
& && \boldsymbol{A}_{n_{t,a}\times n_e}\boldsymbol{x} \geq k - M\boldsymbol{y}, \\
& && \boldsymbol{A}_{n_{t,a}\times n_e}\boldsymbol{x} \leq M(\boldsymbol{y}-1), \\
& && \boldsymbol{B}_{n_{t,d}\times n_e}\boldsymbol{x} \leq 0, \\
& && \boldsymbol{C}_{n_v\times n_e}\boldsymbol{x} \leq 1, \\
& && n_c - \boldsymbol{D}_{n_r\times n_e}\boldsymbol{x} \geq k + M(\boldsymbol{z}-1) \\
& && n_c - \boldsymbol{D}_{n_r\times n_e}\boldsymbol{x} \leq M\boldsymbol{z} \\
& && \forall i, x_i, y_i, z_i \in \{0,1\}, \\
&\text{where} && \boldsymbol{x} = (x_1, x_2, \ldots, x_{n_e})^\top \\
& && \boldsymbol{y} = (y_1, y_2, \ldots, y_{n_{t,a}})^\top \\
& && \boldsymbol{z} = (z_1, z_2, \ldots, z_{n_r})^\top \\
& && a_{i,j}, b_{i,j} = \begin{cases} 1 & \text{edge } e_j \text{ is of type } t_i \\ 0 & \text{otherwise} \end{cases} \\
& && c_{i,j} = \begin{cases} 1 & \text{edge } e_j \text{ is connected} \\ & \text{to vertex } v_i \\ 0 & \text{otherwise} \end{cases} \\
& && d_{i,j} \text{ is the number of vertices of reference} \\
& && i \text{ that edge } e_j \text{ is connected to.}
\end{aligned}
\tag{5}
$$

Here we delineate the process of producing a set of constraints with a real world example. Consider a netlist to be lifted for security k. We first identify its vertices $\boldsymbol{v} = (v_1, v_2, \ldots, v_{n_v})^\top$; each vertex v_i refers to one gate.Based on the standard cell model of the gate (referred to as "reference" in Equation 5), the type $t(v_i)$ of vertex v_i can be determined. Based on identified vertices, edges of the netlist can be defined as $\boldsymbol{e} = (e_1, e_2, \ldots, e_{n_e})^\top$, where each edge e_i is defined as a pair of vertices (v_{i0}, v_{i1}) and has a direction from v_{i1} to v_{i0}. Similar to vertices, type of edge e_i can be determined as $(t(v_{i0}), t(v_{i1}))$.

We can thus begin constructing equations to determine $\boldsymbol{x} = (x_1, x_2, \ldots, x_{n_e})^\top$, whose each binary element x_i represents whether edge e_i should be kept in FEOL part of the layout. We immediately know some ($n = n_{t,d}$) types of edges are fewer than k; they all have to be eliminated. We then construct a binary matrix $\boldsymbol{B}$ whose elements $b_{i,j}$ denote whether edge j is of edge type $i \in \{1, \ldots, n_{t,d}\}$, i.e. one of the edge types that need to be eliminated. This matrix $\boldsymbol{B}$ constitutes the third constraint $\boldsymbol{B}_{n_{t,d}\times n_e}\boldsymbol{x} \leq 0$. Conversely, some other ($n = n_{t,a}$) types of edges have more instances than k, and we can construct another binary binary matrix $\boldsymbol{A}$ whose elements $a_{i,j}$ denote whether edge j is of edge type $i \in \{1, \ldots, n_{t,a}\}$ that can be kept. This matrix $\boldsymbol{A}$ constitutes the first and second constraints $\boldsymbol{A}_{n_{t,a}\times n_e}\boldsymbol{x} \geq k - M\boldsymbol{y}, \boldsymbol{A}_{n_{t,a}\times n_e}\boldsymbol{x} \leq M(\boldsymbol{y}-1)$, where M is a natural number much larger than k, and $\boldsymbol{y}$ is a vector of supplemental variables, whose element y_i denotes whether a corresponding edge type i is to be kept in FEOL.

A further consideration is to not let any vertex become connected to more than one edge; this is ensured by constraining the sum of variable x of all edges connected to any same vertex to not exceed 1 for all vertices, as expressed in the fourth constraint $\boldsymbol{C}_{n_v\times n_e}\boldsymbol{x} \leq 1$. Finally, no type of vertices should have less than k vertices with 0 edge remaining in FEOL, or have less than k vertices with 1 edge remaining. This results in fifth and sixth constraints $n_c - \boldsymbol{D}_{n_r\times n_e}\boldsymbol{x} \geq k + M(\boldsymbol{z}-1), n_c - \boldsymbol{D}_{n_r\times n_e}\boldsymbol{x} \leq M\boldsymbol{z}$ and concludes construction of the binary programming problem.

2) Pin-based definition of edges: The prior definition based on cells impacts both security and/or difficulty of implementation in a real industrial design. First, it disregards the actual difference between pins. In a cell-based definition, two edges might both be leading from an inverter to an AND vertex, while in the netlist one wire is connected to the A pin and the other is connected to the B pin of their respective AND gate. This indicates actual number of indistinguishable wires may be much lower than the algorithm reports, which constitutes a leak of information.

Another problem is with multiple output cells, a most common example is flip-flops. Typical flip-flops offer two outputs, Q and QN, where one is the inverted signal of the other. A cell-based definition will be unable to distinguish different wires in this scenario and treat all of them as the same edge.

It is possible to modify the greedy wire lifting algorithm to work with pin-based definitions, but this will further exponentially increase already extremely long processing time. On the other hand, the proposed BP-based wire lifting algorithm can accommodate this with superior processing speed. Therefore, on top of being faster, provably optimal, the proposed BP-based algorithm is also free from a leak of information and can be applied to designs that uses gates with multiple outputs.

C. Implementation Flow

The proposed BP-based approach of wire lifting greatly alleviates the time complexity of wire lifting solution generation. However, binary programming remains an NP-complete problem.Therefore, implementation of proposed OBISA technique needs to provide solution to two specific problems:

1) Implementing the proposed OBISA technique on a reasonably-sized layout;
2) Converting any given design to layouts of the first kind.

For the first problem, we show a layout design flow modified from the original BISA implementation flow in Section III-C1; For the second problem, we propose to divide the layout along logic module boundaries and apply OBISA flow on each logic modules, shown in Section III-C2; in corner cases where this is not realistic or efficient, we present an alternative approach where the layout is divided using geometrical boundaries, and shown in Section III-C3.

1) Implementation flow on a reasonably-sized layout: The proposed OBISA flow is shown in Figure 6. Boxes shaded with blue slashes represent procedures already present in BISA flow, while boxes shaded with red crosses represent new procedures in this approach. Our need for security requires gate type compensation as well as random placement for maximal obfuscation. Cells of the rare gate models are placed before others to

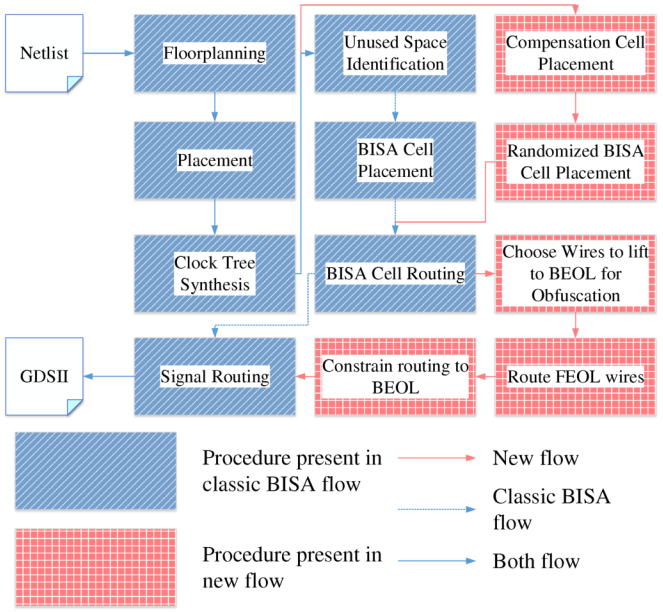
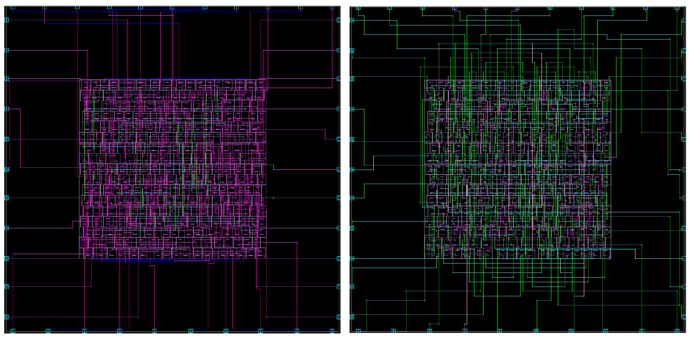


Fig. 6: Implementation flow of proposed OBISA technique on a reasonably-sized layout.



(a) Circuit432 layout without wire lifting optimization. **(b) Circuit432 layout using $k = 46$ wire lifting solution.**

Fig. 7: Circuit432 layout, with and without wire lifting optimization.

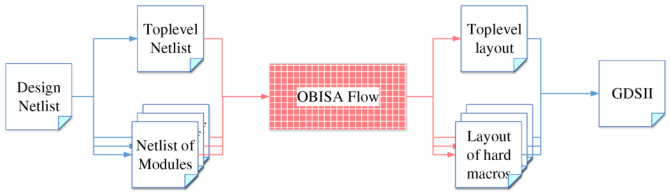


Fig. 8: Hierarchical wire lifting: Apply OBISA flow to each logic module, then integrate into final GDSII.

compensate gate model distribution. The locations of these gates are chosen randomly to reduce possible leakage of information in FEOL layout; for the same purpose, remaining white spaces are filled with BISA cells with random gate models. After that, classic BISA cell routing is performed. Before wire routing, an optimized wire lifting solution is found for the complete layout, using the BP-based technique we have discussed in Section III-B1. The rest of the design flow does not differ from conventional back-end design flow.

One sample result of this procedure is shown in Figure 7. In this example, Circuit432 benchmark from ISCAS'85 is used, and split is performed between M3 and M4 layers. The layout without wire lifting shown in Figure 7a shows most wires in purple (e.g., wires connecting core area to virtual pins), which is the color assigned to M2 layer, while the layout with $k = 46$ wire lifting shown in Figure 7b shows most wires in green (e.g., wires connecting core area to virtual pins), which is the color assigned to M4 layer. Compared to similar layout presented in [19], layout in Figure 7b does not appear to have significantly more congestion than layout in Figure 7a, likely due to the fact that placement optimization is not done blindly and therefore does not suffer from wire length overhead likely caused by an under-optimized placement.

2) Hierarchy-based partitioning: Designs larger than tens of thousands of gates likely need to be partitioned for wire lifting to be efficient. In this subsection we illustrate reuse of partitions already existing in a hierarchical layout design flow by simply performing OBISA insertion and wire lifting to each logic modules it instantiates. A flow diagram of the proposed hierarchy-based partitioning method is shown in Figure 8. In order to manage the amount of computation needed for wire lifting, designs are first partitioned into hard macros (Figure 8). If recurring circuit subgraphs is discovered, they can also be extracted into logic modules and follow the same routine.

3) Geometry-based partitioning: In addition to size, real industrial-scale designs pose unique challenges to efficient wire lifting which require further attention. For example, some of these challenges may include:

- Numbers of instantiations among logic modules differ. This leads to the need of compensation in gate types and security levels among non-uniformly instantiated modules.
- Some logic modules are consisted of few types of gates. This leads to need to hide this unique composition with OBISA cells.
- Some logic modules can be too large; some other logic modules may be too small to provide enough white space.

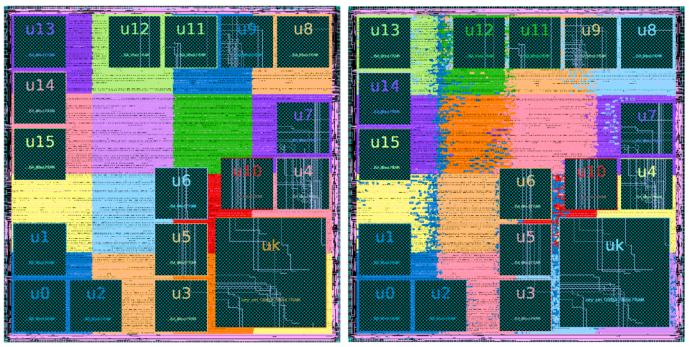
Some of these challenges can be addressed with clever applications of constraints and partition rules. For such challenges the following arrangements are made in our implementation:

- Use gate types from other logic modules with more types of gates for OBISA gate type compensation on logic modules with fewer types of gates, in order to hide the *standard-cell composition bias* present in such modules.
- Use lower utilization ratio for very small modules to accommodate OBISA cells.
- Assign lower security level k for more frequently instantiated modules.

However, it remains a possibility that a logic module may be too large and too indivisible. To prepare for such eventualities, we present a simple geometry-based partitioning scheme to complement hierarchical-based partitioning.

This geometric partitioning simply partitions cells in the layout into $n \times n$ rectangular regions based on their location, as shown in Figure 9a. Wire lifting can then be performed for each partition with updated security level divided by the number of partitions.

This method of partition leads to two more questions to be answered: One, how to determine the wire lifting solution of wires connecting cells belonging to different partitions; Two, edges that are not rarer than the security level of the module may become rarer than that of each partition, whose lifting should be decided independent from partitions. To address the first problem, we introduce the concept of *fringe cells*, defined



(a) Example: toplevel of a DES geometrically. **Partitioned (b) Partitions updated to include core fringe cells.**

Fig. 9: Layout partitioning for simplified wire lifting.

as cells belonging to other partitions that are connected to cells of current partition through edges. They, along with edges connecting them to cells in current partition, will be included when solving wire lifting problem of each partition. If any edge connecting a fringe cell is kept in any partition, the constraint representing that fringe cell in Equation 3 is changed to zero for all future partitions so that no other edge leading to that cell will be kept. We term edges featured in the second problem as *rare edges* and pulled from the consideration of each partition, and decided globally after solution of all partitions are found. To avoid solution of each partition and solution of rare edges from affecting solution of the other, constraints from Equation 3 and 4 involving rare edges are modified so that no matter the rare edges are lifted or kept, no cell will have more than one edges kept, and isolated gate counts of each gate model will be at least as large as security level of that partition. Specifically, constraints of cells connected to rare edges become

$$\begin{aligned} \boldsymbol{C'}_{n_{vre}\times n_e}\boldsymbol{x} \quad &\leq 0, c'_{i,j} = \begin{cases} 1 & \text{rare edge } e_j \text{ is} \\ & \text{connected to vertex } v_i \\ 0 & \text{otherwise} \end{cases} \\ n_c - \boldsymbol{D}_{n_{cre}\times n_e}\boldsymbol{x} &\geq k + \boldsymbol{n}_{n_{cre}\times 1} \end{aligned} \tag{6}$$

Where element $d_{i,j}$ of matrix $\boldsymbol{D}_{n_{cre}\times n_e}$ is the number of vertices of reference i that edge e_j is connected to, n_{vre} (number of rows of matrix $\boldsymbol{C'}_{n_{vre}\times n_e}$) is the number of vertices connected to rare edges, n_{cre} (number of rows of matrix $\boldsymbol{D}_{n_{cre}\times n_e}$) is the number of gate models that have vertices connected to rare edges, and element n_i of vector $\boldsymbol{n}_{n_{cre}\times 1}$ is the number of vertices of gate model i that are connected to rare edges. A diagram that elaborates on the entire process of partitioned wire lifting is shown in Figure 10.

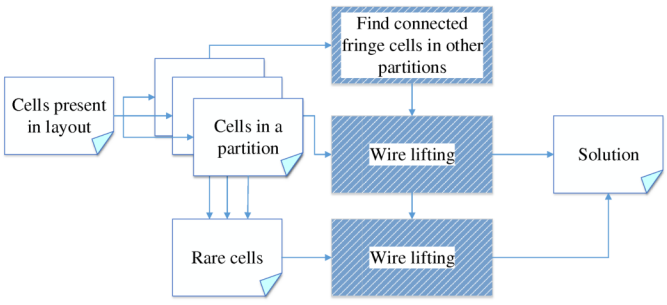


Fig. 10: Flow of partitioned wire lifting.

TABLE I: Number of nets and gates of used benchmark circuits at RTL stage

Benchmark	c432	c880	c1908	c3540
#RTL nets	499	588	766	1,571
#RTL gates	263	528	733	1,521
Benchmark	c5315	c6288	des	aes
#RTL nets	2,379	6,688	38,523	487,489
#RTL gates	2,201	6,656	34,264	448,136

IV. Experimental Evaluation

In this section we present experimental evaluation results to support our claims about the proposed technique, as well as explore implementation costs in terms of timing, area, power, and implementation time. Specifically, the following topics will be discussed:

1) Comparison of processing time and number of wires kept between by greedy wire lifting algorithm and proposed BP-based wire lifting algorithm;
2) Comparison of wire lifting performance between cell-based and pin-based definition of edges;
3) Evaluation of security of layout protected using proposed technique, in terms of known layout-based split manufacturing security metrics;
4) Demonstration of application on designs of industrial dimensions and evaluation of design overhead in terms of area, power, and path delays.

Results presented are collected using following benchmark circuits: Circuit432 from ISCAS'85 benchmark suite, DES and AES crypto-cores from www.opencores.org. Additionally, c880, c1908, c3540, c5315, c6288 from ISCAS'85 circuits are processed to investigate how fast the time complexity climbs as layouts become larger. Their respective sizes are shown in Table I. Note that these figures only reflect the number of gates when their RTL design are uncompiled, and may reduce depending on leeway given to synthesizer. .

Circuit432 is used to evaluate performance of proposed technique with regard to existing greedy wire lifting approach since it was used for this purpose in [19]. The small size of this benchmark circuit poses a particular challenge to OBISA insertion, which is not enough white spaces are left when normal floorplanning density is used, forcing a trade-off between area overhead and restriction on number of standard cell models. To study this limitation, two netlists of Circuit432 benchmark circuits are synthesized: one where only 3 standard cell models are allowed, and one without such restriction[5]. DES and AES cores are used in demonstration of application on designs of large scale and evaluation of design overhead. For each synthesized netlist, three layout are created: *Ctrl* is the control group where neither OBISA insertion nor wire lifting is performed; *OBISA-Only* has OBISA cell occupying white spaces, but routed normally; *OBISA-Lifted* underwent both OBISA cell insertion and wire lifting.

Proposed BP-based wire lifting algorithm is implemented by first generating the problem formulation using a script within the layout editor and then solved with a third-party integer linear programming solver. The presented results are collected using Synopsys IC Compiler and/or Design Compiler environment for the script, and solved with SCIP Optimization Suite [29].

[5]For Circuit432, not restricting standard cell models lead to 12 standard cells being used.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2877012, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

MiniSat, as was used in [19], is used as the Boolean satisfiability problem (SAT) solver in greedy wire lifting algorithm.

A. Performance of BP-based Wire Lifting Algorithm

All comparisons for the purpose of comparing processing speed were made on Circuit432 benchmark circuit synthesized with only 3 standard cell models. The definition of edges used in proposed BP-based wire lifting algorithm is also restricted to cell-based definition. Such restrictions were made to accommodate the greedy algorithm based wire lifting. Both evaluations took place on same server computer featuring 24-core 1995.216 MHz Intel CPUs, 384 GB total memory at 1333 MHz.

From Table II, we can see even under favorable circumstances, the greedy algorithm based approach is inferior in terms of processing speed by 1.74e5 to 1.08e6 times. Another observation is that while the BP-based approach does not appear particularly affected by requested security level k, high security level k significantly impacts the time taken by greedy algorithm based approach. This is likely resulting from the difference both approaches approach security levels. For the BP-based approach, a higher security level means only a larger integer being used on the right side of the constraint equation; indeed, higher security level often reduce the number of possible solutions and improve its speed. On the other hand, the greedy algorithm based approach evaluates security level of each candidate solution by enumerating k different isomorphic mappings between FEOL and the complete graph, a process that becomes exponentially more difficult as k increases.

A final row of data in Table II gives the percent of number of edges kept by the proposed BP-based approach as compared to greedy algorithm approach. The worst case performance in this metric gives us 75% , while best case performance ranges between 155% and 185%. This result has two implications: one, that for most security levels the performance of BP-based approach in terms of edges kept is sufficient, seeing that only in three occasions it yields a worse result than 90%, and one among them was 89%; two, that the result of greedy approach in this regard is much more erratic than that of BP-based approach. This likely results from fact that quality of solutions produced by BP solver is *mathematically guaranteed* under given constraints, while the result of the greedy approach relies on the quality of its earlier choices of kept edges. Therefore it is very much likely, and corroborated by results in Table II, that wire lifting solutions provided by the greedy approach are not optimal.

A few more benchmark circuits from ISCAS'85 benchmark suite have thus been processed, and their processing time are shown in Table III. In the table, "Total time" refers to the sum of both generation of BP constraints and the actual time involved in solving the problem with SCIP solver (i.e., same as "Time" in Table II), while "BP time" only refers to the later. Both results are averages of 100 repetitions. "BP time" is more relevant here since time it takes the EDA tool to retrieve relevant data is unlikely NP-complete. We can see from the table that item exceeds 1 second between one and two thousand gates, making layouts of around ten thousand gates likely upper bound of practicality by extrapolation.

B. Pin-based vs. Cell-based Definition of Edges

Shown in Table IV are number of edges kept n_e when cell-based definition and pin-based definition of edges are used, as well as evaluated level of security k using pin-based definition of edges on cell-based wire lifting results. As can be gathered from the results, not only does n_e differ when the definition of edge is changed, so does the security level. Since it is imprudent to assume the attacker is unable to distinguish pins from the layout, we must assume that cell-based definition of edges in fact leads to lower level of security than requested, as is evidenced by results in Table IV.

Having shown the superiority of pin-based definition of edges, we switch to pin-based definition of edges for results shown in the remainder of this section.

C. Security Evaluation with Known Layout-based Metrics

In this sub-section we present evaluations of proposed method in terms of existing layout-based security metrics for split manufacturing techniques. We are presenting results taken with the following metrics:

- Security against proximity attack is evaluated, as well as neighborhood connectedness ratio $C(R)$.
- Security against identification of functionality through standard cell composition bias is computed with the metric of the same name as defined in [22].

The metric of entropy in FEOL standard cells will not be evaluated as its definition overlaps and contradicts the principle of definition of security level as number of possible mappings from FEOL graph to graph of the complete layout.

1) Security against proximity attack: This metric is studied by simulating a proximity attack on sample layouts and calculating percentage of correct guesses. Layouts at various stages of implementation in the proposed OBISA flow were created to evaluate impact of each measure on success rate of proximity attack. In the table, *only* columns indicate layouts that underwent OBISA insertion only (i.e., without wire lifting), while *lifted* columns indicate layouts that underwent both OBISA insertion and wire lifting. Evaluations on a Circuit432 layout secured with wire lifting solutions produced with greedy algorithm and place-routed without BEOL information is also provided in column *anonymized* for comparison. In addition to Circuit432, *key_sel* module of DES core (to be elaborated in Section IV-D) is also shown as an example of effect on larger benchmarks. The results are shown in Table V. A first impression from the results as shown in Table V is that the number of successful guesses for each layout can be rather stochastic. Indeed, number of successful guesses of all layouts are below 5 except for two cases. This likely results from number of nets that actually had been routed as short as possible, an understandable objective of placement optimization. However, the number of open pins in FEOL does indeed become greatly improved by OBISA cell insertion as well as wire lifting. This on the other hand is likely more significant than possibilities of proximity attack being successful, as guess-based attack might not be always based on proximity, but all guess-based attack are universally more difficult as number of open pins in FEOL increase. If necessary, number of open pins in the FEOL can be further increased *arbitrarily* by adding dummy vias to BEOL layers that do not lead to BEOL wires. This is probably made more significant as larger k is requested - the lifted layout for 3-standard-cell-netlist is 46, much higher than $k = 10$ for the lifted layout of the 12-standard-cell-netlist.

2) Security against netflow attack: Recently, research interest has been focused on improving proximity attack [23], [24], likely due to its potential at producing valid successful attacks

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2877012, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE II: Comparison of Binary Programming (BP) and greedy algorithm-based wire lifting in terms of n_e kept and time consumption.

Method		k=46	k=32	k=20	k=16	k=12	k=8	k=4
BP	n_e kept	48	52	96	121	123	123	123
	Time (sec)	1.3	1.35	3.65	1.43	1.34	1.68	1.38
Greedy	n_e kept	≥ 26	56	101	78	152	138	165
	Time (day)	> 29	12.27	7.34	3.38	16.75	6.05	3.65
Speed improvement		> 1.92e6X	7.85e5X	1.74e5X	2.04e5X	1.08e6X	3.11e5X	2.29e5X
% of edges kept		$\leq 185\%$	93%	95%	155%	81%	89%	75%

TABLE III: Time consumption of proposed wire lifting algorithm on ISCAS'85 benchmark circuits with OBISA insertion

Benchmark	c880	c1908	c3540	c5315	c6288
Achieved k	10	10	19	20	27
FEOL edges	10	10	112	252	252
Total edges	323	264	990	1355	3475
Total time (sec)	0.77	0.69	2.99	3.49	14.88
Total cell count	248	209	631	864	2140
# Repetition	100				
BP time (sec)	0.06	0.07	0.11	0.77	8.28

TABLE IV: Comparison of security and n_e between cell-based and pin-based definition of edges.

Security level k		46	32	20	16	12	8	4
n_e kept	Cell-based	48	52	96	115	119	121	123
	Pin-based	48	50	68	105	117	120	123
Security level of cell-based		14	13	7	5	2	2	4

against split fabrication schemes. Therefore, it makes sense to further verify the security of our proposed OBISA scheme against a state-of-art attack. We have opted to replicate the *net-flow attack* as was described in [23], since the other alternative [24] was performed on routing benchmarks, whose conversion into hardware description language (HDL) would involve quite a lot of effort beyond the scope of this paper. The netflow attack makes use of four more hints in addition to geometric proximity, which are 2) acyclic combinational logic circuit; 3) load capacitance constraint; 4) directionality of dangling wires; 5) timing constraint.

Similar to the treatment in [23], we implemented netflow attack as a set of linear programming problem. Proximity and directionality of dangling wires were implemented as weights to potential connections, hint 3 and 5 are implemented as constraints, and hint 2 was implemented by detecting timing loops in netlist according to linear programming solution, and then adding constraints to prohibit connections responsible for detected timing loops and rerun the attack. Hint of directionality of dangling wires was not implemented as a hard constraint as was done in [23], because it was discovered that directions of dangling wires do not always fit the direction of the correct connection, and excluding all pins in "wrong" direction may leave the problem with no valid solution. The complete statement of the linear programming problem is shown in Equation 7:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum w_{i,j} x_{i,j} \\ \text{subject to} \quad & \\ & \boldsymbol{C}\boldsymbol{x} \geq 0, \\ & \boldsymbol{T}\boldsymbol{x} \geq 0, \\ & x_{i,j} \leq 0 \text{ if } i \text{ and } j \text{ share the same gate}, \\ & \textstyle\sum_s \sum_{i,j \in s} x_{i,j} \leq 0, \\ & \textstyle\sum_{j=1}^{J} x_{i,j} = 1 \\ & \textstyle\sum_{i=1}^{I} x_{i,j} > 0 \\ & \forall i, j, x_{i,j} \in \{0, 1\}, \end{aligned} \tag{7}$$

where $i \in \{1, 2, \ldots, I\}, j \in \{1, 2, \ldots, J\}$
I is total number of unconnected output pins,
J is total number of unconnected input pins,
$w_{i,j} = d_{i,j} - l_{i,j} - l_{j,i}$
$d_{i,j}$ = Euclidean distance between output pin i and input pin j
$l_{k,l}$ = Length of dangling wire of pin k in same direction as pin l
$c_{i,j}$ = Available capacitance allowance of output pin i minus capacitive load of input pin j
$t_{i,j}$ = Required arrival time of input pin j minus arrival time of output pin i
s is a set of all unconnected pins that are found in a timing loop

TABLE V: Success rate of proximity attacks.

Circuit432	Ctrl	Anonymized	OBISA		Ctrl	OBISA	
			Only	Lifted		Only	Lifted
#Std-cell	3				12		
OBISA insertion	No		Yes		No	Yes	
Lifted	No	Yes	No	Yes	No	No	Yes
Success	5.38%	0.38%	1.99%	6.42%	0.00%	0.54%	0.27%
#Cell	220	220	293	293	115	309	309
#OBISA cell	0	0	73	73	0	194	194
Open pins	93	523	184	680	102	952	954
Hit pairs	5	2	3	28	0	1	2

key_sel, DES Core	Ctrl	OBISA	
		Only	Lifted
OBISA insertion	No	Yes	Yes
Lifted	No	No	Yes
Success	0.24%	0.0003%	0.006%
#Cell	1608	3801	3801
#OBISA cell	0	2193	2193
Open pins	5461	13069	14957
Hit pairs	9	4	69

Experiments with thus described netflow attack was performed on c432 circuit. Since c432 circuit did not have sequential gates, required arrival time of unconnected input pins were implemented by taking the sum of visible gate delays between said input pin and output port, then subtracted with longest path delay of the circuit (serves as substitute to clock period). Three layout were created and evaluated: "Normal" was the control group where layout is placed and routed normally without human intervention; "Anonymized" has all its BEOL edges removed prior to placement, then routed without placement optimization (i.e., as was described in [19]); "OBISA" is placed normally, then underwent OBISA insertion flow as described in Figure 6. For *normal* layout, unconnected pins were extracted from layout by choosing all routed shapes of metal layer M3 and above; for the other two layouts, lifted edges were used.

Results from this evaluation are shown in Table VI. It can be gathered from the demonstrated results that OBISA is slightly less secure than anonymizing the layout placement, likely due to proximity hints not being entirely eliminated; however, both OBISA and anonymized significantly outperforms unaltered layout. Since OBISA circuitry often has short timing path, similar fan-out capability as functional circuitry, and no more likely to form timing loops than candidates in functional circuit, these hints are unlikely able to distinguish between OBISA and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2877012, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE VI: Results of netflow attack [23] on layouts of c432 benchmark with normal placement, anonymized placement, and OBISA insertion.

Circuit432 benchmark	Normal	Anonymized	OBISA
#Correct guesses	35	4	10
#Correct guesses in functional circuit			8
#Edges in BEOL	264	277	387
#Functional edges in BEOL			273

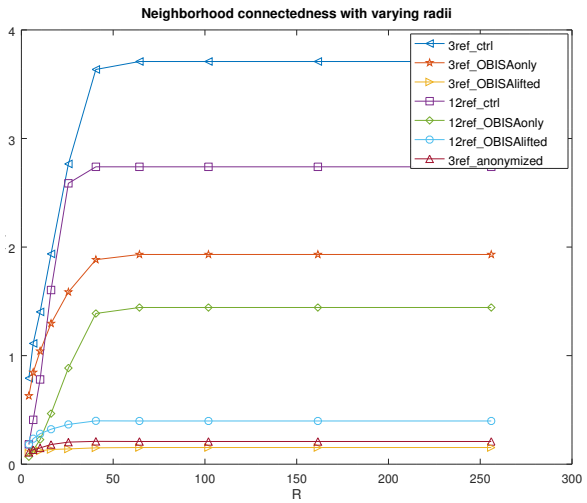


Fig. 11: Neighborhood connectedness ($C(R)$) curve as radii (R) increases, on Circuit432 layouts.

functional gates. Directions of dangling wires is less obvious, but the result seems to suggest even if that hint could distinguish OBISA and functional gates, its effect is small. We are of the opinion that dangling wires can likely be eliminated with relative ease by pre-inserting vias and wire shapes from pins to BEOL layers before letting automatic router to route BEOL edges, however proving it would be beyond the scope of this paper.

3) Neighborhood Connectedness: The neighborhood connectedness ($C(R)$) plot of the same layouts investigated in Table V is shown in Figure 11. As can be seen from the figure, all $C(R)$ curves saturates as radii increases, but both OBISA insertion and wire lifting reduces the eventual saturated $C(R)$. As have been pointed out in [22], the lower the measure, the more "spread out" the circuit is, and less functional information is leaked, resulting in a more secure FEOL layout. Figure 11 also shows $C(R)$ of a Circuit432 layout anonymized (using technique described in [19]) with the trace named *3ref_anonymized* closely follows $C(R)$ of the layouts with OBISA insertion AND wire lifting (traces *3ref_anonymized* and *12ref_anonymized*) at all ranges but lower than $C(R)$ of layouts without wire lifting, likely due to wire lifting.

4) Standard cell composition bias: In this evaluation, *key_sel* module and toplevel module of DES core are examined for its particular design characteristic. Being a control module, functional cells in both modules consists only of flip-flop and multiplexers. Thus, either unsecured module will be very weak in terms of Standard cell composition bias. This is compensated by inserting OBISA cells that are common in other modules of the same DES core. As shown in Table VII, standard cell composition bias of both modules decreased more than 50% after OBISA insertion.

TABLE VII: Standard cell composition bias of *key_sel* and DES toplevel.

	key_sel		des	
	ctrl	OBISA	ctrl	OBISA
Flip-Flops	840	840	1144	1144
Muxes	768	768	0	984
XORs/XNORs	0	0	562	2324
#Cells	1608	3800	1761	29276
bias	6.67E-01	2.82E-01	6.53E-01	5.74E-02

D. Implementation and Overhead on Large Designs

Two particular benchmarks were used in this study: an Advanced Encryption Standard (AES) and a Data Encryption Standard (DES) core. Crypto-cores are selected on the grounds that they are more likely targeted by attacks and usually require higher security reinforcements. After synthesis, the 256-bit AES core we have selected has $657,292$ gates, while the 64-bit DES core has $15,651$. Further, DES was also chosen in [19], and will likely serve as a good basis of comparison. Both designs are large enough to make lifting of a flattened netlist computationally heavy, and therefore necessitates partitioning. Both AES and DES cores are from opencores.org.

Each DES core in the design is consisted of 16 instances of *crp* module and 1 instance of *key_sel* module. The 256-bit AES core is consisted of 16 instances of *one_round* module, 7 instances of *expand_key_type_A_256* module, 6 instances of *expand_key_type_B_256* module, and 1 instance of *final_round* module. Finally, both DES and AES core instantiates interface cells such as flip-flops and multiplexers on their toplevel.

In our implementation we chose a security level $k = 16$ for *one_round* of AES and $k = 10$ for *crp* of DES core. These coefficients were chosen following the guideline as was discussed in Section III-C3, so that the overall security level can be made higher. This leads to an overall security level of $k = 208$ for AES core and $k = 160$ for DES core. To help improve efficiency, geometry-based partitioning was performed on both toplevel modules and *one_round* module of AES core. Implementation overheads in terms of power, timing delay, and area of each module are summarized in Table VIII and IX.

Both tables provide two sets of comparisons:

1) In terms of total wire length, number of OBISA cells inserted as compared to that of functional cells, as well as number of standard cell models instantiated: Close total wire length results between OBISA-inserted layout with (*Lifted* column) and without (*Only* column) wire lifting help to explain why little power and path delay difference were observed between these two types of layouts.

2) In terms of area, power, and path delays: OBISA-reinforced layout that underwent wire lifting (*Lifted* column under *OBISA* column) is compared against similarly OBISA-reinforced layout without wire lifting (*Only* column under *OBISA* column) as well as layout of same module without any security enhancement (*Ctrl* column). Area result are the same for all three scenarios since the same utilization ratio 0.6 was used for all layouts during their floorplanning stage. There is a slight increase in terms of power and path delays in the *Lifted* column with regard to the *Ctrl* column, but in all implementations quite small, and the worst-case path delay overhead in both cores are 3.64% and 4.08% respectively, while the total power overheads are 12.73% and 6.96%. Based on these results, we are confident

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2018.2877012, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

TABLE VIII: Power, timing, and area overheads of wire-lifted DES modules.

Module		key_sel			crp		
Layout		OBISA		Ctrl	OBISA		Ctrl
		Only	Lifted		Only	Lifted	
Power (W)	Internal	4.80E-03	4.25E-03	3.79E-03	1.52E-03	1.51E-03	1.50E-03
	Switching	7.48E-04	7.07E-04	5.71E-04	1.30E-03	1.18E-03	1.09E-03
	Leakage	2.88E-04	2.88E-04	2.13E-04	3.77E-05	3.77E-05	2.85E-05
	Total	5.84E-03	5.24E-03	4.58E-03	2.86E-03	2.72E-03	2.62E-03
Path Delays (ns)	Min	0.4	0.5	0.32	0.87	0.82	0.8
	Median	0.82	0.64	0.48	0.9	0.86	0.83
	Max	1.02	0.78	0.59	1.05	0.98	0.94
Total wire length(μm)		4.25E+05	3.37E+05	1.40E+05	6.86E+04	6.74E+04	2.70E+04
Area (μm^2)		67599.4			10354.2		
#Std-cell		9			28		
#Cell		4381		1608	1099		745
#OBISA cell		2773		0	354		0
Security Level k		1	160	1	1	10	1

Module		des		
Layout		OBISA		Ctrl
		Only	Lifted	
Power (W)	Internal	2.93E-03	2.94E-03	2.86E-03
	Switching	9.92E-03	9.39E-03	8.78E-03
	Leakage	1.32E-03	1.32E-03	2.41E-04
	Total	1.42E-02	1.37E-02	1.19E-02
Path Delays (ns)	Min	0.37	0.39	0.35
	Median	0.41	0.44	0.37
	Max	0.61	0.64	0.58
Total wire length(μm)		4.12E+06	3.99E+06	1.12E+06
Area (μm^2)		753423		
#Std-cell		34		
#Cell		29293		1778
#OBISA cell		27515		0
Security Level k		1	160	1

TABLE IX: Power, timing, and area overheads of wire-lifted AES modules.

Module		final_round			one_round		
Layout		OBISA		Ctrl	OBISA		Ctrl
		Only	Lifted		Only	Lifted	
Power (W)	Internal	6.70E-03	6.64E-03	6.44E-03	1.02E-02	1.01E-02	9.81E-03
	Switching	7.70E-03	7.49E-03	6.53E-03	1.16E-02	1.14E-02	1.12E-02
	Leakage	4.24E-04	4.24E-04	3.08E-04	6.40E-04	6.40E-04	6.39E-04
	Total	1.48E-02	1.46E-02	1.33E-02	2.25E-02	2.21E-02	2.16E-02
Path Delays (ns)	Min	1.37	1.32	1.2	1.83	1.79	1.71
	Median	1.42	1.37	1.24	1.92	1.88	1.79
	Max	1.7	1.62	1.42	2.46	2.28	2.2
Total wire length (μm)		1.08E+06	1.07E+06	4.90E+05	1.65E+06	1.64E+06	1.12E+06
Area (μm^2)		119882			177073		
#Std-cell		31			37		
#Cell		12377		8236	17688		11856
#OBISA cell		4141		0	5832		0
Security Level k		1	208	1	1	16	1
Module		expand_key_type_A_256			expand_key_type_B_256_OBISA		
Layout		OBISA		Ctrl	OBISA		Ctrl
		Only	Lifted		Only	Lifted	
Power (W)	Internal	3.53E-03	3.51E-03	3.06E-03	3.28E-03	3.09E-03	3.26E-03
	Switching	2.40E-03	2.23E-03	1.99E-03	2.09E-03	1.91E-03	1.96E-03
	Leakage	2.39E-04	2.39E-04	1.74E-04	2.31E-04	2.31E-04	1.74E-04
	Total	6.17E-03	5.98E-03	5.23E-03	5.61E-03	5.23E-03	5.39E-03
Path Delays (ns)	Min	1.15	1.12	1.09	1.13	1.13	1.11
	Median	1.23	1.19	1.16	1.2	1.19	1.17
	Max	1.69	1.56	1.48	1.55	1.53	1.47
Total wire length (μm)		5.00E+05	4.89E+05	2.18E+05	2.16E+05	2.41E+05	2.14E+05
Area (μm^2)		58680.1			58602.7		
#Std-cell		30			30		
#Cell		4662		2636	4760		2636
#OBISA cell		2020		0	2124		0
Security Level k		1	30	1	1	35	1

Module		aes_256_hier1		
Layout		OBISA	Ctrl	
			Only	Lifted
Power (W)	Internal	3.49E-02	3.27E-02	2.71E-02
	Switching	1.11E-01	7.89E-02	8.78E-02
	Switching	1.18E-02	1.18E-02	1.92E-04
	Total	1.58E-01	1.23E-01	1.15E-01
Path Delays (ns)	Min	0.33	0.33	0.35
	Median	0.49	0.49	0.44
	Max	1.63	1.14	1.27
Total wire length (μm)		1.19E+07	1.10E+07	1.06E+07
Area (μm^2)		5674310		
#Std-cell		106		
#Cell		108087		2636
#OBISA cell		107036		0
Security Level k		1	208	1

to conclude the proposed wire lifting based OBISA technique introduces no significant performance overhead to the original circuitry.

Implementation results shown in Table VIII and Table IX points at two improvements of significance that were achieved on top of the performance reported in [19]:

1) a much larger and more standard design (AES) achieved a much higher level of security; and
2) overheads in area, delay, and power are reduced from tens to hundreds percent to around ten percent in power, less than five percent in delay, and zero percent in area; further, limitation on number of standard cell models was also removed.

The first difference between the AES module and DES module is their difference in size: *one_round* module of AES has more than ten times as many gates as *crp* module of DES, even before we consider additional cells brought about by insertion of OBISA circuitry. All things considered, the OBISA-inserted AES core consisted of $385,001$ gates, more than 25 times as many gates as a DES core without OBISA insertion. Another difference is in the fact that DES core is a very unique design: only its *key_sel* module and its topmodule have flip-flops, both of which are instantiated only once. Therefore, implementation on AES core, whose modules are all clocked, demonstrates the ability to be applied on synchronous design, as we have predicted during our introduction of our pin-based definition of edges in Section III-B2. A final observation is that our proposed BP-based wire lifting approach allowed presented implementation to reach security levels such as $k = 160$ and $k = 208$ with ease, much higher than previously reported $k = 64$ [19]. This supported our early observation that satisfying an arbitrarily high security level is not only easy for the proposed BP-based approach, it often takes it even less time to conclude than lower security levels which may have more viable solution candidates.

Equally significant is the reduction in overheads. As was theorized previously in Section III-C1, the huge overhead[6] in [19] was most likely result of the approach of eliminating layout cues by preventing place and route tool to optimize the design according to its function. Our evaluation in terms of known layout-based split manufacturing security metrics supported our hypothesis that it would not greatly impact security performance. Our theory that OBISA insertion would help remove the restriction on number of standard cell models was also supported by our implementation result: only design where any such restriction was felt was *crp* module of only 745 gates, where we achieved $k = 10$, and could have further improved that number had we allowed ourselves overhead in area.

E. Comparison with contemporary research

Since after the submission of this work, another work [30] have been accepted at a conference, which is similar to our work in also seeking to improve upon the time complexity of wire lifting algorithm using mixed-integer linear programming, and improving the security level by introducing dummy vertices and edges. We find it encouraging that the idea that timing complexity of wire lifting algorithm can be improved has received support.

[6] 54% to 92% in power, 73% to 114% in delay, 167% to 502% in area were reported in [19].

The primary difference between these two works, on the other hand, is that OBISA is intended as an improvement to existing BISA technique, and therefore carries limitations along with advantages of BISA, as opposed to the technique reported in [30], which is intended as an improvement to k-security. One example of this difference is that all additional gates inserted by the OBISA technique will only occupy white-space and therefore do not incur additional area, power, or timing overhead. Further, the OBISA technique occupies all available white spaces and prevents untargeted Trojan insertion, which is not always possible with split manufacturing alone.

V. Conclusion

In this paper, we have presented a novel implementation approach of Obfuscated Built-In Self-Authentication (OBISA) technique that combines hardware Trojan deterrence through Built-In Self-Authentication (BISA) circuit insertion as well as optimized split manufacturing through wire lifting. The resulting technique is shown to be efficient, secure, and introduces very low performance overhead to the functional design that it is fit for industrial level of integration. The presented implementation flow is tailored to work with all mainstream EDA tools. In the future, the proposed flow could be further improved by expanding the presented technique to further reduce overhead and improve solution generation efficiency.

References

[1] U. Guin, D. Forte, and M. Tehranipoor, "Anti-counterfeit techniques: from design to resign," in *2013 14th International Workshop on Microprocessor Test and Verification*. IEEE, 2013, pp. 89–94.

[2] M. M. Tehranipoor, U. Guin, and D. Forte, "Counterfeit integrated circuits," in *Counterfeit Integrated Circuits*. Springer, 2015, pp. 15–36.

[3] U. Guin, Q. Shi, D. Forte, and M. M. Tehranipoor, "Fortis: A comprehensive solution for establishing forward trust for protecting ips and ics," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 4, p. 63, 2016.

[4] U. Guin, "Establishment of trust and integrity in modern supply chain from design to resign," 2016.

[5] K. Xiao, "Techniques for improving security and trustworthiness of integrated circuits," 2015.

[6] "IARPA Trusted Integrated Circuits (TIC) program announcement," http://www.fbo.gov.

[7] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware trojan detection and reducing trojan activation time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 1, pp. 112–125, 2012.

[8] J. Li and J. Lach, "At-speed delay characterization for ic authentication and trojan horse detection," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 8–14.

[9] Y. Jin, N. Kupp, and Y. Makris, "Dftt: Design for trojan test," in *Electronics, Circuits, and Systems (ICECS), 2010 17th IEEE International Conference on*. IEEE, 2010, pp. 1168–1171.

[10] J. Rajendran, V. Jyothi, O. Sinanoglu, and R. Karri, "Design and analysis of ring oscillator based design-for-trust technique," in *29th VLSI Test Symposium*. IEEE, 2011, pp. 105–110.

[11] H. Salmani and M. Tehranipoor, "Layout-aware switching activity localization to enhance hardware trojan detection," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 1, pp. 76–87, 2012.

[12] R. S. Chakraborty and S. Bhunia, "Security against hardware trojan through a novel application of design obfuscation," in *Proceedings of the 2009 International Conference on Computer-Aided Design*. ACM, 2009, pp. 113–116.

[13] M. Banga and M. S. Hsiao, "Odette: A non-scan design-for-test methodology for trojan detection in ics," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 18–23.

[14] R. S. Chakraborty and S. Bhunia, "Harpoon: an obfuscation-based soc design methodology for hardware protection," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, no. 10, pp. 1493–1502, 2009.

[15] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 83–89.

[16] K. Xiao and M. Tehranipoor, "Bisa: Built-in self-authentication for preventing hardware trojan insertion," in *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 45–50.

[17] K. Xiao, D. Forte, and M. Tehranipoor, "A novel built-in self-authentication technique to prevent inserting hardware trojans," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, pp. 1778–1791, 2014.

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

[19] F. Imeson, A. Emtenan, S. Garg, and M. Tripunitara, "Securing computer hardware using 3d integrated circuit (ic) technology and split manufacturing for obfuscation," in *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, 2013, pp. 495–510.

[20] X. Yang, B.-K. Choi, and M. Sarrafzadeh, "Routability-driven white space allocation for fixed-die standard-cell placement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 4, pp. 410–419, Apr 2003.

[21] J. J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 2013, pp. 1259–1264.

[22] M. Jagasivamani, P. Gadfort, M. Sika, M. Bajura, and M. Fritze, "Split-fabrication obfuscation: Metrics and techniques," in *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 7–12.

[23] Y. Wang, P. Chen, J. Hu, and J. J. Rajendran, "The cat and mouse in split manufacturing," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 165.

[24] J. Magaa, D. Shi, J. Melchert, and A. Davoodi, "Are proximity attacks a threat to the security of split manufacturing of integrated circuits?" *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 12, pp. 3406–3419, Dec 2017.

[25] C. T. O. Otero, J. Tse, R. Karmazin, B. Hill, and R. Manohar, "Automatic obfuscated cell layout for trusted split-foundry design," in *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*. IEEE, 2015, pp. 56–61.

[26] Y. Xie, C. Bao, and A. Srivastava, "Security-aware design flow for 2.5d ic technology," in *Proceedings of the 5th International Workshop on Trustworthy Embedded Devices*, ser. TrustED '15. New York, NY, USA: ACM, 2015, pp. 31–38. [Online]. Available: http://doi.acm.org/10.1145/2808414.2808420

[27] Q. Shi, K. Xiao, D. Forte, and M. M. Tehranipoor, "Obfuscated built-in self-authentication," in *Hardware Protection through Obfuscation*. Springer International Publishing, 2017, ch. 11, pp. 263–289.

[28] R. J. Turk *et al.*, *Cyber incidents involving control systems*. Idaho National Engineering and Environmental Laboratory, 2005.

[29] G. Gamrath, T. Fischer, T. Gally, A. M. Gleixner, G. Hendel, T. Koch, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, S. Schenker, R. Schwarz, F. Serrano, Y. Shinano, S. Vigerske, D. Weninger, M. Winkler, J. T. Witt, and J. Witzig, "The scip optimization suite 3.2," ZIB, Takustr.7, 14195 Berlin, Tech. Rep. 15-60, 2016.

[30] M. Li, B. Yu, Y. Lin, X. Xu, W. Li, and D. Z. Pan, "A practical split manufacturing framework for trojan prevention via simultaneous wire lifting and cell insertion."