Rethinking Split Manufacturing: An Information-Theoretic Approach with Secure Layout Techniques

Abhrajit Sengupta[†][*], Satwik Patnaik[†][*], Johann Knechtel[‡], Mohammed Ashraf [‡], Siddharth Garg[†], and Ozgur Sinanoglu[‡]

[†] Tandon School of Engineering, New York University, New York, USA
[‡] New York University Abu Dhabi, Abu Dhabi, United Arab Emirates
{as9397, sp4012, johann, ma199, sg175, ozgursin}@nyu.edu

***Abstract*—Split manufacturing is a promising technique to defend against fab-based malicious activities such as IP piracy, overbuilding, and insertion of hardware Trojans. However, a network flow-based proximity attack, proposed by Wang *et al.* (DAC'16) [1], has demonstrated that most prior art on split manufacturing is highly vulnerable. Here in this work, we present two practical layout techniques towards secure split manufacturing: (i) gate-level graph coloring and (ii) clustering of same-type gates. Our approach shows promising results against the advanced proximity attack, lowering its success rate by 5.27*x*, 3.19*x*, and 1.73*x* on average compared to the unprotected layouts when splitting at metal layers M1, M2, and M3, respectively. Also, it largely outperforms previous defense efforts; we observe on average 8*x* higher resilience when compared to representative prior art. At the same time, extensive simulations on ISCAS'85 and MCNC benchmarks reveal that our techniques incur an acceptable layout overhead. Apart from this empirical study, we provide—for the first time—a theoretical framework for quantifying the layout-level resilience against any proximity-induced information leakage. Towards this end, we leverage the notion of mutual information and provide extensive results to validate our model.**

I. Introduction

Nowadays, more and more companies rely on external foundries for cost-effective access to advanced fabrication technologies. However, as this trend towards globalization of integrated circuit (IC) manufacturing consolidates, companies are forced to share their valuable intellectual property (IP) with potentially untrusted parties. This dependency coupled with currently inadequate protection measures has led to many security vulnerabilities such as IP piracy, overbuilding, and insertion of hardware Trojans [2]–[4]. These threats are becoming an increasing concern for both commercial and military organizations. In fact, it is estimated that several billions of dollars are lost each year owing to IP piracy [5].

A. Split Manufacturing and Proximity Attack

Split manufacturing was proposed by the IARPA agency [6] to thwart the aforementioned threats. Leveraging the asymmetry of the metal layers, the design is split into two parts: the front-end-of-line (FEOL), consisting of the active device layer and lower metal layers (e.g., $\leq$ M3), and the back-end-of-line (BEOL), that is the remaining higher metal layers (e.g., $\geq$ M4).[1] The FEOL is manufactured in a high-end, third-party foundry which is *untrusted*, whereas the BEOL is fabricated at a *trusted* facility on top of the incomplete wafer(s) provided by the FEOL foundry. This two-step approach helps to hide the overall functionality of the design from an attacker residing at the FEOL foundry, thereby hindering her/him from pirating the design or maliciously modifying it via hardware Trojans. Recently, different works have successfully demonstrated the feasibility of split manufacturing [7]–[10].

Unfortunately, naive split manufacturing falls short of ensuring security. Commercially available physical-design tools apply certain heuristics to minimize power, performance, and area, which may leak certain information. An attacker in the foundry can leverage this information to retrieve the missing BEOL connections, possibly undermining the defense intended by split manufacturing. In fact, Rajendran *et al.* [11] exploit the physical proximity between the cells to be connected; they demonstrated a *proximity attack* that connects nearby cells to retrieve the missing BEOL connections. Recently, an advanced network-flow attack was presented by Wang *et al.* [1]—this attack has been shown to render most prior protection schemes for split manufacturing insecure.

The threat model for split manufacturing is depicted in Fig. 1. The attacker has access to the technology libraries but is oblivious of the functionality of the IC. Naturally, she/he also cannot obtain a working IC; the IC is yet to be manufactured.

B. Prior Art and Our Contributions

To thwart proximity attacks, a pin-swapping based countermeasure was proposed in [11]. In practice, however, many swapped connections can still be correctly inferred. Jagasivamini *et al.* [12] showed that splitting at a lower layer renders the design more secure against proximity attacks; they propose to split the design at M1. Though splitting at M1 may render the design secure against such attacks, it also necessitates state-of-the-art manufacturing facilities at the trusted BEOL foundry. As a result, the cost of production significantly increases, defeating one of the promises of split manufacturing, i.e., affordable (but secure) IC production [7]–[9]. Besides, Wang *et al.* [1] proposed an algorithm for heuristic placement perturbation towards layout protection. Wang *et al.* [13] further proposed a routing-based scheme targeting for 50% Hamming distance between the original and the reconstructed netlist (to induce the maximal ambiguity for an attacker). Magaña *et al.* [14] insert routing blockages to lift wires and, thus, to mitigate routing-centric attacks. To counter Trojan insertion, a formal method for "k-security by wire lifting" was proposed by Imeson *et al.* [15]; it comes along with a high overhead, e.g., $\approx 200\%$ for delay.

Apart from this, there has been little to no effort towards a theoretical model which can quantify the resilience of a design against proximity attacks in general. In this work, we propose such a model based on the concepts of entropy and mutual information. Although the notion of entropy was previously advocated by Jagasivamani *et al.* [12], their study lacks specific formulations and, thus, fails to measure resilience in both theory and practice.

Building up on our theoretical framework, we propose several *placement-centric* techniques aiming to make split manufacturing secure against any proximity attack while ensuring practicality. As for our baseline approach, i.e., full randomization of the placement, it provides the highest level of security, but also incurs the highest

[*] A. Sengupta and S. Patnaik contributed equally.

[1] In accordance with [1], our notion of splitting, e.g., at M2, implies that metal layers M1 and M2 as well as V23 (i.e., the vias between M2 and M3) are readily available to fab-based attackers.

The definitive Version of Record is published in Proc. International Conference On Computer Aided Design (ICCAD) 2017
http://dx.doi.org/10.1109/ICCAD.2017.8203796

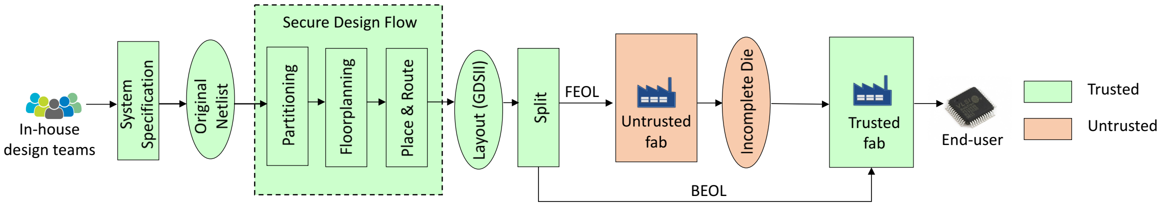


Fig. 1. Threat model for split manufacturing, along with our secure design flow (dashed). Note that the untrusted FEOL fab may want to pirate some IP and/or insert hardware Trojans. In this work, we primarily focus on the former aspect of the threat model.

layout overhead. Thus, to reduce overhead, we propose two novel techniques based on graph coloring and clustering gates of the same type. We show empirically that these techniques can attain notably better trade-offs for layout cost and security.

The contributions of our work can be summarized as follows:

- An information-theoretic framework to gauge the resilience of a given layout against any proximity attack (Section II).
- Two placement-centric techniques which help to render split manufacturing-based layouts secure against any proximity attack at acceptable overhead (Section III).
- A thorough investigation based on the well-known ISCAS'85 and MCNC benchmarks, demonstrating the effectiveness of our techniques and contrasting with naive randomization and prior art (Section V). Here we also investigate the cost-security trade-offs for split manufacturing induced by different split layers; we look into splitting at M1 up to M6.

II. Information-Theoretic Metric

Recall that split manufacturing is meant to offer protection against fab-based attacks such as IP piracy, overbuilding, and/or insertion of hardware Trojans. While the intended protection is based on the fact that the FEOL and BEOL of the chip are manufactured by different parties, physical design tools still operate on the entire design holistically, driven by the strong need for design and cost optimization. As a result, any partial, FEOL-level layout might leak certain information which can be leveraged by an attacker to infer the hidden BEOL connections. Indeed, the notion of physical proximity between connected cells, among other hints, has been leveraged in multiple attacks [1], [11], [14].

An attack-based, empirical security evaluation has two major drawbacks: (1) it can be time-consuming and, thus, ineffective for large layouts; (2) it is naturally specific to the employed attack and, thus, fails to quantify the layout's protection (or the lack of) against other attacks. Surprisingly, however, there has been very little to no effort towards measuring the resilience of a layout against an advanced or even an optimal attack. In this regard, we introduce *an information-theoretic metric to quantify the amount of information that can be extracted by an attacker from the physical layout.*

A. Measures of Information Leakage

To measure the uncertainty of an attacker about the missing connectivity of a given layout, we leverage the concept of *entropy*, which was famously introduced by Shannon. Note that the concept of entropy has been extensively employed to assess the vulnerability of cryptosystems in the context of side-channel attacks, such as power analysis or timing attacks [16], [17].

The entropy of a variable $X : \mathbb{X}$ is defined as

$$H[X] = -\sum_{x \in X} \Pr[X = x] \cdot \log \Pr[X = x] \quad (1)$$

Given another variable $Y : \mathbb{Y}$, the conditional entropy of X denoted as $H[X|Y]$ can be expressed as

$$H[X|Y] = -\sum_{y \in Y} \Pr[Y = y] \cdot H[X|Y = y] \quad (2)$$

The attacker's initial uncertainty about X is $H[X]$, and given a leakage model denoted by Y, the amount of information leakage—formally termed as *mutual information (MI)* [17]—is expressed as

$$I(X;Y) = H[X] - H[X|Y] \quad (3)$$

B. Resilience Against Proximity Attacks

Any proximity attack leverages the fact that the distance between cells reveals information about their connectivity. Thus, by analogy, the distance between cells constitutes the leakage model which an attacker tries to exploit. Hence, we define two variables, X and D, capturing the connectivity and distance between cells as

$$X = \begin{cases} 1 & \text{if two cells } u \text{ and } v \text{ are connected;} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

$$D = \mathtt{distance}(u, v) \quad (5)$$

Without loss of generality, we apply the notion of Manhattan distance (sum of horizontal and vertical distance) between cells.

To quantify the amount of information revealed by their distance about the connectivity between cells and, thus, to quantify the resilience of a layout against proximity attacks, we determine the mutual information *MI*

$$MI = I(X;D) = H[X] - H[X|D] \quad (6)$$

Note that the conditional entropy $H[X|D]$ itself can serve a similar purpose, but it fails to capture the notion of information leakage.

To compute $H[X]$ and $H[X|D]$, we determine the distribution of X and D for a given layout in a pairwise manner for all gates, allowing a straightforward and efficient computation of $I(X;D)$.

The *MI* quantifies the inherent protection of a layout against proximity attacks; the lower the *MI*, the lower the correlation between connectivity X and distance D and, thus, the better the protection. This correlation is apparent from Fig. 2 where the graph for correctly recovered connections (by running the proximity attack of [1]) is plotted over the normalized *MI* for the *c7552* benchmark split at M1. Here we shuffle the placement of randomly selected cells (from 0 to 100% of all cells, in steps of 10%). This way, we obtain 11 different layouts with varying and unbiased distributions for the *MI*. The plot reveals a linear relation between the *MI* and the correct connections (i.e., the attacker's success rate), validating our hypothesis that a lower *MI* implies higher security.

The goal of a security-aware designer is thus to generate layouts in such a way that the *MI* is minimized. Also, another interesting measure could be $I(D;X)$, i.e., the amount of information revealed

The definitive Version of Record is published in Proc. International Conference On Computer Aided Design (ICCAD) 2017
http://dx.doi.org/10.1109/ICCAD.2017.8203796

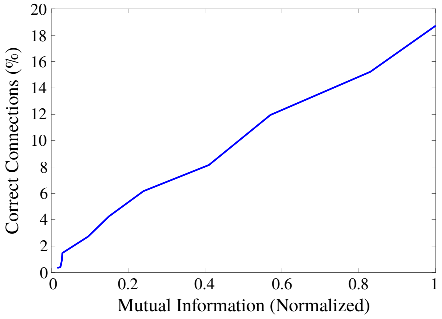


Fig. 2. Correct connections over normalized mutual information for stepwise randomized layouts of *c7552*, split at M1.

by the connectivity about the distance of gates, but it turns out that

$$\begin{aligned} I(D;X) &= H[D] - H[D|X] \\ &= H[D] - ([H[X|D] - H[X] + H[D]) \\ &= H[X] - H[X|D] = I(X;D) \end{aligned}$$

So far, we have considered the distance between cells; thus, the *MI* only quantifies the layout's security when split at M1. Such findings may not translate well for splitting at higher metal layers (see also Section V). Nonetheless, the proposed information-theoretic metric is generic in the sense that it is still applicable to higher split layers as well—one has to simply consider the distances of *open pins/vias* rather than the distances between cells.[2] Since these pins/vias represents parts of the overall routing infrastructure, which is typically optimized towards short interconnects, they will leak additional information beyond the placement of cells.

Notably, applying our metric at different layers will guide the designer which layer he/she should split at, using a precise and quantified trade-off between security (higher when split at lower layers) and cost (lower when split at higher layers). As we focus on placement-centric techniques, here we compute the *MI* considering gate distances—our metric readily and accurately evaluates the layout protection (or lack of) when splitting at M1.

III. Our Secure Layout Techniques

Next, we present different placement-centric techniques for making a layout secure in the context of split manufacturing and proximity attacks. Our analysis above elucidates the need to minimize the layout's mutual information (*MI*) of connectivity and distances, to mitigate any proximity attack.

One obvious and straightforward (thus naive) idea is to completely randomize the placement of cells in the layout to achieve the desired effect. The intuition here is that randomizing a layout would stretch the connected cells apart in an unpredictable manner, thus successfully eliminating any proximity-induced information leakage. This is illustrated in Fig. 3 where the distribution of connectivity is plotted against distance for the original and randomized layout of *c7552*, respectively. It is easy to see that the connectivity in the randomized layout is nearly uniformly distributed over the distance, unlike the original one which is heavily correlated with distance. The random layout exhibits a very low *MI*, and is expected to be secure even against advanced proximity attacks. However, it also incurs excessive overhead regarding power, performance, area, and wirelength, sometimes up to 600%. In Fig. 4, for example, we plot the correct connections and *MI* against wirelength overhead for *c7552* when split at M1. The grey-shaded region in the plot

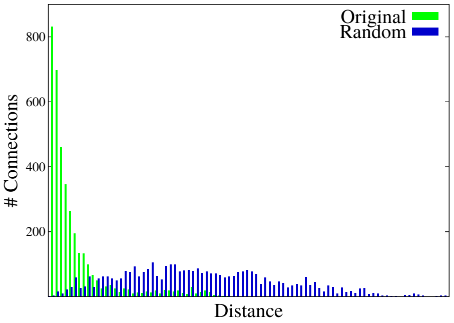


Fig. 3. Distribution of connectivity over distance of *c7552* for original (green) and randomized layout (blue).

marks the desirable solution space having better trade-offs for security and layout cost when compared to randomization. This raises the following question: *can we develop layout techniques that may approximate or even improve the security/resilience level of layout randomization yet at a reasonable cost?*

Here we take on this challenge and present two novel layout techniques, called *g-color* and *g-type*. As illustrated in Fig. 4, our techniques can achieve a similar level of security when compared to randomization, with much lower wirelength overheads at the same time (see also Section V-B for more details on layout cost). In the next two subsections, we present our techniques.

A. g-color

We leverage *graph coloring* to hide the connectivity information; coloring a netlist mandates that there be no connectivity between gates of the same color. The "colored netlist" is then partitioned by clustering all cells of same colors together and the placement of cells is confined within their respective clusters. These constraints naturally mitigate the information leakage to a great extent, thereby making the layout more secure, albeit in a cost-effective manner.

The coloring technique is described in Algorithm 1. (The reader is also referred to Section IV for further details on layout generation.) We extend the greedy coloring strategy discussed in [18]. The process is illustrated in Fig. 6 where we show the coloring of a full-adder circuit (see also Fig. 5 for the latter). For the sake of simplicity, the inputs and outputs are also considered as vertices/gates, as they are likely connecting to other cells in the overall design. The first vertex is selected at random and the rest of the vertices are colored iteratively.[3] After coloring all the vertices/gates, they are clustered

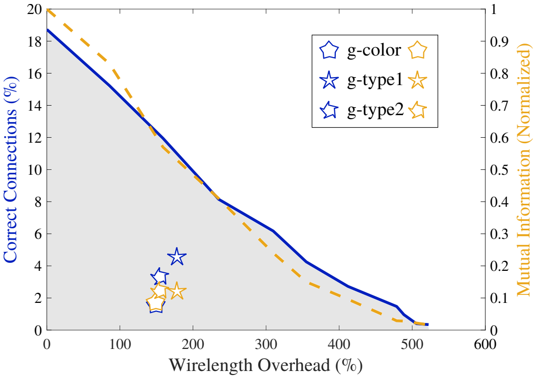


Fig. 4. Correct connections and mutual information versus wirelength overheads, when splitting *c7552* at M1. For layout randomization (represented by the blue line and the dark-yellow, dashed line), gates are randomly selected in steps of 10% and have their locations shuffled.

[2] Open pins/vias describe the open ends of "dangling wires" remaining in the FEOL after split manufacturing; see also [13].

[3] Note that this random selection of the first vertex allows us to obtain different versions of protected layouts for the same design.

The definitive Version of Record is published in Proc. International Conference On Computer Aided Design (ICCAD) 2017
http://dx.doi.org/10.1109/ICCAD.2017.8203796

Algorithm 1: Algorithm for g-color

```
Input  : Flattened netlist $N$
Output: Partitioned netlist $N'$
$G \leftarrow convertToDAG(N)$ /*convert $N$ to a directed acyclic graph $G$*/
$L \leftarrow getListOfVertices(G)$ /*parse the list of vertices*/
$C \leftarrow \phi$ /*initialize the set of colors*/
while $isNotEmpty(L)$ do
    $u \xleftarrow{\$} getNextVertex(L)$ /*pick start vertex randomly*/
    if $notColorable(u)$ then
        $C \leftarrow addNewColor()$ /*requires new color for $u$*/
    $u.color \leftarrow getMinColor(C)$ /*find the color with fewest cells and color $u$*/
    $L' \leftarrow getAdjacencyList(u)$ /*list neighbors of $u$*/
    while $isNotEmpty(L')$ do
        $v \leftarrow getNextVertex(L')$ /*color all neighbors of $u$*/
        if $notColorable(v)$ then
            $C \leftarrow addNewColor()$
        $v.color \leftarrow getMinColor(C)$
        $delete(v, L')$
        $delete(v, L)$
    $delete(u, L)$
$N' \leftarrow partitionByColor(N, C)$ /*partition the netlist according to color*/
return $N'$
```

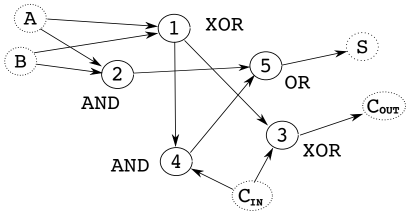
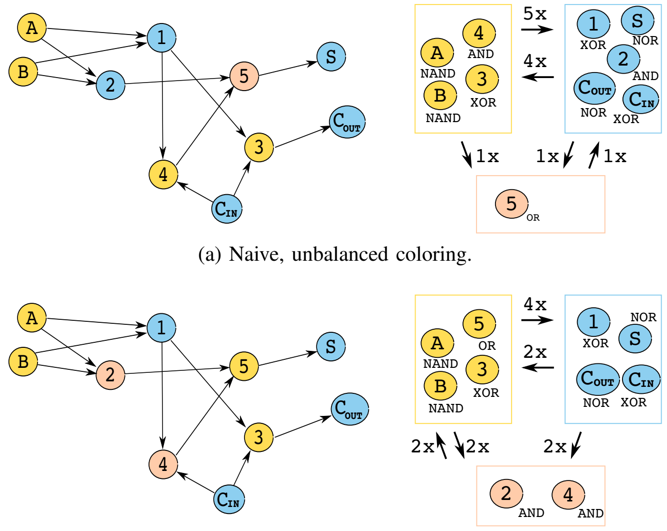


Fig. 5. Graph representation of a full-adder circuit.

together according to their colors as indicated by the encapsulating boxes in Fig. 6.

Initially, we observe that selecting the first available color for the next vertex may produce largely unbalanced clusters (Fig. 6a). In turn, this can cause the design tool to place the different types of partitions (small/large, little/largely interconnected with other partitions) in a manner which may leak information about the underlying connectivity of the gates. We thus adapt the algorithm to select the color corresponding to that with the so-far lowest number of associated cells, yielding more balanced partitions in practice (Fig. 6b).

So far, we select the same color for all the neighbors of a vertex v; all the neighbors/cells are consequently assigned to the same partition. However, as these neighbors/cells are all driven by the same cell (the vertex v), any layout tool seeks to place them in close proximity within their partition. Thus, we further adapt the algorithm to explicitly color all the neighbors differently, thereby "decoupling" cells from their driver.[4] The assignment of different colors to neighbors is streamlined with the balance-aware color selection; see Fig. 6b for an example.



(a) Naive, unbalanced coloring.

(b) Balanced coloring, also with different colors for neighbours.

Fig. 6. Applying g-color to a full adder, with the resulting design partitions and their system-level connectivity (right).

Algorithm 2: Algorithm for g-type

```
Input  : Flattened netlist $N$
Output: Partitioned netlist $N'$
$L \leftarrow parseNetlist(N)$ /*list the set of vertices*/
$H \leftarrow \phi$ /*initialize gate-types*/
while $isNotEmpty(L)$ do
    $u \leftarrow getNextVertex(L)$
    if $u.type ==$ BUF or INV then
        $H \xleftarrow{\$} u$ /*place $u$ uniformly randomly*/
    $H \leftarrow hash(u, u.type)$ /*partition $u$ according to its type*/
    $delete(u, L)$
$N' \leftarrow partitionByType(N, H)$ /*partition the netlist by gate-types*/
return $N'$
```

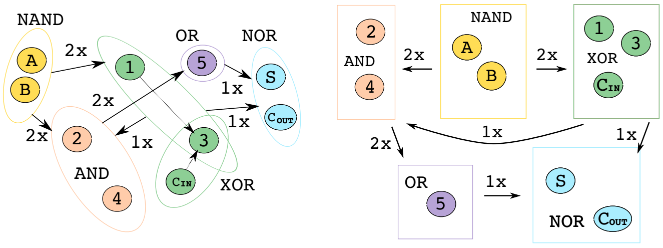


Fig. 7. Applying g-type to a full adder, with the partitions.

B. g-type

We observe empirically that the connectivity amongst the same type of gates is rather low; even in case where particular structures such as "AND trees" are present, they seldom dominate the overall design. Thus, our second approach (independent of g-color) is to cluster and partition all gates of the same type.

Our technique called g-type is outlined in Algorithm 2 and illustrated in Fig. 7. It comes in two flavors—we either consider (*i*) only the functionality of the gates (*g-type1*), or (*ii*) both the functionality of the gates as well as the number of their inputs (*g-type2*), e.g. here we do differentiate between a 2-input NAND gate and a 3-input NAND gate. The latter is motivated by our experimental results which indicate that utilizing more partitions renders a design more resilient against proximity attacks in practice. Note that we do not account for driving strengths during partitioning. Doing so would be superfluous since design tools scale up gates as needed (and/or insert buffers) during later stages.

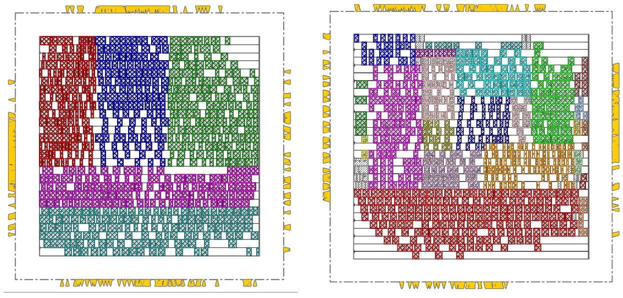
IV. Methodology

Here we describe the steps for the layout generation utilizing our secure techniques. The steps are generic and can be easily embedded into any design flow. As an example, two protected, cell-level layouts of *c7552* are shown in Fig. 8.

First, we obtain the gate-level, technology-mapped netlist of the design to protect (using the *Cadence RTL compiler* along with the *NanGate Open Cell Library* [20]). Next, we apply (one of) our proposed techniques on that netlist to obtain the related design partitions. Given these partitions, we generate a layout where all partitions are mapped to mutually exclusive layout regions called *fences*. A fence confines the corresponding partition's cell placement within its boundaries. This is an important step as it ensures that placement optimization cannot undermine the physical separation of cells dictated by partitions. The actual system-level arrangement of all fences—which can be considered as floorplanning—is done automatically using *Cadence Innovus*. Finally, the layout is routed

[4] Conceptually, we now realize coloring of a *hyper-graph* [19].

The definitive Version of Record is published in Proc. International Conference On Computer Aided Design (ICCAD) 2017
http://dx.doi.org/10.1109/ICCAD.2017.8203796



(a) g-color

(b) g-type2

Fig. 8. Protected layouts for *c7552*. Cells corresponding to particular clusters/partitions have identical colors.

and finalized. We like to emphasize the fact that we target for a *DRC-clean* layout by adapting the utilization target as needed. We resolve any outstanding DRC issues, if any, and report the power, performance, and area (PPA) numbers.

As the concept of split manufacturing hinges on two foundries, we split the DEF file into the FEOL and BEOL parts using a custom script. One aspect of our study is to investigate the cost-security implications at different split layers; we thus obtain multiple sets of FEOL and BEOL parts, for splitting from M1 up to M6.

V. Experimental Results

Setup: Our secure layout techniques are implemented using *Java OpenJDK 1.8.0_121 64-bit Server VM*. The layouts are generated using custom in-house scripts for *Cadence Innovus 15.1* using the *NanGate 45nm Open Cell Library* [20] with ten metal layers. Note that all metal layers are leveraged across all benchmarks for the sake of fair comparison. The PPA analysis is carried out at 0.95V for the slow process corner. We evaluate the resilience of our protected layouts against the network-flow attack by Wang *et al.* [1]. For the latter, we run experiments on an 8-core *Intel Xeon i7-4790 CPU*, at 3.60GHz and with 16GB RAM. The operating system is *Ubuntu 16.04.2 (xenial)*. We conduct our experiments on the *ISCAS'85* and *MCNC* benchmark suites (Table I). While all those benchmarks are fully combinatorial, our techniques can be readily applied to sequential circuits as well.

A. Security Analysis

Reduction in mutual information: The reduction in *MI* for our different techniques, when compared to the original layouts, is presented in Table II. As expected, random placement enables the largest reduction in *MI* and, thus, presumably the best protection. However, recall that this specific assessment is only applicable for splitting at M1. As for higher split layers, one should rather consider the distance between remaining open pins/vias. These distances will be shorter on average, due to design tools (routers) seeking to shorten interconnects wherever possible, thereby bringing the open pins/vias closer together (or even routing some of the nets already completely within the FEOL). Hence, the *MI* as calculated for splitting at M1 will become less expressive for higher layers.

TABLE I
Benchmarks Selected from the ISCAS-85 and MCNC Suites, Along with Their Properties

Benchmark	**Inputs**	**Outputs**	**Gate Count**
apex2	39	3	610
apex4	10	19	5,360
c432	36	7	160
c880	60	26	383
c1908	33	25	880
c2670	233	140	1,193
c5315	178	123	2,307
c7552	207	108	3,512
des	256	245	6,473
ex1010	10	10	5,066

TABLE II
Reduction in *MI* (in %) for the Proposed Techniques Compared to Original Layouts, When Split at M1

Benchmark	**Random**	**g-color**	**g-type1**	**g-type2**
apex2	96.11	75.00	89.44	92.22
apex4	96.67	90.00	96.67	93.33
c432	93.44	91.03	82.41	89.31
c880	96.84	88.42	86.84	89.47
c1908	95.00	79.29	85.71	85.71
c2670	97.22	85.56	89.44	94.44
c5315	98.00	92.00	94.00	92.00
c7552	98.89	91.11	90.00	88.89
des	98.25	92.5	90.00	90.00
ex1010	96.67	93.33	93.33	93.33
Avg.	**96.61**	**87.43**	**87.33**	**89.56**

The above expectation—random placement is most secure, at least while splitting at lower layers—is corroborated while conducting the attack [1] across various split layers (Fig. 9; see also below for further discussion). While random placement is the most secure technique at lower split layers, it becomes less and less effective for higher layers, until the point (at M6) where even the original, unprotected layouts are more resilient. In general, we observe the higher the split layer, the more connections are correctly inferred and, thus, the lower is the actual resilience.

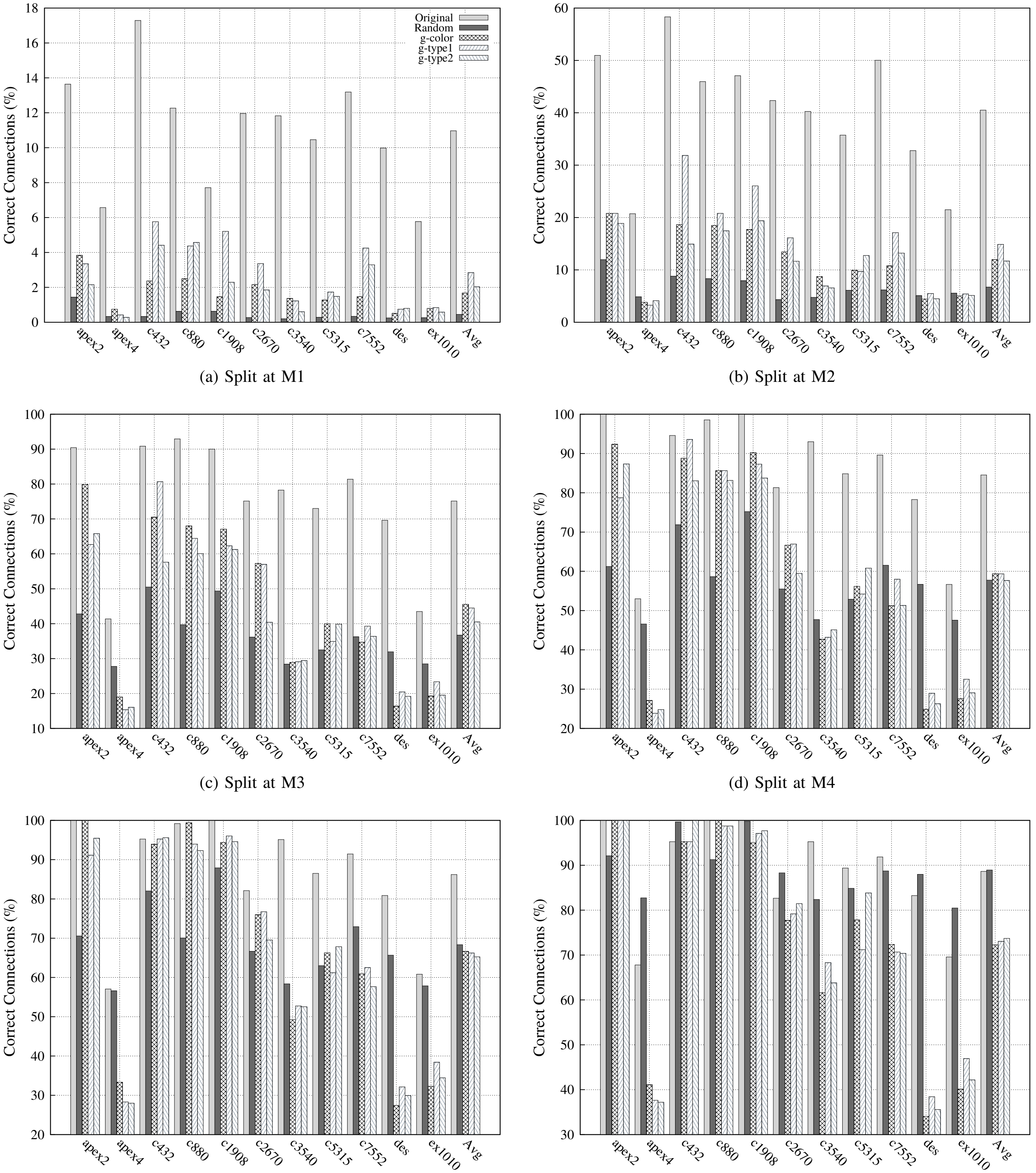
Resilience at M1: It is evident from Fig. 9a that only a few connections are recovered correctly across all benchmarks. Thus, the resilience when splitting at M1 is generally high. Still, as expected, we find that the original layouts are easiest to attack. This reiterates the fact that design tools shall be reinforced with the help of security metrics (such as *MI*) once split manufacturing is considered.

We observe that randomization enables the highest resilience. This is in agreement with our findings above, i.e., randomization achieves the largest reduction in *MI*. Again, this is expected as we "dissolve" the hints of connectivity by randomly perturbing the placement of all gates. Unfortunately, randomization comes at a hefty cost for PPA (up to 600%); see also Section V-B.

Our proposed partitioning techniques perform quite well for security; we are able to significantly reduce the percentage of correct connections when compared to original layouts. In fact, we observe on average reductions of 6.54×, 3.86×, and 5.41× for g-color, g-type1, and g-type2, respectively. As illustrated in Fig. 9a (and Fig. 4), we can achieve similar resilience when compared to randomization (with lower wirelength overheads at the same time).

Resilience at M2 and M3: As expected, the resilience generally decreases across all techniques and benchmarks when compared to M1 (Figs. 9b and 9c). Interestingly enough, the advances of our techniques still carry over to a great extent. On average, we reduce the correct connections by 2.73–3.47× and 1.64–1.85× while splitting at M2 and M3, respectively, as opposed to original layouts. Moreover, for relatively large benchmarks under consideration (i.e., *apex4*, *des*, and *ex1010*), our techniques are even on a par with randomization.

Resilience at M4 and above: Once we split at layer M4 or above, we still achieve average reductions by 1.4×, 1.3×, and 1.2× over original layouts (Figs. 9d, 9e, and 9f). For relatively large benchmarks (*apex4*, *des*, and *ex1010*), we even achieve reductions by 1.52–1.75×. Also note that our techniques are on average on a par with layout randomization for M4 and M5, and notably excel it for M6, by 1.2×. This clearly indicates that only thoughtful placement-centric protection schemes can imply some resilience also for higher split layers and relatively large benchmarks.



(a) Split at M1

(b) Split at M2

(c) Split at M3

(d) Split at M4

(e) Split at M5

(f) Split at M6

Fig. 9. Correct connections (representing the attacker's success rates) for original and varyingly protected layouts, evaluated against the attack [1] when split at different layers.

The definitive Version of Record is published in Proc. International Conference On Computer Aided Design (ICCAD) 2017
http://dx.doi.org/10.1109/ICCAD.2017.8203796

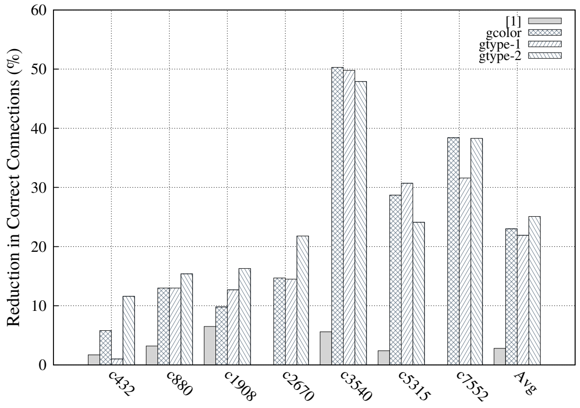


Fig. 10. Reduction in correct connections achieved for the protection scheme in [1] and our techniques, for splitting at M4. The data for [1] is quoted from their recent publication [13] which is also based on the same attack as in [1].

Comparison with Wang *et al.* [1]: We compare our work to the most recent in *placement-centric protection* by Wang *et al.* [1] in Fig. 10. Note that we compare for splitting at M4 since the layouts provided to us indicate this split layer. We lower the number of correct connections on average by 21.9–25.1% when compared to the original layouts—this is an improvement of ≈8× over [1].

Besides that, we cannot directly compare with other studies such as [13], [14]; these are *routing-centric protection* techniques and we are also not made aware of all essential details of their protected layouts (such as the technology files). However, the data presented in [13] indicates that our techniques are still competitive.

B. Layout-Level Cost Analysis

Area overheads: Recall that we adapt the utilization rates as needed to enable DRC-clean layouts; the reported area cost accordingly captures the effect of upscaling die outlines. While layout randomization enables the most resilient layouts on average, it incurs prohibitive overheads (Fig. 11). We note that area cost scales up significantly for relatively large benchmarks under consideration (i.e., *apex4*, *des*, and *ex1010*); however, these benchmarks are still decent in size when compared to state-of-the-art industrial designs. Hence, randomizing layouts is not scalable. In contrast, we observe that our techniques g-color and g-type1 induce on average 60% area cost. Applying g-type2, however, results in larger overhead, sometimes comparable to randomization. Since g-type2 induces on average more partitions, the system-level routing for those partitions becomes more challenging and congested, which can only be managed by larger die outlines. It is easy to see that routability poses a major challenge for any protection scheme "dissolving" the connectivity of gates and their placement. Naturally, a larger die outline also lengthens wires to some degree, which *may* also impact power and performance.

Power and performance overheads: As for layout randomization, both the power and delay overhead are prohibitive. Again, recall that randomization deliberately and uncontrollably "rips apart" connected gates. Even sophisticated design optimization in later stages (timing-driven routing, clock gating, etc., see also [19]) may handle the related overhead only to some degree.

As for our novel layout techniques, we obtain significantly lower overheads. We observe average power overheads of 50% across all benchmarks, which is an improvement of 1.6× over layout randomization. Further, we observe average delay overheads less than 18% in all the benchmarks under consideration; this translates to an improvement of 5× over randomization.

Comparison with Wang *et al.* [1]: While the respective layouts are available to us, we have not been made aware of the technology files and the specifics of the physical-design setup. Hence, we cannot reasonably contrast the PPA cost at the layout level. Further, PPA cost is also omitted in [1] itself. While the wirelength numbers reported in [1] may be lower, it is well known that wirelength is only one aspect among many others to impact PPA cost [21].

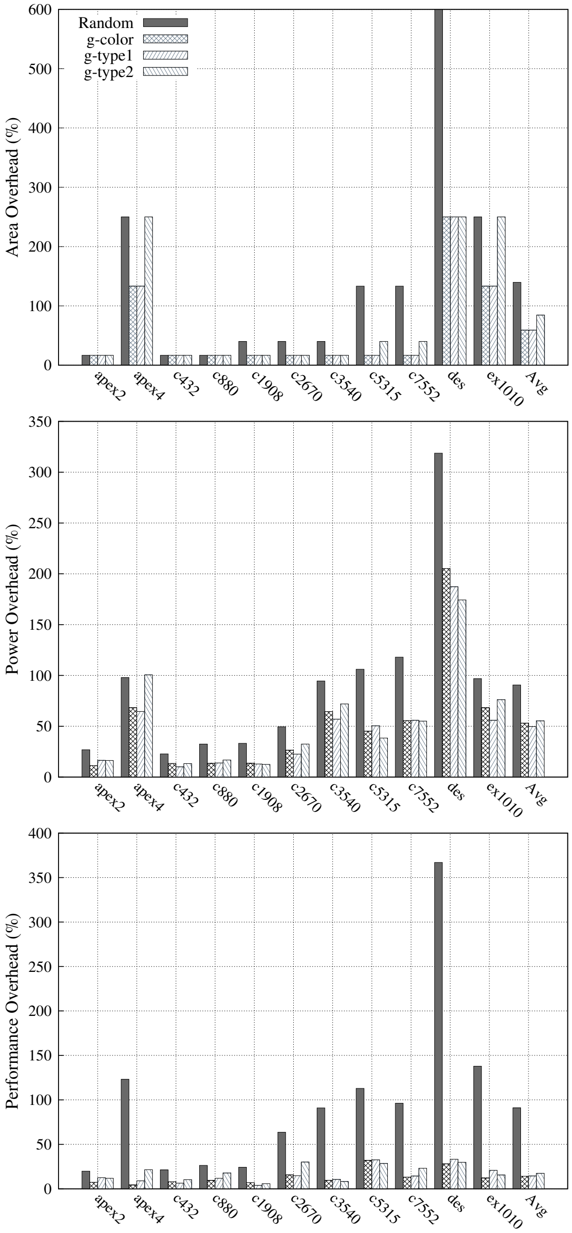


Fig. 11. Layout overheads for various techniques in contrast to the original, unprotected layouts.

C. Discussion

Impact of technology libraries: We also investigate the impact of different technology libraries on both security and cost when using our placement-centric techniques. For these experiments, we additionally synthesized the *c7552* circuit using a library constrained to the three essential gates: NAND, NOR, and inverters.

First, note that g-color is agnostic with respect to the library; the partitioning is solely dictated by connectivity. For g-type, however, there is an interdependency between the library and the final layout since partitions are based on gates of the same type. We observe a direct relation between protection and the number of partitions (i.e.,

The definitive Version of Record is published in Proc. International Conference On Computer Aided Design (ICCAD) 2017
http://dx.doi.org/10.1109/ICCAD.2017.8203796

TABLE III
CORRECT CONNECTIONS (IN %) FOR *c7552*, PROTECTED USING G-TYPE, AND FOR DIFFERENT LIBRARIES AND SPLIT LAYERS

Split layer	Full library		Constrained library	
	g-type1	g-type2	g-type1	g-type2
M1	4.3	3.3	16.1	13.9
M2	17.1	13.2	45.5	37.8
M3	39.3	36.4	59.5	53.7
M4	57.9	51.3	75.5	69.3
M5	62.5	57.7	76.9	71.4
M6	70.7	70.4	89.6	85.6

gates in the library): the fewer partitions, the lower the resilience (in terms of more correctly recovered connections, see Table III). Besides the adverse impact on security, it should be noted that such a significantly constrained library is not practical as it offers very little room for design optimization. In short, an enriched library not only offers significantly more room for design optimization, but it also enables higher resilience while using g-type.[5]

Trade-off for cost and security: We note that the layout resilience varies greatly across the different split layers and different benchmarks. While splitting at M1, M2, or even M3 still offers a reasonable protection also for relatively small benchmarks (which are easier to attack in general), splitting at M4 or above can only protect relatively large benchmarks in comparison. In general, splitting at higher layers implies lower commercial cost, since the trusted BEOL fab is then only required to handle few metal layers having relatively large pitches [6], [11].

Determining the design-specific "sweet spot" for cost and security is thus an essential challenge for split manufacturing. Towards this end, we advocate our metric (mutual information) as another design criteria for future, security-aware tools. Moreover, we like to emphasize the fact that only thoughtful placement-centric schemes like ours (and unlike layout randomization) can provide some degree of protection at higher split layers as well.

Towards better protection at higher split layers: While our techniques already provide comparable protection to randomization at lower layers and even translate to better protection at higher layers, we still observe the general trend of increasingly successful recovery once the attack targets at higher layers. Thus, an interesting question arising is whether one can further strengthen our placement-centric schemes also for higher layers. We believe that this requires applying both placement- and routing-centric techniques in conjunction; this will be the scope of future work.

VI. CONCLUSION

In this work, we first formulate an information-theoretic metric—the mutual information between the connectivity and distances of gates—which helps to analyze the protection of physical layouts against proximity attacks. Our metric can measure the security in an objective and efficient way, as compared to empirical and attack-based evaluation schemes. We show further that randomizing the layout/placement can reduce the mutual information, but only at an excessive overhead. Thus, we also present two effective, placement-centric techniques (namely, g-color and g-type) which enable competitive (sometimes even superior) protection along with an acceptable layout cost. For future work, we plan to extend our approach towards protection at both the FEOL *and* BEOL end.

ACKNOWLEDGMENTS

The authors are grateful to Tri Cao and Jeyavijayan (JV) Rajendran (University of Texas at Dallas) for providing their network-flow attack and their protected layouts of [1].

This work was supported in part by the National Science Foundation (NSF) under Grant 1553419, the Computing and Communication Foundations (NSF/CCF) under Grant 1319841, a grant from the Semiconductor Research Corporation (SRC), and the New York University/New York University Abu Dhabi (NYU/NYUAD) Center for Cyber Security (CCS). Any views expressed are the authors' own and do not necessarily reflect the views of the NSF or SRC.

REFERENCES

[1] Y. Wang, P. Chen, J. Hu, and J. J. Rajendran, "The cat and mouse in split manufacturing," in *Proc. Des. Autom. Conf.*, 2016, pp. 165:1–165:6.

[2] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[3] "iPhone 5 A6 SoC reverse engineered, reveals rare hand-made custom CPU, and tri-core GPU," 2012. [Online]. Available: http://www.extremetech.com/computing/136749-iphone-5-a6-soc-reverse-engineered-reveals-rare-hand-made-custom-cpu-and-a-tri-core-gpu

[4] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[5] "Innovation is at risk as semiconductor equipment and materials industry loses up to $4 billion annually due to IP infringement," 2008. [Online]. Available: http://www.marketwired.com/press-release/innovation-is-risk-as-semiconductor-equipment-materials-industry-loses-up-4-billion-850034.htm

[6] "Trusted integrated chips (TIC) program," Intelligence Advanced Research Projects Activity, 2011. [Online]. Available: https://www.fbo.gov/utils/view?id=b8be3d2c5d5babbdffc6975c370247a6

[7] K. Vaidyanathan *et al.*, "Building trusted ICs using split fabrication," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2014, pp. 1–6.

[8] ——, "Efficient and secure intellectual property (IP) design with split fabrication," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2014, pp. 13–18.

[9] K. Vaidyanathan, B. P. Das, and L. Pileggi, "Detecting reliability attacks during split fabrication using test-only BEOL stack," in *Proc. Des. Autom. Conf.*, 2014, pp. 1–6.

[10] C. T. O. Otero *et al.*, "Automatic obfuscated cell layout for trusted split-foundry design," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2015, pp. 56–61.

[11] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proc. Des. Autom. Test Europe*, 2013, pp. 1259–1264.

[12] M. Jagasivamani *et al.*, "Split-fabrication obfuscation: Metrics and techniques," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2014, pp. 7–12.

[13] Y. Wang, P. Chen, J. Hu, and J. Rajendran, "Routing perturbation for enhanced security in split manufacturing," in *Proc. Asia South P. Des. Autom. Conf.*, 2017, pp. 605–610.

[14] J. Magaña, D. Shi, and A. Davoodi, "Are proximity attacks a threat to the security of split manufacturing of integrated circuits?" in *Proc. Int. Conf. Comp.-Aided Des.*, 2016, pp. 90:1–90:7.

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

[16] B. Köpf and D. Basin, "An information-theoretic model for adaptive side-channel attacks," in *Proc. Comp. Comm. Sec.*, 2007, pp. 286–296.

[17] F.-X. Standaert, T. G. Malkin, and M. Yung, "A unified framework for the analysis of side-channel key recovery attacks," in *Proc. Eurocrypt*, 2009, pp. 443–461.

[18] D. B. West, *Introduction to Graph Theory*, 2nd ed. Prentice Hall, 2000.

[19] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer, 2011.

[20] "NanGate FreePDK45 Open Cell Library," 2011. [Online]. Available: http://www.nangate.com/?page_id=2325

[21] R. S. Shelar and M. Patyra, "Impact of local interconnects on timing and power in a high performance microprocessor," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, vol. 32, no. 10, pp. 1623–1627, 2013.

[5] Depending on both the design to protect and the library, there may be cases where only a few gates remain within a partition. This might undermine the resilience of g-type to some degree, as the arrangement of very small partitions may leak their underlying connectivity. As a countermeasure, these partitions could be balanced by adding dummy gates as needed. Note that this would also prevent leaking the functional composition of the design [10], [12].

The definitive Version of Record is published in Proc. International Conference On Computer Aided Design (ICCAD) 2017
http://dx.doi.org/10.1109/ICCAD.2017.8203796