Raise Your Game for Split Manufacturing: Restoring the True Functionality Through BEOL

Satwik Patnaik, Mohammed Ashraf, Johann Knechtel, and Ozgur Sinanoglu
Tandon School of Engineering, New York University, New York, USA
Division of Engineering, New York University Abu Dhabi, United Arab Emirates
{sp4012,ma199,johann,ozgursin}@nyu.edu

ABSTRACT

Split manufacturing (SM) seeks to protect against piracy of intellectual property (IP) in chip designs. Here we propose a scheme to manipulate both placement and routing in an intertwined manner, thereby increasing the resilience of SM layouts. Key stages of our scheme are to (partially) randomize a design, place and route the erroneous netlist, and restore the original design by re-routing the BEOL. Based on state-of-the-art proximity attacks, we demonstrate that our scheme notably excels over the prior art (i.e., 0% correct connection rates). Our scheme induces controllable PPA overheads and lowers commercial cost (the latter by splitting at higher layers).

1 INTRODUCTION

Proposed in 2011 by the IARPA agency, *split manufacturing (SM)* seeks to protect chip design companies against piracy of their intellectual property (IP) by third-party manufacturing facilities [1]. SM can also help to mitigate related threats such as insertion of hardware Trojans or unauthorized over-manufacturing [2].

In the most common threat model [3–9], the front-end-of-line (FEOL) is handled by an outsourced, high-end fab which is considered competitive but *untrustworthy*, whereas the back-end-of-line (BEOL) is subsequently manufactured (on top of the FEOL) at a *trusted* integration facility. Besides, Wang *et al.* [10] addressed another threat model where the BEOL facility is untrusted; here the adversaries seek to infer the gates from the whole BEOL stack (i.e., all wires/vias are available, except the intra-cell wiring in M1).

The security promise of SM is based on two assumptions: (*i*) third-party manufacturers do not have access to the complete design but only to either FEOL or BEOL and (*ii*) those third parties are not colluding. For the conventional threat model (untrusted FEOL fab), there is an additional risk: adversaries in the fab can leverage the physical implementation details of the FEOL layout. More specifically, since design automation tools optimize for power, performance, and area (PPA), the FEOL part itself contains various hints on the missing BEOL interconnects. Most importantly, gates to be connected are typically placed close to each other. This hint on proximity (among others such as delay constraints or routing paths) is leveraged by various *proximity attacks* [3, 5–7, 9], raising concerns regarding the security promise offered by SM.

In this work, we "raise the game" for SM to protect against malicious fab adversaries. The fundamental idea is to manipulate both placement and routing in an intertwined, holistic and misleading manner, thereby increasing the resilience of FEOL layouts. More specifically, we randomize the design at the netlist level, place and route the resulting erroneous netlist, and restore the true functionality only in the BEOL. This paper can be summarized as follows:

- We initially review state-of-the-art proximity attacks, related metrics, prior protection schemes, and associated shortcomings (Sec. 2). We also outline how our concept can mislead proximity attacks, to begin with, even for attacks that presumably achieve perfect scores (Sec. 3).
- We propose and implement (in *Cadence Innovus*) a protection scheme for SM (Sec. 4). Our scheme is based on holistic placement and routing perturbation in the FEOL and subsequent correction in the BEOL. The scheme allows for controllable impact on PPA. Moreover, we can limit the commercial cost of SM, by splitting after higher layers (e.g., after M6).
- We design custom *correction cells* for our scheme. These cells allow us to handle wire detours in a well-controlled manner, which is essential to (*i*) induce misleading placement and routing in the FEOL and (*ii*) restore the true functionality later on by re-routing in the BEOL.
- We evaluate our scheme in terms of PPA cost and security (Sec. 5). We consider various benchmarks, including the industrial *IBM superblue* benchmarks. We thoroughly contrast with prior art. We also make our DRC-clean protected layouts and SM scripts available to the community, along with the library definitions for the correction cells [11].

2 BACKGROUND AND MOTIVATION

Wang *et al.* [5] proposed an advanced proximity attack which utilizes multiple hints from the FEOL layouts: (*i*) physical proximity of gates, (*ii*) avoidance of combinatorial loops, (*iii*) constraints on load capacitances, (*iv*) direction of "dangling wires"[1], and (*v*) timing constraints. Magaña *et al.* [6, 7] proposed different attack schemes, whereupon they empirically observe that attacks considering routing paths/utilization are more effective than placement-centric attacks. They also observe that the *IBM superblue* suite is considerably more challenging to attack than "traditional," small-scale benchmarks. Note that their attacks do not recover actual netlists, but only list possible candidates for each net to reconnect.

Key attack metrics, as discussed in [3, 5, 12], are the *Hamming distance (HD)*, the *correct connection rate (CCR)*, and the *output error rate (OER)*. The HD quantifies the mismatch between the outputs of an original and the outputs of a recovered/stolen netlist during test stimulation. An HD of 0% (or 100%) denotes attack success. The

DAC '18, June 24–29, 2018, San Francisco, CA, USA

[1] There are metal segments left open/unconnected in the topmost FEOL layer, namely where the vias connecting upward to the BEOL are to be placed. These metal segments are referred to as "dangling wires." Moreover, we refer to the locations for those vias connecting with the BEOL as *virtual pins (vpins)*, as in [6, 7].

CCR is the ratio of successfully recovered nets over all protected nets. Hence, the higher the CCR, the more effective the attack. The OER reflects on the probability of some output bits being incorrect when stimulating the recovered/stolen netlist with test patterns. The routing-centric metrics proposed in [6, 7] gauge the solution space of SM. The *number of vpins* counts the overall vias/pins in the topmost FEOL layer (which are to be reconnected); the *candidate list size* is the average number of nets to consider for each vpin; and the *match in list* reflects for how many vpins the correct net is among those in the candidate list.[2] Thus, while the related attacks help to carefully confine the solution space of SM, they can only be considered as a complementary stage for other attacks.

Prior art [3, 5–10, 12] proposes different measures to render FEOL layouts resilient against proximity attacks. For example, Wang *et al.* [5] as well as Sengupta *et al.* [8] perturb the placement of gates. However, in [5, 8] it has been shown that splitting after higher layers—which is essential to limit the commercial cost of SM [13]—can undermine the protection. Interestingly, to some degree, this even holds true when layout randomization is applied [8]. Such limitation of placement-centric schemes is due to the fact that any placement perturbation is eventually resolved by routing.

Routing-centric protection schemes like those in [3, 6, 7, 9, 12] are typically post-processing the original layouts and thus subject to constraints in routing resources and PPA budgets. Hence, such schemes can be limited to relatively few and/or short wiring detours, which may be easy to attack. For example, Rajendran *et al.* [3] propose to swap the pins of IP modules and reroute them to mislead an attacker. Since the related perturbations only cover the system-level interconnects, this scheme (*i*) cannot protect against gate-level IP piracy and (*ii*) imposes a relatively small solution space for the attacker. As reported in [3] itself, on average 87% of the connections can still be recovered. Besides, enforcing routing detours in the BEOL requires customizing the design flow, which itself can be challenging. For example, the schemes in [6, 7] only allow for implicit detours, namely by inserting routing blockages.

3 OUR CONCEPT (UNDER ATTACK)

Unlike most prior art which manipulates the placement and/or routing at the layout level in a post-processing manner, our concept targets on the netlist itself—namely by partially randomizing it. This helps us retain the misleading modifications throughout any regular design flow, thereby obtaining more resilient FEOL layouts. These layouts are corrected only in the BEOL (Fig. 1). Next, we outline how our concept misleads state-of-the-art attack schemes. We confirm our intuition in Sec. 5.2, where we report on superior values for key security metrics.

Any attack—even when perfect recovery rates are presumed—is bound to observe logic and timing paths of the erroneous netlist. Regarding the attack proposed by Wang *et al.* [5], the physical

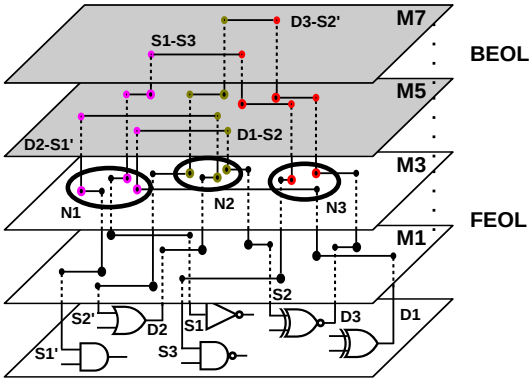

Figure 1: The original netlist is randomized, to N1 (D1 driving S1 and S1'), N2, etc. We place and route the erroneous design, resulting in a FEOL layout which is misleading regarding both placement and routing. Next, we restore the true functionality through the BEOL by rerouting; now D1 is driving S2, D2 is driving S1', etc. (The driver for sinks S1 and S3 is not illustrated in this conceptional figure.)

proximity of gates, the load and timing constraints, as well as the direction of dangling wires are all capturing the erroneous netlist, not the original one. For example, a large buffer such as *BUFX8* typically hints that its sink(s) is/are relatively far away. In the original netlist, however, this buffer may actually drive some nearby sink(s). As for the routing-centric attack in [6], we note that randomizing the netlist helps to enlarge the solution space significantly when compared to the original layouts (and even when compared to *naive lifting*, Sec. 5.2). This will naturally hinder any subsequent attacks (again, which are yet to be demonstrated).

We shall assume that the attacker knows the principle of our protection scheme; hence she/he expects misleading FEOL layouts. Without the BEOL disclosed to her/him, however, a naive attacker can arguably only resort to brute-force, i.e., enumerating all possible netlists, which is computationally prohibitive (Sec. 2). A more sophisticated attacker may seek to exclude "unreasonable" logic and timing paths arising due to randomization. Doing so, however, is subject to (*i*) significant experience on layout design, (*ii*) the size and the (yet unknown) scope of the original netlist, and (*iii*) the "degree of unreasonableness" of paths, which is random. We believe that such attacks are an open but interesting challenge.

4 METHODOLOGY

Our methodology is implemented as an extension to *Cadence Innovus* with custom in-house scripts and library customization. The steps can be summarized as follows: we (*i*) randomize the netlist, (*ii*) place and route the erroneous and misleading netlist, and (*iii*) restore the true functionality by re-routing in the BEOL (Fig. 2).

Next, we provide some details. For (*i*), we iteratively randomize the netlist by swapping the connectivity between randomly selected pairs of drivers and their sinks.[3] While doing so, we ensure that no combinatorial loops arise in the modified netlist by any of the random swaps—loops would help an attacker to identify those modifications [5]. We perform swapping until the OER approaches

[2] Consider the following example, where an attacker observes 1,000 vpins. In accordance with [6], also consider that only two-pin nets are available in the design, i.e., 500 two-pin nets are to be reconnected. The size of the solution space covering all possible netlists is the number of perfect matchings in a complete bipartite graph (representing the 500 drivers and 500 sinks), which is simply $500! = 1.22 \times 10^{1143}$. After conducting the routing-centric attacks, assuming the best possible *match in list* (i.e., 100%) and a *candidate list size* of 1.4 on average, there are at most "only" $\approx 1.4^{500} = 1.16 \times 10^{73}$ possible netlists remaining. (This number is coincidentally approaching the estimated number of atoms in the universe.) It is important to note that for any match in list below 100%, the true netlist is not even covered by that large number.

[3] In case the netlist imposes some alignment constraints, e.g., for datapaths, the related gates have to be ignored for randomization.

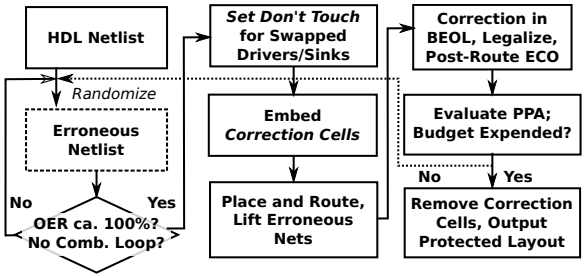

Figure 2: The flow of our protection scheme.

100%, which means that the modified netlist will induce some errors for any input. We also keep track of the original connectivity and the swapped drivers/sinks. For (*ii*), the randomized netlist is loaded into *Cadence Innovus*. Initially, the swapped drivers/sinks are marked as *do not touch* to avoid logic restructuring/removal of the related nets. The netlist is then placed and optimized for timing, power, and congestion. Before routing, the nets connecting the swapped drivers/sinks are prepared for lifting to M6 (or M8) with the help of customized *correction cells*. Note that these correction cells are not impacting the FEOL layout; their scope is lifting and correction of nets in the BEOL (see also below). Next, the design is placed and routed again in *ECO* mode, to implement lifting of the swapped nets. For (*iii*), the true connectivity is restored in the BEOL with the help of the correction cells and the tracked original connectivity. The design is rerouted and taken through *postRoute* optimization to improve timing and resolve DRC issues. In case the PPA budget is not expended yet, we repeat the steps to introduce more randomization. Otherwise, we remove the correction cells, and export DEF/Verilog files for further layout/security analysis.

Key points for the physical design of *correction cells* are given next. We provide our cell implementation on top of the *Nangate 45nm* library in [11]. Figure 3 illustrates our cells.

- The correction cells are modeled as 2-input-2-output OR gates; C, D are input pins, and Y, Z are output pins.
- There are four possible *arcs* (C to Y, C to Z, D to Y, and D to Z). The arc C to Z is used to implement the erroneous netlist during initial place and route. When restoring the true functionality, the arcs C to Z and D to Y are disabled (*set_disable_timing*) so that only true paths are considered for proper timing and power optimization and evaluation.
- All pins are set up in a higher metal layer (here M6 or M8) to enable lifting and routing of wires in the BEOL. The dimensions and offsets for the pins are chosen such that they can be placed onto the tracks of the respective metal layers—this helps to minimize the routing congestion.
- The correction cells can freely overlap with standard cells. That is because standard cells have their pins exclusively in lower metal layers, whereas correction cells neither impact those layers nor the device layer. We implement custom legalization scripts accordingly. These scripts further prevent different correction cells from overlapping with each other.
- The buffer cell *BUFX2* is leveraged for power and timing characteristics for the correction cells. We can refrain from detailed library characterization since the correction cells only implement some BEOL wires.

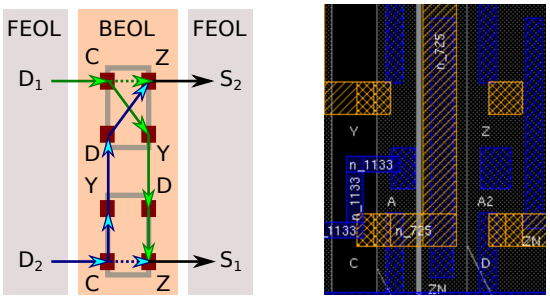

Figure 3: Our correction cells, in conceptional view (a) and layout view (b). For (a), dashed arrows indicate the misleading arcs for initial place and route, whereas regular arrows represent the true paths implemented later by re-routing. It is important to note that re-routing is always between pairs of correction cells. In (b), a correction cell with its pins in M6 is seen overlapping with an inverter (pins in M1).

- To enable proper ECO optimization, the correction cells are set up for load annotation at design time. That is required to capture the capacitive loads of (*i*) the wires running from the correction cells to the sinks and (*ii*) the sinks themselves.

It is important to note that re-routing (to restore the original netlist) is always between pairs of correction cells, not only within one cell. This implies that an attacker may know the scheme of correction cells (i.e., true paths are between C and Y and between D and Z), but she/he cannot derive the original netlist from that knowledge alone. Instead, she/he has to identify the correct pairs of cells, which is hampered by two facts: (*i*) retracing the BEOL-centric correction cells in the FEOL is challenging, and even when the attacker succeeds, then (*ii*) the distances between correct pairs of cells are randomized, based on the erroneous netlist being placed and routed. For (*ii*), recall that this misleads any state-of-the-art proximity attack to begin with (Sec. 3). For (*i*), note that the pins of correction cells are in higher layers (M6 or M8) whereas the layout is split after lower layers, necessitating some wiring paths in between. Hence, the dangling wires related to the correction cells are unlikely to be as distinct as in Fig. 3(b) but rather spread out.

5 RESULTS

5.1 Experimental Setup

Test cases: We evaluate our proposed defense on 12 benchmarks, seven from the *ISCAS-85* suite and five from the industrial *IBM superblue* suite [14]. We convert the *superblue* benchmarks (initially defined in *Bookshelf* format) to *Verilog* files using scripts from [15].

Setup for layout evaluation: Our techniques are implemented for *Cadence Innovus 16.15* using custom in-house *TCL* scripts, which impose negligible runtime overheads. We leverage the *Nangate 45nm Open Cell Library* [16] with ten metal layers. Correction cells are set up in M6 for *ISCAS-85* and in M8 for *superblue* benchmarks. Conservative PPA analysis is carried out for the slow process corner and a supply voltage of 0.95V. We ensure that all layouts are free of congestion by choosing appropriate utilization rates. We allow PPA budgets of 20% for *ISCAS-85* and 5% for *superblue* benchmarks.

Setup for security evaluation: In line with the prior art, we assume that the attacker has access to the FEOL layout and the

technology libraries, but she/he cannot access working chips yet to be manufactured. We utilize the network-flow attack [5] for *ISCAS-85* benchmarks and the routing-based attack *crouting* [6] for *superblue* benchmarks.[4] The OER and HD are computed using *Synopsys VCS* upon applying 1,000,000 test patterns for each netlist; functional equivalence is validated using *Synopsys Formality*.

Comparative study: Besides the *correction cells*, we also implement another set of custom cells, called *naive lifting cells*. These cells are implementing the same principle of lifting wires, but without inducing erroneous connections. We apply these cells using our flow on original layouts to obtain a baseline called *naive lifting*.

The protected layouts of [5, 12] have been made available to us as DEF files. Since there are no definite indications of the split layer in the respective publications, we average the security metrics (CCR, OER, and HD) for splitting after layers M3, M4, and M5 respectively.

Open source: We make our *correction cells* and *naive lifting cells* available to the community, along with the protected layouts, in [11]. We also provide our DEF splitting and conversion script.

5.2 Security Evaluation

Protection of the placement: Recall that our scheme is based on randomly modified netlists, leading to erroneous FEOL layouts which are corrected only in the BEOL. It is intuitive that the distances of *truly* connected gates will be randomly distributed, thereby misleading proximity attacks. In fact, we achieve a significant increase of the distances along with a widely varied distribution (Fig. 4 and Table 1). This finding is corroborated by the superior resilience for *ISCAS-85* benchmarks (Tables 4 and 5); see further below for a comparative study on placement protection.

Protection of the routing: Recall that placement-centric protection schemes are offset by routing, especially once splitting is conducted after higher metal layers (Sec. 2), rendering routing-centric schemes more promising in that context. Figure 5 contrasts the contribution of each metal layer towards the wirelength for *superblue* benchmarks. For original layouts, the majority of wiring is found in the lower metal layers which provides significant leverage for an attacker. As for naively lifted layouts, the wiring is more evenly distributed across the metal layers which *may* help to protect the layouts. However, it is important to note that naive lifting cannot help dissolve the distances between connected gates (Fig. 4). In contrast, our scheme holds the majority of wiring in the higher layers *and* dissolves the true connectivity in the FEOL, offering less leverage for an attacker *and* enabling splits after higher layers.

We observe that our scheme is significantly more effective than naive lifting in terms of increasing vias/vpins in higher layers (Table 2). Additional vias/vpins induce more nets in the BEOL, rendering routing-centric attacks more challenging. For example, taking M5 as the split layer, our scheme increases the vias V56 by 30.65% on average when compared to naive lifting. We observe accordingly that the *crouting* attack [6] is impaired by larger lists of candidates and more vpins (Table 3). Recall that even seemingly small increases in those metrics imply a large-scale, polynomial increase of complexity—our scheme increases both metrics compared to original as well as lifted layouts.

[4]For the latter, note that the advanced attack as proposed by Magaña *et al.* in [7] has not been available to us at the time of writing. Also, note that we have to split the DEF files obtained by *Cadence Innovus* and convert them to *.rt/.out* files using custom scripts [11], as the scripts provided by Magaña *et al.* are tailored for academic routers.

Table 1: Distances between connected gates (in microns).

Benchmark	*Layout*	Mean	Median	Std. Dev.
superblue1	***Original***	14.31	2.85	54.84
	Lifted	14.37	2.92	54.83
	Proposed	198.46	48.41	318.88
superblue5	***Original***	14.38	2.99	49.16
	Lifted	14.39	2.99	49.17
	Proposed	244.73	96.9	328.84
superblue10	***Original***	12.66	2.73	49.59
	Lifted	12.71	2.8	49.58
	Proposed	254.06	71.03	372.07
superblue12	***Original***	19.06	3.18	75.37
	Lifted	19.08	3.23	75.37
	Proposed	263.21	81.28	395.26
superblue18	***Original***	12.91	2.54	41.74
	Lifted	12.93	2.54	41.74
	Proposed	208.47	119.51	244.81

In short, our scheme (*i*) keeps the major share of wirelength in the BEOL (Fig. 5) and (*ii*) significantly increase the via counts (Table 2), all while inducing misleading routing in the FEOL.

Comparison with the prior art: We contrast the resilience of our scheme against various prior art in Tables 4 and 5. As expected, the original layouts are most prone to proximity attacks: on average 94% CCR and 7% HD can be achieved when running the attack of [5]. Selective gate-level placement perturbation as proposed in [5] offers only a marginal improvement over unprotected layouts, with an attacker making 92% correct connections and experiencing 15% HD. Sengupta *et al.* [8] proposed four different protection strategies, and while the CCR is reduced to 63% on average, it should be noted that those techniques can become impractical to protect larger designs (in terms of excessive PPA overheads). Swapping of block pins as proposed in [3] is also limited; the attacker can still correctly infer 87% of the missing system-level interconnects. The scheme proposed by Wang *et al.* [12] reduces the CCR to about 72%. More recently, Feng *et al.* [9] proposed a routing-based scheme which can reduce the CCR significantly to about 21%.

Our scheme offers the best protection as it can reduce the CCR to the ideal value of 0%—none of the nets randomized in the FEOL are correctly inferred by an attacker. We attribute this superior result to the holistic mitigation of placement *and* routing hints. Besides CCR, our scheme also achieves an OER of ≈100% (only [12] reports similar numbers). Finally, our average HD is 40.4% which is another significant improvement over the prior art.

Besides the *crouting* attack, Magaña *et al.* [6, 7] also proposed routing-centric protection schemes. As discussed, a key metric for their evaluation is the number of vias/vpins. In Table 6, we compare their most recent results [7] with ours, whereas we set up the *correction cells* for lifting wires to M8. Since our scheme increases the via count in those higher layers to a much larger degree, we believe that it is more resilient than that of Magaña *et al.*[5]

5.3 Layout Evaluation

We estimate the area cost with regard to die outlines. That is because our correction cells do not impact the device layer, but they mandate re-routing which may require larger outlines to maintain DRC-clean layouts. In practice, however, we obtain no area cost at all (Fig. 6).

[5]Recall that the attacks in [6, 7] do not provide actual netlists, hence we cannot compare for related metrics such as CCR, OER, and HD. Besides, note that Magaña *et al.* [7] cautioned on wire lifting, since routers may be misguided by lifting which, in turn, can impact PPA. Our scheme exhibits reasonable PPA cost despite lifting (Sec. 5.3).

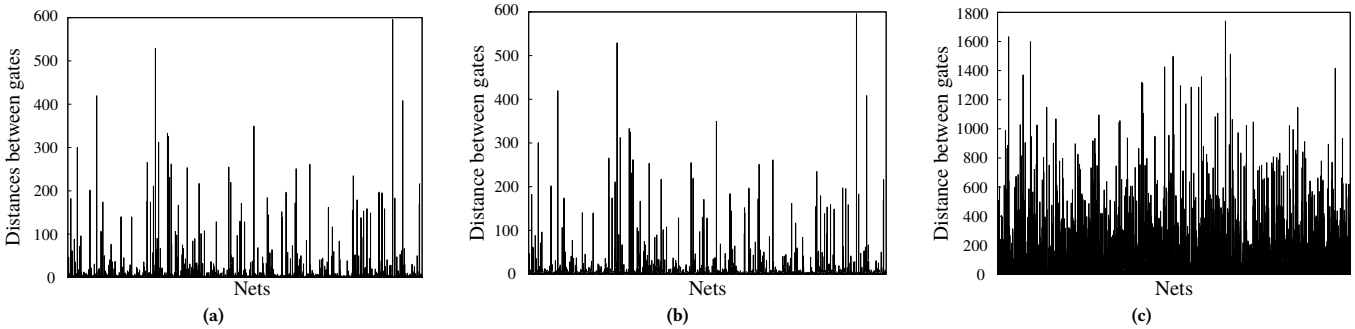

Figure 4: Distances between drivers/sinks for original (a), naively lifted (b), and our layouts (c), for *superblue18* benchmark.

Table 2: Comparison of additional vias for naively lifted layouts and our proposed scheme over original *IBM superblue* layouts. For a fair comparison, we randomize the same set of nets. We ensure zero die-area overhead and all layouts are DRC-clean.

Benchmark	Nets	I/O Pins	Util.	*Layout*	V12	V23	V34	V45	V56	V67	V78	V89	V910	Total Vias
superblue1	873,712	8,320/13,025	69	***Original***	3,016,748	2,334,923	664,292	239,550	170,423	82,762	56,170	34,164	16,249	6,615,281
				Lifted (%)	0.10	0.56	1.29	2.44	3.45	3.28	1.69	0.68	0.37	0.61
				Proposed (%)	2.1	4.13	10.82	18.38	29.86	31.79	34.2	27.3	40.93	5.87
superblue5	754,907	11,661/9,617	77	***Original***	2,430,541	1,866,252	553,843	217,394	157,046	75,306	50,970	30,714	15,227	5,397,293
				Lifted (%)	0.1	0.8	1.8	3.3	4.9	5.0	2.4	1.2	0.7	0.9
				Proposed (%)	3.2	7.3	12.9	23.9	40.0	55.1	59.5	51.3	67.6	9.2
superblue10	1,147,401	10,454/23,663	75	***Original***	3,871,474	3,048,375	875,305	329,549	238,533	111,507	76,885	45,408	22,721	8,619,757
				Lifted (%)	0.04	0.49	1.11	2.11	3.02	3.18	1.49	0.53	0.24	0.52
				Proposed (%)	2.06	6.29	12.43	22.92	32.27	55.22	57.16	59.29	69.74	7.90
superblue12	1,520,046	1,936/4,629	56	***Original***	5,368,332	3,995,438	1,130,079	445,635	316,038	141,141	100,358	55,097	31,301	11,583,419
				Lifted (%)	0.03	0.16	0.42	0.87	1.33	1.37	0.58	0.31	0.2	0.2
				Proposed (%)	1.59	6.99	19.09	30.56	30.19	34.67	22.92	30.93	-1.19	7.78
superblue18	670,323	3,921/7,465	67	***Original***	2,298,823	1,686,525	480,099	179,088	121,277	51,187	28,950	18,345	4,319	4,868,613
				Lifted (%)	0.05	0.55	1.56	3.69	5.62	5.82	3.10	0.72	1.09	0.73
				Proposed (%)	1.73	5.98	10.50	20.03	39.24	61.11	90.08	71.08	287.84	7.34

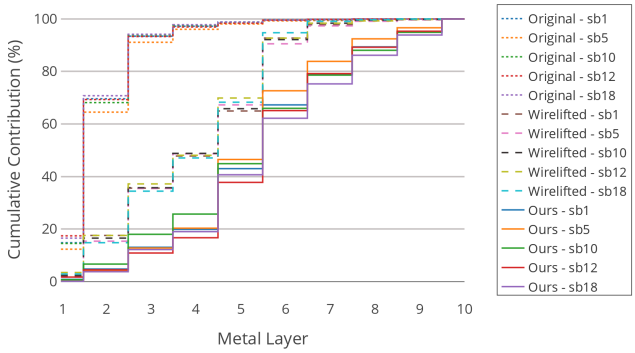

Figure 5: Contribution of the metal layers to wirelength for randomized nets in *superblue* benchmarks.

Since nets are lifted to BEOL layers (M6/M8) and rerouted, we naturally observe some increases in wirelength. These overheads, however, translate only to some degree to power and delay cost. On average, 11.5% and 10% power and delay cost arise for the *ISCAS-85* benchmarks. For the *superblue* benchmarks, average overheads are 3.5% and 2.7% for power and delay, respectively. We found that these overheads are 3.4% and 2.6% higher than those induced for *naive lifting*. In this context, recall that our scheme outperforms naive lifting as well as prior protection schemes in terms of security.

Note that most prior studies do not report on detailed layout evaluation and PPA results. In a recent study by Sengupta *et al.* [8], however, the authors report on PPA results for their protection scheme. We contrast their results with ours in Fig. 6; our scheme induces on average lower overheads on all area, power, and delay.

Table 3: Results for *crouting* attack [6]. Comparison of vpins and candidate list size (E[LS]) as a function of bounding box. N/A denotes attack failures.

Benchmark	*Layout*	#VPins	E[LS] for Bounding Box		
			15	30	45
superblue1	***Original***	73,110	4.63	13.25	23.46
	Lifted	73,810	4.65	13.27	23.47
	Proposed	75,754	4.69	13.46	23.83
superblue5	***Original***	67,194	4.86	13.99	24.87
	Lifted	67,676	4.85	13.9	24.73
	Proposed	N/A	N/A	N/A	N/A
superblue10	***Original***	155,180	5.05	14.54	25.75
	Lifted	N/A	N/A	N/A	N/A
	Proposed	157,106	4.88	14.1	25.07
superblue12	***Original***	127,112	4.84	13.85	24.45
	Lifted	127,610	4.83	13.79	24.35
	Proposed	165,106	6.29	17.95	32.04
superblue18	***Original***	50,026	3.76	10.86	19.17
	Lifted	51,970	3.87	11.09	19.54
	Proposed	54,154	4.26	12.22	21.74

6 CONCLUSION

Multiple studies recently questioned the security of split manufacturing. In this work, we raise the designer's game to protect against malicious FEOL parties. Our idea is to randomize the functionality and connectivity in the netlist, place and route that misleading design, and restore the true functionality only through the BEOL.

We contrast our scheme to recent state-of-the-art defense techniques, while leveraging both placement- and routing-centric attacks. We observe that our protected layouts are significantly more resilient while exhibiting reasonable PPA cost, even on large-scale industrial benchmarks. Another contribution in our scheme is that we readily support splitting after higher layers (e.g., after M6),

Table 4: Comparison with placement perturbation schemes. The metrics for our scheme are averaged for splitting after M3, M4, and M5; the metrics for the prior art are quoted. All values are in percentage. The attacks are based on [5].

Benchmark	Original Layout			Placement Perturbation [5]			Placement Perturbation [8]				Proposed		
	CCR	OER	HD	CCR	OER	HD	Random CCR	G-Color CCR	G-Type1 CCR	G-Type2 CCR	CCR	OER	HD
c432	92.4	75.4	23.4	90.7	98.8	41.8	68.1	84.4	89.8	78.8	0	99.9	48.4
c880	100	0	0	96.8	15.8	1.2	56.1	84.3	81.4	78.5	0	99.9	43.4
c1355	95.4	59.5	2.4	93.2	94.5	8	N/A	N/A	N/A	N/A	0	99.9	40.1
c1908	97.5	52.3	4.3	91	97.8	17.7	70.8	83.9	81.9	79.9	0	99.9	46.2
c2670	86.3	99.9	7	86.3	100	7.5	52.8	66.6	66.9	56.5	0	99.9	39.8
c3540	88.2	95.4	18.2	82.6	98.8	27.9	44.8	40.3	41.7	42.4	0	99.9	47.9
c5315	93.5	98.7	4.3	91.1	98.7	12.5	49.5	54.1	50.1	56.2	0	99.9	38.3
c6288	97.8	36.8	3	97.6	74.2	16.5	N/A	N/A	N/A	N/A	0	99.9	31.6
c7552	97.8	69.5	1.6	97.9	81.7	3.1	56.9	48.9	53.3	48.5	0	99.9	27.8
Average	94.3	65.3	7.1	91.9	84.5	15.1	57.0	66.1	66.4	62.9	0	99.9	40.4

Table 5: Comparison with routing perturbation schemes. The setup is the same as in Table 4.

Benchmark	Original Layout			Pin Swapping [3]			Routing Perturbation [12]			Synergistic SM [9]			Proposed		
	CCR	OER	HD	CCR	OER	HD	CCR	OER	HD	CCR	OER	HD	CCR	OER	HD
c432	92.4	75.4	23.4	92.5	N/A	39.8	78.8	99.4	46.1	N/A	N/A	N/A	0	99.9	48.4
c880	100	0	0	85	N/A	26	47.5	99.9	18	N/A	N/A	N/A	0	99.9	43.4
c1355	95.4	59.5	2.4	86	N/A	40	77.1	100	26.6	N/A	N/A	N/A	0	99.9	40.1
c1908	97.5	52.3	4.3	86.2	N/A	25	83.8	100	38.8	N/A	N/A	N/A	0	99.9	46.2
c2670	86.3	99.9	7	N/A	N/A	N/A	58.3	100	14	33.3	N/A	20.5	0	99.9	39.8
c3540	88.2	95.4	18.2	83.5	N/A	50	77	100	36.1	11.5	N/A	35	0	99.9	47.9
c5315	93.5	98.7	4.3	92.5	N/A	41	74.7	100	18.1	14.9	N/A	23.6	0	99.9	38.3
c6288	97.8	36.8	3	N/A	N/A	N/A	80.9	100	42.1	33.1	N/A	40.6	0	99.9	31.6
c7552	97.8	69.5	1.6	91	N/A	48	73.9	100	20.3	21.3	N/A	24.7	0	99.9	27.8
Average	94.3	65.3	7.1	88.1	N/A	33.4	72.4	99.9	28.9	20.8	N/A	28.9	0	99.9	40.4

Table 6: Comparison with [7] w.r.t. additional via count. Layouts are split after M6 and true connectivity restored in M8.

Benchmark	Routing Blockage [7]		Proposed Scheme	
	Δ_+V67 (%)	Δ_+V78 (%)	Δ_+V67 (%)	Δ_+V78 (%)
superblue1	23.28	65.07	36.32	49.22
superblue5	12.74	24.01	55.12	59.47
superblue10	64.85	84.09	62.09	73.12
superblue12	16.99	35.59	79.34	70.59
superblue18	24.73	58.66	61.87	124.16
Average	28.52	53.48	58.95	75.31

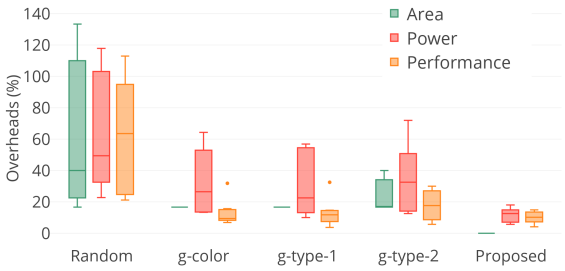

Figure 6: Comparison with [8] on *ISCAS-85* benchmarks.

thereby limiting commercial cost. In future work, we seek to protect against further threats such as insertion of hardware Trojans and to formulate strategies towards "provably secure SM."

ACKNOWLEDGEMENTS

The authors are grateful to Jeyavijayan (JV) Rajendran (Texas A&M) for providing the network-flow attack and protected layouts of [5, 12]. This work was supported in part by the Center for Cyber Security (CCS) at NYU New York/Abu Dhabi (NYU/NYUAD).

REFERENCES

[1] C. McCants, "Trusted integrated chips (TIC)," Intelligence Advanced Research Projects Activity (IARPA), 2011. [Online]. Available: https://www.iarpa.gov/index.php/research-programs/tic
[2] D. Forte, S. Bhunia, and M. M. Tehranipoor, Eds., *Hardware Protection through Obfuscation.* Springer, 2017.
[3] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proc. Des. Autom. Test Europe*, 2013, pp. 1259–1264.
[4] B. Hill *et al.*, "A split-foundry asynchronous FPGA," in *Proc. Cust. Integ. Circ. Conf.*, 2013, pp. 1–4.
[5] 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.
[6] 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.
[7] J. Magaña, D. Shi, J. Melchert, and A. Davoodi, "Are proximity attacks a threat to the security of split manufacturing of integrated circuits?" *Trans. VLSI Syst.*, vol. 25, no. 12, pp. 3406–3419, 2017.
[8] A. Sengupta *et al.*, "Rethinking split manufacturing: An information-theoretic approach with secure layout techniques," in *Proc. Int. Conf. Comp.-Aided Des.*, 2017, pp. 329–336.
[9] L. Feng *et al.*, "Making split fabrication synergistically secure and manufacturable," in *Proc. Int. Conf. Comp.-Aided Des.*, 2017, pp. 313–320.
[10] Y. Wang, T. Cao, J. Hu, and J. J. Rajendran, "Front-end-of-line attacks in split manufacturing," in *Proc. Int. Conf. Comp.-Aided Des.*, 2017, pp. 321–328.
[11] (2018) DfX Lab, NYUAD. [Online]. Available: http://sites.nyuad.nyu.edu/dfx/research-topics/design-for-trust-split-manufacturing/
[12] Y. Wang, P. Chen, J. Hu, and J. Rajendran, "Routing perturbation for enhanced security in split manufacturing," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2017, pp. 605–610.
[13] K. Xiao, D. Forte, and M. M. Tehranipoor, "Efficient and secure split manufacturing via obfuscated built-in self-authentication," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2015, pp. 14–19.
[14] N. Viswanathan *et al.*, "The ISPD-2011 routability-driven placement contest and benchmark suite," in *Proc. Int. Symp. Phys. Des.*, 2011, pp. 141–146.
[15] A. B. Kahng, H. Lee, and J. Li, "Horizontal benchmark extension for improved assessment of physical CAD research," in *Proc. Great Lakes Symp. VLSI*, 2014, pp. 27–32. [Online]. Available: http://vlsicad.ucsd.edu/A2A/
[16] (2011) NanGate FreePDK45 Open Cell Library. Nangate Inc. [Online]. Available: http://www.nangate.com/?page_id=2325