Chapter 10
Split Manufacturing

Siddharth Garg and Jeyavijayan (JV) Rajendran

10.1 Introduction

The idea behind split manufacturing (or split fabrication) is to partition (or "split") an IC netlist into multiple "parts" and fabricate each part at a separate foundry. Intuitively, since no one foundry gets access to the full design of the IC, its ability to either pirate the design or maliciously modify it in a targeted way is hindered.

In its simplest instantiation, an IC is split into two parts. One part has of all the active components (transistors) and some of the interconnect (wires), while the other part has the remaining interconnections. As we will discuss, more sophisticated instantiations of split manufacturing might even involve splitting active components across gates.

Technologically, split manufacturing can be achieved in one of two ways: either using an FEOL/BEOL split, or using 3D integration technology. These are discussed below.

- FEOL/BEOL splitting: this technique, shown in Fig. 10.1a, involves splitting the front-end of line (FEOL) and the back-end of line (BEOL) fabrication steps across two foundries. The FEOL part consists of transistors as well as lower metal layers (for example Metal 4 and below), and the BEOL part consists of the upper metal layers [1]. The untrusted, high-end foundry manufactures the FEOL part (including the lower BEOL layers), since these steps involve the smallest feature sizes and require access to advanced fabrication technology. Next, the trusted, low-end, in-house foundry manufactures the remaining BEOL layers. Clearly, the attacker in the high-end untrusted foundry only has access to a partial netlist; he has only

S. Garg (✉) · J. Rajendran
New York University, New York City, NY, USA
e-mail: siddharth.garg@nyu.edu; sg175@nyu.edu

S. Garg · J. Rajendran
The University of Texas at Dallas, Richardson, TX, USA

D. Forte et al. (eds.), *Hardware Protection through Obfuscation*,
DOI 10.1007/978-3-319-49019-9_10

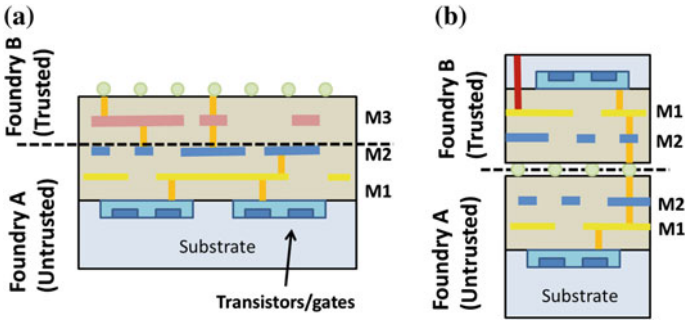


Fig. 10.1 Two approaches to split manufacturing. **a** FEOL/BEOL split manufacturing and **b** 3D integration-based split manufacturing

the FEOL part but not the BEOL part. The feasibility of this approach has been demonstrated by Vaidyanathan et al. [2] in a 0.13 μm technology node.

- 2.5D/3D integration: As shown in Fig. 10.1b, the netlist is split across two or more wafers, each containing a part of the netlist. The wafers are fabricated in different foundries and integrated through 3D integration technology. When the top part consists of only metal layers, the technology is more commonly referred to as 2.5D integration and the top tier is referred to as an interposer.

Split manufacturing is advantageous over other IP protection techniques as it does not require any key-storage mechanisms, as logic encryption does. In addition, as we will see, split manufacturing can be used to defend against strong attack models in which the attacker has access to the netlist the defender wishes to fabricate and aims to maliciously modify targeted parts of the netlist. On the other hand, hand split manufacturing is susceptible to proximity attacks that exploit physical design information, while logic encryption is not. Mitigating this vulnerability potentially introduces high overheads.

10.1.1 Split Manufacturing Flow

Figure 10.2 shows an exemplar split manufacturing flow that leverages 2.5D integration. The designer starts with the design netlist and first *partitions* the netlist into two tiers—the bottom tier consists of all gates and some wires, while the top tier consists of the remaining wires. The top tier is also referred to as the *hidden* tier since it is hidden from the view of the untrusted foundry. Note that partitioning for 2.5D/3D integration is a well-studied problem in the EDA community [3]. However, these partitioning strategies try to optimize metrics like delay and power, not security.

After partitioning, the next step is physical design. At the end of this step, the GDSII files for the bottom and top tiers are ready to be sent to their respective foundries. As mentioned earlier, the attacker should not have access to the layout

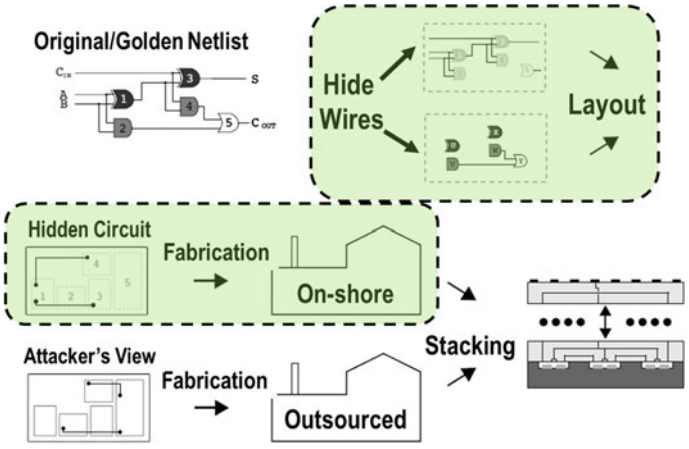


Fig. 10.2 Split manufacturing flow using 2.5D integration. The steps in the *green boxes* must be performed securely, while all others are potentially subject to attack. The flows for single-wafer and full 3D integration-based split manufacturing are similar

of wires in the top tier. Traditional physical design tools optimize for metrics like average wire length and can potentially leak information to the attacker, thus compromising security. In Sect. 10.4, we discuss a security-aware layout strategy for split manufacturing.

The two tiers are manufactured at their respective foundries. The assumption is that the two foundries cannot collude—for this reason, the top tier must be fabricated in a trusted, in-house foundry. Finally, the top and bottom tiers are stacked and packaged by a trusted integrator.

An inherent assumption in any split manufacturing process is that the IC has not been manufactured before, or at the very least, previously manufactured versions cannot be purchased commercially. If this were the case, the untrusted foundry could simply purchase the IC from the market and reverse engineer the wiring in the hidden tier.

The rest of this chapter is organized as follows. Section 10.2 details two specific threat models in the context of split manufacturing. The weak threat model assumes an attacker who is interested in thieving the designer's IP. In contrast, the strong threat model strengthens the weak attacker with apriori knowledge of the IC's netlist (hence, IP theft is moot)—the strong attacker wishes to maliciously modify the chip's functionality.

Next, Sect. 10.3 discusses a specific security metric, k-security, that quantifies the amount of security that split manufacturing buys in the context of the strong attack model. Intuitively, k-security measures the extent to which the attacker is confused as to the functionality of each gate in the (partial) netlist she observes.

Section 10.4 discusses how the traditional design automation tool flow should be modified to obtain a certain level of security while minimizing cost and information

leakage. In the secure partitioning step, the netlist is partitioned so as to maximize k-security within a cost constraint. In the secure layout step, the placement of gates in the netlist outsourced to the untrusted foundry is decided so as to ensure that the layout/placement does not reveal information about the wiring in the part of the netlist hidden from the attacker. We conclude in Sect. 10.6.

10.2 Threat Models

We now discuss two specific threat models in the context of split manufacturing. In the first, we assume that the attacker only has access to the GDSII file of the bottom tier. We call this the weak attack model. The attacker's goal is to reverse engineer the full netlist when the designer is fabricating. In the second, we assume that in addition to the GDSII file of the bottom tier, the attacker also has access to the full netlist when the designer is fabricating. Here, the attacker's goal is not IP theft, but instead is hardware Trojan insertion.

10.2.1 Weak Attack Model

In the weak attack model, the goal of the designer is to keep the IC's netlist hidden from the attacker. Formally, let the designer's private netlist be $C = (V, E)$. The part of the netlist sent to the untrusted foundry is referred to as $C_{PART} = (V_{PART}, E_{PART})$, where $|V| = |V_{PART}|$ for FEOL/BEOL splitting or 2.5D integration-based split manufacturing. We assume that the attacker can reliably recover the netlist C_{PART} from its layout file. Can a determined attacker recover C knowing only C_{PART}?

Rajendran et al. [4] have shown that attacker's can potentially infer the hidden connectivity in the top tier by leveraging the proximity of gates in the bottom tier. This is referred to as a *proximity attack*. Proximity attacks can be particularly successful if the defender uses conventional layout techniques. Such techniques try to minimize sum wire length; thus, connected gates are likely to be proximal. This is illustrated in Fig. 10.3 using a simple example in which connecting proximal bond points recovers the hidden wires correctly. Figure 10.4 shows the histogram of Euclidean distances between pairs of connected and unconnected gates obtained using a commercial layout tool—observe that connected gates are far more likely to be proximal than unconnected gates.

Using even a simple strategy in which the attacker connects each unconnected gate input to its closest unconnected gate output results in >90% correct recovery of hidden connections when a conventional netlist partitioning technique such as hMetis [5] and a commercial layout tool are used in the split fabrication flow.

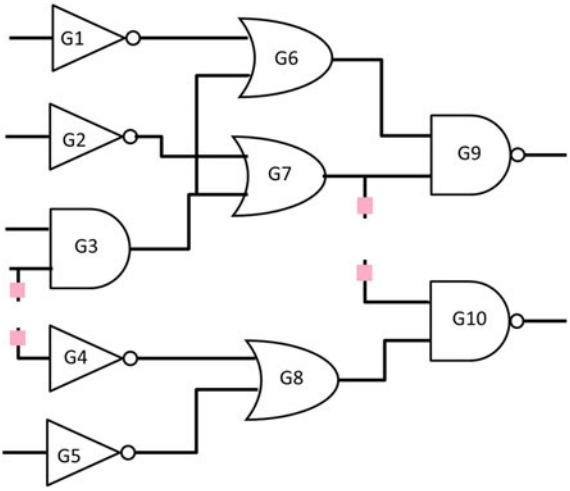


Fig. 10.3 c17 benchmark circuit with two hidden wires. Connecting proximal bond points recovers the correct netlist

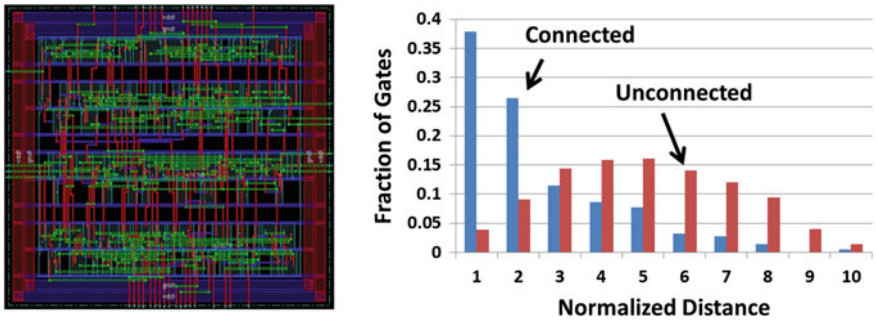


Fig. 10.4 Layout of a sample benchmark and corresponding wire length distribution for unconnected and connected gates

10.2.2 Strong Attack Model

In the weak attack model, the designer's netlist is private. However, what if the designer's goal is to fabricate logic for which the functionality, and perhaps even the netlist, is public knowledge? Examples of such functions are abundant. Most cryptographic protocols are publicly known, for instance advanced encryption standard (AES) and the data encryption standard (DES) protocols, and often have known optimized hardware implementations [6, 7]. The primary threat in such a scenario arises from hardware Trojan insertion. As noted by [8], hardware Trojans can have a disastrous impact on IC security, from unauthorized privilege escalation [9] to secret key leakage [10]. Hardware Trojans are broadly categorized into: (1) always active,

(2) trigger and payload, and (3) reliability-based [11], i.e., those that use device degradation as an implicit trigger.

Specifically, in the strong attack model, we assume that attacker has access to the full netlist, C, that the designer wishes to fabricate. In addition, as in the weak attack model, the attacker also has the partial netlist C_{PART}. We assume that the designer has scrubbed the layout file from which C_{PART} is derived of all identifying labels, and therefore, the labels in C_{PART} are arbitrary and unrelated to those in the original netlist C.

The attacker wishes to modify the design, i.e., insert a hardware Trojan, in a certain *targeted* way. For instance, for the privilege escalation attack [9], the attacker's goal is to modify the gates that control the bits that determine whether the processor executes in user or kernel mode. That is, the attacker needs to determine where in the design to insert the hardware Trojan payload. Similarly, to insert a Trojan that triggers when a certain sequence of instructions is observed [12], the attacker needs to identify certain wires/gates in the decode logic. As another example, the reliability attack discussed in [11] also requires modifications of certain targeted parts of the netlist.

To succeed in its objective, therefore, the attacker must first correctly identify the gate(s) in C_{PART} that it wishes to modify (recall that gates in C_{PART} and C are differently labeled). It does so by *matching* the gates in the partial netlist to those in the public netlist. If the match is correct, the attacker succeeds. The attacker's objective can therefore be formulated mathematically as follows. To match gates in C_{PART} to those in C, the attacker wishes to find a bijective mapping $\phi : V \rightarrow V_{PART}$ such that $< \phi(u), \phi(v) > \in E_{PART}$ only if $< u, v > \in E$. That is, the attacker knows that if an edge exists in C_{PART}, the corresponding edge must exist in C. On the other hand, if an edge does not exist in C_{PART}, the corresponding can still exist in C since it might be hidden.

The condition above is equivalent to the attacker determining a sub-graph of C, one that consists of all of the vertices in C but only some of the edges, that is isomorphic to C_{PART}. Two graphs are said to be isomorphic if they have the structure, i.e., there is a way to permute the vertices of the first graph to obtain the second. When a sub-graph of one graph is isomorphic to another, this is referred to as a **sub-graph isomorphism**.

The crux of using split fabrication as a defense mechanism in this setting is that *many such sub-graph isomorphisms might exist*, thus hindering the attacker in identifying a correct mapping.

Note that the proximity attacks discussed above are still relevant in the context of the strong attack model—that is, in addition to finding sub-graph isomorphisms, the attacker could use proximity information to match gates in C_{PART} to C. In this sense, the strong attack model subsumes the weak attack model. The rest of this chapter therefore focuses on the strong attack model.

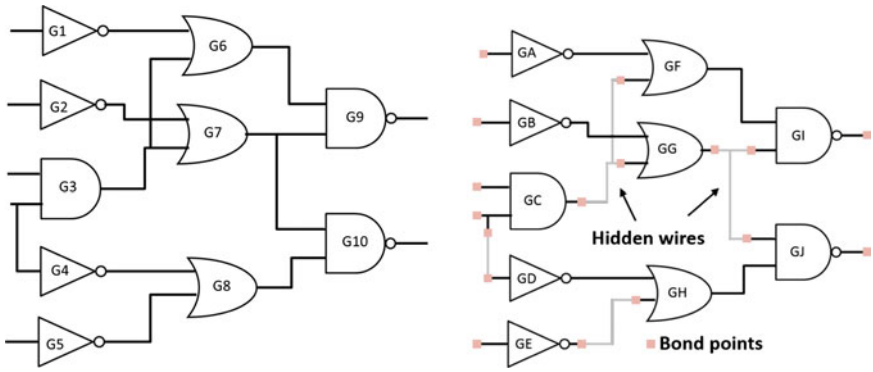
10.3 Security Metric

Imeson et al. [13] have proposed a security metric, *k-security*, that quantifies the security obtained from split manufacturing against targeted hardware Trojan insertion attacks. Consider, for example, the public netlist in Fig. 10.8a and the partial netlist sent to the untrusted foundry, shown in Fig. 10.8b. Five wires from the public netlist have been hidden, and bond points are added to allow these wires to be implemented in the top tier. Now observe that gate G6 in the public netlist can correspond to *either* gate GF and GG in the partial netlist. We thus say that gate G6 is 2-secure. To attack G6, the attacker can either pick one of GF/GG and fail with probability 0.5 (modifying both gates will change the nature of the attack). On the other hand, note that the attacker can uniquely identify that gate GC in the partial netlist is gate G3 in the public netlist. Gate G3 is therefore only 1-secure. The definition of k-security is formalized below (Fig. 10.5).

Definition 1 (*k-security*) A gate $u \in V_{PUB}$ is k-secure if there exist k distinct subgraph isomorphisms $\{\phi_1, \phi_2, \ldots, \phi_k\}$ between C_{PUB} and C_{PART} where $\phi_i(u) \neq \phi_j(u)$ for all $i, j \in [1, k]$ and $i \neq j$. A partial netlist C_{PART} is k-secure with respect to the public netlist C_{PUB} if each vertex $u \in V_{PUB}$ is k-secure.

10.3.1 Relevance of k-Security

To further understand the relevance of the k-security metric, consider the hardware implementation of the DES protocol shown in Fig. 10.6a. It has been shown that if an attacker can modify the LSB output of the 14^{th} round, then she can easily recover



(a) c17 Benchmark Netlist (b) Secure Partitioning

Fig. 10.5 The c17 benchmark netlist (**a**), and the part of the netlist sent to the untrusted foundry after secure partitioning (**b**)

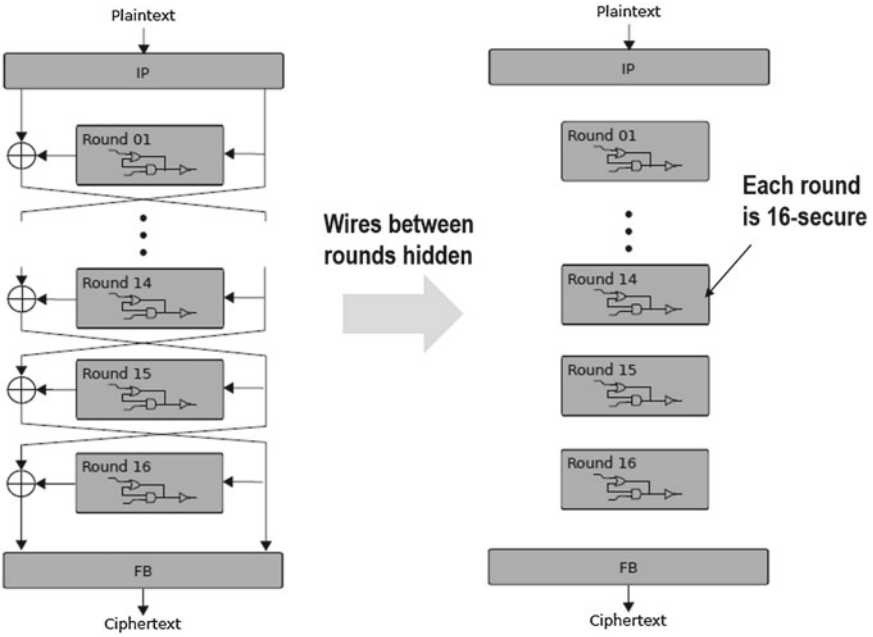


Fig. 10.6 A hardware implementation of DES encryption (*left*) and the partial netlist sent to the untrusted foundry after partitioning. Each round of the DES implementation is 16-secure since it cannot be distinguished from any other round

the DES key from plaintext–ciphertext combinations [14]. Without obfuscation, an untrusted foundry might try to leverage this vulnerability by maliciously modifying the DES implementation such that the least significant bit (LSB) output of the 14^{th} round flips when a certain trigger condition occurs. Obviously, to carry out such a targeted attack, the foundry must first identify the wire that corresponds to this vulnerable bit.

Now consider the implementation in Fig. 10.6b where all wires between rounds are hidden. Since the functionality (and netlist) of each round is identical, any one of the 16 modules could correspond to the one that implements the 14^{th} round. Indeed, in this case, the LSB output of the 14^{th} round is 16-secure.

Other examples of security-critical gates that an attacker might wish to target in a netlist include:

- The gate that outputs the privilege bit in a microprocessor. By modifying the output of this gate, the attacker can launch a privilege escalation attack [15].
- Bits that indicate the type of instruction in a processor decode unit. Rarely occurring instruction types, or more generally, gate outputs that rarely switch, can be used as triggers for attacks [16].

10.3.2 Computing k-Security

For a given partitioning, that is, given C and C_{PART}, how can the defender determine the security level k? As indicated by the definition of k-security, the problem determining the security level is closely related to the sub-graph isomorphism problem, which is NP-complete. Indeed, Imeson et al. [13] formally prove that the problem of determining whether a given partitioning meets a security constraint, k, is also NP-complete.

Having characterized the complexity of the problem, the next step is to devise a concrete algorithm to determine the security level for a given partitioning solution. To do so, the defender iterates through vertices in C. For each vertex in C, the defender iteratively checks if it can be mapped to each vertex in C_{PART}. Specifically, to check whether vertex $u \in V$ can map to vertex $v \in V_{PART}$, she checks if a sub-graph of C is isomorphic to C_{PART} with the constraint that u must map to v ($\phi(u) = v$).

The check above can be performed in one of two ways: (1) directly using sub-graph isomorphism solvers (since it is an instance of a sub-graph isomorphism problem); or (2) reducing the check to an instance of a CNF-SAT and calling a SAT solver. The reason to try the second approach is that fast, off-the-shelf solvers are available for the SAT problem.

The reduction to SAT approach (which is the one recommended by Imeson et al. [13]) introduces Boolean variables ϕ_{ij} that are 1 if node v_i in C_{PART} maps to node r_j in C, and 0 otherwise. Constraints are then introduced that ensure that a node in C_{PART} maps to only one node in C and vice versa. Finally, constraints are also introduced to ensure that an edge in C_{PART} only maps to an edge in C. The three sets of constraints are conjoined and input to a SAT solver.

Given graphs C and C_{PART}, and a bijective mapping ϕ as defined above, we now construct a Boolean formula that is true if and only if graphs C and C_{PART} are sub-isomorphic for the mapping ϕ. We will construct the formula in parts.

First, we ensure that each vertex in C maps to only one vertex in C_{PART}:

$$F_1 = \prod_{i}^{|V_{PART}|} \sum_{j}^{|V|} \left(\phi_{i,j} \prod_{k \neq i}^{|V|} \neg \phi_{i,k} \right)$$

and vice versa:

$$F_2 = \prod_{j}^{|V|} \sum_{i}^{|V_{PART}|} \left(\phi_{i,j} \prod_{k \neq i}^{|V_{PART}|} \neg \phi_{k,j} \right)$$

Finally, we need to ensure that each edge in C_{PART} maps to an edge in C. Let $E_{PART} = \{e_1, e_2, \ldots, e_{|E_{PART}|}\}$ and $E = \{f_1, f_2, \ldots, f_{|E|}\}$. Furthermore, let $e_k = \langle q_{src(e_k)}, q_{dest(e_k)} \rangle \in E_{PART}$ and $f_k = \langle r_{src(f_k)}, r_{dest(f_k)} \rangle \in E$. This condition can be expressed as follows:

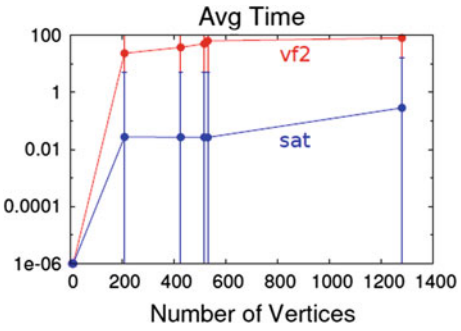


Fig. 10.7 Run-time of SAT-based a domain-specific sub-iso solver-based approaches for computing security. The sub-iso solver used is VF2

$$F_3 = \prod_{k}^{|E_{PART}|} \sum_{l}^{|E|} \phi_{src(e_k),src(f_l)} \wedge \phi_{dest(e_k),dest(f_l)}$$

The formula F that is input to the SAT solver is then expressed as a conjunction of the three formulae above: $F = F_1 \wedge F_2 \wedge F_3$. The formula F has $O(|V_{PART}||V|)$ variables and $O(|E_{PART}||E|)$ clauses.

Empirically (and perhaps surprisingly), the SAT approach is faster than using a domain-specific sub-iso solver. This is illustrated in Fig. 10.7.

10.4 Defense Mechanisms

Designer's must mitigate the threat from strong attackers in two ways: (1) secure partitioning to maximize k-security so as to defeat sub-iso attacks and (2) secure layout to defeat proximity attacks. Note that to defend against weak attackers, only the second method would be required. We now describe these defense mechanisms in detail.

10.4.1 Secure Partitioning

Split manufacturing incurs a cost—wires that cross from one tier to the other use large, capacitive bond points. This increases the area, delay, and power consumption of the chip. Thus, on the one hand, increasing the number of hidden wires increases security, but also increases area, delay, and power. The goal of secure partitioning is to explore the trade-offs between security and cost. As a starting point, we adopt a simple notion of cost, the number of hidden wires. Security is measured using the k-security metric described earlier.

Given C and constraint on maximum number of hidden wires, H, the goal of secure partitioning can be formulated as finding C_{PART} as follows:

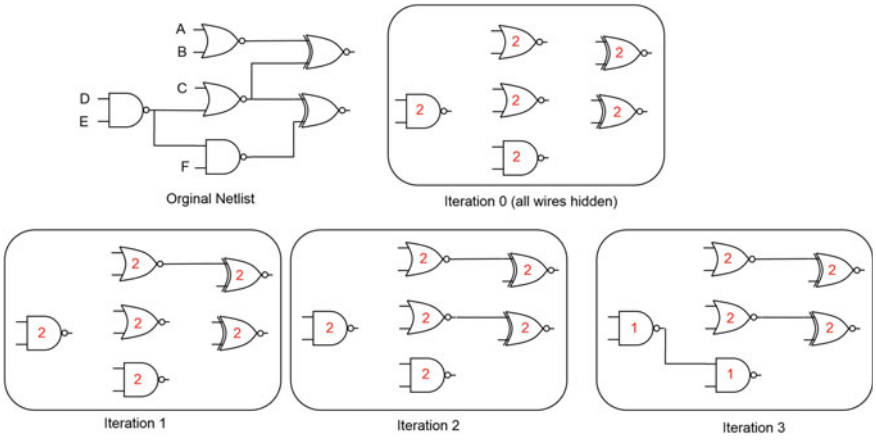


Fig. 10.8 Example of greedy secure partitioning heuristic. In each iteration, a new edge is added in a way to minimize the reduction in security. Each gate is annotated with its security level in each iteration

$$\max_{C_{PART}} k(C, C_{PART})$$

such that

$$|E| - |E_{PART}| \leq H$$

and

$$C_{PART} \subseteq C,$$

where $k(C, C_{PART})$ returns the security level of a partitioning solution.

El Massad [17] has shown that there exists no polynomial time approximation scheme for the secure partitioning problem. As an alternative, Imeson et al. [13] have proposed a greedy heuristic to solve this problem.

The greedy heuristic initializes C_{PART} to have all the gates in C but none of the wires. That is, C_{PART} is initialized to have maximum security, but at maximum cost. Then, in each iteration, a new edge/wire is added to C_{PART}, specifically, one that results in the smallest reduction in security. These iterations continue till $E_{PART} = E - H$, at which point the procedure terminates.

An example illustrating the greedy procedure is shown in Fig. 10.8. The original netlist has 6 gates. Each gate can be at best 2-secure because it can, at best, be confused for the other gate of the same type. Starting with the maximally secure netlist in which all wires are hidden, we observe that the new wire added in the first iteration does not reduce security. The same is true for the second iteration. In the third iteration, adding any new wire will result in a drop in security—the wire that is added results in the least drop in security (the security of the two NAND gates goes down to 1). Adding any other wire would have resulted in even larger reduction in

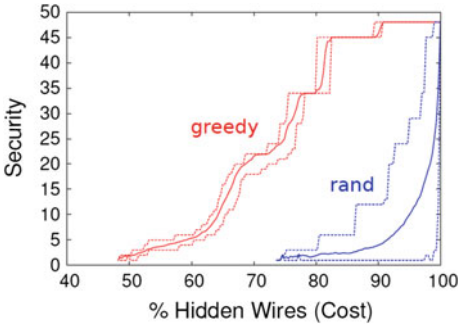


Fig. 10.9 Security versus cost trade-offs obtained using the greedy secure partitioning and random partitioning approaches

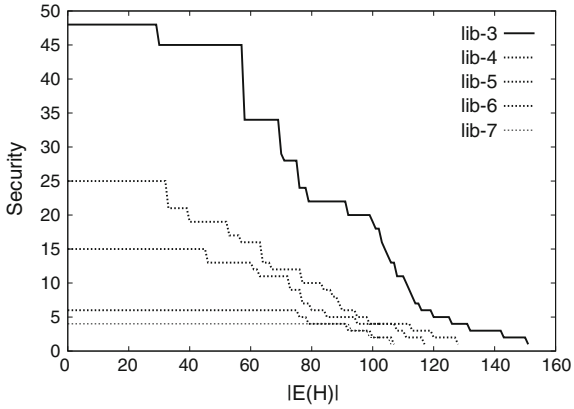


Fig. 10.10 Impact of choice of technology library on security. More diverse technology libraries yield greater security

security. Adding any further wires makes all the gates 1-secure, i.e., at this point we obtain no security at all.

Figure 10.9 shows the security versus cost trade-offs obtained by using the greedy approach on the c432 benchmark circuit from the ISCAS-85 benchmark suite. The results obtained from the greedy heuristic are compared to *randomly* hiding wires from the netlist, a strategy suggested in a white paper by Tezzaron. Note that the greedy secure partitioning approach significantly outperforms random partitioning.

The upper bound on security in Fig. 10.9 is set by the gate type that appears the *fewest* times in the netlist. In theory, if there is a unique gate in the netlist (one that appears only once), it will always be 1-secure, and correspondingly, the entire netlist would only be 1-secure. This suggests that the diversity of the technology library has a role to play in security—the more diverse the technology library, the less effective we would expect secure partitioning to be. This is indeed the case, as shown in Fig. 10.10.

10.4.2 Secure Layout

Figure 10.11 shows a secure layout tool flow that provably defends against proximity attacks [4]. In this flow, the bottom tier layout is performed independently of the top tier—since the layout tool has no information about connectivity in the top tier, this information cannot be embedded in the resulting layout. After layout, conventional routing is performed for the top and bottom tiers.

Figure 10.12 compares the results after conventional 2D layout to secure layout for the bottom and top tiers. Of note, the routing in the top tier is more "convoluted" than in the optimized 2D layout—this reflects the fact that the proximity of gates in the bottom tier reveals no information about their connectivity in the top tier. This assertion was empirically verified using a statistical hypothesis test for the layout in Fig. 10.12.

We note that although the secure layout tool flow guarantees security, it comes at extra cost because of increased wire length in the top tier. Increased wire length implies greater delay and power consumption. Table 10.1 reproduces data from Imeson et al. [13] that illustrates the relationship between delay, power, total wire

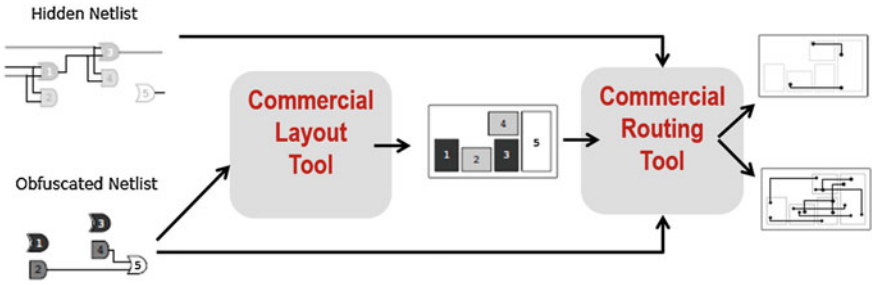


Fig. 10.11 Secure layout tool flow. The bottom tier layout is performed separately from the top tier after partitioning using a commercial layout tool. The resulting layout is independent of the connectivity of the hidden (top) tier

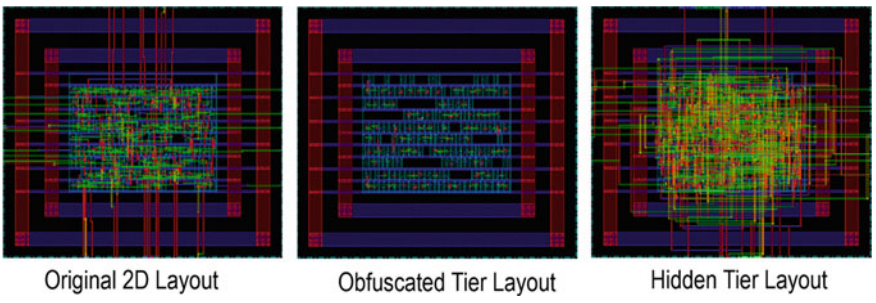


Fig. 10.12 Layout of the bottom and top tiers after secure layout, compared to the original, optimized 2D layout

Table 10.1 Power, delay, wire length, and area analysis for different levels of security on the c432 circuit. 1^* is the base circuit with no wires lifted and 48^* has all of the wires lifted

Security	Power ratio	Delay ratio	Total wire length (μm)	Total area (μm^2)
1^*	1.00	1.00	2739	1621
2	1.54	1.73	6574	4336
4	1.55	1.76	7050	4416
8	1.61	1.82	8084	4976
16	1.62	1.86	8161	5248
24	1.71	1.98	9476	6048
32	1.73	1.99	9836	6368
48^*	1.92	2.14	13058	8144

length and area as the k-security level is increased. These results are for a relatively small benchmark (c432); for large benchmarks, it is possible that beyond a certain security level the design becomes unroutable.

10.4.3 Raising the Bar on the Attacker

As a consequence of the defense mechanisms studied in this section, the attacker is now confounded. For any gate in the original netlist that the attacker wishes to modify, there are $k - 1$ other gates that can conceivably correspond to that specific gate.

An attacker could now choose one of those k gates at random and fail in her objective with a high probability. Or, the attacker could try and modify all gates, modifying one at a time (since modifying all together will not result in the malicious functionality the attacker desires). Figure 10.13 shows the minimal hardware the attacker would need to sequentially modify each of the k gates. The additional hardware significantly increases the Trojan area and likelihood of being detected in post-fabrication testing.

10.5 Future Opportunities and Challenges for Split Manufacturing

10.5.1 Reducing Cost

As we have seen, existing split manufacturing approaches can incur high cost, although they do provide strong security guarantees. Several opportunities exist to reduce the cost without compromising security.

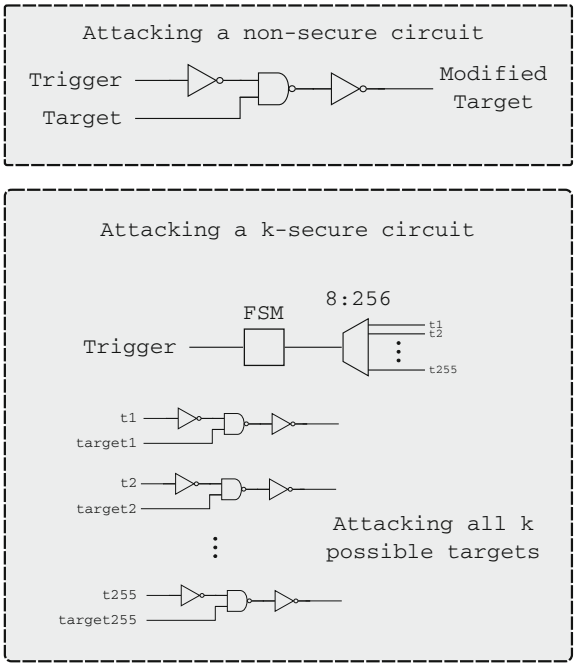


Fig. 10.13 Attack scenarios of 1- and k-secure circuits

Using decoy gates: simply duplicating a netlist and connecting only one of the netlists to the chip's IO pins (the IO pins are implemented in the hidden tier) automatically provides 2-security with low overhead in terms of number of bond points, power, and delay. The trade-off is in the area of the bottom tier, but these trade-offs might be acceptable especially in the era of dark silicon [18]. In this solution, one of the two netlists acts a decoy. The same idea can be deployed at the gate level instead of at the netlist level, i.e., the security level of individual gates can be increased by introducing decoys in the bottom tier.

Leveraging full 3D integration: Full 3D integration allows each tier to have both gates and wires. Using full 3D would allow gates to be hidden in the top tier along with wires. The top tier is still fabricated by a trusted foundry, but one with access to more mature CMOS fabrication technology. Security-critical gates in the design, for example, the sub-circuit that controls super-user privileges in a microprocessor, can be selectively implemented on the top tier.

Reducing cost of secure layout: A significant contributor to the cost of split manufacturing is secure layout, especially as the number of hidden wires exceed. The secure layout approach discussed so far guarantees that the placement of gates in the bottom tier leaks no information their connectivity—in practice, a solution that allows designers to trade off a limited amount of information leakage for reduced cost is desirable. Xie et al. [19] have taken in a step in that direction using simulated annealing-based layout flow. However, the authors do not quantify the amount of

information leaked by the proposed approach. More details on this approach can be found in Chap. 12.

10.5.2 Alternative Security Metrics

A criticism of the k-security metric is that it is perhaps too conservative. For one, it assumes that the attacker knows the original netlist, a potentially unrealistic assumption. Second, the metric requires that *every* gate in the netlist be k-secure while this might only be required of certain security-critical gates.

Jagasivamani et al. [20] have proposed alternative metrics for split manufacturing that might be relevant in the weak attack model. These metrics depend only on the partial netlist that the attacker observes, C_{PART} and not on the original netlist, C_{PUB}. Two specific metrics proposed by [20] and their relationship to k-security are discussed below:

Standard cell entropy: this metric measures the diversity of standard cells in the design using its entropy; a design that only uses cells of one type (say NANDs) has entropy 0, while one that has the same number of cells of each type has the highest possible entropy. Counter-intuitively, however, the authors advocate for lower entropy as being *beneficial* for security. This is antithetical to the traditional interpretation of greater entropy (i.e., greater disorder) being useful from a security perspective [21]. A simple example illustrates why using entropy in the way that Jagasivamani et al. suggest might be misleading.

Consider a netlist with $N/2$ gates of type 1 and $N/2$ gates of type 2 versus one with $N - 1$ gates of type 1 and a single gate of type 2. Based on the entropy metric, the former netlist has higher entropy than the latter and is therefore *less* secure. On the other hand, k-security suggests the opposite: The former netlist is $N/2$-secure while the latter is only 1-secure.

This discussion illustrates the potential danger in simply adopting metrics, like entropy, that are used in entirely different security contexts. While entropy is a useful metric for side-channel vulnerability assessment, for instance, it is not clear how it directly relates to the split manufacturing problem. In contrast, k-security has a precise attack model and relates directly to the success probability of the attacker in this model.

Neighbor connectedness: To address the threat from proximity attacks, Jagasivamani et al. [20] suggest the use of a metric that measures how likely proximal (neighboring) gates are to be connected. While this metric captures, abstractly, resilience against proximity attacks, metrics that can precisely estimate the attackers success probability are needed.

Table 10.2 Overview of split manufacturing-based obfuscation techniques

Work	Domain	Attack model	Attacker intent	Methods
Reference [13]	Logic	Strong attacker	Trojan insertion	2.5D integration Provably randomized layout
Reference [22]	Logic	Strong attacker	Trojan insertion	Obfuscated built-in self-authentication Optimized layout with filler extra filler cells
Reference [2]	Logic	Weak attacker	IP theft	FEOL/BEOL split (M1 and above) Optimized layout
Reference [23]	Logic	Weak attacker	IP theft	FEOL/BEOL split (poly and above) Obfuscated layout of standard cells
Reference [24]	SRAM	Weak attacker	Trojan insertion	FEOL/BEOL split (M1 and above) Partially randomized layout nonconventional design decoys
Reference [25]	RF	Weak attacker	IP theft	FEOL/BEOL split (M4–M7 on) obfuscated inductors and capacitors

10.5.3 Complementary Uses of Split Manufacturing

Split manufacturing can be used in a number of security-related setting that are complementary or orthogonal to the setting discussed in this chapter. Vaidyanathan et al. [24] for SRAM blocks and analog IP, while Bi et al. [25] have proposed similar ideas for RF ICs.

In particular, Vaidyanathan et al. [2] identify hard IP blocks such as SRAM arrays and analog IP as specific sources of weakness because they typically have very regular layout patterns. Even with only FEOL and M1 access, attackers can easily reverse engineer these patterns.

To address this vulnerability, the authors propose to use (a) randomized placement of peripheral logic (akin to the secure layout approach discussed before), (b) nonconventional design approaches for common logic blocks like decoders, and (c) nonstandard, application-specific features to confound the attacker (in part, similar to decoy cells discussed earlier).

Bi et al. [25] observe that for RF designs, removing the top metal layers has the effect of hiding inductors from the design. Further removing lower metal layers hides

capacitors from the design. As a consequence, the inductance and/or capacitance values in the design can be hidden from the attacker. The authors then posit that without that an attacker cannot recover these values without knowing the original design intent, for example, the center frequency. In many practical settings, RF designs for standard operating bands, for example, the designer's objectives are readily apparent from the standards documentation. Whether or not the hidden inductance and capacitance values can be reverse engineered for this stronger attack model remains to be addressed.

Otero et al. [23] have proposed techniques to obfuscate connections within standard cells, instead of across cells as we have discussed so far in this chapter. While this fine-grained level of obfuscation enables even distinct standard cells to look identical, it raises the bar on the capabilities of foundry entrusted with the BEOL connections.

Finally, Xiao et al. [22] have proposed to leverage split manufacturing in a different way, i.e., to obfuscate the implementation of built-in self-authentication circuits (BISA) on the chip. BISA cells occupy what would otherwise be nonfunctional filler cells and deter a foundry from using these cells for malicious purposes. Obfuscating BISA using split fabrication makes it even harder for a foundry to maliciously modify the original netlist without triggering an alert. More details of this approach are discussed in Chap. 11.

Table 10.2 provides a summary of the different proposals for the use of split manufacturing to achieve obfuscation.

10.6 Conclusion

Split manufacturing is an emerging technique to defend against the threat of outsourced semiconductor fabrication at untrusted foundries. By hiding a part of the design from the attacker, split manufacturing can be used to prevent IP theft and targeted hardware Trojan insertion. In this chapter, we have discussed existing threat models in the context of split manufacturing and presented state-of-the-art defense mechanisms and associated security metrics to mitigate these threats. We have also provided pointers to outstanding challenges that remain to be addressed and opportunities to further improve the effectiveness of split manufacturing.

References

1. Intelligence Advanced Research Projects Activity. Trusted Integrated Circuits Program
2. Vaidyanathan K, Das BP, Sumbul E, Liu R, Pileggi L (2014) Building trusted ics using split fabrication. In: IEEE international symposium on hardware-oriented security and trust (HOST), 2014. IEEE, pp 1–6
3. Goplen B, Spatnekar S (2007) Placement of 3d ics with thermal and interlayer via considerations. In: Proceedings of the 44th annual design automation conference. ACM, pp 626–631

4. Rajendran J, Sinanoglu O, Karri R (2013) Is split manufacturing secure? In: Proceedings of IEEE/ACM conference on design automation and test in Europe, pp 1259–1264
5. Selvakkumaran N, Karypis G (2006) Multiobjective hypergraph-partitioning algorithms for cut and maximum subdomain-degree minimization. IEEE Trans Comput-Aid Des Integr Circuits Syst 25(3):504–517
6. Hoornaert F, Goubert J, Desmedt Y (1985) Efficient hardware implementation of the des. In: Advances in cryptology. Springer, pp 147–173
7. Ichikawa T, Kasuya T, Matsui M (2000) Hardware evaluation of the aes finalists. In: AES candidate conference, pp 279–285
8. Tehranipoor M, Koushanfar F (2010) A survey of hardware trojan taxonomy and detection. IEEE Des Test Comput 27(1):10–25
9. King ST, Tucek J, Cozzie A, Grier C, Jiang W, Zhou Y (2008) Designing and implementing malicious hardware. In: Proceedings of the 1st usenix workshop on large-scale exploits and emergent threats. USENIX Association, p 5
10. Lin L, Kasper M, Güneysu T, Paar C, Burleson W (2009) Trojan side-channels: lightweight hardware trojans through side-channel engineering. In: Cryptographic hardware and embedded systems-CHES 2009. Springer, pp 382–395
11. Shiyanovskii Y, Wolff F, Rajendran A, Papachristou C, Weyer D, Clay W (2010) Process reliability based trojans through nbti and hci effects. In: 2010 NASA/ESA conference on adaptive hardware and systems (AHS). IEEE, pp 215–222
12. Yang K, Hicks M, Dong Q, Austin T, Sylvester D (2016) A2: analog malicious hardware
13. Imeson F, Emtenan A, Garg S, Tripunitara M (2013) Securing computer hardware using 3D integrated circuit (IC) technology and split manufacturing for obfuscation. In: Proceedings of USENIX security, pp 495–510
14. Boneh D, DeMillo RA, Lipton RJ (1997) On the importance of checking cryptographic protocols for faults. In: Proceedings of the international conference on theory and application of cryptographic techniques, pp 37–51
15. Hicks M, Finnicum M, King ST, Martin MMK, Smith JM (2010) overcoming an untrusted computing base: detecting and removing malicious hardware automatically. In: IEEE symposium on security and privacy, pp 159–172
16. Bhunia S, Hsiao MS, Banga M, Narasimhan S (2014) Hardware trojan attacks: threat analysis and countermeasures. Proc IEEE 102(8):1229–1247
17. El Massad M (2014) On the complexity of the circuit obfuscation problem for split manufacturing
18. Shafique M, Garg S, Henkel J, Marculescu D (2014) The eda challenges in the dark silicon era. In: Design automation conference (DAC), 2014 51st ACM/EDAC/IEEE. IEEE, pp 1–6
19. Xie Y, Bao C, Srivastava A (2015) Security-aware design flow for 2.5 d ic technology. In: Proceedings of the 5th international workshop on trustworthy embedded devices. ACM, pp 31–38
20. Jagasivamani M, Gadfort P, Sika M, Bajura M, Fritze M (2014) Split-fabrication obfuscation: metrics and techniques. In: 2014 IEEE international symposium on hardware-oriented security and trust (HOST). IEEE, pp 7–12
21. Cachin C (1997) Entropy measures and unconditional security in cryptography. PhD thesis, Swiss Federal Institute of Technology Zurich
22. Xiao K, Forte D, Tehranipoor MM (2015) Efficient and secure split manufacturing via obfuscated built-in self-authentication. In: 2015 IEEE international symposium on hardware oriented security and trust (HOST). IEEE, pp 14–19
23. Otero CTO, Tse J, Karmazin R, Hill B, Manohar R (2015) Automatic obfuscated cell layout for trusted split-foundry design. In: 2015 IEEE international symposium on hardware oriented security and trust (HOST). IEEE, pp 56–61
24. Vaidyanathan K, Liu R, Sumbul E, Zhu Q, Franchetti F, Pileggi L (2014) Efficient and secure intellectual property (ip) design with split fabrication. In: 2014 IEEE international symposium on hardware-oriented security and trust (HOST). IEEE pp 13–18

25. Bi Y, Yuan J-S, Jin Y (2015) Split manufacturing in radio-frequency designs. In: Proceedings of the international conference on security and management (SAM), p 204. The steering committee of the world congress in computer science, computer engineering and applied computing (WorldComp)