How Secure is Split Manufacturing in Preventing Hardware Trojan?

Zhang Chen, Pingqiang Zhou
School of Information Science and Technology
ShanghaiTech University
Shanghai, P. R. China
{chenzhang, zhoupq}@shanghaitech.edu.cn

Tsung-Yi Ho
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 30013
tyho@cs.nthu.edu.tw

Yier Jin
Department of Electrical & Computer Engineering
University of Central Florida
Orlando, Florida 32816
yier.jin@eecs.ucf.edu

***Abstract*—With the trend of outsourcing fabrication, split manufacturing is regarded as a promising way to both provide the high-end nodes in untrusted external foundries and protect the design from potential attackers. However, in this work, we show that split manufacturing is not inherently secure. A hardware trojan attacker can still discover necessary information with a simulated annealing based attack approach at the placement level. We further propose a defense approach by moving the insecure gates away from their easily-attacked candidate locations. Experimental results on benchmark circuits show the effectiveness of our proposed methods.**

I. Introduction

Due to the high cost of owning a state-of-the-art manufacturing foundry and the complexity of integrated-circuit (IC) design, globalized IC production flow has become the mainstream. Although the separation of design and fabrication brings economic benefits to design companies, it also incurs significant concern on the security of fabricated circuits since potential threats come from all stages of the supply chain in the form of hardware attacks. To counter these attacks, all stages of the design and fabrication flow need to take security into account [1].

Hardware trojan (HT) [2], one of the most common attacks on ICs, may not only cause huge economic losses but also bring tremendous harm to the military, governmental, and public safety [3]. As HT can be implanted across the whole supply chain from functional design to fabrication [4], its defense comprises of both pre-silicon prevention and post-silicon detection/diagnosis methods. Split manufacturing [5]–[9], as a pre-silicon prevention method, is currently one of the main defense techniques. In split manufacturing, the whole design is split into two parts, the Front End of Line (FEOL) transistors and lower metal layers and Back End of Line (BEOL) top metal layers. Only FEOL layers need to be fabricated in high-end foundries while BEOL layers can be fabricated in a trusted low-end foundry. In other words, due to split manufacturing, only the transistors and the limited number of connections in lower metal layers are exposed to potential threats, while the connections in top metal layers are hidden from attackers. Therefore, there is a tradeoff between security level and cost – lower splitting layer means less information exposed to attackers, but higher cost for manufacturing more lower metal layers in trusted foundries.

Recent research works [5]–[7], [9] show that, even with part of information hidden by split manufacturing, attackers can still successfully attack a split fabricated design by exploiting the heuristics used in physical design. To make split fabricated design more secure, additional efforts need to be made in EDA tools during physical design process. In this work, we come up with two metrics to quantitatively measure the security of a circuit under HT attack, where the attacker can implant HT at multiple spots to guarantee success. To illustrate the vulnerabilities of split fabricated circuits to targeted HT attack, we first assume the most secure scenario of split manufacturing in our work – the splitting layer is the M1 metal layer where the attacker can only see a sea-of-gates with no inter-gate connections. Then we propose an effective simulated annealing (SA) based attack approach that incorporates global information such as logic connection, total wirelength and leverages the common knowledge that minimizing total wirelength is a fundamental objective during placement. Furthermore, to counter our proposed attack, we provide a defense method that moves gates out of their easily-attacked candidate locations.

The contributions of this work are as follows:

- We propose two metrics to evaluate the security level of a circuit under HT attacks.
- We present a SA-based attack method that integrates global information with global heuristics. Our results show that the proposed attack method is effective even when a circuit is split after the M1 metal layer.
- We also propose a corresponding gate-swapping-based defense approach. Experimental results show that it can significantly improve the security of a circuit.

II. Related Work

To carry out a targeted HT attack, one needs both the complete gate-level netlist and the mapping of the gates in netlist to their physical locations in the layout [7]. An example for such attack is proposed in [10], where the state of hardware registers is modified to maliciously raise privilege level. To achieve successful attack, the gate and wire that corresponds to the privilege bit needs to be determined.

To hide the connections in layout from attackers, [7] proposes to enhance security by lowering the splitting layer to be after M1 so that the attacker can only see a sea-of-gates with no inter-cell connections. However, the cost of BEOL foundries will largely increase in this case. For general split manufacturing with several lower metal layers included in FEOL, the attacker can reconstruct all the connections in the design by greedily connecting nearby pins to each other, where the proximity heuristic [5] exploits the fact that connected pins should be placed close to each other. [9] further improves proximity attack by incorporating several other heuristics such as load capacitance constraint, timing constraint. As for defense, pin swapping techniques [5] are adopted to reduce the correctness of the reconstructed connections, while [9] further minimizes the wirelength overhead arisen in security improvement. The

978-1-5090-5701-6/16/$31.00

problem with proximity attacks is that they do not take global information such as total wirelength into account. Besides, there is no guarantee that the reconstructed connection graph from the layout is isomorphic to the connection graph of the original netlist, so it cannot be directly used in HT attacks.

In [6], a graph isomorphism method is proposed to obtain the netlist-layout mapping. Two connection graphs are respectively constructed – one for the netlist and the other for the physical layout of FEOL layers. Then the mapping is obtained by graph isomorphism between the two connection graphs. To evaluate the effectiveness of the attack, k-security metric is proposed: If a gate in the logical netlist can be mapped to k candidate locations in the layout, then this gate is k-secure, meaning that it cannot be distinguished from the other $k-1$ gates. A wire lifting technique is then demonstrated to uplift the security of the circuit. However, the security analysis involved in k-security assumes that the attacker only exploits logical connection information. But in practice, the attacker can also exploit proximity information from physical layout.

III. Problem Formulation

In this section, we first discuss our threat model, then present our problem formulation on HT attack.

A. Threat Model

As discussed in Section II, the attacker wants to carry out a targeted HT attack, which requires to get the mapping between the gates in netlist and their locations in layout. While the attacker can implant HT at all possible locations, such strategy increases the workload of the attacker as well as the risk of implanted HT being detected. Therefore, the attacker should reduce the number of implantations as much as possible.

In our threat model, the attacker can get help from two roles in two stages: A rogue element in the untrusted foundry who can modify the FEOL layout during fabrication and a malicious observer in the design stage who cannot do malicious changes on the design, but has access to the precise gate-level netlist of the entire circuit. Our threat model aligns with the one used in [6].

The reason that the attacker is assumed to be able to obtain gate-level netlist is that unlike software attackers, organizations that intend for hardware attack are resourceful and are willing to pay for the related cost because successfully implanted HT is capable of executing valuable attacks [11].

Moreover, to show the vulnerabilities of split-fabricated circuits to HT attacks, we assume that the circuit is split after M1, which gives least information to the attacker. We also assume that primary inputs and outputs in the layout can be uniquely identified based on the specification of the design and can be correctly mapped to their counterparts in netlist without further efforts.

Finally, if chips of a design are fabricated in batches, then the attacker can buy an instance from the market and reverse engineer the wire connections in BEOL, nullifying the effort of split manufacturing on hiding connections [6]. Thus, we assume that designers are aware of this and have all FEOL parts fabricated before releasing their products to the market.

B. Problem Formulation

With the physical layout of the FEOL layers and the gate-level netlist of the entire circuit, the goal of the attacker is to map the gates in netlist to their physical locations in layout so as to implant HT. We use Fig. 1 to illustrate our problem definition. Fig. 1(a) is the graph corresponding to the gate-level netlist of a circuit. Fig. 1(b) is the complete physical layout of this circuit, which shows the correct mapping between the gates in netlist (as shown in Fig. 1(a)) and the physical gates in layout. Fig. 1(c) is the layout that the attacker sees, with no inter-cell connections and all physical gates labelled differently.

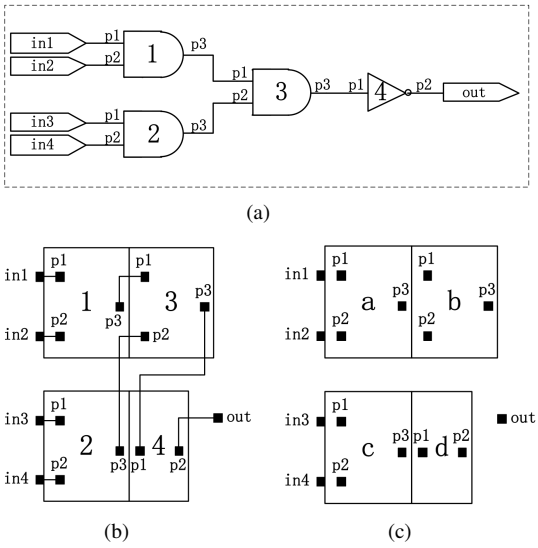

Fig. 1. (a) Logical connection corresponding to the gate-level netlist. (b) The complete physical layout with all gates correctly mapped to their counterparts in netlist. (c) The physical layout that the attacker sees.

Let $V_n = \{1, 2, 3, 4\}$ be the set of the gates in the netlist and $V_l = \{a, b, c, d\}$ be the set of the gates in the layout that the attacker sees, then the mapping problem is to find a mapping $\phi : V_n \rightarrow V_l$ that is close to the correct mapping $\phi_c : V_n \rightarrow V_l$. From Fig. 1(b)(c), the correct mappings are all bijective: $\phi_c(1) = a, \phi_c(2) = c, \phi_c(3) = b, \phi_c(4) = d$.

For the attacker, since he can get information from the untrusted foundry, he can reverse engineer all the components in FEOL [9], after which he knows the gate type of all the physical gates in layout. Consequently, the initial mapping for the attacker is $\phi(x) = \{a, b, c\}$ for $x = 1, 2, 3$ and $\phi(4) = d$. Based on the initial mapping, if his target is gate 3, then he has to implement HT at all three AND gates. Note that ϕ is not restricted to be bijective. The reason for this is obvious: the main goal of the attacker is to successfully implant HT, so if the location for the target cannot be uniquely identified, multiple implantations are acceptable. We call $\phi(V_n(i))$ the mapped set of the i-th gate in netlist, which contains all the possible physical gates for $V_n(i)$ to map to. Obviously, $|\phi(V_n(i))|$ may not always be 1, but the attacker can try to prune $\phi(V_n(i))$ so as to reduce the cost and risk of the attack.

Generally, let m be the number of gate types in a circuit, the mapping problem becomes finding m mappings $\phi_1, \phi_2, \phi_3, ..., \phi_m$ such that $\phi_j : V_n^j \rightarrow V_l^j$ is the mapping between the gates of type j in netlist and the physical gates of the same type in layout. For any gate $V_n^j(i)$ of type j, its initial mapped set is the set of all the physical gates of type j. The problem is then to prune the initial mapped sets in an appropriate way. We will present our solutions to the

mapping and pruning problems in Section IV. After that, we will present our defense method in Section V.

IV. Attack Approach

We propose a simulated annealing (SA) based attack approach that takes advantage of the global wirelength information and also explores multiple mapping solutions. The attack comprises of two steps:

- First, we use the SA engine that minimizes total wirelength to get multiple netlist-layout mapping solutions.
- Second, we integrate all the mapping solutions. By extracting the most possible locations for each gate, we further prune the mapped set of each gate to obtain the final mapping solution.

The aim of this attack is to prune the mapped set of each gate as much as possible so that the cost for implanting HT is reduced, with the constraint that most mapped sets should contain the real locations for corresponding gates. If it were not the case, the attack may not take effect as expected because the correct locations are not implanted with HT. To quantitatively measure the effectiveness of the attack, we propose two metrics:

1) Effective Mapped Set Ratio (EMSR). The mapped set of a gate is effective only when it contains the real location of this gate. If a mapped set does not contain the real location, it misleads the attacker to miss the targeted spot and harms the effectiveness of the attack. EMSR is defined as the percentage of effective mapped sets among all mapped sets. Formally,

$$\text{EMSR} = \frac{\sum_{i=1}^{|V_n|} |\phi(V_n(i)) \cap \phi_c(V_n(i))|}{|V_n|} \tag{1}$$

where $\phi(V_n(i))$ is the mapped set of gate $V_n(i)$ and $\phi_c(V_n(i))$ is the real physical gate for $V_n(i)$. $|V_n|$ is the total number of gates in the circuit.

2) Average Mapped Set Pruning Ratio (AMSPR). AMSPR is the pruning ratio of mapped sets, calculated as the average ratio between the reduced sizes of the sets after attack and the sizes of the initial sets. Formally,

$$\text{AMSPR} = \frac{1}{|V_n|} \cdot \sum_{i=1}^{|V_n|} \Big(1 - \frac{|\phi(V_n(i))|}{|\phi_{ini}(V_n(i))|}\Big) \cdot |\phi(V_n(i)) \cap \phi_c(V_n(i))| \tag{2}$$

where $\phi_{ini}(V_n(i))$ is the initial mapped set for $V_n(i)$ that contains all the physical gates with the same gate type as $V_n(i)$. Note that if a mapped set $\phi(V_n(i))$ becomes ineffective during pruning, i.e., $\phi(V_n(i)) \cap \phi_c(V_n(i))$ is empty, this mapped set would only mislead the attacker no matter how much it is pruned. Thus, if a mapped set becomes ineffective, we set the pruning ratio of this mapped set to be 0, meaning that the pruning is fruitless.

To achieve high AMSPR while not harming EMSR, instead of looking for local heuristics that may work in fewer scenarios, we propose an attack based on the global heuristic that placement generally minimizes the total wirelength of a circuit. Further, due to the suboptimality and diversity of placement solutions, obtaining an accurate mapping based on a single mapping solution is hard, so our work captures the diversity of placement solutions by solving the mapping problem for a number of times, after which all solutions are merged to form a better solution.

A. Mapping by Simulated Annealing (SA)

Since the primary objective of placement tools is to minimize the total wirelength, we use SA method to obtain netlist-layout mappings under the objective of total wirelength minimization:

- First, a random bijective mapping is generated, with each gate in netlist mapped to a random physical gate of the same type. Note that if a gate type has only one instance in the circuit, it is inherently correctly mapped. Due to the property of bijective mapping, each gate in layout is also uniquely mapped to a gate in netlist. Consequently, the connections in the physical design can be fully reconstructed by following the connections in netlist. For the calculation of total wirelength, we use half perimeter wirelength (HPWL) as the estimated wirelength for each net.
- Second, to drive the mapping towards smaller total wirelength, a SA framework is adopted to iteratively improve wirelength by randomly swapping the mappings of two gates. For example, if $V_n(i)$ and $V_n(j)$ previously map to $V_l(p)$ and $V_l(q)$ correspondingly, then after swapping, they map to $V_l(q)$ and $V_l(p)$ correspondingly. It is worth mentioning that swapping can only happen between two gates of the same type.
- In a mapping solution after SA process, if gate $V_n(i)$ in netlist is mapped to gate $V_l(j)$ in layout, then we add $V_l(j)$ to $\phi(V_n(i))$ if $\phi(V_n(i))$ does not contain it.
- We run the above SA process for a number of times, aiming to eliminate the improbable locations for each gate in netlist. To avoid eliminating the real location of each gate, we should run the SA process for enough times to allow the correct mapping of each gate to emerge. All the solutions, with each representing a possible placement that minimizes wirelength, show the most possible locations of a gate in netlist. If a gate in netlist is never mapped to some physical gates in these wirelength-optimized solutions, then it is unlikely for the gate to map to these physical gates in the correct mapping. In this way, $\phi(V_n(i))$ for each gate is effectively pruned to only include the possible locations in terms of global wirelength minimization.

B. Mapped Set Pruning

To further reduce the number of possible locations of each gate, a statistical method is applied. We illustrate this method using the following example. Assume a gate $V_n(i)$ in netlist is mapped to the physical gate $V_l(j)$ in M of the total N SA solutions. Let $\phi(V_n(i))$ be the mapped set for $V_n(i)$ after all SA process and $S(V_n(i))$ be the set of the physical gates with the same gate type as $V_n(i)$. Without any heuristic-based inference, the probability of each gate $V_l(j)$ in $S(V_n(i))$ being mapped to $V_n(i)$ will be $P_i(V_l(j)) = \frac{1}{|S(V_n(i))|}$. With heuristics considered, the distribution of the probabilities is biased towards the more possible choices. We use $\frac{M}{N}$ to decide whether to prune $V_l(j)$ from $\phi(V_n(i))$: If $\frac{M}{N}$ is smaller than $\frac{1}{|S(V_n(i))|}$, then the likelihood of $V_l(j)$ being mapped to $V_n(i)$ is even smaller than randomly choosing a gate in $S(V_n(i))$ and mapping it to $V_n(i)$. This reflects that a placement that optimizes wirelength may not put $V_n(i)$ at the location of $V_l(j)$ and the existence of $V_l(j)$ in $\phi(V_n(i))$ is largely due to the vast exploration of solution space in SA process. So we prune $V_l(j)$ from $\phi(V_n(i))$.

Formally, for any $V_l(j)$ in $\phi(V_n(i))$, we calculate $P_i(V_l(j))$ as $\frac{n_{ij}}{N}$, where n_{ij} is the number of the mapping solutions that map $V_n(i)$ to $V_l(j)$. To give more flexibility to this method, we use $\frac{\alpha}{|S(V_n(i))|}$ rather than $\frac{1}{|S(V_n(i))|}$ for comparison, where $\alpha \in [0, 2]$ is the parameter that controls the pruning power: Higher value prunes more from $\phi(V_n(i))$ and lower value prunes less. The appropriate value for α is determined in experiment.

The difference between pruning process and the previous mapping process discussed in Section IV-A lies in their different goals: the mapping process aims to find all possible locations for a gate in wirelength-opmitized placement so that the attacker will not miss the real location; the pruning process aims to further prune the possible locations in order to reduce the cost and risk of attacks. We refer to the possible locations after pruning as the candidate locations for a gate. The attacker may implant HT either at all or stochastically at any subset of these locations, based on his attacking capability.

V. Defense

To fulfill the security provided by split manufacturing, physical design techniques need to be deployed in placement stage to reduce the information revealed by design heuristics. Local movement of gates [9] is usually applied to disrupt the proximity of connected gates. While it performs well against greedy proximity attack, its performances are not naturally generalized to global information based attack. To see this, we use the example circuit in Fig. 2. There are one NOR2,

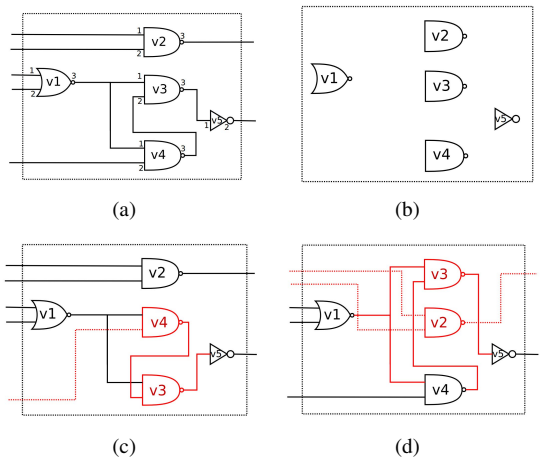

Fig. 2. Example circuit for defense: (a) original circuit, (b) FEOL for split after M1, (c) circuit with v3, v4 swapped, and (d) circuit with v2, v3 swapped.

one INV and three NAND2 gates in this circuit. Fig. 2(a) is the original layout. We assume that the fabrication is split after M1 layer, so the attacker cannot see any inter-cell connections just as in Fig. 2(b). Despite the absence of connections, the attacker still can correctly reconstruct most connections even without gate-level netlist because the circuit is well organized in terms of pin positions. Now, if we only consider disrupting pin proximity and swap v3 with v4 as in Fig. 2(c), the attack effectiveness is mitigated. But for a HT attacker who has netlist, he can get around the intentionally misleading pin positions by using the global wirelength as the heuristic: When swapping v3 with v4, the global wirelength of the circuit is only affected smally, so the attacker will treat v3 and v4 as mutually interchangeable gates and implant HT at both locations. Although the cost of the attack increases, the effort in defense is much more nullified. On the other hand, if the defender swaps v2 with v3 as shown in Fig. 2(d), the global wirelength is significantly changed and the attacker will not regard this placement as a possible one, which prevents v3 from being correctly attacked. Therefore, although it is undesirable to have increased wirelength, for the sake of security in critical chips, there is a tradeoff for us to explore sometimes.

To counter the threat from a more powerful HT attacker, we propose a defense method that incorporates global wirelength information. The goal of this defense is to hide gates from their candidate locations, which means to decrease EMSR. As the defender is required to know the candidate locations obtained by the attack, he first needs to go through the entire attack process to collect candidate locations for each gate. Then a greedy gate-swapping-based defense algorithm is used, which is shown in Algorithm 1. The goal is to swap the locations of gates so that they are not in one of their candidate locations obtained by the attack.

Algorithm 1 Greedy Gate-Swapping-Based Defense

Input:
The candidate locations for each gate ϕ, the original placement
Output:
The placement with improved security

$G \leftarrow V_n$
Ascendingly sort all the gates in G based on the number of their candidate locations
while $G \neq \emptyset$ **do**
  ToSwap $\leftarrow \emptyset$
  Pop the first gate $V_n(f)$ from G and add it to ToSwap
  Find all the gates in G whose number of candidate locations equals $|\phi(V_n(f))|$, pop them from G and add into ToSwap
  while ToSwap $\neq \emptyset$ **do**
    for each gate g in $ToSwap$ **do**
      if $g \notin \phi(g)$ or $|\phi(g)| = S(g)$ **then**
        Pop g from ToSwap
      else
        for each gate g_o such that $g_o \in S(g)$ and $g_o \notin \phi(g)$ **do**
          Get the security elevation and wirelength increase if g swaps its location with g_o
        end for
      end if
    end for
    Get the pair of g and corresponding g_o that gives highest nonzero security elevation. If multiple pairs have the same security elevation, get the one with least wirelength increase
    Swap the locations of g and g_o
    Pop g from ToSwap
  end while
end while
return The placement with improved security;

We start from the gates with fewest possible candidates because they are most insecure [6]. For each gate g, we find its out-of-candidate gates, which is defined as the union set of each gate g_o that is with the same gate type as g but not in the candidate locations of g. Then the security

elevation for the location swapping between g and any g_o is measured using Algorithm 2. The reason we only consider swappings between same-type gates is that in this case, no matter how swappings are made, the candidate locations for each gate remain unchanged. So if a gate is swapped to an out-of-candidate location, its security elevation will not be nullified by swappings of other gates. The calculation of security elevation is based on the goal that we want to move as many gates to their out-of-candidate locations as possible. For a gate g, if it is not in an out-of-candidate location, then its probability of being correctly mapped by random guessing is $\frac{1}{\phi(g)}$. After being swapped to an out-of-candidate location, its probability of being correctly mapped becomes 0. So its security elevation is computed as $\frac{1}{\phi(g)}$. However, we want to enforce more gates to out-of-candidate locations, so we add security elevation by 1 if a gate is moved from an in-candidate location to an out-of-candidate location.

Algorithm 2 Security Elevation Calculation

Input:
Two gates, g and g_o

Output:
The amount of security elevation if the locations for g and g_o are swapped

1: **if** $g \notin \phi(g_o)$ and $g_o \notin \phi(g)$ **then**
2: SecurityElevation$= 2 + \frac{1}{|\phi(g)|} + \frac{1}{|\phi(g_o)|}$
3: **else if** $g_o \notin \phi(g_o)$ and $g \in \phi(g_o)$ **then**
4: SecurityElevation$= \frac{1}{|\phi(g)|} - \frac{1}{|\phi(g_o)|}$
5: **else**
6: SecurityElevation$= 1 + \frac{1}{|\phi(g)|}$
7: **end if**
8: **return** SecurityElevation;

VI. Experimental Results

A. Experimental Setup

The proposed techniques are evaluated on 8 circuits from ISCAS-85 benchmarks [12]. OSU technology library [13] is used and placement is performed by FastPlace3 [14].

B. Number of Simulated Annealing Process

In the first phase of attack, we run the SA process for a number of times to eliminate improbable locations while letting the correct locations to be included in mapped sets. If the number of run times is too small, correct locations are likely to be excluded from mapped sets. On the other hand, if the number of run times is too large, then it would be a waste of time after correct locations are already included.

TABLE I
Correlation between maximum number of same-type instances and the required number of SA runs to reach $> 95\%$ EMSR

Circuit	c432	c499	c1908	c2670	c3540	c5315	c6288	c7552
#max same-type	38	190	208	459	547	1016	623	1172
#SA runs	54	196	455	673	944	1357	881	2398

In our experiments, we found that the proper number for run times is related to the maximum number of same-type instances in the circuit because they have most possible locations to choose from. Table I shows the maximum number of same-type instances and the required number of SA runs to reach $> 95\%$ EMSR for benchmarks. A suitable choice for $\#SA\ runs$ is two times of $\#max\ same\text{-}type$.

C. Effectiveness of Attack

To save computation time, each SA process is run with a fast mode without exhaustively searching the solution space. For mapped set pruning, we tries out different value of α for each benchmark. Fig. 3 shows the EMSR and AMSPR of each

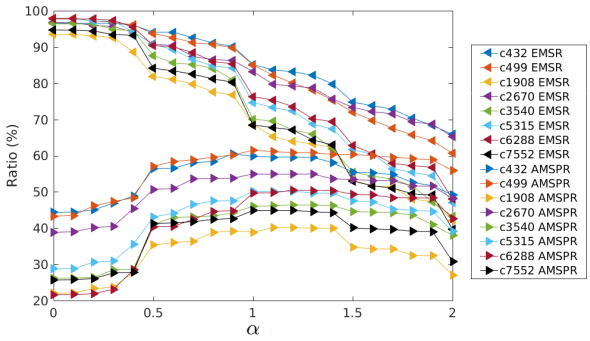

Fig. 3. Effective Mapped Set Ratio (EMSR) & Average Mapped Set Pruning Ratio (AMSPR) w.r.t. the pruning parameter α.

benchmark, from which we can see that $\alpha = 0.9$ gives a good tradeoff between EMSR and AMSPR. If α increases more, then EMSR will suffer from quick reduction while AMSPR grows slowly. This is because many correct locations are not often included in a mapping solution. If we prune the mapped sets too much, then large amount of correct locations will be pruned out. The average EMSR at $\alpha = 0.9$ is 85% while the average EMSR at $\alpha = 0$ is 95%, which means only 10% mapped sets become ineffective during pruning. On the other hand, the average AMSPR of all benchmarks rises to nearly 50%, which means half of the initial possible locations can be pruned out, halving the cost and risk of the attacker.

Note that the effectiveness of the attack is not influenced much by circuit scale. For example, c6288 has more than 5 times the number of gates than c1908 but the attack is more effective on c6288 in terms of both metrics. The underlying impact on the effectiveness of the attack comes from the diversity of a circuit. c1908 has only 8 gate types including primary input/output and almost half of the gates belong to a same gate type. In comparison, c6288 has 15 gate types with the maximum number of same-type instances being only $\frac{1}{5}$ of the total number of instances.

Our attack approach not only performs well on global metrics such as EMSR and AMSPR, but also prunes many mapped sets to a small size without turning the mapped sets into ineffective. Fig. 4 shows the changes in the distribution of the sizes of mapped sets of c432. Fig. 4(a) is the distribution before attack while Fig. 4(b) is the distribution after attack. For ineffective mapped sets, their set sizes are restored to their initial sizes, i.e., the numbers of gates with corresponding types. In Fig. 4(b), most mapped sets reside in the region of $size < 10$, which largely reduce the cost and risk of the attacker. For each benchmark circuit, Table II shows the number of mapped sets of different sizes before and after attack, where many mapped sets have their sizes reduced to be under 30 after attack.

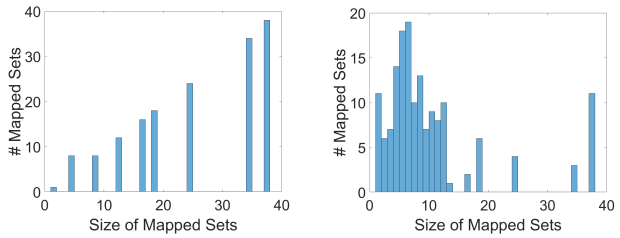

Fig. 4. Distribution of the sizes of mapped sets of c432: (a) before attack, (b) after attack.

TABLE II
THE NUMBER OF MAPPED SETS OF DIFFERENT SIZES: B MEANS BEFORE ATTACK, A MEANS AFTER ATTACK

Circuit	Size of mapped set									
	$[0, 10)$		$[10, 30)$		$[30, 100)$		$[100, 200)$		$[200, +\infty)$	
	B	A	B	A	B	A	B	A	B	A
c432	17	105	70	40	72	14	0	0	0	0
c499	0	80	116	272	256	179	190	31	0	0
c1908	1	6	0	100	162	270	150	40	208	105
c2670	0	14	0	106	126	392	271	493	779	171
c3540	0	0	0	28	169	256	112	931	1365	431
c5315	0	3	0	11	0	343	168	723	2676	1764
c6288	0	27	47	149	204	452	294	1270	2411	1058
c7552	0	2	0	15	0	268	0	653	3733	2795

D. Effectiveness of Defense

The effectiveness of defense is directly measured by EMSR since the goal of the defender is to hide gates away from their mapped sets. Figure 5 shows the reduction of EMSR after using the proposed defense method. We can see that the EMSR of all benchmarks goes below 30%, with an average reduction of 63.4%. With only a small number of effective mapped sets left, the defense method moves most gates to locations that are immune to the global wirelength based attack. Besides, since we assign higher priority to more insecure gates, i.e., the gates with smaller mapped sets, the number of small effective mapped sets will decrease, significantly elevating the cost and risk of the attacker.

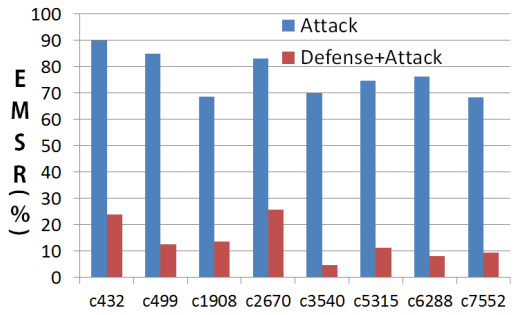

Fig. 5. Effectiveness of our defense approach.

E. Security–Wirelength Overhead Tradeoff

Since our defense method can move gates to places that wirelength-driven placement does not choose, the security against attacks that utilize global wirelength heuristic is guaranteed. Also, the gates in out-of-candidate locations are secure in a solid way, which means the best chance for the attacker is to randomly guess their locations. Yet, extensively applying this method to all gates is expensive.

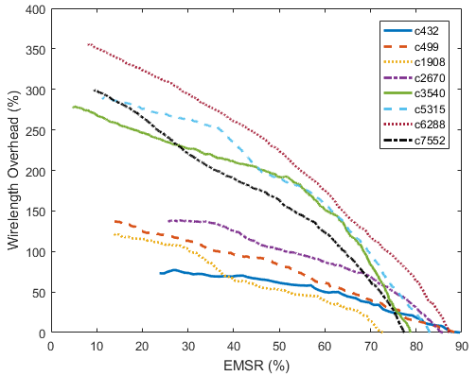

Fig. 6. Security vs. Wirelength Overhead.

Fig. 6 shows the tradeoff between EMSR and wirelength overhead for all benchmarks. As EMSR decreases during gate swapping, the wirelength overhead increases approximately linearly with EMSR. Thus, it is more cost-effective to use this defense on the most insecure gates because the wirelength overhead is not related to the security of the gates to be swapped. The defender can also set a wirelength overhead budget and only allow defense within this budget.

VII. CONCLUSION

In this work, we have shown that split manufacturing is not secure and the attacker can get necessary information through simulated annealing based attack approach at the placement level. We further proposed an effective defense approach to protect the gates away from the attack.

REFERENCES

[1] Y. Jin, "Introduction to hardware security," Electronics, vol. 4, pp. 763–784, October 2015.
[2] M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, "Hardware security: Threat models and metrics," in ICCAD, 2013, pp. 819–823.
[3] H. Li, Q. Liu, and J. Zhang, "A survey of hardware trojan threat and defense," Integration, the VLSI Journal, 2016.
[4] N. Jacob, D. Merli, J. Heyszl, and G. Sigl, "Hardware trojans: current challenges and approaches," IET Computers & Digital Techniques, vol. 8, no. 6, pp. 264–273, 2014.
[5] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in DATE, 2013, pp. 1259–1264.
[6] 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 USENIX, 2013, pp. 495–510.
[7] K. Vaidyanathan, B. P. Das, E. Sumbul, R. Liu, and L. Pileggi, "Building trusted ICs using split fabrication," in HOST, 2014, pp. 1–6.
[8] M. Jagasivamani, P. Gadfort, M. Sika, M. Bajura, and M. Fritze, "Split-fabrication obfuscation: Metrics and techniques," in HOST, 2014, pp. 7–12.
[9] Y. Wang, P. Chen, J. Hu, and J. J. Rajendran, "The cat and mouse in split manufacturing," in DAC, 2016, pp. 165:1–165:6.
[10] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and implementing malicious hardware," in USENIX, 2008, pp. 5:1–5:8.
[11] J. Francq and F. Frick, "Introduction to hardware trojan detection methods," in DATE, 2015, pp. 770–775.
[12] M. C. Hansen, H. Yalcin, and J. P. Hayes, "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering," IEEE Design & Test, vol. 16, no. 3, pp. 72–80, 1999.
[13] "System on Chip (SoC) Design Flows," available at http://vlsiarch.ecen.okstate.edu/flow/.
[14] N. Viswanathan, M. Pan, and C. Chu, "Fastplace 3.0: a fast multilevel quadratic placement algorithm with placement congestion control," in ASP-DAC, 2007, pp. 135–140.