This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2957359, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Reverse Engineering for 2.5D Split Manufactured ICs

Wei-Che Wang, Yizhang Wu and Puneet Gupta
Department of Electrical and Computer Engineering, University of California, Los Angeles

***Abstract*—Integrated circuit (IC) split manufacturing has been shown to be one of the most effective protection schemes to prevent reverse engineering from malicious foundries. Among the existing split manufacturing approaches, the 2.5D split manufacturing using silicon interposer has much less fabrication and testing costs compared to layer splitting approaches. In this paper we propose a Boolean Satisfiability(SAT)-based attack to reconstruct the wire connections of the 2.5D split manufacturing netlists. Our SAT-based attack can fully reconstruct the missing wires between modules with 100% correctness and therefore the functionality of the chip can be completely reverse engineered. In addition, we show that the runtime of attack is significantly reduced compared to existing 2.5D split manufacturing SAT attacks by applying grouping hints obtained from a Satisfiability Modulo Theories (SMT)-based grouping algorithm, which is purely depending on the circuit functionality, so no physical defensive mechanisms can prevent such attack. In our experiments we show that our grouping algorithm can speed up the existing SAT attack runtime by more than 1,000X and can successfully reverse engineer reasonable size benchmarks even when the split nets contains more than one fanouts and the total cut size is close to 1,000.**

I. Introduction

The globalization of Integrated circuit (IC) supply chain due to the higher fabrication cost and increasing complexity of modern designs has led to new security threats including IC overproduction and reverse engineering [1]. In the cost-effective fabless model, the foundries that the IC/IP owner outsourced the design to might not be trustworthy. Once the foundry obtains the whole GDSII of the design, it can overproduce or perform reverse engineering to obtain all design details, which leads to significant revenue lost. Split manufacturing, as one of the most promising defensive mechanisms to prevent foundry reverse engineering, has been studied intensively in recent years.

A. Layer-based Split Manufacturing

To protect ICs from the malicious foundries or attackers, Layer-based Split Manufacturing (LSM) has been proposed as a protecting mechanism to minimize the aforementioned risks [2]. LSM divides a design into Front End of Line (FEOL) and Back End of Line (BEOL) parts, and different parts are fabricated at different foundries. The FEOL (higher complexity and cost) part is fabricated at an untrusted foundry. Since the complete connections of the circuit are unknown to the untrusted foundry, the design cannot be fully reverse engineered. After the FEOL fabrication, the wafer is shipped back to an onshore trusted foundry for the BEOL fabrication and integration. While LSM may fit well with the advanced 3D IC fabrication model, however, the yield loss due to wafer transportation, integration, and the requirement of design rule compatibility of two foundries are still remaining as the major challenges [3]. Also, the cost of splitting lower metal layers can induce even higher cost [4], while splitting at higher layers may not provide sufficient security [5].

B. Module-based Split Manufacturing

Another split manufacturing strategy is the Module-based Split Manufacturing (MSM), which is a promising IC integration technology that is designed to improve system performance by using silicon interposers [6]. MSM offers a high density and low cost package system [7] with inter-chip bandwidth benefits and power reduction [8]. 2.5D interposer products are already commercially available, such as AMD Radeon Fury GPU [9] and NVIDIA Tesla Accelerator [10]. Therefore, the security of MSM has become more and more important for the maturity of the technology, and research efforts have been devoted to this area, including security-purpose 2.5D integration [11], attacking, and defending techniques of MSM [12], [13].

Compared to LSM, the advantages of MSM include:

1) Alignment and integration: The integration of LSM is more challenging than MSM because of the larger pitch size of the interposer compared to metal layers. Also, each module of MSM can be packaged and tested as normal chip before being sent to the integrator.
2) Fabrication compatibility: No design rule compatibility requirements for MSM since the connection is done through chip-to-chip interposers.

C. Threat Model of the Proposed Attack

We focus on reconstructing the missing connections of the MSM strategy given the following assumptions:

1) The adversaries are malicious foundries who have access to all the MSM parts including both FEOL and BEOL of the design.
2) Once the design is integrated at a trusted facility and sold to the open market, the adversaries can obtain the functional chip to find out correct input/output pairs.

Our threat model is the same as existing SAT-based MSM attacks [12], [13], [14], where no physical reverse engineering of the functional chip is required. Please note that for the proximity attack on LSM [15], the BEOL and complete function of the chip are unknown and 100% correctness is not guaranteed. Our attack is on MSM connections and guarantees 100% correctness assuming that the complete function is available. The only thing the adversary does not know is the connection or interposers between the splitting parts. Since the splitting parts are known to the adversary, the intermediate input/output at the splitting interface are also known by the adversary.

D. Problem Formation

The goal of our attack is to reconstruct the missing connections of the MSM components without costly reverse engineering of the interposers. Figure 1 shows an example of a circuit split using the MSM strategy. Partition 1 and partition 2 can be either fabricated at the same or different foundries but none of them know the final connections between the two parts as indicated by the circles. The split can have no-fanout as indicated in Figure 1 (a) or include the fanout (3x4) as shown in Figure 1 (b). The cut nets of both fanout and no-fanout splits can be the same but the cut sizes are different. For both splits it is impractical for the adversary to brute-force all connections to find a correct solution simply because of the size of the solution space.

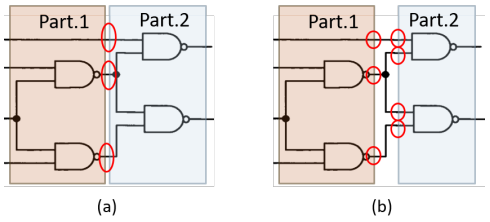

Fig. 1: MSM example (a) without fanout and (b) with fanout split. Partition 1 and partition 2 can be fabricated at the same or different foundries but the connections between them are hidden. The goal of the adversary is to connect outputs of partition 1 to the inputs of partition 2 correctly.

To address this problem, we propose to use a Boolean Satisfiability (SAT) solver with hints obtained from circuit function analysis. Different from the SAT attack proposed in [14], we introduce grouping hints that can significantly reduce the runtime of the SAT attack. The main contributions of this paper include:

- Two SAT-based attacks are compared. Results show that the grouping hint is much more effective than the no-fanout hint,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2957359, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

which turns out to be even worse than without applying any hint due to the increase of virtual gate counts.
- Hard grouping algorithm and soft grouping strategies are proposed to significantly reduce the runtime of SAT attack. The hard grouping algorithm is independent of physical implementation of the split therefore no physical defensive mechanism, such as place and route perturbation, can be effective. The use of soft grouping strategies can further reduce attack runtime using the hints obtained from physical implementations of the design.

II. The SAT-Based Attack

A. SAT Attack Modeling

In this section we model the reconstruction of missing connections of split manufacturing as SAT-based attack proposed in [16], which has been shown to be an effective attack to retrieve the correct key of many logic obfuscation schemes. The SAT-based attack algorithm allows the adversary to decrypt an obfuscated netlist using a small amount of input patterns and their corresponding outputs (distinguishing input/output pairs or DIPs) from a functional circuit. The algorithm iteratively finds such DIPs and formalize them as a sequence of SAT formulas to be solved by a SAT solver. Each DIP can rule out a subset of wrong keys and the algorithm is guaranteed to find an equivalent class of the correct key.

To formulate the problem the first step is to model the missing connections with virtual multiplexer (mux) or demultiplexer (demux) gates with selection keys. As shown in Figure 2, there are two possible ways to model the connections with a cut size of 3 MSM. Figure 2 (a) shows a connection network using 3 mux gates where each of the mux is configured by a key k_i. The network models that m_i can be connected to anyone of the n_i wires. Figure 2 (b) shows a connection network using 3 demux gates representing a model that each n_i can only connect to one m_i, which is the ORed value of all demuxes. One of the demux outputs will be n_i and others will be zero depending on the key k_i of the demux. This model constrains the connection to be no-fanout while maintaining the same size of keys as in Figure 2 (a), which is $n * log(n)$ bits for a cut size of n.

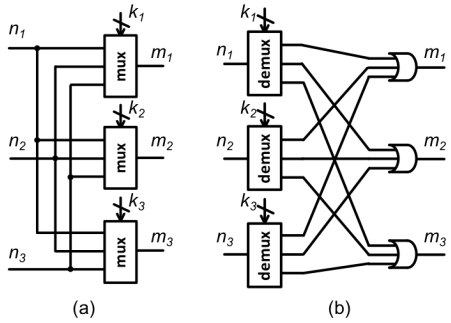

Fig. 2: Modeling example of no-fanout MSM with cut size of 3. (a) mux network (b) demux network.

Combining the virtual connection network and the split parts we can now apply SAT attack to the design to solve the keys. The objective is to retrieve the correct values of all key bits in order to reconstruct the missing connections. The attacking model assumes we have access to following aspects:

1) The gate-level netlists of both partition 1 and partition 2. Along with the model for missing wires from partition 1 to partition 2, the conjunction normal form (CNF) $C(X, K, Y)$ of whole design can be obtained and the function of the design is represented as $Y = f(X, K)$, where X is the primary input of the circuits and $K = (k_1, k_2, ..., k_i)$ is the selection keys to all mux or demux gates.
2) A fully functional chip obtained from the market, from which an adversary can observe the correct output of the circuit given an input $Y = eval(X)$

TABLE I: mux and demux network runtime results in seconds.

circuits	Cut size	mux	demux	Cut size	mux	demux
c3540	52	90	159	115	1,588	TO
c5315	93	690	3,108	120	TO	TO
c7552	50	257	604	108	TO	TO
seq	70	165	682	165	TO	TO
apex4	47	26	25	251	20,666	TO
ex1010	72	2,007	1,060	281	TO	TO
DES	85	38	219	346	8,339	TO

TABLE II: mux network runtime results in seconds with grouping hints.

circuits	Cut size	No Hint	50%	33%	20%
c3540	115	1,588	379	143	79
c5315	120	TO	907	128	39
c7552	108	TO	TO	10,824	378
seq	165	TO	TO	895	244
apex4	251	20,666	4,574	2,791	974
ex1010	281	TO	TO	6,568	3,923
DES	346	8,339	2,696	1,386	527

B. Runtime Results

In our experiments we use ISCAS85 and Microelectronics Center of North Carolina (MCNC) benchmarks to evaluate the runtime of two connection networks. All runtime results in our experiments are measured on a 2.7GHz Intel Core i5 CPU with 8GB memory. The attack terminates either when correct keys are found or the runtime is larger than 24 hours (TO). Once a correct key is found, the circuit will behave exactly the same as the original circuit before split. Different sizes of cut nets are tried and the attack tries to find a key that matches all outputs. Table I shows the runtime of the mux and demux networks. There are multiple ways to cut nets for a design while maintaining the balance of size of partitions but in general the runtime is proportional to the cut size. For some benchmarks if the cut size is larger than 100 then it becomes difficult to find the correct key. The demux network models the connection in such a way that every output of partition 1 can only connect to one input of partition 2, which exploits the information to the adversary that the connections are without fanout for the split shown in Figure 1 (a). However, except for some small cut sizes, most runtimes of demux networks are much larger than the mux network even though the size of the keys are the same. One possible reason is that there are n OR gates each with n inputs in the demux network, which increases the number of clauses significantly in CNF and can slow down the SAT solver [17]. In the rest of the paper we will focus on the mux connection work attacks.

C. Grouping Hints

From Table I we know that the runtime of solving the key can be effected significantly by the complexity of the connection network in addition to the cut size. Therefore, one way to reduce the runtime is to use a simpler connection network that translates to fewer CNFs. In other words, if the connections can be represented by smaller mux gates the runtime can be significantly reduced. One approach to reduce the mux network complexity is to apply a grouping hint to each of the mux, which contains the information of the candidates from partition 1 to partition 2.

With grouping hints the adversary can model the connection with a smaller mux network because now the candidate connections of m_i are not all n_i but can be a sub-group of n_i. For example, the key length for the no-fanout split is not $n * log(n)$ anymore but becomes $n * log(pn)$ where p is the grouping hint percentage, which means that an input of partition 2 can only be connected to p portion of connections from partition 1. Table II shows the results when different p's are imposed to the mux connection network. We can see that some testbenches show significant runtime reduction when 50% of p is imposed, and as p keeps getting smaller, all benchmarks can be solved and most of the runtime are almost hundred times smaller compared to the no-hint cases.

From Table II we know that grouping can help reduce the runtime significantly. However a wrong grouping hint can cause the SAT solver a long runtime yet still cannot find the correct solution,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2957359, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

therefore the grouping hints should be carefully computed. To address this issue we propose an algorithm to find hard grouping hints that are guaranteed to include correct groupings irrespective to physical constraints and routing heuristics. Details of soft grouping hints, which do not guarantee to include the correct connections, will be discussed in Section IV.

III. Hard Grouping Hints and Results

A. Hard Grouping Algorithm

In this section we present the algorithm to find hard grouping hints which guarantees to include correct connections. Protective schemes that are effective to proximity attack [18] do not protect the design from hard grouping algorithm because hard grouping hints are completely independent of how the circuit is physically implemented. For a cut size of n, define the n-bit output from partition 1 as ***Z1***=$(z1_1, z1_2, ...z1_n)$ and input to partition 2 as ***Z2***=$(z2_1, z2_2, ...z2_n)$. The goal of the hard grouping algorithm is to find the candidate connections of each $z2_i$ from ***Z1***.

The hard grouping algorithm is implemented in Python as a Satisfiability Modulo Theories (SMT) problem and solved by an existing SMT solver [19], which is a verification engine that understands a satisfiability problem at a higher level of abstraction other than Boolean formulas while still retaining the speed and efficiency of modern SAT engines.

The hard grouping algorithm first assigns ***Z1*** with fixed number of bits being one (or zero), which we note as *hot bits*, to SMT solver and it finds a valid input-output pair (X, Y) of the whole complete design obtained from open market to generate such ***Z1*** for partition 1. Next step is to find all possible patterns of ***Z2*** with the same number of hot bits which can reproduce the same Y of the whole complete design. The grouping information can then be found by mapping hot bits in ***Z1*** to hot bits in each of ***Z2***.

In our attacking model, the adversary has access to (1) gate-level netlists of both partition 1 and partition 2 such that partitions can be represented as a set of SMT formula, $S(X, Z1, Z2, Y)$, and (2) a fully functional chip obtained from the market which can be used to observe the correct output given an input, $Y = eval(X)$. The algorithm for finding hard grouping hints for n-bit ***Z1*** using a specified number of hot bits hb, and hot bit type (1 or 0), hb_type, is shown in Algorithm 1. It returns a map from each net in ***Z1*** to all possible candidates in ***Z2***.

A simple example to illustrate the idea of the algorithm is given in Figure 3: a ***Z1***=(11000) and its corresponding (X, Y) is found. Under such constraints, assume ***Z2***=(00110) is the only solution found to generate the same Y for partition 2. Now we know that $z1_1$ and $z1_2$ can only connect to $z2_3$ and $z2_4$ (all locations with 1's in ***Z2***), and all other connections of ***Z1*** can only connect to $z2_1$, $z2_2$, and $z2_5$ (all locations with 0's in ***Z2***). If there are multiple solutions of ***Z2*** for the same ***Z1*** and (X, Y), the union of the groupings should be the final grouping found for the ***Z1***. For every ***Z1*** with different solutions we take the intersection of the groupings found so far to obtain smaller grouping sizes because a correct connection should always exist no matter what inputs or ***Z1***s are.

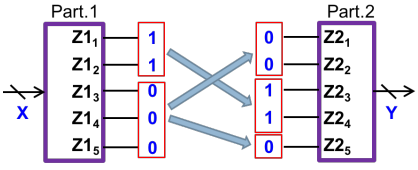

Fig. 3: Hard grouping example.

To enhance the speed of this algorithm, we introduce the concept of Distinguishing ***Z2***($Z2^d$). For a fixed ***Z1***, it is only necessary to find ***Z2*** which can reveal new grouping information instead of all possible solutions of ***Z2***. For instance, if $Z1 = 01100$ and we have found $Z2 = 00110$ and 01010, from the perspective of bits of value 1 in ***Z1***, we know that possible candidates are $z2_2$, $z2_3$ and $z2_4$. The next ***Z2*** to be found is distinguishing if and only if it can reveal new candidates, which are $z2_1$ and $z2_5$. Thus, $Z2 = 10100$ is a $Z2^d$ as it reveals $Z2_1$ as an additional possible candidates while $Z2 = 01100$ is not distinguishing. Applying the constraints of finding distinguishing ***Z2*** after a new ***Z2*** is found speeds up the algorithm significantly.

Algorithm 1 SMT Find_Grouping Algorithm

```
1:  function FIND_GROUP(eval, hb_type, hb)
2:      i = 1
3:      F = S(X, Z1, Z2, Y) ∧ (Y = eval(X))
4:      F = F ∧ (number of hb_type in Z1 and Z2 = hb)
5:      while sat[F] do
6:          Z1_i = smt_assignment_Z1[F]
7:          X_i = smt_assignment_X[F]
8:          F_new = F ∧ (Z1 = Z1_i) ∧ (X = X_i)
9:          j = 1
10:         while sat[F_new] do
11:             Z2_j = smt_assignment_Z1[F_new]
12:             for 1's in Z1_i do
13:                 Group_one = Group_one ∪ (1's in Z2_j)
14:             for 0's in Z1_i do
15:                 Group_zero = Group_zero ∪ (0's in Z2_j)
16:             F_new = F_new ∧ (Z2 ≠ Z2_j^d) ∧ (Z2 is a Z2^d)
17:             j = j + 1
18:         F = F ∧ (Z1 ≠ Z1_i)
19:         i = i + 1
20:     for all z1_k in Z1 do
21:         Group[z1_k] = Group_one[z1_k] ∩ Group_zero[z1_k]
        return Group
```

B. Number of Hot Bits

In cases of large cut size, constraints in line 4 of Algorithm 1 are usually unsatifiable if desired number of hot bits hb in ***Z1*** and ***Z2*** is small. For example in DES with no-fanout cut size 346, it is unsatifiable to find an input X to generate only one 1 and 345 0's at the output of partition 1. Therefore starting with one hot bit may be an inefficient approach.

Figure 4 shows the decoupled runtime of different ending hb of DES with 346 cut size starting from 173 hot bits to the ending hot bits indicated. Note that we start with number of hot bits being half of the cut size because we try both 1's and 0's as hot bit types. We can see that the smaller the ending hb is, the faster it is for the SAT attack to find the connection because the size of groups are smaller, however the time spent on the grouping algorithm also become longer simply because the number of iterations the algorithm is executed. For some small ending hb the grouping time itself is already longer than the total time. Therefore from empirical observations we propose to start Algorithm 1 with hb being half of the cut size and decrease hb by one until more than half of group sizes are smaller than 20% or when the overall group sizes are not getting smaller. The complete algorithm for finding hard grouping is shown in Algorithm 2.

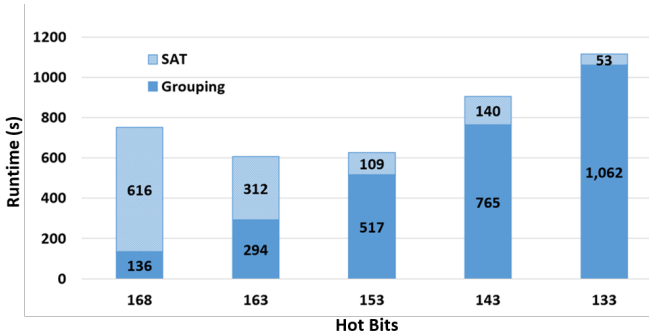

Fig. 4: DES runtime with 346 cut size and different ending hot bits.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2957359, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Algorithm 2 Complete SMT Grouping Algorithm

Input: SMT_formula and *eval*
Output: *Final_Group*

1: **for** i =*half of cut size* **do**
2: $Group_{one\_i} = FIND\_GROUP(eval, 1, i)$
3: $Group_{zero\_i} = FIND\_GROUP(eval, 0, i)$
4: $Group = Group \cap Group_{one\_i} \cap Group_{zero\_i}$
5: **if** $Group \leq 20\%$ **then**
6: *break*
7: $i = i - 1$

TABLE III: Runtime (seconds) and hot bits of hard grouping hints and reduction ratio compared to no hints. The total runtime compared is the sum of grouping and SAT time.

circuits	Cut size	hot bits	Grouping	SAT	Ratio
c3540	115	53	51	5	28.4
c5315	120	55	23	37	>1,440.0
c7552	108	50	15	210	>384.0
seq	165	71	169	493	>130.5
apex4	251	117	482	233	28.9
ex1010	281	125	648	6,173	>12.7
DES	346	163	294	312	13.8

C. Hard Grouping Results without Fanout

In Table III we show the runtime of the hard grouping algorithm and SAT attack algorithm after applying hard grouping hints. The reduction ratio is defined as the runtime of SAT time with no hints divided by the summation of grouping runtime and the SAT runtime after grouping. The reduction ratio for designs with original runtime larger than 24 hours (TO) is calculated as if the original runtime is 24 hours, which gives the lower bound of the reduction ratio. Compared to original SAT runtime in Table I, the total runtime with hard hints has been improved by 13X to more than 1,440X. Number of hot bits are also presented in the table. In general, the algorithm stops when the number of hot bits is within 10 bits less than the starting point, which is half of the cut size.

Figure 5 shows the group size distribution of DES with 346 cut size after running Algorithm 2. Most group sizes are smaller than 20% of 346. More than 58% of nets have size less than 10 and about 40% of the connections have group size of 1, which means that these connections are already determined during the grouping procedure.

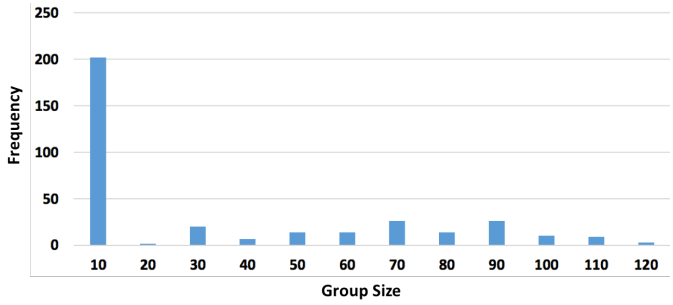

Fig. 5: Group size distribution of DES.

D. Hard Grouping with Fanout

When splitting with fanout with number of bits in ***Z2*** is larger than ***Z1*** as shown in Figure 1 (b), line 4 in Algorithm 1 can be modified to $\boldsymbol{F} = \boldsymbol{F} \wedge$ (number of hb_types in Z1$= hb) \wedge$ (number of 1's/0's in Z2>=number of 1's/0's in Z1) to solve for the grouping. The idea is that the number of 1's/0's in ***Z2*** should be greater or equal to the number of 1's/0's in ***Z1***. For example, if $Z1 = (100)$, and there are five bits of ***Z2***, then all possible solutions of $Z2$ containing one to three 1's need to be found to construct the grouping from ***Z1*** to ***Z2***.

Table IV shows the runtime of fanout split with and without hard hints. Compared to Table II, the runtime of split with fanout on the same nets are much longer than split without fanout. For most benchmarks the key cannot be resolved in 24 hours without hints, but with hard grouping hints the runtime can be significantly improved.

TABLE IV: Runtime (second) of fanout split with hard grouping hints and reduction ratio compared to no hints. The total runtime compared is the sum of grouping and SAT time.

circuits	Cut size	No hint	Grouping	SAT	Ratio
c3540	115x187	4,706	186	27	22.1
c5315	120x269	TO	157	877	>83.6
c7552	108x188	TO	108	299	>212.3
seq	165x239	TO	226	2,839	>28.2
apex4	251x710	TO	36,501	9,727	>1.9
ex1010	281x677	TO	70,240	5,838	>1.1
DES	346x455	TO	1,794	245	>42.4

IV. Soft Grouping Hints and Results

A. Soft Grouping Strategy

Besides hard grouping hints, another way to reduce the mux network complexity is to apply soft hints from physical implementation constraints. Similar to proximity attacks [15], the adversary knows that a wire of partition 1 is likely to connect to the wires that are physically close to itself in partition 2 due to the interposer delay. Figure 6 shows simulated results of interconnect delay and transition slew using a commercial 65nm technology. Each interconnect connecting to the input of an inverter cell is driven by the largest buffer cell available in the standard cell library. Metal 8 with $5um$ width is used to emulate the interposer interconnect wire. We can see that as the wire becomes longer the delay and slew become larger. This information can be exploited by the adversary to narrow down possible connections and thus simplify the connection network. For example, a 2GHz design would require the path delay to be smaller than 500ps. In the 65nm technology we used, a normal setup time for a D flip-flop is about 100ps, which leaves 400ps margin for the total gate delay. As shown in Figure 6 if the wire is longer than 20mm, the wire delay is already larger than 400ps, therefore such connection can be pruned out in the connection network model. Another information is the transition slew. The library defines that the max slew is about 385ps, and if a wire is longer than 10mm the slew becomes larger than 385ps, which tells the adversary that a connection longer than 10mm is not likely to be made.

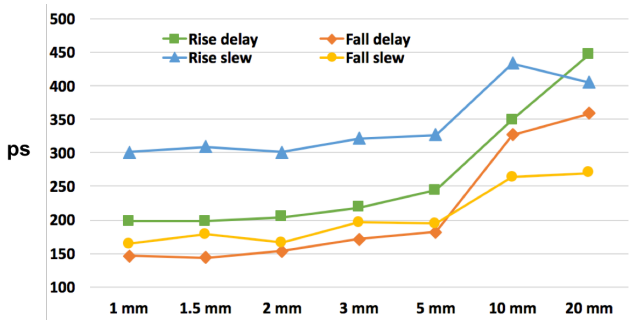

Fig. 6: Interconnect delay and transition slew.

Given the modern GPU design specifications [10], which contains billion of gates and die size as large as about 600mm^2, having a cut size of hundreds of thousands between the two partitions is expected from our empirical observations. Assume the design is split into two parts, each with the dimension of 10mm by 10mm. With existing interposer technology of 50um pitch [20], the allowable number of interposers on each partition is about 40,000, therefore it is possible that most interposer sites will be used after split. The exemplary analysis of delay and transition constrains present in Figure 6 tells the adversary that connections from the left edge of partition 1 to the right edge of the partition 2 is not likely to be made as illustrated in Figure 7, which shows an unlikely connection marked as "X" and possible connections marked as "O".

The difference between hard hints and soft hints is that when soft hints are applied, the correct connections are no longer guaranteed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2957359, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

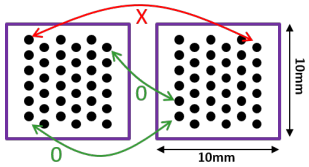

Fig. 7: Interposer soft grouping example. The "X" connection is not likely to happen due to interconnect delay and transition constraints.

TABLE V: Runtime (seconds) of fanout split of hard grouping first followed by 50% soft grouping.

circuits	Cut size	Solved Nets	SAT	Ratio
c3540	115x187	30%	15	23.4
c5315	120x269	12%	401	>154.8
c7552	108x188	16%	99	>417.4
seq	165x239	11%	290	>167.4
apex4	251x710	50%	2,591	>2.2
ex1010	281x677	64%	4,409	>1.2
DES	346x455	56%	60	>46.6

to be included after the grouping, and the keys of the connection network may not be found. This is because the IC/IP designer can perform routing or placement perturbation to violate the physical design principles [15].

B. Results

To integrate soft grouping with hard grouping, the adversary can first apply hard grouping to obtain connections for partial nets and then apply soft hints to the rest of the nets to further reduce the runtime. Table V presents the percentages of solved nets (nets with grouping size one) after hard grouping and runtime results when 50% soft grouping is applied after hard grouping. The 50% soft grouping means that each net can only be connected to the closest 50% possible candidates. We can see that about 11% to 64% of the connections are already solved without even applying SAT attack, and these connections are guaranteed to be correct because they are found by hard grouping algorithms. For those unsolved nets, the runtime of SAT attack is significantly reduced compare to Table IV because of the soft grouping.

Table VI shows the results of 50% soft grouping hints applied before executing the hard grouping algorithm. The runtime is further reduced compared to hard grouping only in Table IV, but the grouping is not guaranteed to include the correct connections. In practical the adversary can apply hard grouping hints first to find solutions for partial nets that are guaranteed to be correct and then apply soft hints to the rest of the nets, or apply the soft groupings that are highly likely to be true before the hard grouping to reduce the runtime. The grouping sequences can be applied in an arbitrary order depending on the actual implementation of the victim design.

V. Defense Strategy

Create floating connections. To defend the hard grouping algorithm, the IC/IP designer can create redundant floating connections at the output of partition 1 to cause confusion or even create unsolvable grouping solutions of Algorithm 1. For example, say ***Z1***=(1001) for a no-fanout split, $z1_4$ is the redundant floating net that does not connect to partition 2 and a corresponding input X of partition 1 is found.

TABLE VI: Runtime (seconds) of fanout split of 50% soft grouping first followed by hard grouping.

circuits	Cut size	No hint	Grouping	SAT	Ratio
c3540	115x187	4,706	179	10	24.9
c5315	120x269	TO	140	158	>289.9
c7552	108x188	TO	99	39	>626.1
seq	165x239	TO	224	407	>136.9
apex4	251x710	TO	4,277	8,978	>6.4
ex1010	281x677	TO	37,961	4,061	>2.3
DES	346x455	TO	1,723	71	>48.2

The algorithm tries to find ***Z2*** with two 1's to generate $Y = eval(X)$, but since $z1_4$ is floating, such solution for ***Z2*** may not exist, so the grouping hints may not be generated. From our experiments we can see that the ability of SAT attack itself is limited, therefore without the help of hard grouping algorithm the overall performance of the attack is significantly weakened.

Split the design into more partitions. To model the connection network the adversary needs to know the topological order of the partitions. When there are only two partitions the order of the partitions can be easily figured out. If there are more than two partitions, finding the topological order becomes a more difficult task and there is no straightforward way to translate our attacking algorithm to solve partitions with unknown orders. The complexity may be too high for the attack to solve the key in practical runtime.

VI. Conclusion and Future Work

In this paper we present SAT attacks to 2.5D split manufacturing based on the hard grouping hints obtained from SMT-based grouping algorithms. We first show that the runtime of SAT attack can be significantly affected by the complexity of the connection network, therefore a simplified network should be used to reduce the runtime. Then we propose hard grouping algorithms to find grouping hints that guarantee to include correct connections to effectively simplify the connection network and reduce the runtime of SAT attack significantly. Our experiments are done on both fanout and no-fanout splits and results show that the runtime is improved by more than hundred times for some testbenches compared to SAT attack without hints. Finally we discuss defense strategies to protect the split manufacturing from our attack. Our future work aims to experiment larger circuits and solve such defense strategies with more powerful grouping algorithms, such as incorporating the LSM proximity attack to the MSM soft grouping algorithm.

References

[1] Randy Torrance and Dick James. The State-of-the-Art in IC Reverse Engineering. In *CHES*, Sep. 2009.
[2] J. Valamehr et al. A 3-D Split Manufacturing Approach to Trustworthy System Development. *IEEE TCAD*, April 2013.
[3] M. Jagasivamani et al. Split-fabrication obfuscation: Metrics and techniques. In *IEEE HOST*, May 2014.
[4] K. Xiao, D. Forte, and M. M. Tehranipoor. Efficient and secure split manufacturing via obfuscated built-in self-authentication. In *IEEE HOST*, May 2015.
[5] K. Vaidyanathan others. Building trusted ICs using split fabrication. In *IEEE HOST*, May 2014.
[6] V. Sundaram et al. Low cost, high performance, and high reliability 2.5D silicon interposer. In *ECTC*, May 2013.
[7] Y. Sun et al. Modeling and fabrication of the redistribution layer on the 2.5D Si interposer. In *ICEPT*, Aug 2017.
[8] Yang Xie, Chongxi Bao, and Ankur Srivastava. Security-Aware Design Flow for 2.5D IC Technology. In *International Workshop on TrustED*, October 2015.
[9] C. Lee et al. An Overview of the Development of a GPU with Integrated HBM on Silicon Interposer. In *ECTC*, May 2016.
[10] NVIDIA Tesla P100. The Most Advanced Datacenter Accelerator Ever Built. In *NVIDIA White Paper*, 2016.
[11] Yang Xie, Chongxi Bao, and Ankur Srivastava. 3D/2.5D IC-Based Obfuscation. In *Hardware Protection through Obfuscation*, Jan 2017.
[12] Satwik Patnaik et al. Best of Both Worlds: Integration of Split Manufacturing and Camouflaging into a Security-Driven CAD Flow for 3D ICs. In *Proc. ICCAD*, 2018.
[13] Peng Gu et al. Cost-efficient 3D Integration to Hinder Reverse Engineering During and After Manufacturing. In *Proc. AsianHOST*, 2018.
[14] Y. Xie, C. Bao, and A. Srivastava. Security-Aware 2.5D Integrated Circuit Design Flow Against Hardware IP Piracy. *Computer*, May 2017.
[15] Y. Wang et al. The cat and mouse in split manufacturing. In *Proc. ACM/IEEE Design Automation Conference*, June 2016.
[16] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *IEEE HOST*, May 2015.
[17] C. Yu et al. Incremental SAT-Based Reverse Engineering of Camouflaged Logic Circuits. *IEEE TCAD*, Oct 2017.
[18] S. Patnaik, M. Ashraf, J. Knechtel, and O. Sinanoglu. Raise Your Game for Split Manufacturing: Restoring the True Functionality Through BEOL. In *Proc. DAC*, June 2018.
[19] Leonardo de Moura and Nikolaj Bjrner. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, April 2008.
[20] K. Cho et al. Signal Integrity Design and Analysis of Silicon Interposer for GPU-Memory Channels in High-Bandwidth Memory Interface. *IEEE TCPMT*, Jan. 2018.