SECURITY ATTACK MODELS FOR SPLIT MANUFACTURING OF INTEGRATED CIRCUITS

A Thesis

by

PU CHEN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Jiang Hu
Committee Members,	Peng Li
	Eun Jung Kim
Head of Department,	Miroslav M. Begovic

December 2016

Major Subject: Computer Engineering

ABSTRACT

Split manufacturing of integrated circuits reduces vulnerabilities introduced by an untrusted foundry by manufacturing only a part of design at an untrusted high-end foundry and the remaining part at a trusted low-end foundry. Unfortunately, a naïve spilt manufacturing alone does not ensure security. An attacker can use proximity attack to undermine the security offered by split manufacturing. However, this attack is applicable only to hierarchical designs.

We propose a physical attack model for split manufacturing for industry-standard/relevant flattened designs. Our attack uses heuristics of physical design tools, which outperform previous attack. We also develop a logic-aware physical attack considering logic redundancy, which identifies incorrect connections effectively. The effectiveness of proposed techniques is demonstrated by simulations on benchmark circuits. Our attack success rate is ~10× that of the proximity attack; our attack predicts 80% of the missing BEOL connections correctly, while the proximity attack predicts only 8% for flattened designs.

ACKNOWLEDGEMENTS

I would first like to thank my advisor Prof. Jiang Hu who enabled me to research on this project and guided me with patience all the time. Whenever I ran into a trouble spot, he would steer me in the right direction with his insightful suggestions. Without his persistent help this thesis would not have been possible.

I would also thank my committee members, Prof. Kim, and Prof. Li, for their valuable comments on this thesis.

Thanks also go to Dr. Jeyavijayan Rajendran and Yujie, for the helpful discussions, and for the sleepless nights we were working together before deadlines.

Finally, I must express my gratitude to my mother, father for their support and encouragement.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Professor Jiang Hu and Professor Peng Li of the Department of the Electrical & Computer Engineering and Professor Eun Jung Kim of the Department of Computer Science.

All other work conducted for the thesis was completed by the student independently.

Funding Source

This work was made possible in part by NSF under Grant Number 1618824. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of NSF.

NOMENCLATURE

IC	Integrated Circuit
IP	Intellectual Property
EDA	Electronic Design Automation
FEOL	Front-end-of-line
BEOL	Back-end-of-line
AT	Arrival Time
RAT	Required Arrival Time
STA	Static Timing Analysis
BDD	Binary Decision Diagram
ROBDD	Reduced Ordered Binary Decision Diagram
ATPG	Automatic Test Pattern Generation
VIA	Vertical Interconnect Access

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

I-A. Motivation

The cost of owning and maintaining a state-of-the-art semiconductor manufacturing facility has become enormously expensive, even several billion dollars [1]. Consequently, only high-end commercial foundries now manufacture high performance, mixed system integrated circuits (ICs), especially at the advanced technology nodes [2]. Without the economies of scale, many of the design companies cannot afford owning and acquiring expensive foundries; hence, outsourcing their fabrication process to these "one- stop-shop" foundries becomes a necessity. Globalization of IC production flow has reduced design complexity and fabrication cost, but it has introduced several security vulnerabilities [3]. An attacker anywhere in the IC supply chain can perform the following attacks: reverse engineering, malicious circuit insertion, counterfeiting, and intellectual property (IP) piracy [2,4–8]. Due to these attacks, the semiconductor industry loses billions of dollars annually [9]. This is because designers have no control over their design in this distributed supply chain, and, more importantly, current electronic design automation (EDA) tools do not consider security as a design objective.

Split manufacturing of integrated circuits reduces vulnerabilities introduced by an untrusted foundry by manufacturing only the front-end-of-line (FEOL) layers at an untrusted high-end foundry and the back-end-of-line (BEOL) layers at a trusted low-end

foundry [2,10–13]. An attacker in the untrusted foundry has access only to an incomplete design, i.e., the FEOL but not the BEOL. Thus, he can neither pirate nor insert Trojans into it. Recently, researchers have successfully fabricated split-manufactured designs with ~0% faults and 5% performance overhead [11,12,14,15], including a 1.3-million-transistor asynchronous FPGA [15]. Moreover, research from industry has shown that split manufacturing can help improve yield [14]. Although promising and feasible, split manufacturing still cannot guarantee security. Heuristics of physical design tools can be utilized to undermine the security offered by split manufacturing, as demonstrated in [9].

I-B. Threat model

The objective of the attacker is to retrieve the missing BEOL connections from the FEOL connections. Since the attacker is in the FEOL foundry, he has access to the technology library. Consequently, he can obtain the following information about logic gates: layout structure, delay, capacitance load, and wire capacitance. Based on this information, an attacker can reverse engineer the FEOL components and thereby, obtains the incomplete gate-level netlist (this netlist lacks the BEOL information). For this purpose, he can use existing reverse-engineering tools [6]. The attacker neither knows the functionality implemented by the design nor has access to an IC that performs that function.

I-C. Previous work

The semiconductor industry proposed split manufacturing in the early 2000s to improve yield by using only defect-free FEOL parts [14]. Recently, Intelligence

Advanced Research Project Agency (IARPA) proposed split manufacturing for security [2]. Split manufacturing is feasible, as several research groups have successfully demonstrated fully functional split-manufactured designs: 32-bit multiplier, DES, and SRAM circuits [11,12,16]; asynchronous FPGA [17]; and RRAM-based split manufacturing [18]. Split manufacturing for analog designs has been proposed [19].
An attack called proximity attack has been in proposed [10]. This attack aims to recover the missing BEOL connections using the physical proximity of the FEOL components and the heuristics of the physical design tools. To thwart this attack, a pin-swapping technique is proposed to swap the block pins in the layout such that the Hamming distance between the outputs of the original design and the design recovered by proximity attack is close to 50% [10]. The disadvantages of this work is that it is applicable only to hierarchical designs, while most designs used by industry are flattened designs.

I-D. Contributions

In this work, we develop a physical attack for flattened designs using a network-flow model. In addition to the proximity heuristic, our framework considers timing constraint, load capacitance constraint and dangling wire hint. Note that most of the hints described in [10] are for hierarchical designs and cannot be used for flattened designs. Apart from that, we develop a logic-aware physical attack by incorporating logic redundancy detection into our network-flow model. Experiments on ISCAS-85 and ITC-99 benchmark circuits demonstrate that our physical attack outperforms proximity attack

[10] for flattened designs by ~10× and the proposed logic-aware physical attack effectively identifies incorrect connections.

CHAPTER II

BACKGROUND

II-A. Split manufacturing

A chip layout can be split into FEOL part, which includes transistor and lower metal layers, and BEOL part, which includes intermediate and top metal layers, as illustrated by Figure 1. A possible split manufacturing-aware IC design flow is shown in Figure 2. A gate level netlist is partitioned into blocks which are then floorplanned and placed. Wires are assigned to different metal layers and routed so that wiring delay and routing congestion are minimized. The layout of the entire design is split into two: one containing only the FEOL layers and the other containing only the BEOL layers. The FEOL part is first manufactured at an untrusted, high-end foundry. Its wafer is then transported to a trusted, relatively low-end foundry, where the BEOL part is manufactured [2].

Split manufacturing aims to improve the security of an IC, by preventing the FEOL foundry from gaining full control of the IC. For instance, without the BEOL layers, it is very difficult for an attacker in the FEOL foundry to identify the "safe" places to insert Trojans or pirate the designs [2], [10] – [13].

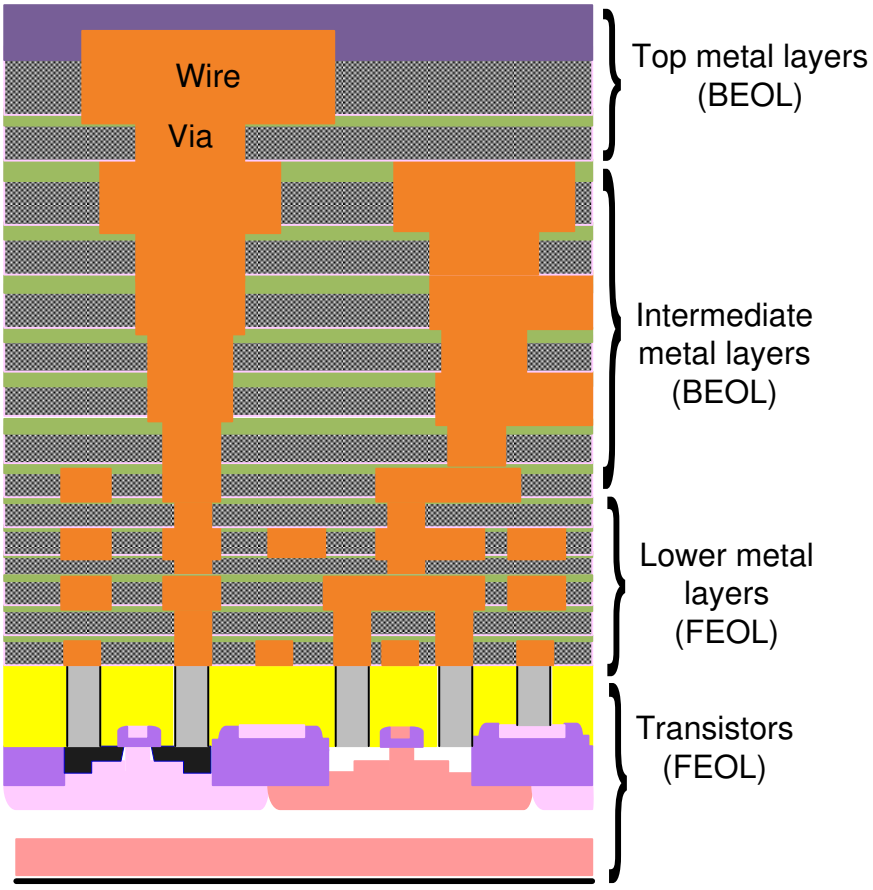


Figure 1: A cross-section of an IC layout consisting of FEOL layers (transistors, lower metal layer) and BEOL layers (intermediate, and top metal layers) [2]

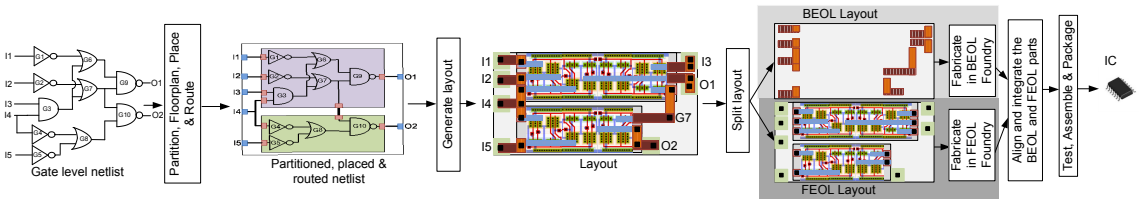
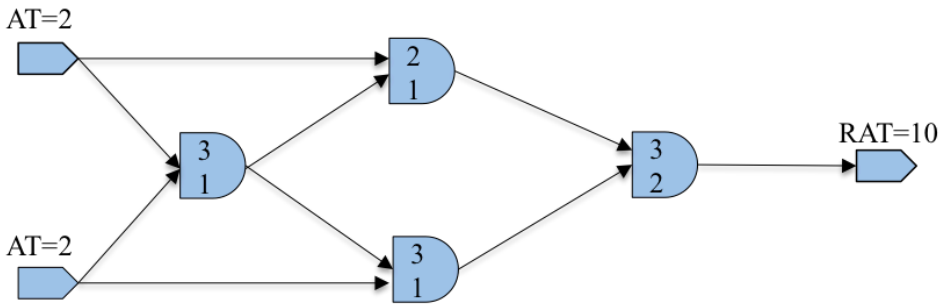


Figure 2: Split manufacturing-aware design flow [10]

II-B. Static timing analysis

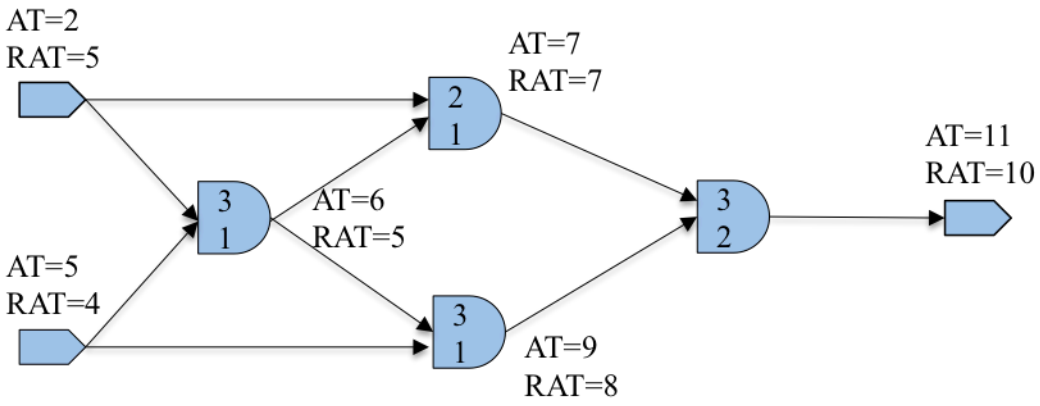
Static timing analysis (STA) is a simulation method to perform timing measurement for a circuit. Instead of simulating the logical operation of the circuit, it determines the worst-case condition of signals at all pins of the circuit.

Two essential concepts associated with STA are arrival time (AT) and required arrival time (RAT). AT is the time when a signal arrives at a certain point and is propagated in topological order. RAT is the latest time at which a signal can arrive without affecting the overall delay of the design and is propagated in reversed topological order. Figure 3 show an example of delay calculation using STA.



(a)

Figure 3: (a) shows timing parameters of a design (b) shows the results after performing STA



(b)

Figure 3 Continued

II-C. Binary decision diagram

Binary decision diagram (BDD) [20] is a compressed representation of Boolean function. It is a data structure consisting of several decision nodes and terminal nodes. A BDD is "ordered" if different decision variables appear in same order. A BDD is "reduced" if it merges isomorphic subgraph and eliminates isomorphic children. Reduced ordered binary decision diagram (ROBDD) is useful in functional equivalence checking because it is canonical.

CHAPTER III

ATTACK IN SPLIT MANUFACTURING

In a common embodiment of split manufacturing, FEOL layers are manufactured by an offshore high-end foundry while BEOL manufacturing and the final integration are conducted in a trusted foundry. The security risk in this scenario arises from the attacker in the offshore foundry.

The objective of the attacker is to retrieve the missing BEOL connections from the FEOL connections. Since the attacker is in the foundry, he has access to the technology library. Consequently, he can obtain the following information about logic gates: layout structure, delay, capacitance load, and wire capacitance. Based on this information, an attacker can reverse engineer the FEOL components and thereby obtains the incomplete gate-level netlist (this netlist lacks the BEOL information). For this purpose, he can use existing tools [21]. The attacker neither knows the functionality implemented by the design nor has access to an IC that performs that function.

An attacker has the disadvantage that the solution space can be astronomically large. If k gate output pins miss their connections, there are 2^{k^2} possible connections in the worst case. An attacker can tremendously reduce this large solution space based on the knowledge that the designer used conventional physical design tools to design the target IC, which has been depicted in [10]. An attacker can take advantage of the following hints, which are public knowledge.

Hint 1: Physical proximity. Physical design tools aim to minimize wirelength, thereby improving performance and reducing power consumption. Therefore, a connection between two pins is rarely very long. Hence, an attacker will prefer to connect two pins that are close to each other rather than the ones that are far apart.

Hint 2: Acyclic combinational logic circuit. With the exception of ring oscillators, flip-flops, and latches, combinational loops are rare in a design.

Hint 3: Load capacitance constraint. A gate can drive only a limited load capacitance to honor slew constraints. The maximum load capacitance of a gate can be obtained from the physical design library, which is public information. Hence, an attacker will consider only connections that will not violate the load capacitance constraints.

Hint 4: Directionality of dangling wires. Physical design tools route wires from a source gate to the sink node along the latter's direction. Hence, the directionality dangling wires at lower metal layers indicates the direction of their destination cell. An attacker can disregard components in the other directions.

Consider the example in Figure 4. There is a dangling metal pointing towards gate A in the FEOL design available to the attacker. Intuitively, the missing upper metal is most likely to be connected with gate A instead of gate B.

Hint 5: Timing constraint. If a connection violates the timing constraints, then this connection can be excluded. An attacker can at least obtain a conservative estimate on timing constraints through educated guess on clock period.

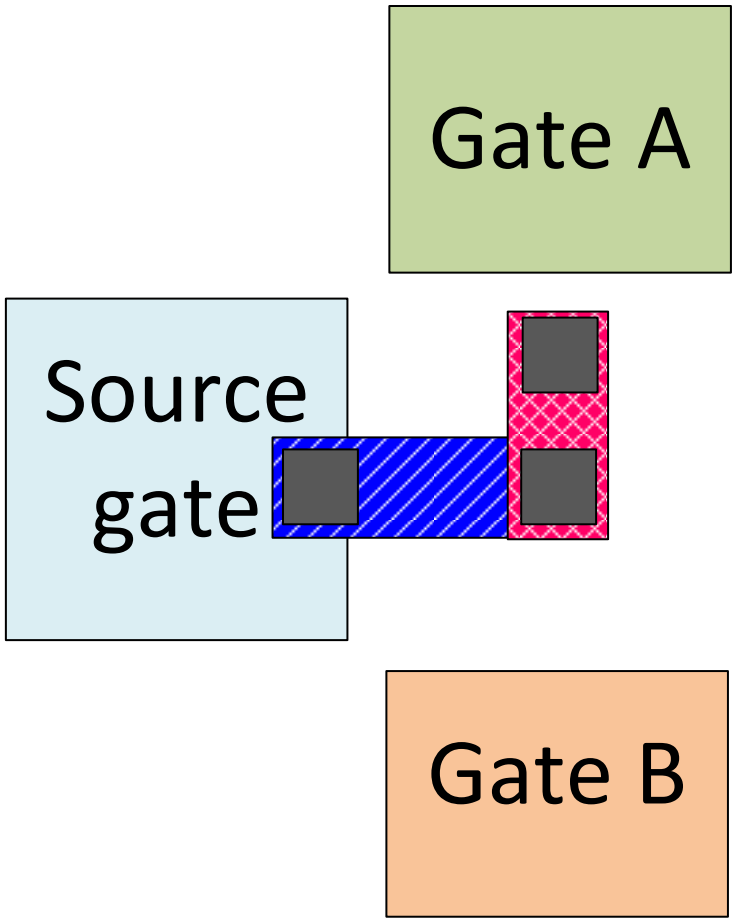


Figure 4: The dangling wire points potential connection from the Source gate towards Gate A.

Hint 6: Logic redundancy. Logic synthesis tools tend to avoid logic redundancy. An attacker can use this heuristic to disregard connections that result in redundant logic.

In Figure 5, the dashed lines indicate the BEOL information, which is not available to the attacker. If one connects the output of G3 to G6, function $f_1 = ab + b'c + ac = ab + b'c$ is redundant. Such a connection will not exist, if the design is

generated by a reasonable logic synthesis tool. Consequently, an attacker will connect the output of G3 to f2, thus deducing the correct connection.

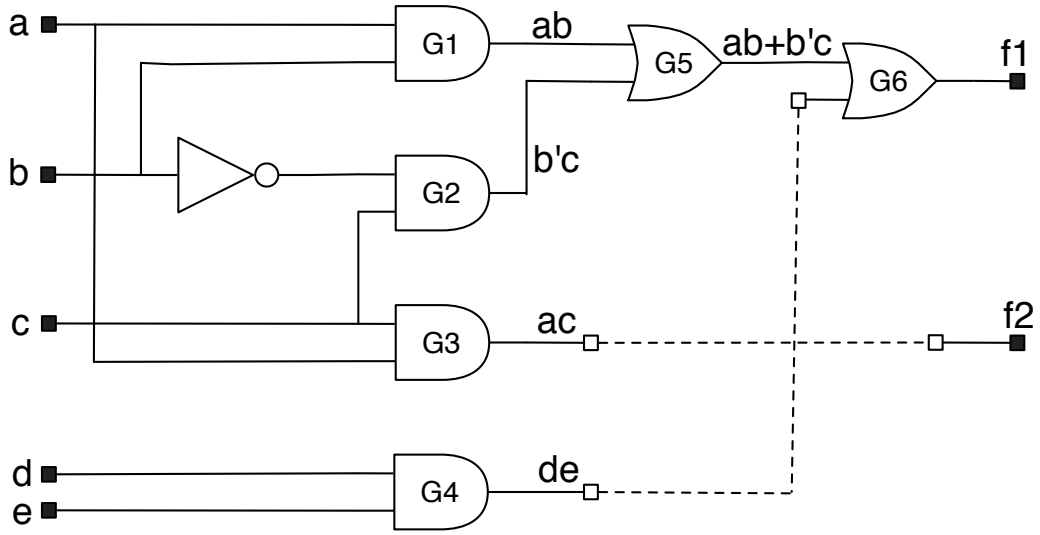


Figure 5: Restoring missing wires by Hint #6

Hint 7: Physical alignment. A Vertical Interconnect Access (VIA) is a small insulating oxide layer that connects different metal layers. Physical design tools tend to align the VIAs, as illustrated in Figure 6. An attacker can use this heuristic to favor connections between aligned VIAs.

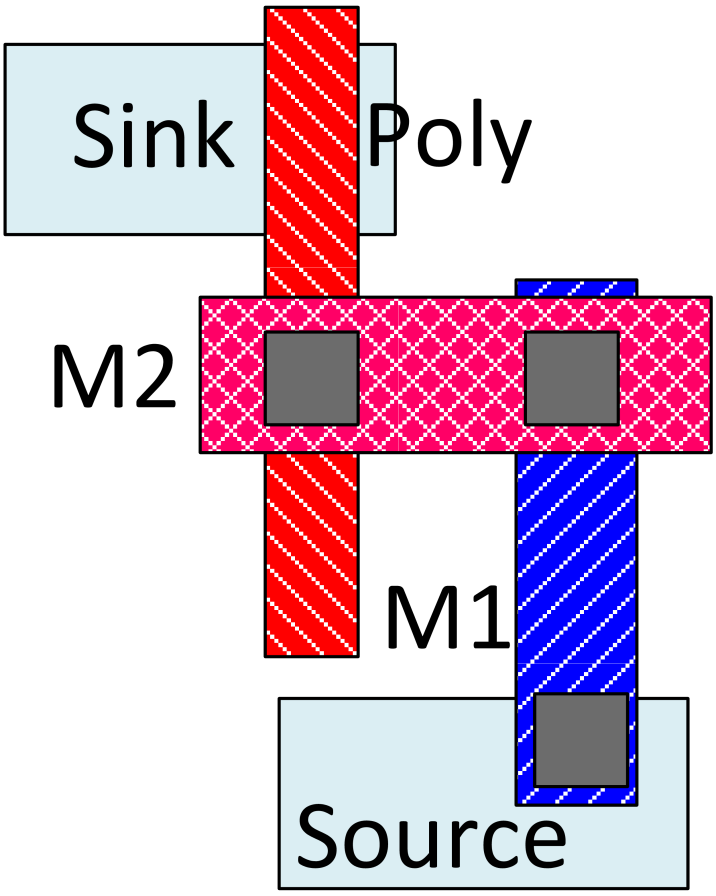


Figure 6: The VIAs between source gate and sink gate are aligned.

III-A. Greedy attack for flattened designs

The greedy attack mainly follows the proximity hint (Hint #1) and the acyclic combinational logic hint (Hint #2) [10]. Unlike in hierarchical designs [10], where each missing net has only 2 pins, the net in flat designs may have multiple fanouts, i.e., more than 2 pins. In the greedy attack, we iteratively connect a gate input pin to its nearest gate output pin. After each connection, we check if that connection results in a combinational loop. If a loop is found, this connection is reverted, the input pin is tried to connect with the next nearest output pin that does not result in a combinational loop.

This procedure is repeated till all gate input pins are connected. At the end, if there is a dangling output (i.e., the output of a gate that is not connected to any input), we find its nearest multi-fanout net and connect the nearest input pin in this net to the dangling output pin. Algorithm 1 describes the steps involves in the greedy attack.

Input: FEOL layers

Output: Netlist with BEOL connections

Reverse engineer FEOL layers

while Unassigned pins exist **do**

    Select an arbitrary unassigned input pin as a TargetPin

    ListOfCandidatePins = BuildCandidatePinsList(TargetPin)

    Select the output pin from ListOfCandidatePins that is closest to TargetPin as a CandidatePin

    Connect TargetPin and CandidatePin

    Update netlist

end

return netlist

BuildCandidatePinsList(TargetPin)

Input: TargetPin

Output: Candidate pins for TargetPin

CandidatePins = Output pins

for each PinI $\in$ CandidatePins do

if CombinationalLoop(TargetPin, PinI) then

CandidatePins -= PinI

end

end

return CandidatePins

Algorithm 1: Greedy attack on flattened design

III-B. Physical attack

We describe a network-flow based physical attack that considers Hints #1—#5, and Hint #7 of aforementioned hints in a holistic manner. This is illustrated by an example in Figure 7, where the attack needs to infer the connections between output pins $\{a, b\}$ and input pins $\{1, 2, 3\}$.

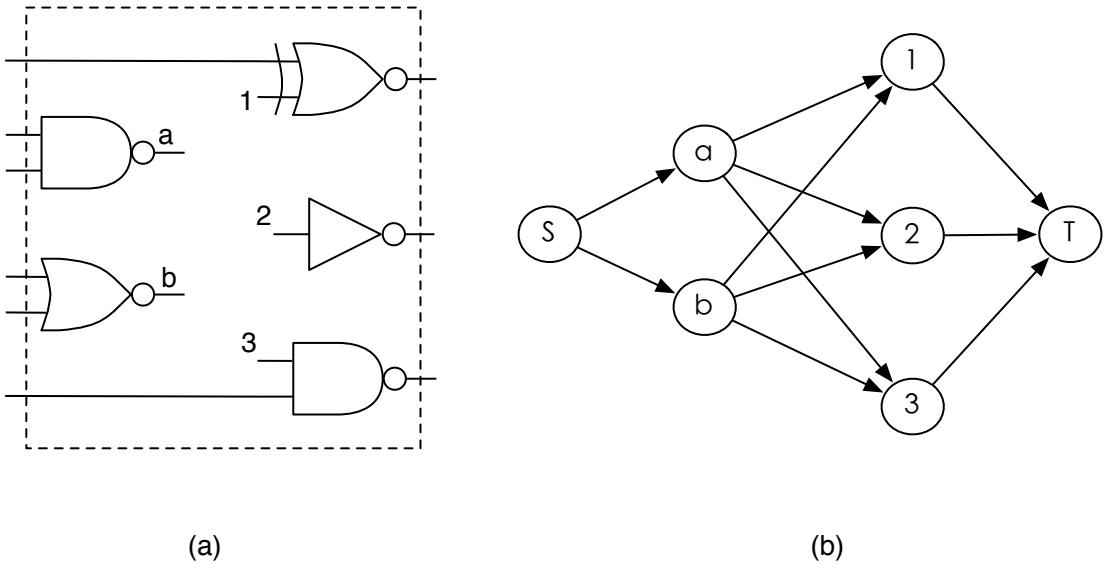


Figure 7: (a) A circuit with missing connections. (b) Network flow model for inferring the missing connections

The network is a directed graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. The set V is composed by a set of vertices corresponding to the output pins (V_0), a set of vertices corresponding to the input pins (V_i), the source vertex (S) and the target vertex (T). The set E consists of E_{So}, edges from S to every output pin vertex, E_{oi}, edges from output pin vertices to input pin vertices, and E_{iT} , which includes edges from every input pin vertex to the target vertex. The network for Figure 7(a) is shown in Figure 7(b). In a network flow solution, certain amount of flow emerges from S, goes through network edges and finally arrives T. The flow through edge $(a, i) \in E_{oi}$ infers wire connection between output pin a and input pin i.

The six hints are addressed by edge construction for E_{oi}, edge capacities, edge costs and dynamic use of the network flow model. A necessary condition for including an edge $(a, i) \in E_{oi}$ is that output pin a is along the direction of input pin i's dangling wire and vice-versa. For the example in Figure 8, edge $(a, 2)$ is included in E_{oi}, but $(a, 1)$ is not. Another condition is that the connection between a and i would not result in timing violation. We can estimate the slack at a by subtracting the arrival time (AT) at a from the required arrival time (RAT) at i. This is an optimistic estimation without considering the delay from a to i. If this optimistic slack is less than zero, then including the delay from a to i would make the violation even worse. Then, the connection between a and i is disallowed, i.e., there is no (a, i) in E_{oi}. Sometimes the AT and RAT are not available due to wire disconnections, then we replace AT with lower bound, which is the AT at the primary input, and replace RAT with upper bound, which is the RAT at the primary output. The estimate obtained as such provides an upper bound for

the slack. By constructing E_{oi} as such, the hint of directionality of dangling wire (Hint #4) and timing constraints (Hint #5) are followed.

The capacity c_{Sa} for each edge in E_{So} is defined as the load capacitance constraint for output pin a. The capacity c_{ai} for each edge in E_{oi} is infinity. The capacity c_{iT} for each edge in E_{iT} is the input capacitance for pin i. A flow solution that satisfies the edge capacity constraints follows the hint of load capacitance constraint (Hint #3).
The cost w_{ai} for each edge in E_{oi} is the wirelength in connecting pin a and i. The other edge costs are set to 0. If we run min-cost flow algorithm on this network, the solution minimizes the total flow cost, which is the total wirelength for all connections. This edge cost definition addresses the proximity hint (Hint #1).

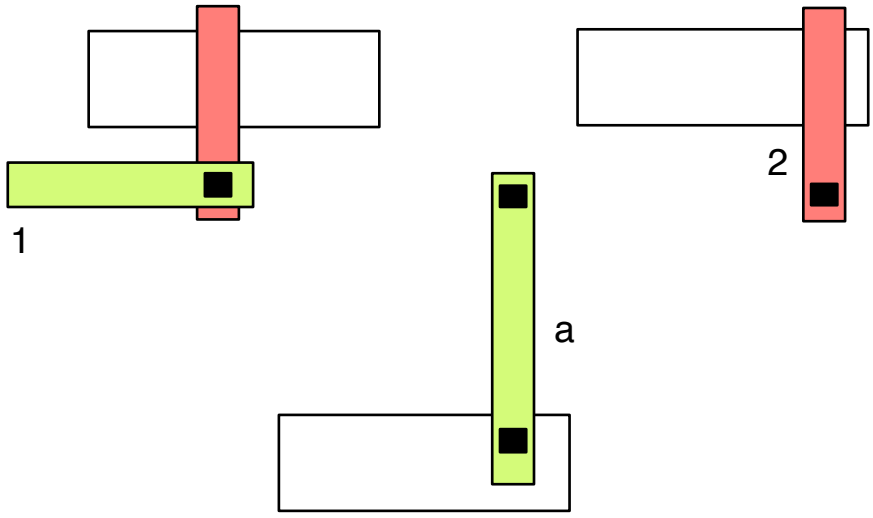


Figure 8: Input pin 1 and 2 are in output pin a's dangling direction. Output pin a is in pin 2' dangling direction, but not in pin 1's dangling direction

The hint of acyclic combinational logic circuit (Hint #2) is difficult, if not impossible, to be handled in a one-shot network flow solution, because a loop can be

detected only after the connection solution is obtained. To solve this issue, we used an iterative network-flow approach. After connections are inferred from a network flow solution, a circuit traversal is performed to check if any loop exists. If so, the longest inferred connection is picked. This connection must correspond to an edge in E_{oi}. The min-cost flow algorithm is conducted again after removing this edge from the network. This procedure is repeated until no loop is detected.

In the min-cost network flow problem, the decision variables are the flow $x_{i\ j}$ going through each edge $(i, j\ ) \in E$ Then the problem is formally formulated as follows.

$$Min \qquad \Sigma_{(i,j)\in E} w_{i,j} \cdot x_{i,j} \qquad (1)$$

$$s.t. \quad \Sigma_{i|(i,j)\in E} x_{i,j} = \Sigma_{k|(j,k)\in E} x_{j,k}, \qquad j \in V_0 \cup V_1 \qquad (2)$$

$$\Sigma_{(i,T)\in E_{iT}} x_{i,T} = \Sigma_{(i,T)\in E_{iT}} c_{i,T} \qquad (3)$$

$$\Sigma_{(S,i)\in E_{S_0}} x_{S,i} = \Sigma_{(i,T)\in E_{iT}} c_{i,T} \qquad (4)$$

$$x_{i,j} \leq c_{i,j}, \qquad \forall (i,j) \in E \qquad (5)$$

This problem can be solved by off-the-shelf algorithms, e.g., the Edmonds-Karp algorithm [22], which can obtain the optimal solution in polynomial time.

Consider the BEOL connections of a combinational logic circuit shown in Figure 9. This design has two primary inputs ($N1$, $N2$), one primary output ($N3$), and four gates ($Gate$ 1-4). $P_{Gate,Net}$ denotes a pin where net X, net Y are the inputs of the gate and net Z is the output of the gate. The RAT at the primary output of this design is requested to be less than 9. The gate delays of gate 1 - 4 are 1, 2, 1, 2.5 respectively. The wire delays

from $P_{1,Z}$ to $P_{3,X}$, from $N2$ to $P_{2,X}$, and from $P_{2,Z}$ to $P_{4,Y}$ are equal to 2, and the wire delay from $P_{4,Z}$ to $N3$ is 0.2.

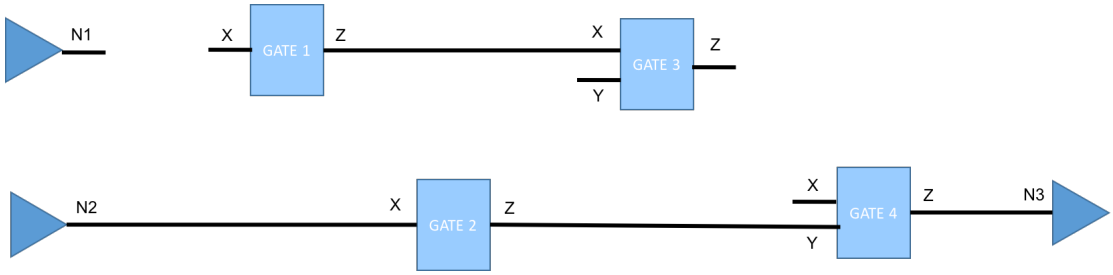


Figure 9: A circuit with FEOL connections only

Table 1 demonstrates the locations, timing parameters and capacitance information of each pin an attacker obtains from the design in Figure 9. The location fields are the X-Y coordinates of the pins in the design. The coordinates are shown in absolute unite for ease of understanding. The direction fields are the directionalities of pins, where E, W, N, and S denote east, west, north, and south respectively. Since the AT at $P_{1,X}$, $P_{3,Y}$, $P_{4,X}$ and the RAT at $P_{3,Z}$ are not available due to the wire disconnections, the AT at those pins are replaced with the AT at primary input, which is equal to 0. The RAT at $N1$ and $P_{3,Z}$ are replaced with the RAT at primary output, which is equal to 9.

Table 1: Pin information obtained from the design in Figure 9

Pin	Location	Direction	Input capacitance/ load capacitance constraint	AT	RAT
$N1$	(0,0)	E	N/A	0	13
$N2$	(55,120)	N	N/A	0	0.3
$N3$	(155,0)	S	N/A	8.7	9
$P_{1,X}$	(30,20)	N	1	0	5
$P_{1,Z}$	(30,15)	N	5	1	6
$P_{2,X}$	(55,115)	W, S	1	2	2.3
$P_{2,Z}$	(50,115)	N, E	5	4	4.3
$P_{3,X}$	(45,15)	E	1	3	8
$P_{3,Y}$	(50,15)	N, S, W	1	0	8
$P_{3,Z}$	(50,10)	N, S, E	5	4	9
$P_{4,X}$	(150,10)	W, N	1	0	6.3
$P_{4,Y}$	(150,5)	E, S	1	6	6.3
$P_{4,Z}$	(155,5)	W, N	5	8.5	8.8

Table 2 shows the costs of each edge in the network flow model. The cost of edges between $P_{1,X}$ and $P_{1,Z}$, between $P_{3,Y}$ and $P_{3,Z}$, and between $P_{4,X}$ and $P_{4,Z}$ are infinite because a gate's output will not be connected with its input. The cost of edges between $P_{1,Z}$ and $P_{3,Y}$, and between $P_{2,Z}$ and $P_{4,X}$ are infinite because the output of one gate will not be connected to both inputs of another gate. The cost of edges between $P_{1,X}$

and $P_{4,Z,}$, and between $P_{3,Y}$ and $P_{4,Z}$ are infinite because those connections result in timing violation. The capacity of each edge in the network is equal to the pin wirelength, except for the edge between $P_{3,Z}$ and $P_{4,X}$, and the edge between $P_{2,Z}$ and $P_{3,Y}$, where the square root of pin distances are used to favor physical alignment.

Table 2: Costs of edges from output pin vertexes to input pin vertexes in the network

	$N1$	$N2$	$P_{1,Z}$	$P_{2,Z}$	$P_{3,Z}$	$P_{4,Z}$
$P_{1,X}$	50	125	∞	115	30	∞
$P_{3,Y}$	65	110	∞	10	∞	∞
$P_{4,X}$	160	205	125	∞	10	∞

Figure 10 is the circuit reconstructed by the attacker with performing Edmonds-Karp algorithm once. As one can see, pin $N1$ is still unsigned, and the connection from $P_{3,Z}$ to $P_{1,X}$ results in a loop in the circuit. Thus, the attacker decreases the costs of edges inferred with $N1$, increases the cost of the edge between $P_{3,Z}$ and $P_{1,X}$, then performs the second iteration of attack. The updated cost of each edge in the network is demonstrated in Table 3 where the costs of edges inferred with $N1$ are decreased by 30%, while the cost of the edge between $P_{3,Z}$ and $P_{1,X}$ is increased by 30%. The reconstructed circuit is shown in Figure 11.

Table 3: Updated costs of edges from output pin vertexes to input pin vertexes in the network

	$N1$	$N2$	$P_{1,Z}$	$P_{2,Z}$	$P_{3,Z}$	$P_{4,Z}$
$P_{1,X}$	35	125	∞	115	39	∞
$P_{3,Y}$	45	110	∞	10	∞	∞
$P_{4,X}$	112	205	125	∞	10	∞

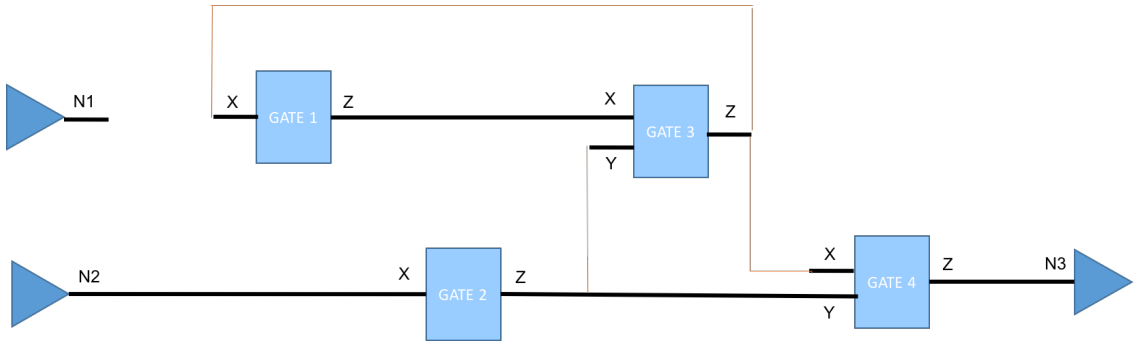


Figure 10: Reconstructed circuit from the first iteration of physical attack

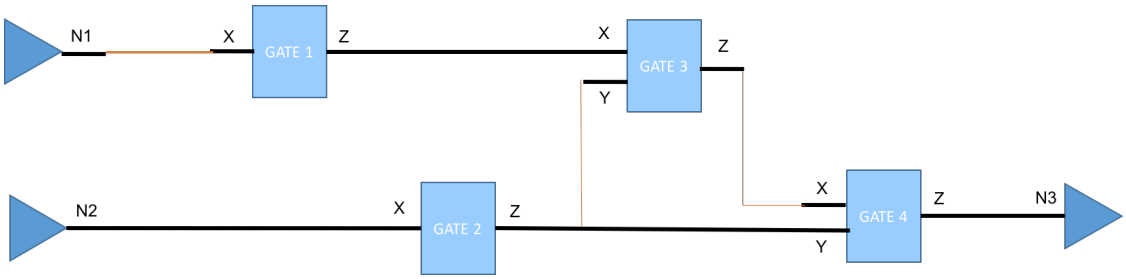


Figure 11: Reconstructed circuit from the second iteration of physical attack

Algorithm 2 illustrates each step involved in physical attack.

Input: FEOL layers

Output: Netlist with BEOL connections

Reverse engineer FEOL layers

Build network for unsigned pins

for each edge Eso where pin o is an unassigned output pin

Set capacity Cso for Eso to the load capacitance constraint for pin o

end

for each edge E_{it} where pin i is an unassigned input pin

Set capacity C_{it} for E_{it} to the input capacitance constraint for pin i

end

for each edge E_{oi} where pin o is an unassigned output pin and pin i is an unassigned input pin

Set capacities C_{oi} for E_{oi} that violates dangling wires hint or timing constraints to 0

Set cost ω_{oi} for E_{oi} to the wirelength in connecting pin o to pin i

end

Run Edmonds-Karp algorithm

while Unassigned pins exist or loop exists **do**

Update flow cost of inferred edges

Run Edmonds-Karp algorithm

end

return netlist

Algorithm 2: Physical attack

III-C. Logic-aware physical attack

We develop a logic-aware physical attack by incorporating logic redundancy (Hint #6) into our network-flow framework with using Automatic Test Pattern Generation (ATPG) [23] and BDD [20]. We utilize ATPG to recognize redundant wires within a design [24], which are potential incorrect connections made by a physical attack. However, ATPG alone is not sufficient since limited untestable faults are allowed in commercial logic synthesis tools, which satisfy testability to achieve the optimal area-delay-testability trade-off instead of optimizing testability individually. Thus, we integrate BDD into our logic-aware physical attack.

Along with the circuit traversal, BDD for the visited part is constructed. We use this BDD to detect following logic redundancies:

(i) A restored circuit with BDD nodes corresponding to constant '0' or '1' indicates that input-independent computing is performed.

(ii) If two circuit pins are mapped to the same BDD node, there exists logic redundancy since ROBDD is canonical [20], as illustrated in Figure 12.

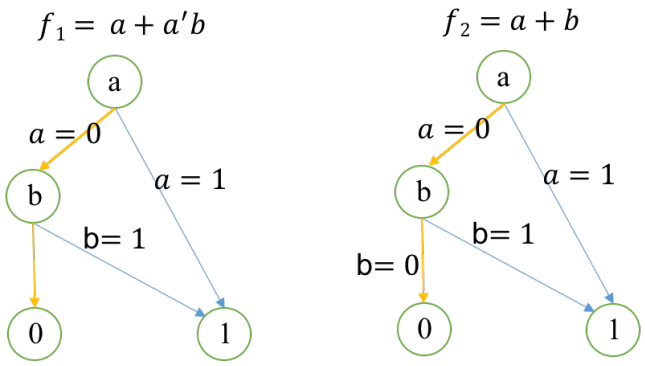


Figure 12: Logic expression f_1 and f_2 have identical ROBDD structure

(iii) For a multiple-input logic gate, if the BDD node of one input pin is an implicant of the BDD node of another input pin, the circuit can be simplified. Without loss of generality, we illustrate this case in Table 4 with two-input basic logic gates as all compound multiple-input logic gates function as the combination of a few basic logic gates. In case of XOR, $f_1 \cdot \overline{f_2} \subseteq f_1 \cdot \overline{f_1} \subseteq \emptyset$. In case of XNOR, $\overline{f_1} \cdot \overline{f_2} = \overline{\overline{f_1} + \overline{f_2}} = \overline{f_2}$.

Table 4: Logic redundancy results from implicant

Type	Input		Output
AND	f_1	f_2	$f = f_1 \cdot f_2 = f_1$
OR	f_1	f_2	$f = f_1 + f_2 = f_2$
XOR	f_1	f_2	$f = f_1 \cdot \overline{f_2} + \overline{f_1} \cdot f_2 = \overline{f_1} \cdot f_2$
XNOR	f_1	f_2	$f = f_1 \cdot f_2 + \overline{f_1} \cdot \overline{f_2} = f_1 \cdot \overline{f_2}$

The logic redundancies detected by BDD is not allowed, because they unnecessarily increase the cost of implementation, i.e., the size of physical implementation, the complexity of Boolean network, and the delay of the design. To prevent a logic redundancy from propagating, we divide a design into several stages where each stage is determined at the point BDD detects redundancy. Algorithm 3 demonstrates the logic-aware physical attack.

Input: FEOL layers

Output: Netlist with BEOL connections

Reverse engineer FEOL layers

Apply Physical attack

Build BDD of the reconstructed circuit and divide the circuit into stages

if logic redundancy exists

    Locate redundant wires using ATPG

    Update inferred edges in network

end

return netlist;

Algorithm 3: Logic-redundancy-aware network-flow attack

CHAPTER IV

EXPERIMENTS

IV-A. Experimental setup

We evaluate our techniques using ISCAS-85 combinational benchmark circuits [25] and ITC-99 benchmark [26]. Each circuit was synthesised by Synopsys Design Compiler tool [27]. Placement and routing were performed using Cadence SoC Encounter tool [28] for 180nm CMOS technology [29].

We assess the effectiveness of the attack model by identifying the number of correct connections it makes. An attacker always tries to make as many correct connections as possible. In addition, we can evaluate the performance of attack techniques through error rate, the number of wrong outputs produced on applying a specific number of inputs [11–13,16]. The objective of the attacker is to minimize the error rate of the recovered design. The error rate between the outputs of the original design and the design reconstructed using the attack was determined by applying 50,000 random input patterns.

IV-B. Experimental results

IV-B.1 Effectiveness of attack

For each benchmark circuit, we performed greedy attack and physical attack respectively. Figure 13 shows the percentage of pins that are correctly connected by using different attack techniques. In case of “Greedy attack,” the average number of

correct connections is around 8%, lower than that of "Physical attack." This verifies that the greedy attack is not applicable to flattened designs while physical attack is effective.

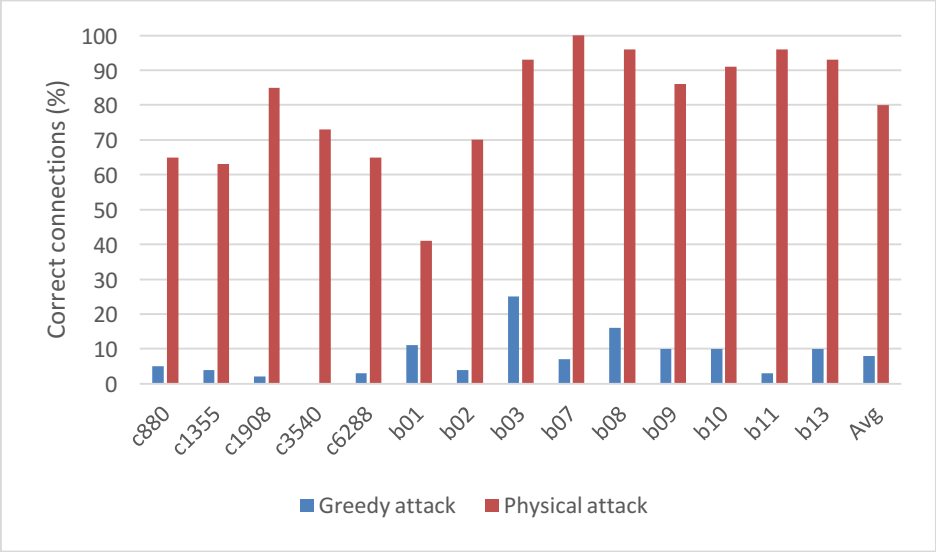


Figure 13: Correct connection rate on performing greedy attack and the proposed physical attack

Figure 14 depicts the error rate of primary output ports between the original design and the design subject to different attack techniques. The error rate of "Physical attack" is 64.36%, lower than that of "Greedy attack," highlighting the effectiveness of the proposed physical attack.

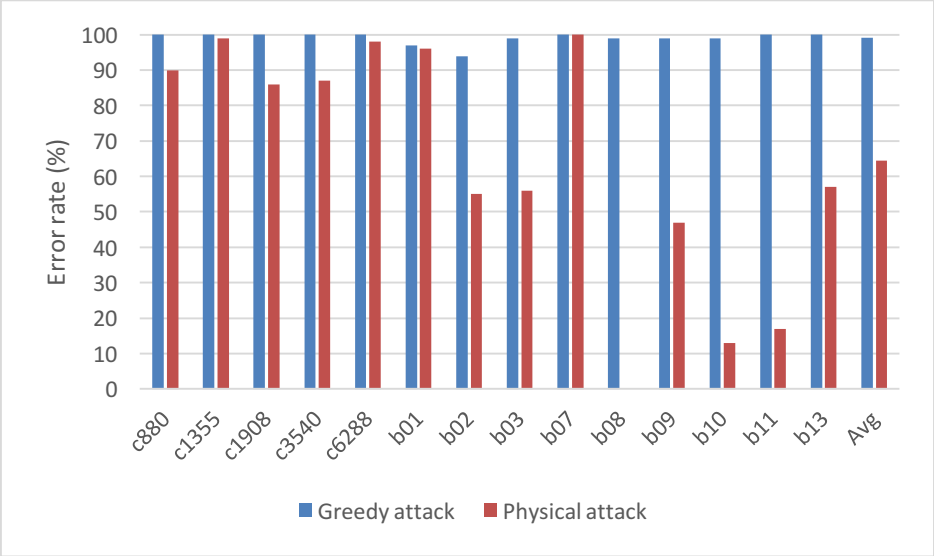


Figure 14: Output error rate on performing greedy attack and the proposed network-flow attack

IV-B.2 Effectiveness of split layer on security

The layer at which the BEOL and FEOL split occurs is called the split layer. If the split layer is M2 or M3, it may guarantee security, but it demands a relatively high-end BEOL facility, and thus increasing its cost. Contrarily, if the split layer is M5 or M6, it may not guarantee security, but it does not demand a relatively high-end BEOL facility, and thus decreasing its cost. Thus, it is necessary to find the effect of split layer on the proposed attack.

Figure 15 shows the correct connection rate and error rate for b09 and b11 on allowing different split layers. It can be seen that when M5 or M6 is the split layer, the proposed attack is highly effective, because there are less number of candidate solutions. It can be seen that when M3 or M4 is the split layer, the proposed attack is ineffective because there are more number of candidate solutions.

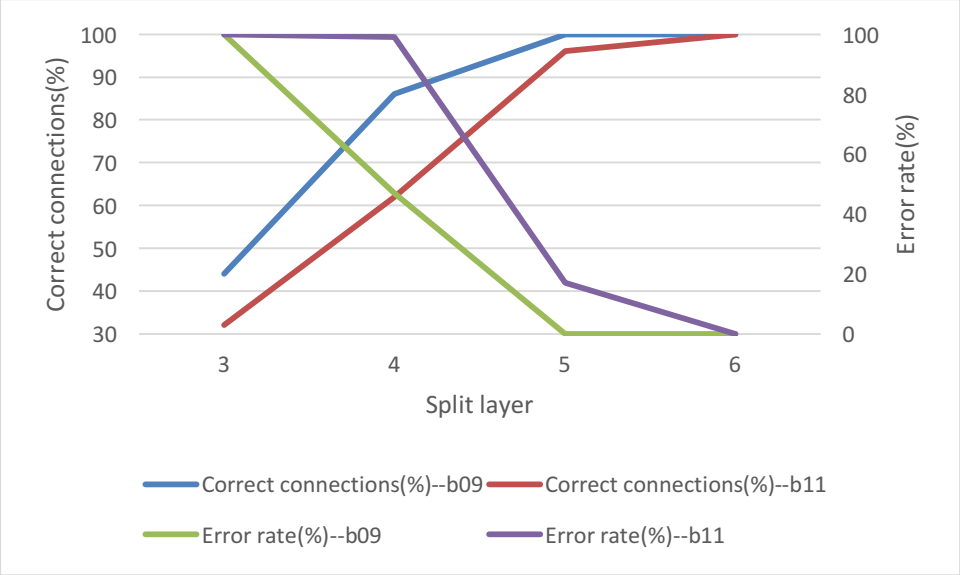


Figure 15: Correct connection rate and error rate vs. split layer for "Physical attack" on circuits b09 and b11

IV-B.3 Effectiveness of logic-aware physical attack

In order to evaluate the effectiveness of the proposed logic-aware physical attack, the physical attack is compared with following logic-aware physical attack techniques:

- **ATPG + Physical** Logic-aware physical attack to circuits with detecting logic redundancy via ATPG alone.
- **ATPG + BDD + Physical** Logic-aware physical attack to circuits with detecting logic redundancy via both ATPG and BDD.

We access the effectiveness of the proposed logic-aware physical attack by identifying the number of incorrect connections detected by "ATPG + Physical" and "ATPG + BDD + Physical" respectively. Figures 16 shows the results of incorrect-connection detection rate (ICDR), which is the number of incorrect connections detected

by logic redundancy detector over all the incorrect connections a physical attack makes. The average ICDR of "ATPG + Physical" is 70.9%, higher than 50%. This demonstrates that ATPG is an effective way to recognize incorrect connections. Meanwhile, the average ICDR of "ATPG + BDD" is 73.9%, higher than that of "ATPG", which verifies that BDD enhances the ability of identifying incorrect connections.

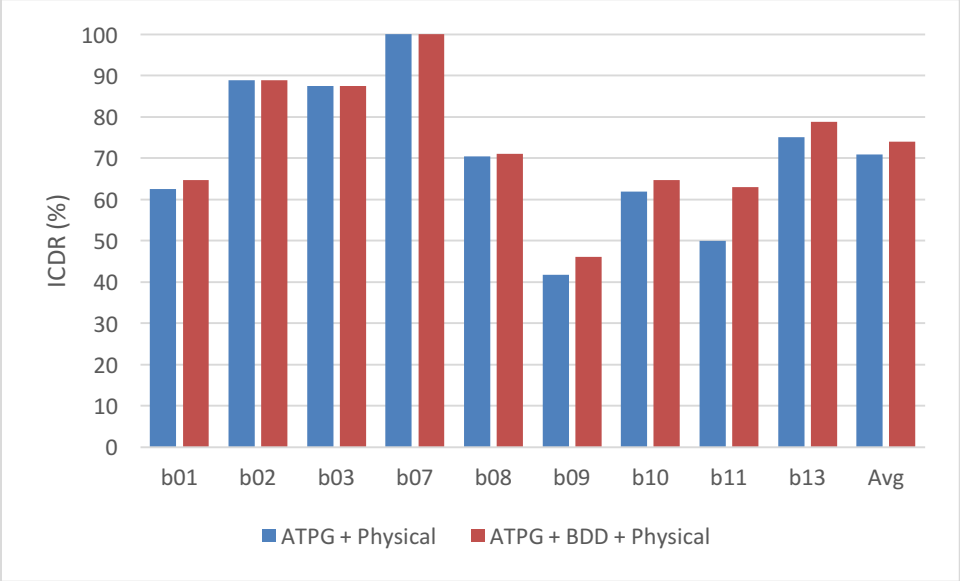


Figure 16: ICDR for "ATPG + Physical" and "ATPG + BDD + Physical"

Figure 17 shows the results of correct connection rate subject to different attack techniques. In case of "Physical," the average correct connection rate is reduced to around 43%, this is because the split layer in this experiment is lower than before. One can see that, for each benchmark circuit, the average connection rate of "ATPG + Physical" and "ATPG + BDD + Physical" are close to each other, and both of them are lower than that of "Physical."

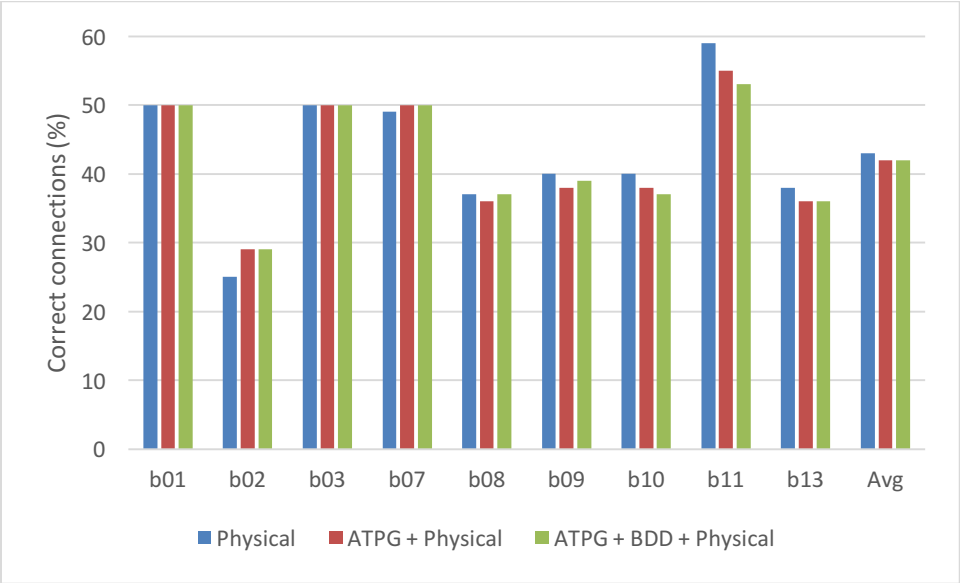


Figure 17: Correct connection rate on performing different attack techniques

Although effectively detecting incorrect connections, the proposed logic-aware physical attack cannot ensure the improve the attack performance. This is because our logic-aware physical attack lacks intelligence to handle incorrect connections. Simply increasing the cost of incorrect connections in the network-flow model is not sufficient.

CHAPTER V

CONCLUSION

Split manufacturing, though not a universal solution for all security problems, it can protect commercial designs from rogue elements in the FEOL foundry. While state-of-the-art attack is applicable only to hierarchical designs [10], we have proposed an attack for industry-relevant flattened designs, using the heuristics of physical designs tools. Our attack success rate is ~10× that of the state-of-the-art algorithm [10]; our attack predicts 80% of the missing BEOL connections correctly, while the state-of-the-art predicts only 8% for flattened designs.

While the logic-aware attack fails to further increase the correct connection rate, we showed that it can identify incorrect connections effectively.

REFERENCES

[1] DIGITIMES Research, “Trends in the global IC design service market.” http://www.digitimes.com/news/a20120313RS400.html?chid=2, 2012.

[2] Intelligence Advanced Research Projects Activity, “Trusted integrated circuits program.” https://www.fbo.gov/utils/view?id=b8be3d2c5d5babbdffc6975c370247a6, 2011.

[3] DARPA, “Defense science board study on high performance microchip supply.” http://www.acq.osd.mil/dsb/reports/ADA435563.pdf, 2005.

[4] M. Tehranipoor and F. Koushanfar, “A survey of hardware trojan taxonomy and detection,” IEEE Design and Test of Computers, vol. 27, no. 1, pp. 10–25, 2010.

[5] J. Rajendran, E. Gavas, J. Jimenez, V. Padman, and R. Karri, “Towards a comprehensive and systematic classification of hardware trojans,” IEEE International Symposium on Circuits and Systems, pp. 1871–1874, 2010.

[6] R. Torrance and D. James, “The state-of-the-art in semiconductor reverse engineering,” IEEE/ACM Design Automation Conference, pp. 333–338, 2011.

[7] U. Guin, K. Huang, D. DiMase, J. Carulli, M. Tehranipoor, and Y. Makris, “Counterfeit integrated circuits: a rising threat in the global semiconductor supply chain,” Proceedings of the IEEE, vol. 102, no. 8, pp. 1207–1228, 2014.

[8] S. Bhunia, M. Hsiao, M. Banga, and S. Narasimhan, “Hardware trojan attacks: threat analysis and countermeasures,” Proceedings of the IEEE, vol. 102, no. 8, pp. 1229–1247, 2014.

[9] SEMI, “Innovation is at risk as semiconductor equipment and materials industry loses up to $4 billion annually due to IP infringement.” www.semi.org/en/Press/P043775, 2008.

[10] J. Rajendran, O. Sinanoglu, and R. Karri, “Is split manufacturing secure?” IEEE/ACM Design Automation and Test in Europe, pp. 1259–1264, 2013.

[11] K. Vaidyanathan, B. P. Das, E. Sumbul, R. Liu, and L. Pileggi, “Building trusted ICs using split fabrication,” IEEE Symposium on Hardware Oriented Security and Trust, pp. 1–6, 2014.

[12] K. Vaidyanathan, R. Liu, E. Sumbul, Q. Zhu, F. Franchetti, and L. Pileggi, “Efficient and secure intellectual property design for split fabrication,” IEEE Symposium on Hardware Oriented Security and Trust, pp. 13–18, 2014.

[13] M. Jagasivamani, P. Gadfort, M. Sika, M. Bajura, and M. Fritze, “Split fabrication obfuscation: metrics and techniques,” IEEE Symposium on Hardware Oriented Security and Trust, pp. 7–12, 2014.

[14] R. Jarvis and M. G. McIntyre, “Split manufacturing method for advanced semiconductor circuits,” US Patent no. 7195931, 2004.

[15] B. Hill, R. Karmazin, C. Otero, J. Tse, and R. Manohar, “A split-foundry asynchronous FPGA,” IEEE Custom Integrated Circuits Conference, pp. 1–4, 2013.

[16] K. Vaidyanathan, B. P. Das, E. Sumbul, R. Liu, and L. Pileggi, “Detecting reliability attacks during split fabrication using test-only beol stack,” IEEE/ACM Design Automation Conference, pp. 156:1–156:6, 2014.

[17] C. Otero, J. Tse, R. Karmazin, B. Hill, and R. Manohar, "Automatic obfuscated cell layout for trusted split-foundry design," IEEE International Symposium on Hardware Oriented Security and Trust, pp. 56–61, 2015.

[18] S. Mitra, H.-S. Wong, and S. Wong, "Stopping hardware trojans in their tracks." http://spectrum.ieee.org/semiconductors/design/ stopping-hardware-trojans-in-their-tracks, 2015.

[19] Y. Bi, J. Yuan, and Y. Jin, "Beyond the interconnections: split manufacturing in RF designs," MDPI Electronics, vol. 4, pp. 541–564, 2015.

[20] S. B. Akers, "Binary decision diagrams," IEEE Transactions on Computers 100, no. 6: 509-516, 1978.

[21] L. Grunwald, "Degate: ein open-source-tool zum auslesen von EMV chips." http://www.degate.org/documentation/, 2011.

[22] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network flows: theory, algorithms, and applications," Upper Saddle River, NJ: Prentice Hall, 1993.

[23] L. Lavagno, G. Martin, and L. Scheffer, "Electronic design automation for integrated circuits handbook," CRC Press, 2006.

Kirkland, Tom, and M. Mercer, "Automatic test pattern generation," 1988.

[24] S.C. Chang, V. Ginneken, L.P. and M. Marek-Sadowska, "Circuit optimization by rewiring," IEEE Transactions on Computers, 48(9), pp.962-970, 1999.

[25] M. Hansen, H. Yalcin, and J. Hayes, "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering," IEEE Design Test of Computers, vol. 16, no. 3, pp. 72–80, 1999.

[26] F. Corno, M. S. Reorda, and G. Squillero, “RTL level ITC’99 benchmarks and first ATPG results,” IEEE Des. Test, vol. 17, no. 3, pp. 44–53, 2000.

[27] Synopsys, “Design compiler.” http://www.synopsys.com/tools/implementation/rtlsynthesis/dcgraphical/Pages/default.aspx, 2016.

[28] Cadence, “SoC encounter.” http://www.cadence.com/products/di/soc_encounter/pages/default.aspx, 2016.

[29] “FreePDK: Unleashing VLSI to the masses.” http://vlsiarch.ecen.okstate.edu/flows/MOSIS_SCMOS/, 2014.