Empowering Hardware Security using IC Testing Principles

Jeyavijayan Rajendran
Polytechnic Institute of New York University E-mail: jrajen01@students.poly.edu Web page: isis.poly.edu/∼ jv

I. INTRODUCTION

Today's System on Chip (SoC) is being incorporated with digital, analog, radio frequency, photonic and other devices [1]. More recently, sensors, actuators, and biochips are also being integrated into these already powerful SoCs. On one hand, SoC integration has been enabled by advances in mixed system integration and the increase in the wafer sizes (currently about 300 mm and projected to be 450mm by 2018 [1]). Consequently, the cost per chip of such SOCs has reduced. On the other hand, support for multiple capabilities and mixed technologies has increased the cost of ownership of advanced foundries. For instance, the cost of owning a foundry will be $5 billion in 2015 [2]. Consequently, only large commercial foundries now manufacture such high performance, mixed system SoCs especially at the advanced technology nodes [3].

Absent the economies of scale, many of the design companies cannot afford owning and acquiring expensive foundries and hence, outsource their design fabrication to these "one-stop-shop" foundries[1]. This globalization of Integrated Circuit (IC) design flow has introduced security vulnerabilities. If a design is fabricated in a foundry that is outside the direct control of the (fabless) design house, reverse engineering, malicious circuit modification, and Intellectual Property (IP) piracy are possible [3]. An attacker, anywhere in this design flow, can reverse engineer the functionality of an IC/IP, and steal and claim ownership of the IP. An untrusted IC foundry may overbuild ICs and sell the excess parts in the gray market. Rogue elements in the foundry may insert malicious circuits (hardware Trojans) into the design without the designer's knowledge [4]–[6]. Because of these and similar hardware-based attacks, the semiconductor industry loses $4 billion annually [7].

A. Approach

While hardware security and trust is an emerging design concern, a somewhat related yet fundamentally different problem of detecting manufacture-time defects has been the focus of VLSI test researchers for the last few decades. On one hand, manufacture-time defects are unintentional and natural. On the other hand, hardware-based attacks are man-made, intentional, and carefully hidden thereby hampering the direct application of existing VLSI testing techniques. However, our research shows that many of the important concepts in VLSI test can be adapted to the hardware security and trust context. Inspired by the design enhancement approach (Design-for-Testability – DfT) to improve testability of manufacturing defects, we propose Design-for-Trust (DfTr) solutions to thwart hardware attacks. The unifying theme that connects all the proposed DfTr solutions is the novel use of VLSI test concepts.

B. Intellectual Merit

We develop Design-for-Trust solutions by leveraging well known VLSI test principles, techniques and tools [8]–[11]. These include:

- logic encryption to protect against reverse engineering, IP piracy, IC overbuilding and Trojans;
- security analysis of logic encryption using an attacker-defender approach;
- secure split manufacturing to protect against reverse engineering, IP piracy, IC overbuilding and Trojans;
- a DfTr framework integrating these approaches, providing layers of defense against a variety of attacks.

We develop a logic encryption scheme to equip designs with built-in keys that are unique to each IC. We analyze the fault activation, propagation, and masking effects in designs and relate them to application of wrong keys to the design, corruption of the outputs of the design and protection of embedded keys. We identify the vulnerabilities in straightforward split manufacturing when implemented with the existing design tools and flows, and strengthen it by leveraging VLSI test concepts. Finally, we integrate all these techniques into a unified DfTr framework that can deliver layers of defense in a cost-controlled manner.

[1]companies that do not own their foundry are termed 'fabless' design houses.

II. TEST PRINCIPLES

Well-known VLSI test principles [12] are leveraged to develop DfTr techniques. Example VLSI test principles that are considered include:

- **Test Principle 1 – Fault Excitation:** A stuck-at-v fault at a site is excited when an input pattern justifies that site to $\overline{v}$.
- **Test Principle 2 – Sensitization:** A site is sensitized to an output if every side input of every gate on a path from the site to the output is justified to the non-controlling value of the gate.
- **Test Principle 3 – Fault Propagation:** The effect of a fault at a site propagates to an output if the input pattern excites the fault and sensitizes the faulty site to the output.
- **Test Principle 4 – Fault Masking:** Multiple effects of the same excited fault or multiple excited faults mask each other when none of their effects manifest at the outputs, as the errors cancel out.

III. APPLICATION OF VLSI TESTING TO LOGIC ENCRYPTION [8], [11]

Encrypting a design while it passes through the untrusted design phases will not reveal the design's functionality to an attacker. Thereby preventing a wide variety of attacks such as reverse engineering, cloning, trojan insertion and overbuilding. A design can be encrypted by using logic encryption techniques.

Logic encryption[2] hides the functionality and the implementation of a design by inserting additional circuit elements into the original design. In order for the design to exhibit its correct functionality (i.e., produce correct outputs), a valid key has to be supplied to the encrypted design. When a wrong key is applied, the encrypted design will exhibit a wrong functionality (i.e., produce wrong outputs). Post-fabrication, the IC is activated by applying the valid key. The secret keys may be stored and embedded in a tamper-evident memory inside the IC to prevent access to an attacker.

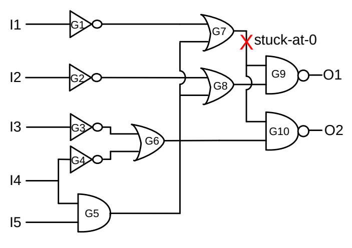
A. Criteria for logic encryption

In an encrypted design, a wrong key should result in a wrong output for all input patterns. If a correct output is produced for a wrong key, then the encryption is weak and the attacker will benefit. If a wrong key affects only one or a few of the output bits, then the attacker might be able to recover most of the outputs. If all the output bits are affected, then the wrong output is the complement of the correct output. Hence, ideally, inspired by the Avalanche criterion [13], a wrong key should affect exactly half of the output bits i.e., *the Hamming distance between the correct and wrong outputs should be 50%* [14]. This 50% Hamming distance renders it difficult for an attacker to recover the key. In another form of an attack, end users can collude by sharing their valid keys. To prevent this collusion attack, *each IC should have a unique key* [15].

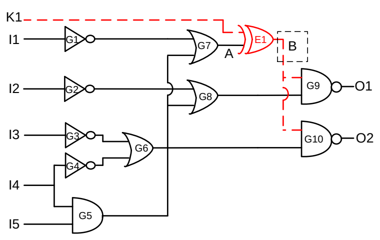
B. Previous work

Logic encryption techniques can be broadly classified into two types—sequential and combinational. In sequential logic encryption, additional logic (black) states are introduced in the state transition graph [5], [16]. The state transition graph is modified in such a way that the design reaches a valid state only on applying a correct

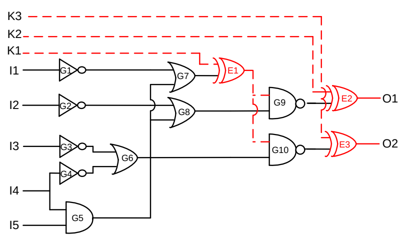
[2]Logic encryption of hardware does not mean encrypting the design file by a cryptographic algorithm, instead it means hiding the hardware's functionality. Researchers have previously used 'logic obfuscation' [4], [5] for this purpose. Obfuscation, however, has a different meaning in software. An obfuscated program is difficult to reconstruct even if its functionality is known. Obfuscation hides the implementation and not the function. To highlight this difference we use logic encryption to denote that the functionality is encrypted when the valid key is not applied to the design.



(a) A faulty circuit



(b) An encrypted circuit with a wrong key (K1 = 1) equivalent to the faulty circuit



(c) A circuit encrypted with three XOR gates (E1, E2, and E3)

Fig. 1. Relation between logic encryption and fault analysis in IC testing – fault excitation, propagation, and masking.

sequence of key bits. If the key is withdrawn, the design, once again, ends up in a black state. However, the effectiveness of these methods in producing a wrong output has not been demonstrated. In combinational logic encryption, XOR/XNOR gates were introduced at random locations in the design to conceal its functionality [4]. One of the inputs in these inserted gates serves as the key inputs. One can configure these inserted gates as buffers or inverters using these key inputs. The values applied to these key inputs are the *keys*. Using VLSI test principles, we show that, when gates are randomly inserted into the design, a wrong key does not necessarily affect the output as its effects may not be propagated to the outputs. To generate unique keys per chip, [17] calibrates the XOR/XNOR gates post-fabrication by tweaking the threshold voltage of the gates. Memory elements may also be inserted into the design [14]. In this case, the circuit will function correctly only when the memory elements are programmed correctly. However, the insertion of memory elements will incur significant area and performance overhead.

C. Proposed Approach – Leveraging VLSI Test Principles to Improve Logic Encryption

The following observations relate logic encryption and fault analysis in IC testing and are leveraged to guide the insertion of XOR/XNOR key gates.

Connection to Test Principle 1: Application of a wrong key is analogous to *excitation* of a fault. For a wrong key, either a stuck–at–0 (s–a–0) or stuck–at–1 (s–a–1) fault will get excited, when XOR/XNOR gates are used for encryption. This is illustrated for the C17 circuit encrypted with one XOR gate (E1) as shown in Figure 1(b). If a wrong key (K1=1) is applied to the circuit, the value of net B is the negated value of net A. This is the same as exciting a s-a-0 (when A=1) or s-a-1 (when A=0) fault at the output of G7 as shown in Figure 1(a).

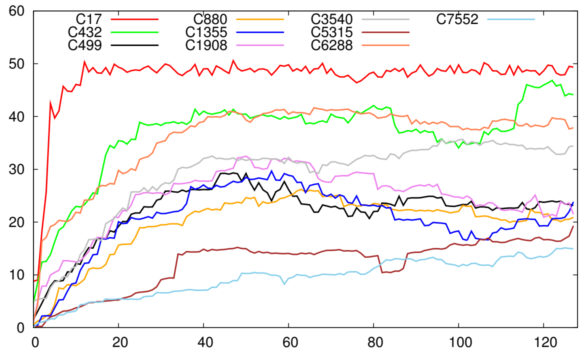
Connection to Test Principle 3: Corruption of an output due to a wrong key is analogous to the *propagation* of an excited fault. This is illustrated for the circuit shown in Figure 1(b). Let a wrong key (K1 = 1) be applied to the circuit. For the input pattern 00000, a s–a–0 fault gets excited at the output of E1 and propagated to both outputs. The value at the output of the gate E1 is 0 instead of 1, and the output is 11 instead of the correct output 00. For the input pattern 01110, even though the s–a–0 fault gets excited at the output of E1, the output is 00, which is the same as the functional output, as the fault effects have been blocked.

Connection to Test Principle 4: Cancellation of the effect of multiple wrong key bits is analogous to the *masking* of multiple excited faults in a faulty design. Consider the encrypted circuit in Figure 1(c). When key bits (K1, K2, and K3) are 000, the correct functional output is 00 for the input pattern ‘00000’. However, if the key bits are 111 (wrong key), the effect introduced by the XOR key gate E1 is masked by the XOR key gates E2 and E3, resulting in the undesired correct output 00.

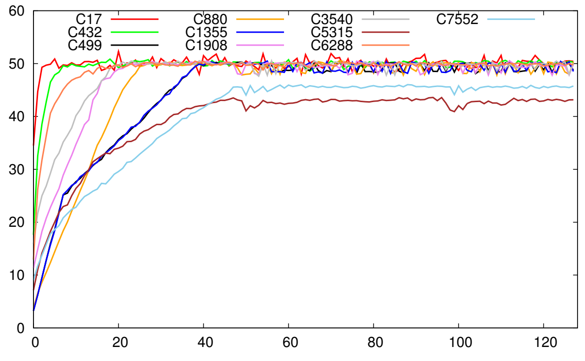
Restating Logic Encryption as a VLSI Test Problem: Insert XOR/XNOR gates such that a wrong key will affect 50% of the outputs. In terms of fault simulation, this goal can be stated as finding a set of faults, which together will affect 50% of the outputs when excited.

D. Results

Fault analysis-based key gate insertion is compared with random insertion [4] in Figure 2. When the XOR/XNOR key gates are



(a)



(b)

Fig. 2. Hamming distance comparison of (a) random insertion [4] and (b) proposed fault analysis based logic encryption for different ISCAS-85 benchmarks. X-axis is the number of key gates. Y-axis is the Hamming distance.

inserted at random locations in the circuit (as proposed in [4]), a Hamming distance of 50% is not achieved except for the smallest benchmark C17. The main reason for this poor performance is blocking of fault effects and fault masking. Fault analysis based insertion of key gates achieves the 50% Hamming distance goal for almost all the benchmarks as it considers the fault masking effects. The slopes of the lines in Figures 2(a,b) indicate the performance of logic encryption. If the line is steeper (as in the fault analysis based encryption), 50% Hamming distance is achieved with a smaller number of key gates.

IV. Security Analysis of Logic Encryption [9]

Envision an attack where the attacker applies specific input patterns, observes the outputs for these patterns, and deciphers the secret embedded key. To perform this attack, the attacker needs the encrypted netlist and a functional IC; these can be obtained in a variety of ways as shown in Figure 3.

Connection to Test Principle 2: is illustrated in Figure 4 (a). Determining the value of an unknown embedded key is analogous to

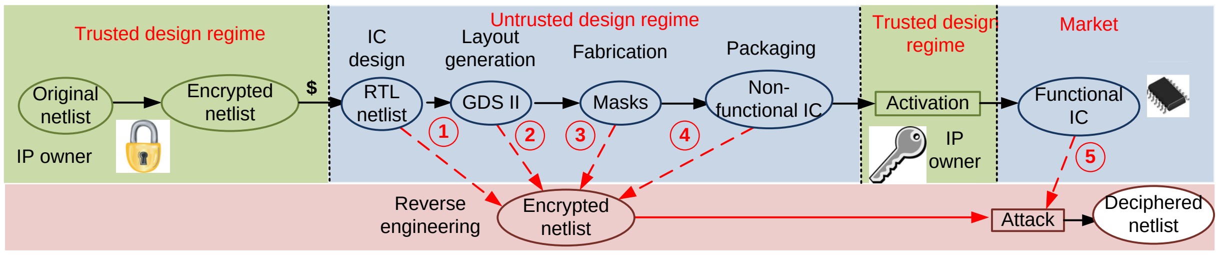


Fig. 3. The top part represents the naive logic encryption flow [4]. The design is in the encrypted form in the untrusted design regime. In the untrusted regime, an attacker can obtain the encrypted netlist from (1) the IC design, or by reverse engineering the (2) layout, (3) mask, or (4) a fabricated IC, and (5) the functional IC from the market. Using this attack, the attacker can get a deciphered netlist and make pirated copies.

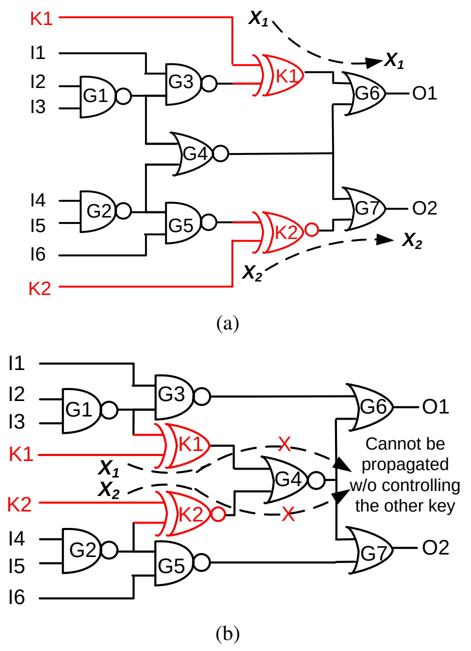


Fig. 4. (a) A circuit encrypted using key gates K1 and K2. By applying the input pattern 100000, an attacker can sensitize key bits K1 and K2 to the outputs O1 and O2, and observe their values. (b) The attacker cannot propagate the effect of key bits K1 and K2 individually to the outputs. Hence, the attacker has to brute force to determine the values of K1 and K2.

sensitizing[3] the key bit to an output without being masked/blocked by the other key bits/inputs. By observing the output, the sensitized key bit can be determined, given that other key bits (similar to unknown X-sources[4] in VLSI test) do not interfere with the sensitized path.

A. Strengthening Logic Encryption

To prevent attacks that aim at leaking the logic encryption key, key sensitization has to be hampered by inserting key gates judiciously. The attacker should be forced into decrypting the key bits using brute force. *This way, even full access to the netlist won't help the attacker circumvent the defense.* Consider the example circuit in Figure 4 (b) which is functionally identical to circuit in Figure 4 (a) but the two key gates *K1* and *K2* are inserted at different locations. The attacker then has to decrypt the two interfering keys bits together rather than individually.

Logic encryption can be strengthened by inserting key gates with complex interferences[5] among them. For this, we build an

[3]Sensitization of an internal line l to an output O refers to the condition (values applied from the primary inputs to justify the side input of gates on the path from l to O to the non-controllable values of the gates) which bijectively maps l to O and thus renders any change on l observable on O.

[4]X-sources: Uninitialized memory units, bus contentions or multi-cycle paths are the source of unknown response bits, i.e., unknown-Xs in testing. They are uncontrollable.

[5]An interference exists between two key gates if the value of one key bit cannot be propagated or sensitized without controlling or knowing the value of the other key bit.

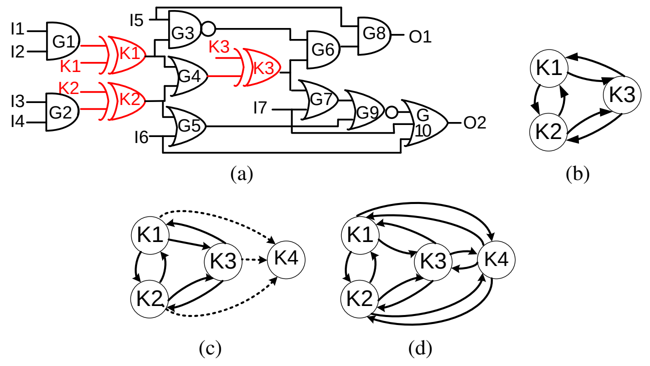


Fig. 5. (a) An example circuit with three key gates. (b) Interference graph of the key gates. Non-erasable interacting keys are denoted by solid edges. (c) If the new key gate is inserted at the output G10, it creates erasable edges (dotted lines) with the other key gates. (d) If the new key gate is inserted at the output G5, it creates non-erasable edges (solid lines) with the other key gates.

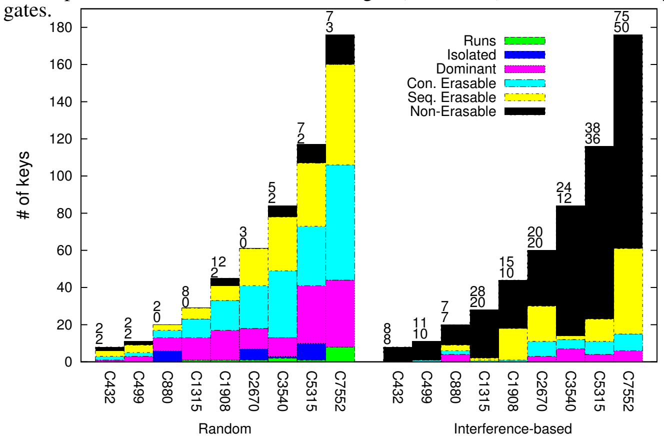


Fig. 6. Types of key gates inserted by different logic encryption techniques. Effective key size from an attacker's perspective (top) and from a defender's perspective (bottom) are shown on top the bars.

interference graph of key gates where each node is a key gate and an edge connects two nodes if the two gates interfere. If there are no input patterns that erase the effect of one key gate on another (i.e., mask of the effect of one key gate and determine the value of the other key-bit), the corresponding strong interference is represented by a solid edge (hard interference) between the two nodes. Otherwise, the corresponding (soft/weak) interference is represented by a dotted edge; this is a *pessimistic* approach by the defender, as sometimes key-effect-erasing patterns do not exist even in the presence of gates with controlling values in between the two key-gates (e.g., conflicting justification requirements, etc.).

To strengthen logic encryption, the number of mutually non-erasable edges in the interference graph should be maximized. The ultimate goal is to eradicate any sensitization opportunities of individual keys for the attacker (i.e., the use of **Test Principle 2**), forcing an attacker to use brute force to determine the values on these key gate inputs. In the example in Figure 5, if the new key gate K4 is inserted at the output of G5, then it creates non-erasable interference with the other key gates as shown in Figure 5(d). Thus, this location is preferred for the insertion of a new key gate.

1) Results: Figure 6 shows the results for different benchmarks and for random [4] and interference-based insertions. In the random insertion, most of the keys can be concurrently erased. Several keys are inserted in back-to-back runs benefiting the attacker. Only 30% of keys cannot be erased or can be erased sequentially requiring brute force by an attacker. In the "interference-based" insertion, runs of keys are not allowed and around 90% of keys are non-erasable. Isolated keys are absent in this case.

Due to the pessimism of the defender, the effective key size is the maximum number of non-erasable keys in a connected key interference graph. However, an attacker may be unable to find the key-effect-erasing patterns even for the erasable edges. Hence, from an attacker's perspective, the effective key size is the largest key size on which brute force is actually attempted.

V. Application of VLSI Testing to Split Manufacturing [10]

A. *Split Manufacturing*

Split manufacturing is another technique to improve the security of an IC, In this technique as the Front End Of Line (FEOL) and Back End Of Line (BEOL) layers are fabricated separately and combined post-fabrication [3]. This prevents a single foundry (especially the FEOL foundry) from gaining full access to the IC. For instance, without the BEOL layers, an attacker in the FEOL foundry can neither identify the 'safe' places within a circuit to insert trojans nor pirate the designs without the BEOL layers.

In split manufacturing [18], the layout of the design is split into two: the FEOL layers and BEOL layers, which are then fabricated separately in different foundries as illustrated in Figure 7. The FEOL layers consist of transistors and other lower metal layers ($\leq$M4) and the BEOL layers consist of the top metal layers ($>$M4). This corresponds to the partitioning of a gate level netlist into blocks where the transistors and wires inside a block form the FEOL layers, and the top metal wires connecting the blocks and the IO ports form the BEOL layers. Post-fabrication, the FEOL and BEOL wafers are aligned, integrated and tested [3], [18]. The asymmetrical nature of the metal layers facilitates split manufacturing. The top BEOL metal layers are thicker and have a larger pitch than the bottom FEOL metal layers. In one embodiment, the fabricated FEOL and BEOL wafers are integrated using electrical, mechanical, or optical alignment techniques and tested for defects by a system integrator [18]. In another embodiment, the FEOL wafer is fabricated in an advanced foundry and then sent to a trusted second foundry where the BEOL layout is built on top of it [18].

Transporting the FEOL wafers to the BEOL foundry or transporting the FEOL and BEOL wafers to the SoC integrator may present a challenge; these wafers are thin and might crack or delaminate during transportation. Alignment of the FEOL and BEOL layers and increase in die area to accommodate alignment structures present another challenge. Split manufacturing may also affect the signal integrity, timing of the signals that span the FEOL and BEOL layers, and other design-for-manufacturability aspects. The economic benefit of split manufacturing comes from performing the low cost BEOL layer fabrication in-house and outsourcing the expensive FEOL layer fabrication. While several projects [3], [18] focus on addressing these manufacturability challenges and make it feasible to reap the benefits of split manufacturing, we show that straightforward split manufacturing is inherently *insecure*, and propose ways to make it secure.

B. *Threat model*

The attacker is in the offshore foundry that manufactures the FEOL wafer. Since the attacker has the GDSII layout file of the design, he/she can reverse engineer it and obtain the gate-level netlist. Such reverse engineering techniques have been demonstrated [19]. The attacker in the FEOL foundry gains knowledge about most of the design (the transistors and the lower metal layers) except for the missing BEOL connections. Once the attacker determines these missing BEOL connections, he/she can reconstruct the original design.

Consider the ISCAS-85 combinational logic benchmark circuit, C17 shown in Figure 8. It is partitioned into partition A (light colored) and partition B (dark colored). Typically, the wires within a partition (local wires except Vdd and clock) are assigned to lower metal layers.

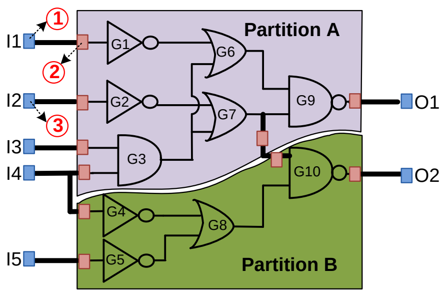


Fig. 8. The C17 benchmark circuit. Thin lines denote FEOL nets while the thick lines denote the BEOL nets.

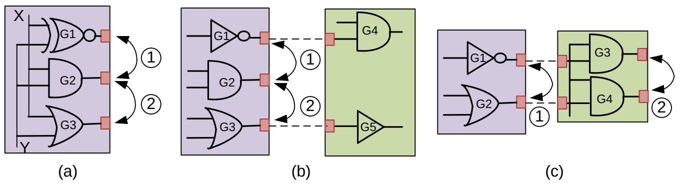


Fig. 9. IC testing principles applied to split manufacturing: (a) Pin that has a logical value opposite to that of the swapped pin is preferred (fault excitation). ① Values at $P_{G1,A,out}$ and $P_{G2,A,out}$ differ only when X=Y=0; ② Values at $P_{G2,A,out}$ and $P_{G3,A,out}$ differ in two cases: X=1, Y=0 and X=0, Y=1. Thus, $P_{G3,A,out}$ is selected as the swapping pin for $P_{G2,A,out}$. (b) If $P_{G1,A,out}$ is selected as the swapping pin for $P_{G2,A,out}$, the wrong value will propagate only when the other input of G4 is 1. However, if $P_{G3,A,out}$ is selected as the swapping pin, the buffer, G5, will always propagate the wrong value (fault propagation). (c) The logical error introduced by swapping $P_{G1,A,out}$ and $P_{G2,A,out}$ is cancelled by swapping $P_{G3,B,out}$ and $P_{G4,B,out}$ (fault masking).

The wires that span the partitions and I/O ports are assigned to higher metal layers. This makes routing easier [20]. The nets connecting the input ports I1-I5 to the corresponding inputs of the gates G1-G5 use the BEOL layers. The nets connecting the output of gates G9 and G10 to output ports O1 and O2, respectively, use the BEOL layers. The net that connects the output of G7 to one of the inputs of G10 also uses the BEOL layers.

C. *Vulnerability of Split Manufacturing to Proximity Attack*

Every missing BEOL connection is a net that connects a target pin (an output pin of a partition or an input port of the design from which a signal originates) and its corresponding candidate pin (an input pin of a partition or an output port of the design at which a signal terminates). A target pin has many candidate pins but the attacker is trying to determine the correct candidate pin for the target pin with the following objective: if he/she can connect every target pin with its correct candidate pin, he/she can recover the original design.

The **proximity attack** exploits the heuristic that floorplanning and placement (F&P) tools use to reduce the wiring (delay) between the pins to be connected [20] – place the partitions close by and orient the partitions. *This heuristic of most F&P tools is a security vulnerability that can be exploited by an attacker in the FEOL foundry who does not have access to the BEOL layers.* Thus, the attacker simply makes the missing connections between the two closest pins.

D. *Leveraging VLSI Fault Analysis Techniques to Strengthen Split Manufacturing*

Proximity attack can be overcome by rearranging the pins such that they are no longer closest to the pins that they are supposed to connect to. A proximity attacker will thereby be deceived into making the wrong BEOL connections.

The objective of a designer then is to swap a sufficient number of pins such that the functionality of the deceiving netlist[6] differs from that of the original netlist. The Hamming distance between the outputs of the original netlist and the deceiving netlist can quantify the functional difference between these netlists. Ideally, the Hamming distance should be 50%.

To find a swapping pin for a target pin, similar to an attacker, the defender can build the list of candidate pins for that target pin. Then, he/she can randomly select the swapping pin from that list.

[6]A deceiving netlist is created by the defender by swapping the pins.

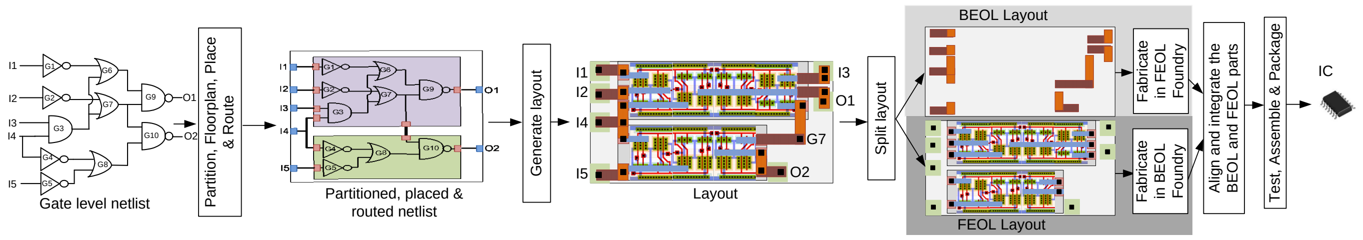

I1
I2
I3
I4
I5
G1
G2
G3
G4
G5
G6
G7
G8
G9
G10
O1
O2
Gate level netlist
Partition, Floorplan, Place & Route
Partitioned, placed & routed netlist
Generate layout
Layout
G7
Split layout
BEOL Layout
FEOL Layout
Fabricate in FEOL Foundry
Fabricate in BEOL Foundry
Align and integrate the BEOL and FEOL parts
Test, Assemble & Package
IC

Fig. 7. Split manufacturing aware design flow: The layout obtained from the traditional design flow is split into two parts– the FEOL part containing the transistors and the lower metal layers and the BEOL part containing the top metal layers– which are then fabricated in two different foundries. Either the designer or the BEOL foundry assembles the FEOL and BEOL wafers into the final IC.

Unfortunately, such random selections might not guarantee that the attacker will get a wrong output on making a wrong connection. Hence, we use VLSI test principles to select the swapping pin for a target pin in order to achieve the 50% Hamming distance objective.

Connection to Test Principle 1: The fact that different values on the swapping and the target pin introduces an error is analogous to the *excitation* of a fault. Ideally, such pins must be swapped to achieve the 50% Hamming distance objective. Otherwise, the resulting design, even with wrong connections, will still produce mostly correct outputs.

Connection to Test Principle 3: Corruption of an output due to two swapped pins is analogous to the *propagation* of an excited fault. Ideally, swapping should be done so that many outputs are corrupted (fault propagates to many outputs).

Connection to Test Principle 4: Cancellation of the effect of multiple swapped pin pairs is analogous to the *masking* of multiple excited faults. Sometimes, logical values corrupted by swapping pins in partition A can be restored to their original value because of swapping pins in partition B.

These scenarios are illustrated in Figure 9. In order to account for the points above, we define a fault impact metric which guided our preliminary pin swapping.

E. Results

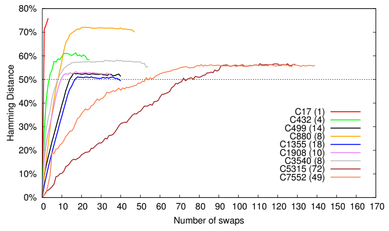
Figure 10(a) shows the Hamming distance between the outputs of the original design and the deceiving netlist obtained after swapping partition pins based on the proposed fault analysis. Fault analysis based swapping achieves at least 50% Hamming distance for all the benchmarks. This is because, it accounts for the fault activation, propagation, and masking effects in pin-swap selections. Furthermore, the curves are steep in fault analysis based swapping. This indicates that fault analysis based swapping will take a small number of swaps to achieve the 50% Hamming distance mark. Figure 10(b) depicts the Hamming distance between the outputs of the original design and the design reconstructed using proximity attack. When proximity attack is performed, the Hamming distance between the outputs of the original and reconstructed designs reduces, which highlights the effectiveness of the proposed attack. This is particularly evident in the case where no defense is used (average Hamming distance is around 10%). The average Hamming distance for ‘Fault-analysis swap defense + Proximity connections attack’ technique is 42%

Figure 10(c) shows the percentage of partition pins and IO ports that are correctly connected by an attacker using proximity attack. In case of ‘No defense + Proximity connections attack’, the average number of correct connections is 96%; for the C17 benchmark, the attacker is able to make all the connections correctly. This verifies our earlier claim that F&P tools place pins, which will be connected in BEOL, closer to each other and indicates that straightforward split manufacturing can be easily compromised. In case of ‘Fault-analysis swap defense + Proximity connections attack’, the attacker connects 87% of the pins correctly, because only a small number of pins were swapped; for most of the designs at most 20 pins were swapped. The remaining 13% of wrong connections create 50% Hamming distance.

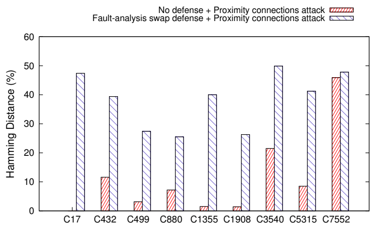
VI. Research Impact

A. Grants

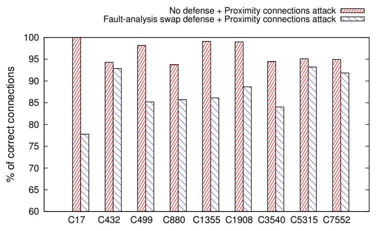
Air Force Research Labs awarded a grant to build a tool that performs logic encryption using IC testing principles. IARPA’s Trust-in-Integrated-Circuits program will be using the proximity attack to assess the security of split manufacturing techniques.



(a)



(b)



(c)

Fig. 10. Split manufacturing security assessment: (a) Hamming distance: original vs swapped-pinned design. (b) Hamming distance: original vs proximity-attack-reconstructed design.(c) Percentage of correct connections by a proximity attacker.

B. Outreach

The proposed concepts on using VLSI testing techniques is being presented as a tutorial in several conferences including ICCD 2012 [21], VTS 2012 [22], DATE 2013 [23], ETS 2013 [24], NATW 2013 [25], ISQED 2013 [26], and LATW 2013 [27].

VII. Conclusion

By relating VLSI testing concepts to hardware security, we:

- transferred control over an IC design back to the fab-less IC designers, who had relinquished their control to reduce cost by out-sourcing,
- ensured safeness and security of electronic components,
- successfully combat the rogue elements as they intrude the supply chain, and
- showed how existing CAD tools can be adopted for security.

References

[1] "International Technology Roadmap for Semiconductors." http://www.itrs.net/Links/2011ITRS/Home2011.htm.

[2] DIGITIMES Research, "Trends in the global IC design service market." http://www.digitimes.com/Reports/Report.asp?datepublish=2012/3/13\&pages=RS\&seq=400\&read=toc.

[3] Intelligence Advanced Research Projects Activity, "Trusted Integrated Circuits Program." https://www.fbo.gov/utils/view?id=b8be3d2c5d5babbdffc6975c370247a6.

[4] J. Roy, F. Koushanfar, and I. Markov, "EPIC: Ending Piracy of Integrated Circuits," *Proceedings of the IEEE/ACM Design, Automation and Test in Europe*, pp. 1069–1074, 2008.

[5] R. Chakraborty and S. Bhunia, "HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.

[6] "http://spectrum.ieee.org/semiconductors/design/creative-winners-in-hardware-trojan-contest,"

[7] 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.

[8] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, " Logic encryption: A fault analysis perspective," *in the Proc. of IEEE Design Automation and Test in Europe Conference*, pp. 953–958, 2012.

[9] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, " Security analysis of logic obfuscation," *in the Proc. of IEEE/ACM Design Automation Conference*, pp. 83–89, 2012.

[10] J. Rajendran, O. Sinanoglu, and R. Karri, " Is Split Manufacturing Secure?," *in the Proc. of IEEE Design Automation and Test in Europe Conference*, 2013.

[11] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, " Applying IC Testing Principles to Secure ICs," *in the Proc. of Government Microcircuit Applications and Critical Technology Conference*, 2012.

[12] M. L. Bushnell and V. D. Agrawal, "Essentials of Electronic Testing for Digital, Memory, and Mixed-Signal VLSI Circuits," *Kluwer Academic Publishers, Boston*, 2000.

[13] H. Heys and S. Tavares, "Avalanche characteristics of substitution-permutation encryption networks," *IEEE Transactions on Computers*, vol. 44, no. 9, pp. 1131 –1139, 1995.

[14] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC Piracy Using Reconfigurable Logic Barriers," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[15] G. Suh and S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," *Proceedings of the IEEE/ACM Design Automation Conference*, pp. 9–14, 2007.

[16] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," *Proceedings of USENIX security*, pp. 291–306, 2007.

[17] W. Griffin, A. Raghunathan, and K. Roy, "CLIP:Circuit Level IC Protection Through Direct Injection of Process Variations," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 20, no. 5, pp. 791–803, 2012.

[18] R. Jarvis and M. G. McIntyre, "Split manufacturing method for advanced semiconductor circuits," *US Patent no. 7195931*, 2004.

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

[20] N. A. Sherwani, "Algorithms for VLSI Physical Design Automation," 2002.

[21] "Hardware security and trust," *International Conference on Computer Design*, 2012.

[22] "Hardware security and trust," *IEEE VLSI Test Symposium*, 2012.

[23] "Hardware security and trust," *IEEE/ACM Design Automation and Test in Europe*, 2013.

[24] "Reconciling the dichotomy between ic testing and security," *European Test Symposium*, 2013.

[25] "Vlsi test and security," *IEEE North Atlantic Test Workshop*, 2013.

[26] "Hardware security and implications on design flows," *IEEE International Symposium on Quality Electronic Design*, 2013.

[27] "Regaining hardware trust by leveraging test techniques and tools," *IEEE Latin American Test Workshop*, 2013.