Chapter 7
Hardware Hardening Approaches Using Camouflaging, Encryption, and Obfuscation

Qiaoyan Yu, Jaya Dofe, Yuejun Zhang, and Jonathan Frey

7.1 Introduction

Under the current global business model, the integrated circuit (IC) supply chain typically involves multiple countries and companies which do not use the same regulation rules. Despite reducing the cost of fabrication, assembly and test, the globalized IC supply chain has led to serious security concerns. Intellectual property (IP) piracy, IC overbuilding, reverse engineering, physical attack, counterfeiting chip, and hardware Trojans are recognized as critical security threats on maintaining the integrity and trust of ICs [6, 7, 14, 24, 27, 29–32, 38, 45, 47, 48, 63]. Those threats could cause companies lose profit and challenge the national security [1]. Hardware counterfeiting and IP piracy cost the US economy more than 200 billion dollars a year on defense and military related chips [33]. Growing hardware IP piracy and reverse engineering attacks challenge traditional chip design and fabrication [24, 29, 32, 47, 48, 50]. Thus, it is imperative to develop countermeasures to address those attacks.

IP piracy is illegal or unlicensed usage of IPs. An attacker can steal valuable hardware IPs in the form of register-transfer-level (RTL) representations ("soft IP"), gate-level designs directly implementable in hardware ("firm IP"), or GDSII design database ("hard IP"), and sell those IPs as genuine ones [10]. Hardware IP

Q. Yu (✉) • J. Dofe
University of New Hampshire, Durham, NH 03824, USA
e-mail: qiaoyan.yu@unh.edu; jhs49@wildcats.unh.edu

Y. Zhang
Ningbo University, Ningbo, Zhejiang, China
e-mail: zhangyuejun@nbu.edu.cn

J. Frey
Charles Stark Draper Laboratory, Inc., Cambridge, MA 02139, USA
e-mail: jfrey@draper.com

© Springer International Publishing AG 2017

P. Mishra et al. (eds.), *Hardware IP Security and Trust*,
DOI 10.1007/978-3-319-49025-0_7

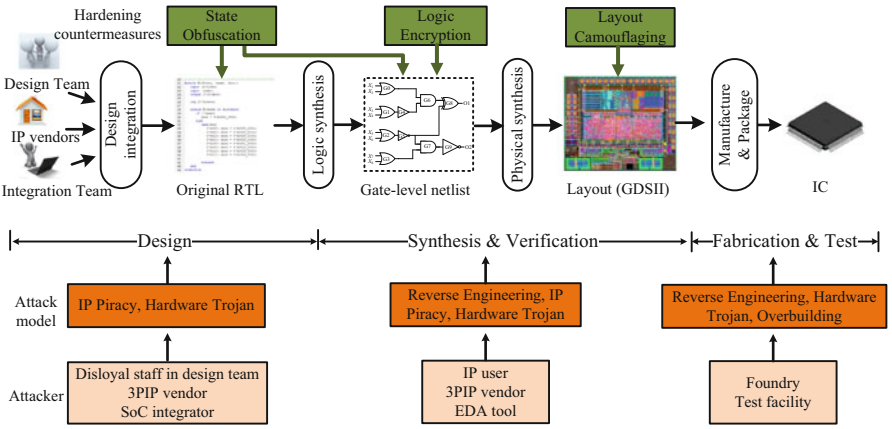

Fig. 7.1 Applying three hardware hardening techniques in the IC design flow

reusing in systems-on-chip (SoCs) design is prevalent practice in silicon industry as it reduces design time and cost dramatically [8]. The IP piracy attack can take place at various stages in the IC supply chain. As shown in Fig. 7.1, the potential IP piracy attackers could be designer, third-party IP (3PIP) vendor, and SoC integrator at design, synthesis, and verification stages. In the fabrication stage, an untrusted foundry may overbuild the IP cores and sell them under a different brand name to make profit.

Reverse engineering of an IC is a process of identifying its structure, design, and functionality [43]. Product teardowns, system-level analysis, process analysis, and circuit extraction are different types of reverse engineering [53]. Using reverse engineering, one can: (1) identify the device technology [29], (2) extract gate-level netlist [53], and (3) infer the chip functionality [24]. Several techniques and tools have been developed to facilitate reverse engineering [15, 37, 53]. Traditionally, reverse engineering is a legal process as per US Semiconductor Chip Protection Act for teaching, analysis, and evaluation of techniques incorporated in mask work [25]. However, reverse engineering is a double-edged sword. One could use reverse engineering techniques to pirate ICs. Reverse engineering attack can be done at various abstraction level of supply chain depending upon the attacker's objectives.

The main focus of this chapter is to review hardware hardening techniques to counteract with IP piracy and reverse engineering attacks. More specifically, we devote this chapter to study active countermeasures like camouflaging, logic encryption, and obfuscation. Different terminology used in this chapter is summarized in Sect. 7.2. The state-of-the-art hardware design hardening methods are reviewed in Sect. 7.3. In Sect. 7.4, a dynamic deflection-based design obfuscation method is presented. Furthermore, we introduce design obfuscation for three-dimensional (3D) ICs in Sect. 7.5. We conclude this chapter in Sect. 7.6.

7.2 Terminology

Before we start to introduce countermeasures for hardening at different levels, we define the terminology that will be used in the remainder of this chapter. Due to similar concepts having different names in the literature published in this field, we categorize the hardware hardening methods into three classes:

(1) *Camouflaging*: Camouflaging is a means of disguising the original presence of the design, more specifically, at layout level. This is a type of countermeasure against the image-analysis-based reverse engineering attack.
(2) *Logic Encryption*: Logic locking, logic obfuscation, and logic encryption are three close concepts that are applied to counteract piracy and reverse engineering attacks of combinational logic circuits. The main principle of logic locking/obfuscation/encryption is to insert key-controlled logic gates to the original logic netlist such that the modified netlist will not function correctly without the right key. Hereafter, we use logic encryption to refer this category. Ideally, the key-controlled logic gates are mixed with the original logic gates; only the countermeasure designer would be able to differentiate the protection gates from the original ones.
(3) *State Obfuscation*: Obfuscation is often utilized to obscure sequential circuits. Similar to logic encryption, a key sequence controls the correct state transitions. State obfuscation typically preserves the original state transitions and adds several key-authentication states before the power-up state. Theoretically, the success rate of IP piracy or reverse engineering attacks is extremely low if the attacker uses brute force methods to find the key.

7.3 State of the Art

This section summarizes three main categories of hardware hardening methods that can resist reverse engineering and IP piracy attacks.

7.3.1 Camouflaging

Circuit camouflaging is a technique that makes subtle changes within the physical layout of a logic cell to obfuscate its actual logic function [18, 19, 43]. The main goal of circuit camouflaging is to disguise the circuit against a reverse engineer who utilizes scanning electron microscopy (SEM) pictures to recover the original chip design. For instance, from the SEM image analysis, a camouflaged logic cell appears as a 2-input NAND gate; however, that logic cell is a 2-input NOR gate in reality. Such a mislead could be achieved by a subtle change on a metal contact. As shown in Fig. 7.2, the contact between metal 1 (M1) and the dielectric is fully connected

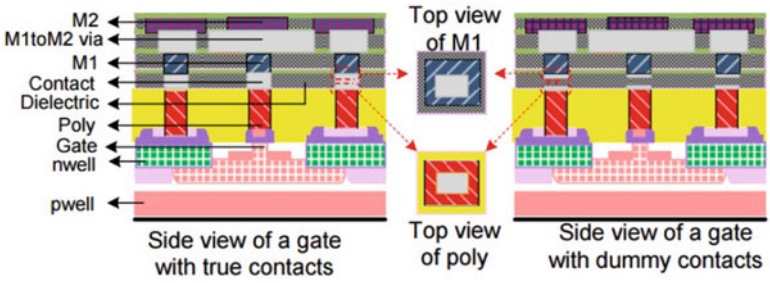

Fig. 7.2 Cross-section views of non-camouflaged contact (*left*) and camouflaged contact (*right*) [43]

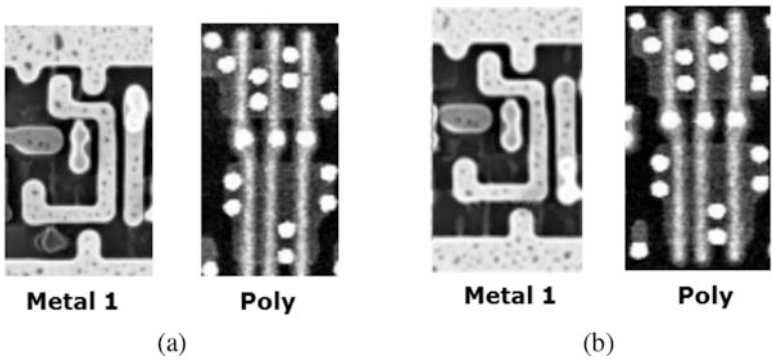

Fig. 7.3 SEM images of (**a**) conventional and (**b**) Camouflaged 2-input AND gate [19]

in the left cell but disconnected after a thin layer of contact material in the right cell. The top views of M1 (and polysilicon) for the gates w/wo camouflaging are identical. The work [19] illustrates that the conventional and camouflaged 2-input AND gates have the same SEM image, as shown in Fig. 7.3. As a result, the attacker could easily extract a wrong netlist if he/she relies on the SEM image analysis. Furthermore, because of the wrong netlist extraction, the attacker cannot precisely modify the netlist to insert hardware Trojans.

The general principle of camouflaging is either to make the connected nodes appear to be isolated, or to have the isolated nodes appear to be connected. In real applications, one can partially camouflage the circuit using a strong custom cell camouflage library [16, 17, 19] or a generic camouflaged cell layout [43]. In the camouflage library [19], every cell looks identical at every mask layer to prevent automated layout recognition [16, 17]. However, making every cell identical in terms of size and spacing with other cells will limit area optimization for the improvement of cell performance. The generic camouflaged cell [43] can perform either as an exclusive-OR (XOR), NAND, or NOR gate, depending on whether true or dummy contacts are used in the cell layout. Rajendran et al. [44] analyze the vulnerability of randomly chosen nodes for camouflaging. Their work shows that

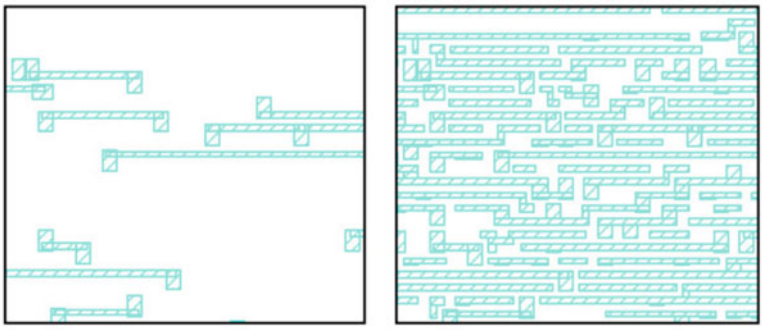

Fig. 7.4 Layout before (*left*) and after (*right*) using camouflaged smart fill of poly, contact, and active layers [19]

the reverse engineering on the camouflaged circuit can still reveal the logic function if the attacker exploits the circuit outputs together with SEM images. Rajendran et al. [43] further propose a smart camouflaging algorithm to increase the difficulty for reverse engineering. The camouflaged standard cells can also be applied to every logic gate [19]. As pointed out above, to camouflage the cells, different logic gates are designed in a way that every cell has the same size and spacing to its neighbor cells. This contradicts the use of transistor sizing for delay optimization. Hence, completely camouflaging the entire chip will sacrifice the system performance, despite the gain of resistance against reverse engineering and piracy attacks. To address this limitation, Cocchi et al. provide elementary block camouflage libraries that are derived from each logic gate layout [17] and could be programmed after the manufacturing process.

In addition to camouflaging, the contacts in a logic cell, metal, active, and device layers can be filled with camouflaged smart fills, which consume all silicon area in the processed layers [19]. Figure 7.4 shows an example of before and after using camouflaged smart fills. As a result, there is no space for hardware Trojan insertion. Meanwhile, the smart fill adds one more task for the attacker to differentiate the real connection trace from the dummy one before he/she identifies the true contacts within one logic cell.

7.3.2 Logic Encryption

Logic encryption methods insert additional logic gates to the original logic netlist, where the additional logic gates are controlled by key bits. An incorrect key leads to netlist mismatch compared to the original netlist and produces different functional outputs than the original ones. The early work [48] introduces the concept of logic encryption as shown in Fig. 7.5. Physical Unclonable Function (PUF) circuit produces a unique challenge. The user key can be stored in a tamper-proof memory.

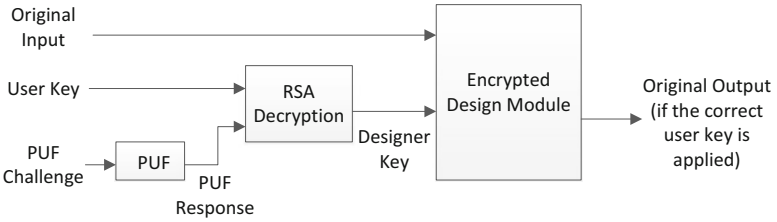

Fig. 7.5 Example of logic encryption structure

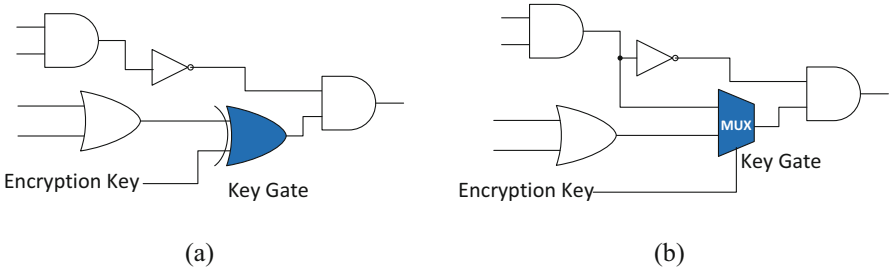

Fig. 7.6 Examples of lightweight logic encryption based on (**a**) XOR and (**b**) MUX key gates. Note, the *solid gates* are key gates and the *unsolid gates* belong to the original netlist

The correct designer key to unlock the encrypted design module is derived from the RSA [49] decryption unit, in which the user key and PUF response serve as ciphertext and crypto key, respectively.

As RSA [49] decryption unit costs 10,000 gates in its ASIC implementation [48], researchers propose to replace RSA using lightweight logic encryption cells, such as XOR/XNOR [40, 51]. Alternatively, multiplexers (MUX) [39, 46, 56] can also be used to encrypt the logic. Figure 7.6 shows examples of simplified logic encryption. In the XOR-based encryption [40, 41, 46, 48, 51, 60], an incorrect key bit may flip the primary output. In the MUX-based encryption [39, 46, 56], an arbitrary internal net is added as an input for the key gate. Without the correct key, the key gate may select the arbitrary internal net for the primary output computation.

7.3.2.1 Attacks Against Logic Encryption

Although combinational logic can be hardened by logic encryption methods mentioned above, there exist several specific attacks that could defeat this type of countermeasures by reducing the number of brute force attempts for key recovery. The existing attack models assume that the attacker owns a locked logic netlist, and he/she either has a copy of the activated chip (as a black box) from the open market or is able to measure the correct chip outputs with given test patterns.

The work [41] demonstrates a sensitization attack, which generates key-leaking input patterns and then passes the key bits to the primary outputs with those input patterns. Lee and Touba [35] propose to perform the brute force attack on the logic cone with the fewest key bits; after the subset of a key vector is retrieved, they continue the same attack iteratively on the other logic cones until all key bits are found. Subramanyan et al. propose a stronger satisfiability checking (SAT) attack [52] that can defeat the encrypted circuit against fault-analysis attacks [41, 46, 61]. The SAT attack is based on the observation that the encrypted netlist with the correct key should generate the same outputs as those from the activated chip, no matter what input pattern is applied to the primary input. Instead of searching for the correct key, the SAT attack looks for a member of the equivalence class of keys that generate the correct output for all input patterns. The SAT attack iteratively searches for the distinguishing inputs (DIP) that cause the inconsistency between the outputs from the netlist using a wrong key and from the unlocked netlist. When no more DIP can be found, the equivalent class of keys is reached. In a hill-climbing attack [39], a key bit is randomly flipped to check whether the output of the netlist with the new key matches to that of the unlocked netlist. The output inconsistency reveals the correct key bit by bit.

7.3.2.2 Logic Encryption Algorithms

The original proposal for logic encryption is EPIC [48], in which XOR/XNOR key gates are randomly placed in the IC. Fault-analysis techniques and tools [34] are exploited to perform stronger logic encryption [41, 46], through which the Hamming distance between the outputs after applying the valid key and an incorrect key is maximized close to 50 %. AND or OR gates can also be used to encrypt the design [23]. In an XOR/XNOR gate locking scheme, a key gate is either an inverter or a buffer. An incorrect key turns the key gate into an inverter. The key gate with the correct key acts as a buffer or a delay unit. Instead of inserting XOR/XNOR key gates, Baumgarten et al. [5] suggest dividing the entire IP into fixed and configurable portions. The configurable portion is implemented as a key-controlled lookup table (LUT). An adversarial foundry is able to observe the geometry of the fixed circuit portion but does not have a clear picture of the configurable portion. This is because the geometry of the configurable circuit portion is composed of generic memory layout, which is programmed by the IP creator after chip fabrication. Encryption using configurable logic is also discussed in the literature [36], in which the logic type can be changed after fabrication.

SAT attacks are capable of breaking most of encrypted logic within a few hours, with the assistance of state-of-the-art SAT solvers. An AES module is utilized to protect the key inputs and thus increase the time for solving a SAT formula [61]. To reduce the hardware cost of AES, a lightweight Anti-SAT unit that outputs 0 (1) for an invalid (valid) key is proposed in [58] to mitigate the SAT attack on logic encryption. Alternatively, key scrambling and logic masking techniques are integrated with the early logic encryption work [41] in their recent work [62].

7.3.3 State Obfuscation

State obfuscation is a countermeasure for sequential circuit protection. The main purpose of the obfuscation is to prevent the adversary from obtaining the clear state transition graph. A remote activation scheme [2] is one of the early works to address IP piracy through the concept of obfuscation. The continuous state transition is granted by a key uniquely generated by a PUF. This method triples one of the existing states; only one of the tripled states protected with the correct key will further drive the state transitions. Hereafter, we refer this method as *DUP*. The piracy-aware IC design flow in [48] enriches the RTL description with an on-chip true random number generator and public-key cryptographic engine. The authors extended their work further in [32], in which a channel from the normal state transition to the black hole states is added to prevent the state from returning to the normal operation.

An SoC design methodology hardware protection through obfuscation of netlist (HARPOON) is presented for IP protection [10]. The main principle of this method is to add a key-controlled obfuscation mode before the finite state machine (FSM) enters the normal mode. Theoretically, an adversary cannot reach the normal mode without a correct key sequence. This method does not protect the normal mode states based on the assumption that the key exploration space is too big for the attacker to reach the true power-up state. Their obfuscation concept is extended to the register transfer level in [11]. The RTL hardware description is converted into control and data flow graphs and then dummy states are added to obfuscate the design with an enabling key sequence. In the gate-level obfuscation work [9], the output of the FSM in the obfuscation mode also flips a few selected logic nodes. Their follow-up work [13] increases the number of unreachable states for the obfuscation mode to thwart hardware Trojan insertion.

The work [20] suggests using obfuscation to thwart hardware tampering. Different than using an external key sequence in [10], this work locks the design with an internally generated Code-Word (similar to a key sequence), which is based on a specific state transition path. This method aims at the key-leaking issue in [10, 11, 13]. Although an incorrect key will cause the FSM temporarily deviate from the correct transition path, the normal state transition will resume after one or several state transitions.

The previous key-based obfuscation works [2, 9, 11, 12, 41] provide a strong defense line against the reverse engineering attacks on the fabricated chip without a correct key. This is true because the number of possible key combinations is too large for brute force attack to hit the right key. As fabless IP cores are prevalent in SoC integration, IP vendors should be aware of the advanced attackers, who may use electronic design automation (EDA) tools to analyze the netlist and obtain information like code coverage, net toggle activities, and thermal activities to accelerate the process of revere engineering. Thus, it is imperative to re-evaluate the previous obfuscation methods under the assumption that EDA tools may reveal obfuscation details.

7.4 Dynamic State-Deflection-Based Obfuscation Method

In this section, we present a new state obfuscation method [21], which considers the reverse engineering attack through EDA tools.

7.4.1 Overview of DSD Obfuscation

Our dynamic state-deflection (DSD) method in [21] protects each normal state ST_x with a black hole state cluster B_x, as shown in Fig. 7.7a. If an incorrect key *!KS* is applied to the FSM, a normal state is deflected to its own black hole cluster. The FSM never returns to the normal state once it enters the black hole cluster. Each black hole cluster is composed of multiple states (only three are shown in Fig. 7.7a), rather than a single one. This is because we map each black hole state to a unique incorrect key (to save the hardware cost, multiple incorrect keys can share one black hole state). As we cannot predict which incorrect keys will be applied by the attacker, we propose a *Mapfunc* function to dynamically assign one black hole state to an incorrect key. The state within the black hole cluster does not remain unchanged; instead, it constantly switches to other black hole states. More details of black hole state creation and dynamic transition are provided in Sects. 7.4.2 and 7.4.3, respectively.

Considering that the gate-level netlist does not provide a clear picture of used and unused states, we use a state flip vector *FlipBit* to selectively invert the state bits using the circuit shown in Fig. 7.7b. Thanks to the inherent feedback loops, the FSM state bits will naturally switch over time without the predetermined transition specification. The modification on the state transition graph can be described with a Markov Chain transition matrix, which indicates the probability for a state transition to another state. Let's use $S_0, \ldots, S_{i-1}$ to represent the states for authentication/obfuscation/black hole cluster and use $S_i, \ldots, S_n$ to represent the normal operation states. The Markov Chain matrix of our obfuscation method is shown in Fig. 7.7c. The non-zero $g_{i(n-1)}$ stands for the single transition path from the obfuscation mode to the normal operation mode. The non-zero sub-matrix H (composed of h_{ij}) in the matrix means that our method creates the channel for each normal state to enter one of the black hole states.

The transition matrices for other methods are shown in Fig. 7.8. According to Fig. 7.8a, the DUP method [2] significantly increases one of the self-transition probability (e.g., p_{ii}) and reduces other probabilities on the same row; no transition paths are strictly blocked by the key sequence. If the attacker runs sufficient amount of simulations, he/she will observe a p_{ii} of high probability. Theoretically, overwriting the state register when the state remains at the same value too long could be an effective tampering technique. The two zero sub-matrices for the HARPOON [10] transition matrix shown in Fig. 7.8b obstruct the transition between the obfuscation states and the true functional states except the reset state in the

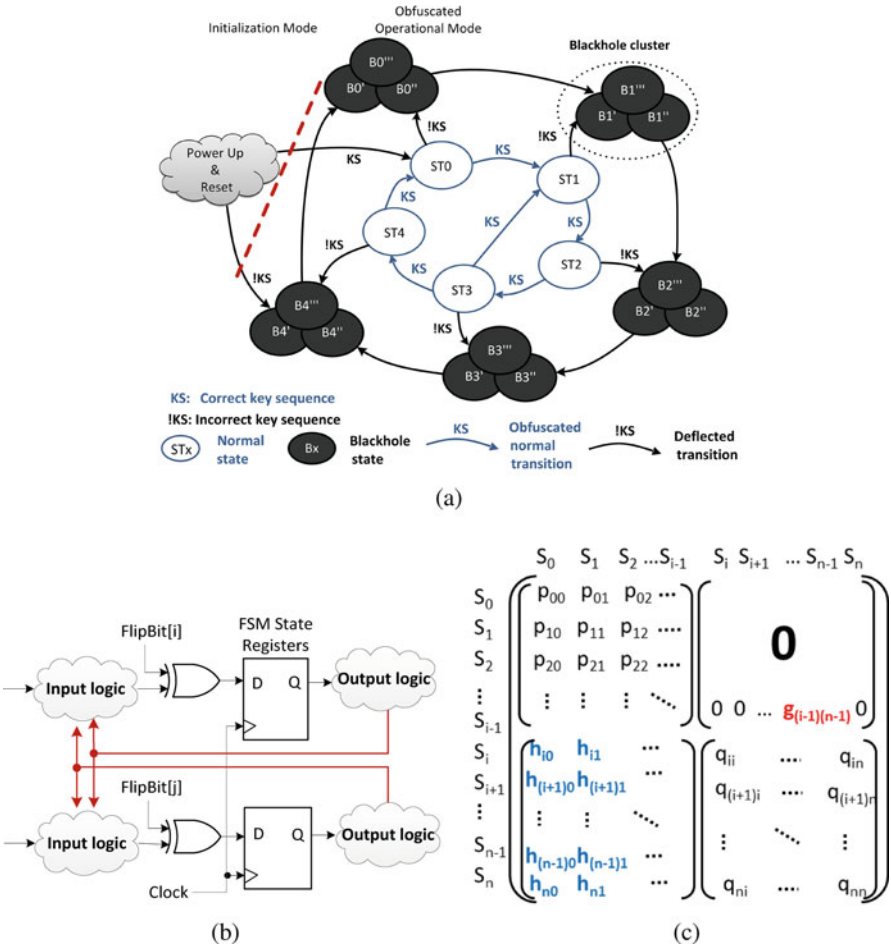

Fig. 7.7 Proposed dynamic state-deflection method for netlist obfuscation. (**a**) State transition graph, (**b**) circuit example of our method, and (**c**) proposed state transition paths represented with a Markov Chain matrix

functional mode (therefore $q_{(i-1)i}$ is the single non-zero probability). The P (element p_{ij}) and Q (element q_{ij}) matrices are strictly confined in its own sub-matrix. If the attacker overwrites the original state, the key obfuscation step can be bypassed. The matrix shown in Fig. 7.8c is the Code-Word-based obfuscation method [20], in which the obfuscated states are multiple ad-hoc states, thus allowing the FSM return to the normal state later.

As can be seen, our method is an extension version of HARPOON and DUP. The main difference is that our H matrix is not a zero matrix, which means the normal states will transition to obfuscation states if an incorrect key is used or if tampering is detected. Intuitively, our method can protect the FSM even if the attacker makes

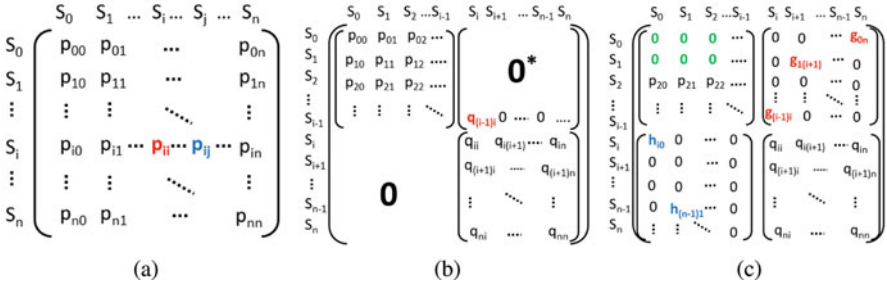

Fig. 7.8 Transition Markov chain matrices. (**a**) DUP [2], (**b**) HARPOON [10], and (**c**) CodeWord [20]

the FSM start in one of the normal states. This is because the FSM could leave the normal mode due to the key authentication before the next state transition. As a result, the workload of reverse engineering dramatically increases than any of the other methods discussed above.

7.4.2 Black Hole State Creation for Gate-level Obfuscation

In the RTL obfuscation methods, the team that adds the dummy states to the obfuscation mode (i.e., the obfuscation mechanism provider) needs to clearly know (1) what states have been used in the normal operation mode, and (2) the exact signals that drive the state transitions. Thus, the obfuscation mechanism provider is assumed to have sufficient knowledge of the original design. The goal of our method is to eliminate this assumption. We propose to obfuscate the gate-level netlist without the prior knowledge of the states in use and signals triggering the state transitions. We believe that the gate-level obfuscation is more attractive and flexible than the RTL obfuscation, as we can decouple the design obfuscation process from the original design.

In our method, we assume that the obfuscation provider is able to learn the number of registers that represent the state binary bits by reading the gate-level netlist. This is reasonable as one can differentiate flip-flop devices from combinational logic gates in the netlist. We form the black hole states shown in Fig. 7.7a with the following procedure [21]. First, each state (e.g., $S3, S2, S1, S0, B3, B2, B1, B0$) in the new FSM is composed of all the original state bits ($S3, S2, S1, S0$) and several newly added flip-flop bits ($B3, B2, B1, B0$). When the correct key sequence is applied, the newly added flip-flops ($B3, B2, B1, B0$) are set to zero and thus the new state keeps consistent with the FSM state before our state extension. This arrangement guarantees that the newly added flip-flops do not change the original function if the key is correct. If $B3, B2, B1, B0$ is non-zero due to an invalid key, then certainly the new state ($S3, S2, S1, S0, B3, B2, B1, B0$) belongs to one of the black hole states in

the cluster. The correct key can be hard-wired in the netlist or saved in a tamper-proof memory. The former one is less secure (but more hardware efficient) than the latter one at a certain degree. Note, more flip-flops added for new states not only increase the difficulty of hardware tampering but also incur a greater hardware cost.

7.4.3 Dynamic Transition in Black Hole Cluster

To raise the difficulty of reverse engineering, we further harden the FSM by creating dynamic transitions within the black hole state clusters. Different than the existing work [2, 10, 13], the transition among the black hole states is not static and predefined. One example of our state transition is shown in Fig. 7.9. If a wrong key sequence (KS) is used, the new state bits are preset to a non-zero vector (e.g., 4'b0001) and the original state bits remain whatever they are defined to be in the original design. After cycle 0, the newly added state bits are modified by a *Rotatefunc* algorithm, which changes the vector $B3, B2, B1, B0$ to a new non-zero vector. Each wrong key sequence will have a unique pattern defined in the *Rotatefunc* algorithm. In the example shown in Fig. 7.9, the *Rotatefunc* is a circular left-shift function. In parallel to the process of switching the black hole state bits, we propose a *Mapfunc* algorithm to deflect the original state bits from the correct state transition. Our *Mapfunc* algorithm takes the newly added state bits as an input to determine how to modify the content of the original state bits. For instance, when $B3, B2, B1, B0$ is 4'b0001, the $S3$ original state bit is flipped. Note, our method selectively inverts the state bit, rather than set the original state bits to a fixed vector. Therefore, the black hole state is not determined in the design time. It depends on which wrong key is used and the corresponding mapping statement specified in the *Mapfunc* algorithm. As shown in Fig. 7.9, the vector of 4'b0010 in the newly added state bits lead to flip $S2$ and $S0$ bits in the original state.

7.4.4 Experimental Results

7.4.4.1 Experimental Setup

To assess the capability against reverse engineering attacks, we applied three representable obfuscation approaches—DUP [2], ISO [12], and HARPOON [10] and our method to the generic gate-level netlist of ISCAS'89 benchmark circuits (http://www.pld.ttu.ee/~maksim/benchmarks/iscas89/verilog/). The obfuscated Verilog codes were synthesized in Synopsys Design Vision with a TSMC 65 nm CMOS technology. The test benches for the synthesized netlists were generated by TetraMax. Verilog simulations were conducted in Cadence NClaunch and the code coverage analyses were performed with the Cadence ICCR tool.

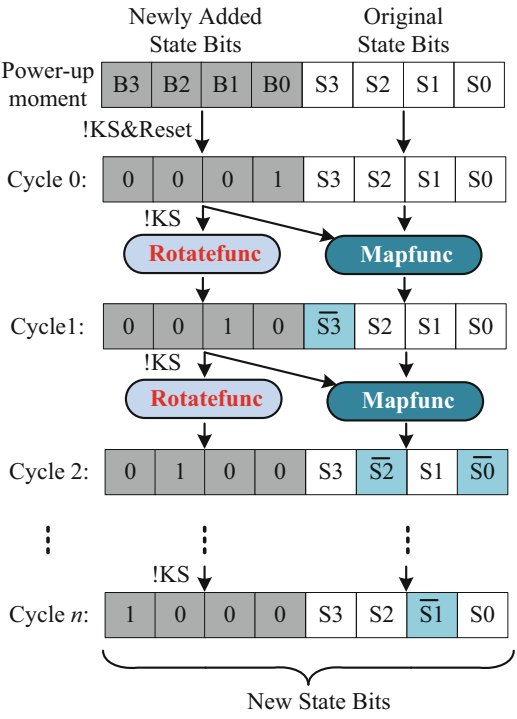

Fig. 7.9 Flow of state transition modification

7.4.4.2 Hardening Capability Against Circuit Switching Activity Analysis Attack

The goal of state obfuscation is to thwart the reverse engineer from retrieving the circuitry used for the normal operation mode. In the existing work, the researchers have not extensively discussed the hardening efficiency of different methods in the scenario that an attacker could exploit the EDA tool to recover the original circuit. In this work, we fill in this gap by studying the net toggle activities and code coverage of the circuits hardened with different obfuscation methods.

Complementary Net Toggle Activities

In the existing methods, the use of an invalid key will either cause the FSM to be stuck in an obfuscation state or an inner loop in a specified obfuscation mode. Thus, the majority of the true circuit under protection will not switch. If the attacker compares the net toggle activities for the cases of using different keys, he/she can sort out the un-toggled nets and recognize those nets are probably used for the true operational circuit. Therefore, one would design the obfuscation method in a way that does not lead to complementary net toggle activities between the scenarios of using the correct key and the wrong keys.

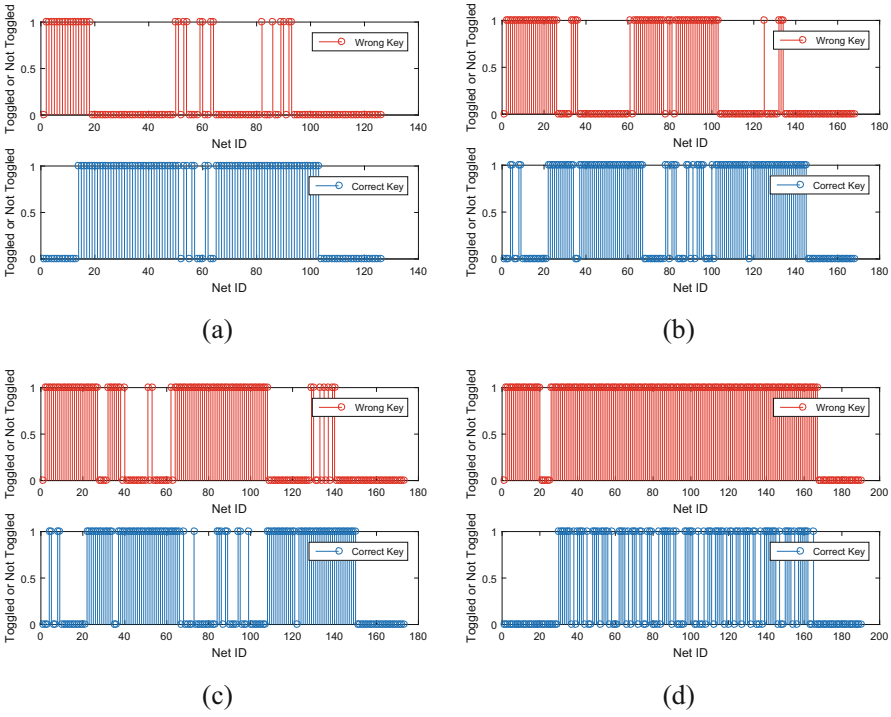

Fig. 7.10 Net toggle activities for s298 obfuscated with (**a**) DUP, (**b**) ISO, (**c**) HARPOON, and (**d**) proposed methods

Figure 7.10a shows the net toggle activities for the s298 circuit protected with the DUP obfuscation method. As can be seen, if incorrect key sequences are applied, the majority of nets from the DUP netlist do not toggle due to the blocked state transitions. In contrast, those un-toggled nets indeed switch if the correct key sequence is used. Thus, the net toggle activities for the case with/without the correct key sequence are nearly complementary. As shown in Fig. 7.10b, c, the number of toggled nets from the ISO and HARPOON netlist is more than that of the DUP netlist, due to the transitions in the obfuscation mode. However, a large number of nets have complementary toggle activities for the obfuscated netlist with a correct key sequence. The proposed obfuscation method examines the key sequence at every state transition and deflects the state to a black hole state. In our method, the FSM does not remain at a small inner loop when an incorrect key is applied to the FSM. Instead, our FSM further changes over time within the black hole state cluster, and triggers the other state registers and output ports to start switching. Consequently, our method obtains nearly the same net toggle activities for the case of with/without the correct key, as shown in Fig. 7.10d. Note, as the test bench generated by TetraMax does not reach 100 % test coverage, some logic outputs in the design module are not toggled for both incorrect and correct key cases.

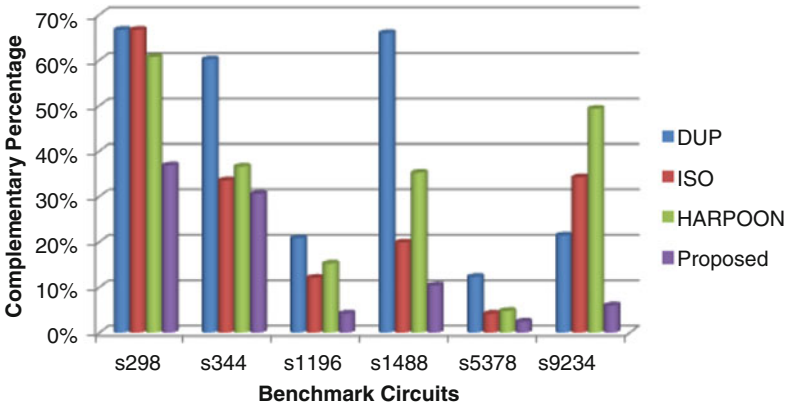

Fig. 7.11 Complementary percentage of the net toggle activities in incorrect and correct key scenarios for different benchmark circuits

To compare the obfuscation efficiency achieved by different methods, we define our comparison metric in Eq. (7.1). This expresses the *complementary degree* of the net toggle activities for the correct and incorrect keys.

$$P_{\text{complement}} = \frac{N_{\text{different toggled nets}}}{N_{\text{total nets in netlist}}} \tag{7.1}$$

A lower complementary degree means that less information will be obtained through code analysis. As shown in Fig. 7.11, the proposed method yields the lowest complementary percentage over the other methods. Based on the experiments performed on the given benchmark circuits, our method reduces the complementary percentage by up to 77 %, 85 %, and 90 % than the DUP, ISO, and HARPOON methods, respectively.

Code Coverage

We used *code coverage* as another metric to compare the hardening degree achieved by different obfuscation methods. If the obfuscated netlist with incorrect key sequences obtains a higher code coverage, this equivalently indicates that obfuscation method will yield a lower net toggling complementary percentage. Our method tightly blends the dummy states with the original states, so that the attacker cannot determine the dummy states by using code coverage analysis through tools like Cadence ICCR. As the netlist under attack is a gate-level netlist, only toggle coverage can be measured. Branch and FSM coverage are not eligible in the code coverage analysis tool. In the method [12], an incorrect key sequence will lead to the FSM enter an isolated mode eventually, thus, the FSM will freeze and the code coverage will be limited. The same reasoning applies to the method [10]. The use of black hole states will lock the chip operation. But, if the attacker has the soft IP or firm IP running in a code analysis tool, the un-toggled statements will

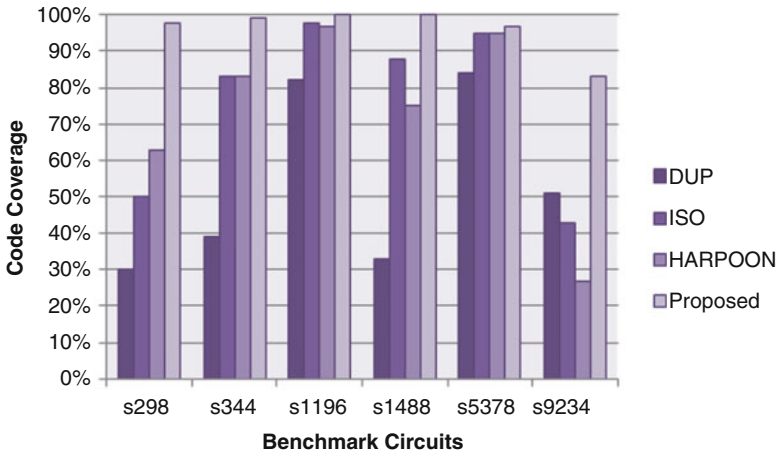

Fig. 7.12 Code coverages for the incorrect key scenarios

reveal the information relevant to the black hole states. As shown in Fig. 7.12, the proposed method always achieves the highest code coverage among the methods under comparison. As our method keeps the state transiting all the time, our method improves the code coverage by up to 56 % over the existing methods if a wrong key sequence is applied. Other methods, either stuck at the wrong states, or enter a small internal obfuscation states, thus having a lower code coverage. Code coverage proves that the proposed method reveals less information with an incorrect key over the other obfuscation methods.

7.4.4.3 Number of Unique State Register Patterns in Obfuscation Mode

In the previous work [10, 11, 13], the state registers either remain same or switch within a loop of different patterns if an incorrect key sequence is applied. Both of the scenarios will facilitate the attacker to identify the states in the obfuscation mode. Then, the attacker can add new logic to overwrite the state registers and thus skip the livelock of the obfuscation state transition. In contrast, our method utilizes *Mapfunc* function to "pseudo-randomly" change the state register values. Even though an attacker may know a *Mapfunc* function is applied in the obfuscation mode, the fact that they do not have knowledge of the exact *Mapfunc* being used will make it difficult to perform an attack where a register is overwritten to a specific value. As shown in Fig. 7.13, our method generates the largest number of unique state patterns over the other methods. On average, the proposed method produces 81, 75, and 75 more unique obfuscation state patterns over DUP, ISO, and HARPOON, respectively. This observation confirms that our method does not only obfuscate the state transitions, but is also capable of resisting the register pattern analysis and thus thwart state register overwrite attacks.

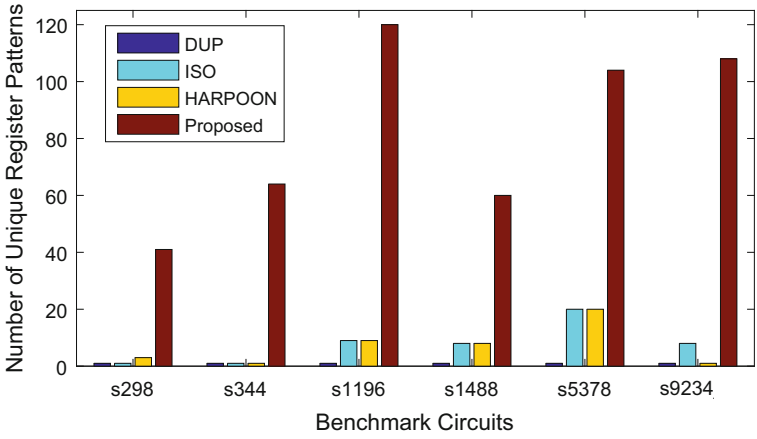

Fig. 7.13 Number of unique register patterns in different benchmark circuits

Table 7.1 Area cost in a 65 nm technology (unit: μm^2)

Benchmark	DUP	ISO	HARPOON	Proposed
s298	283.32	392.04	403.92	345.96
s344	316.08	432.72	445.32	386.64
s1196	737.28	861.12	873.72	825.12
s1488	657.72	826.56	839.16	779.36
s5378	2599.55	3282.48	3298.68	2892.6
s9234	3080.16	2826.72	2840.76	2395.8

7.4.4.4 Area and Power Overhead

The silicon area cost of the benchmark circuits with different obfuscation methods are compared in Table 7.1. The same key size of 12-bits is applied to all the methods. Due to the use of simple logic for state-deflection and dynamic black hole state transitioning, our method reduces the area overhead by up to 12 % over ISO and HARPOON. Compared to DUP, our method consumes more area but provides better IP hardening performance as discussed in Sects. 7.4.4.2 and 7.4.4.3.

The power consumption is compared in Table 7.2. We set up the clock period as 1ns for all designs. As our method checks the key sequence and deflect the FSM state at every state transition cycle, it, on average, consumes 9.5 % and 7.8 % more power than ISO and HARPOON, respectively. Again, the power of DUP is 19 % less than our method, but the DUP method loses obfuscation strength over our approach.

Table 7.2 Total power consumption in a 65 nm technology (unit: mW)

Benchmark	DUP	ISO	HARPOON	Proposed
s298	0.1135	0.1565	0.1597	0.1851
s344	0.1268	0.1704	0.1748	0.1865
s1196	0.1883	0.2341	0.2374	0.2299
s1488	0.063	0.1116	0.1140	0.1139
s5378	1.6886	1.4693	1.4736	1.7650
s9234	1.6713	1.2287	1.2384	1.3355

7.4.5 Discussion

Key-based design obfuscation methods have become attractive to resist reverse engineering and IP piracy attacks. The existing RTL obfuscation methods require the prior knowledge of all the used and unused states and lack protection on the normal operation states. To address the aforementioned limitations, we propose a dynamic state-deflection method for gate-level netlist. Our analyses on the code coverage and net toggle activities show that the proposed method reveals the least amount of information for the adversary who has access to the obfuscated netlist and conducts state register overwrite attacks. Simulation results show that our method achieves up to 56 % higher code coverage than the existing methods if an incorrect key sequences are applied. Hence our method provides a better hardening solution against the adversary who utilizes brute force attacks and EDA code analysis functions to recover the original design. Due to the dynamic deflection feature, on average, our method generates 75 more unique state register patterns than the HARPOON method if an incorrect key is applied, at the cost of 7.8 % power increase in the case study.

7.5 Obfuscation for Three-Dimensional ICs

In this section, we review how three-dimensional (3D) structure can be leveraged to address the security threats in 2D ICs and potential risks of using 3D ICs. Next, we introduce the method [22] that extends the obfuscation concept to 3D ICs.

7.5.1 Leveraging 3D for Security

3D integration has attracted a significant amount of attention during the past two decades to develop diverse computing platforms such as high performance processors, low power systems-on-chip (SoCs), and reconfigurable platforms such as FPGAs. The majority of the existing work has leveraged the unique 3D characteristics to enhance the security of 2D ICs rather than investigating hardware

security within stand-alone 3D ICs [4]. As indicated by leading 3D manufacturing foundries [3], the stacking process of 3D ICs conceals the details of the circuit design and therefore thwarts reverse engineering attacks. Secondly, 3D ICs facilitate *split manufacturing* where the entire IC is distributed throughout multiple dies/planes. Thus, due to the incompleteness of each plane, the design details of a functional block are not revealed.

Existing 3D split manufacturing approaches fall into two primary categories. In the first category, as investigated by Valamehr et al. [54, 55], the entire design is separated into two tiers: one plane is dedicated to being the primary computation plane whereas the second plane is an optional control plane that should be provided by a trusted foundry. This control plane is used to monitor possible malicious behavior within the computation plane and overwrites the malicious signals, if necessary. The second category, as studied by Imeson et al. [28], relies on the interconnects of a trusted die to obfuscate the entire 3D circuit. Thus, the circuit within the untrusted tier cannot be reverse engineered since interconnectivity is unknown. Similar studies have been performed to further enhance the obfuscation level achieved by split manufacturing [42, 57, 59]. As exemplified by these studies, existing approaches rely primarily on the presence of a trusted plane. While effective to enhance security, these existing techniques do not investigate the potential security weaknesses inherent to 3D ICs.

7.5.2 Trustworthiness of Vertical Communication

A non-trivial security weakness within 3D ICs is the trustworthiness of vertical communication. In a heterogeneous 3D stack with dies from different vendors, one of the dies can attempt to extract secret information such as an encryption key, authentication code, or certain IP characteristics. Thus, a die within a 3D stack should not only be protected from external attacks (as is the case in traditional 2D ICs), but also from attacks originating from a nearby die within the same 3D stack. Furthermore, since the authentication level of each die is different, the security of the overall 3D IC is dependent upon the weakest die. The overall 3D IC (including the die with a strong authentication level) can be compromised once an attacker succeeds in accessing the weak die. Note that due to high bandwidth inter-die vertical communication, an attacker has more access points to compromise security, producing additional security threats that do not exist in 2D ICs.

Another potential weakness is the leakage of connectivity information from an untrusted 3D integration foundry. Existing approaches that rely on split manufacturing typically assume that existing vertical communication is inherently secure. Unfortunately, this assumption is not always true. For example, the foundry that manufactures vertical interconnects may leak this connectivity information, resulting in weaker design obfuscation than what is assumed with split manufacturing.

7.5.3 *Proposed Obfuscation Method for 3D ICs*

We propose a new countermeasure for 3D ICs in [22], which is fundamentally different from the split manufacturing approach. In the existing split manufacturing methods, one of the dies and vertical connections must be manufactured by a trusted foundry. Without this assumption, split manufacturing cannot ensure the trustworthiness of a 3D SoC with dies from different vendors. It is predicted that commercial dies, rather than customized dies, will be vertically integrated to develop 3D ICs in the near future. Since the I/O definition and certain specifications of commercial dies are public, an attacker can reverse engineer the 3D SoC design, particularly if each die of the 3D IC can be separated from the stack using a debonding technique.

7.5.3.1 Overview of Proposed 3D Obfuscation Method

To eliminate the need for at least one trusted foundry for 3D ICs, we propose a secure cross-plane communication network [22]. The proposed method is based on the insertion of a *Network-on-Chip (NoC)-based shielding plane* between two commercial dies to thwart reverse engineering attacks on the vertical dimension. As shown in Fig. 7.14, the proposed NoC shielding plane (NOC-SIP) obfuscates the communication among the adjacent dies. As compared to split manufacturing, this method provides higher flexibility and scalability when developing more secure 3D ICs. This characteristic is due to the enhanced modularity and scalability of NoCs as compared to a bus-centric communication network. Ad-hoc algorithms that split IC functionalities are not suitable for a large-scale system. Furthermore, additional wire lifting through split manufacturing leads to a larger number of Through-Silicon-Vias (TSVs), resulting in a larger area overhead and TSV capacitance (and therefore power).

The essence of the proposed NOC-SIP is to provide an obfuscated communication channel between two planes that host commercial dies. Our method makes it significantly more challenging to reverse engineer the 3D IC system. If the proposed shielding layer is sufficiently strong, the 3D system has more flexibility to use low-end dies without sacrificing the overall system's security assurance. We assume each commercial die has a regular I/O pad map. Those I/O pads are connected to the proposed NOC-SIP with a regular node array, as shown in Fig. 7.14. Therefore, the specific I/O connectivity information of the dies is hidden to the 3D foundry. As a result, the proximity attack [42] is less likely to help to reveal the design details from split dies.

An example of the proposed method is depicted in Fig. 7.15. Without the proposed NOC-SIP die, node A in plane 2 would be connected to node B in plane 1 directly with a TSV. If an attacker has the ability to reverse engineer debonded planes 1 and 2, the 3D IC would be compromised. Alternatively, the NOC-SIP plane redirects the signal from a TSV-A on node A to several routing

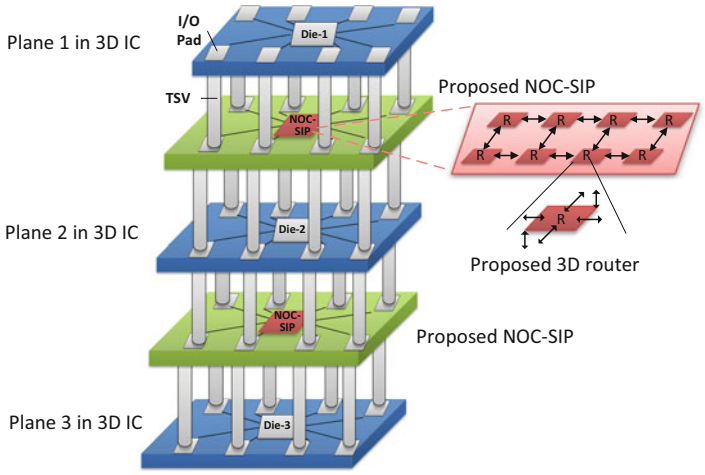

Fig. 7.14 Conceptual representation of the proposed countermeasure for 3D IC security

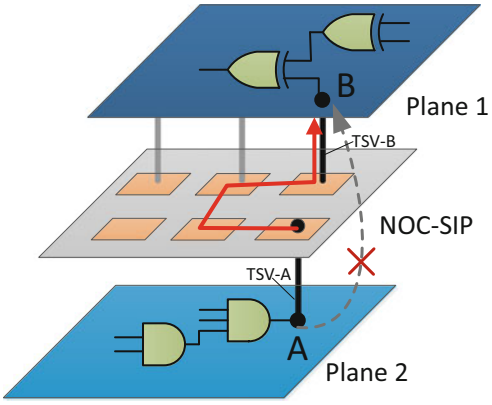

Fig. 7.15 An example of vertical communication through the proposed NOC-SIP

hops before the signal truly reaches TSV-B on node B. Thus, the direct connection from node A to node B is removed. The proposed NOC-SIP enhances security for the following three reasons: (1) even if the adversary successfully separates planes 1 and 2, a scanning electron microscope (SEM) picture of the vertical connection does not reveal useful information for reverse engineers to retrieve the complete system design, (2) the inserted NOC-SIP facilitates the use of 2D security countermeasures to address 3D security threats, and (3) the inherent scalability of the NOC-SIP overcomes the limited flexibility and scalability concerns in the existing split manufacturing algorithms.

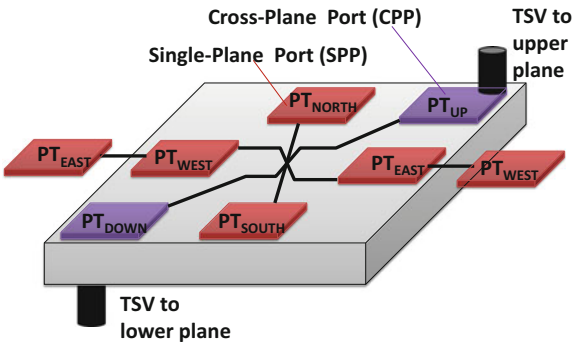

Fig. 7.16 Proposed 3D router architecture

7.5.3.2 Proposed 3D Router Design

The proposed 3D router is shown in Fig. 7.16. The single-plane ports (SPP) PT_{North}, PT_{East}, PT_{South}, and PT_{West} fulfill the communication function within the NOC-SIP. Alternatively, the two cross-plane port (CPP) PT_{UP} and PT_{DOWN} are responsible for cross-plane communication. In contrast, a typical 2D router has five pairs of input and output ports. Figure 7.17a shows the diagram of one input and one output port for a 2D router. The proposed 3D router in the NOC-SIP is different than the router in the traditional 2D NoC. As shown in Fig. 7.17b, the logic of the input and output ports in our 3D router is much simpler than that of a 2D router. Only single-depth input and output FIFOs are required; FIFO controllers are removed; the route computing unit is replaced with a simple routing path request unit (no computation inside); a simpler arbiter can fulfill the routing arbitration. Among the five ports in a 2D router, one port is connected with a network interface (NI) to reach a processing element (e.g., microprocessor or a memory core). However, in the NOC-SIP, no processing element is connected to any port of the 3D router. The packetization function that is executed in the NI of a 2D NoC is part of our vertical router's function.

In our 3D router, the cross-plane ports, PT_{UP} and PT_{DOWN}, are designed for vertical connection from/to other planes. The primary functions of these two ports include: (1) packetize/de-packetize the bit stream from/to the other plane, (2) assign a source-routing path for each data packet, and (3) buffer the bit stream if the previous packets do not have available bandwidth.

As shown in Fig. 7.17c, k-bit data from plane 1 is stored in the buffer before packetization. The packet leaving the cross-plane port is formatted in a way that starts with header information, follows with the detailed routing hops, and ends with the real data. The uniqueness of this cross-plane port is the existence of a router identifier (that is programmable after fabrication) and source routing table, which are provided by the 3D chip designer through one-time read-only memory (ROM) programming equipment. As the router identifier and routing table are initialized after fabrication, placing malicious hardware in the NOC-SIP during the design stage cannot guarantee compromised system. Similarly, the knowledge of the 3D

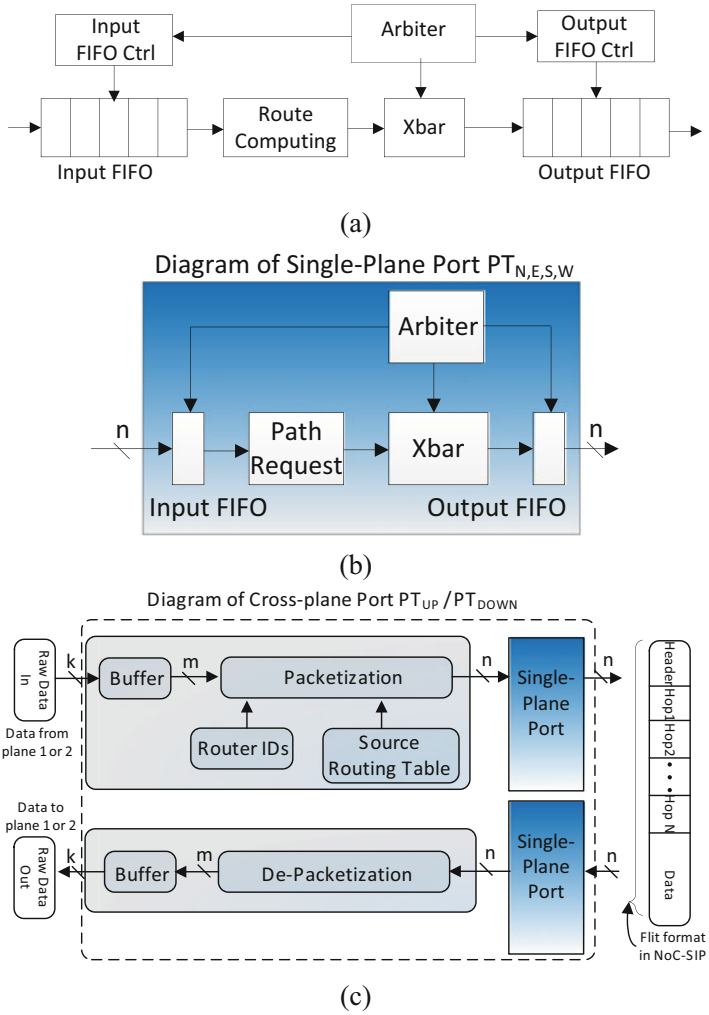

Fig. 7.17 Schematics of (**a**) one pair of input and output ports in a 2D router, (**b**) single-plane port in the proposed 3D router, and (**c**) cross-plane port in the proposed 3D router

router design does not help the reverse engineer, as the router identifier and source routing table are unknown before deployment. Once the router ID or the source routing table is changed, the secret information sniffed by the attack for one case would not be useful.

Source routing introduces unpredictability for the adversary who intends to insert hardware Trojans in the NOC-SIP. As there is no computation unit for route path preparation, the attacker cannot successfully execute a meaningful attack without the knowledge of router identifier assignment. Another advantage of source routing is to manually balance the latency for the transmission between different pairs of

I/O pads on planes 1 and 2, shown in Fig. 7.14. To thwart the delay-based side-channel attacks, the source routing design in the 3D router further facilitates a dynamic routing, which varies the latency of the communication between source and destination ports within the NOC-SIP.

7.5.3.3 Assessment on 3D Obfuscation Method

We implemented the proposed NOC-SIP in Verilog HDL and synthesized the HDL code in Synopsys Design Compiler with a 65 nm TSMC technology. The width of the raw data from a plane (other than NOC-SIP) is set to 5 bits, and the packet width for the NOC-SIP is 32 bits. The input and output FIFOs are 32-bit single-depth buffer. XY routing algorithm is applied to the 2D NoC design. Round-robin arbitration is used in both 2D NoC and NOC-SIP. The NI for the 2D mesh NoC is OCP-IP compatible [26]. The hardware cost of our NOC-SIP is compared with two typical 4×4 mesh NoCs.

The router switching activity of the NOC-SIP is used as a metric to evaluate the difficulty of identifying vertical connectivity through reverse engineering. Three cross-plane communication methods are compared: direct TSV, NOC-SIP with XY routing, and NOC-SIP with source routing. Direct TSV refers to the case where the I/O pads from different dies are statically connected during the 3D IC integration process. NOC-SIP with XY and source routing stand for the proposed shielding layer using XY packet routing and dynamic source routing, respectively.

First, we randomly select a single pair of I/O pads from two planes for vertical communication. The number of switching transitions of each 3D router is recorded in 5000 clock cycles. As shown in Fig. 7.18a, the activity of the TSVs directly connected to two I/O pads indicates that the TSV nodes 4 and 9 are used in the cross-plane communication. Note that the color bar represents the number of node transitions in 5000 cycles. Alternatively, the router activity map (Fig. 7.18b) of the NOC-SIP with XY routing shows that the routers 4, 8, 9, 10, 11, and 12 are used in the cross-plane communication. A reverse engineer, however, cannot know which two TSVs in the routers are actually used for cross-plane communication. The use of source routing further increases the number of involved routers to 10, thereby increasing the degree of obfuscation, as shown in Fig. 7.18c.

If two pairs of vertical communication are applied, the NOC-SIP achieves enhanced obfuscation performance. As shown in Fig. 7.19, the number of active routers in the NOC-SIP is always greater than the direct TSV method. The source routing for NOC-SIP has the ability to distribute the data transmission throughout the entire NOC-SIP. Thus, the proposed approach makes reverse engineering significantly more difficult.

The experiments shown by Figs. 7.18 and 7.19 are extended by increasing the number of communication pad pairs to 10. As shown in Fig. 7.20, the NOC-SIP with source routing engages 12 routers with two pairs of pads. In contrast, the direct TSV method involves 12 routers with 10 pairs of pads. This behavior demonstrates that the proposed NOC-SIP method effectively obfuscates the cross-

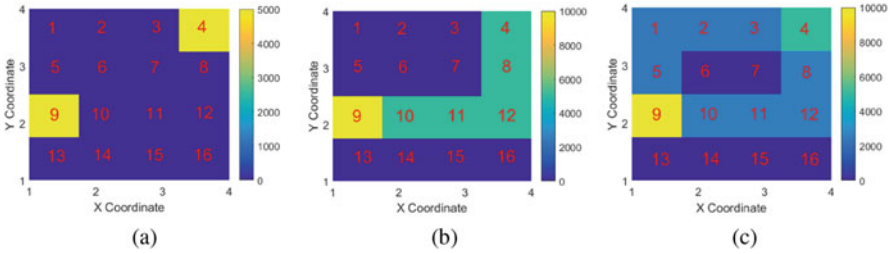

Fig. 7.18 Single pad-to-pad vertical communication. (**a**) Direct TSV, (**b**) NOC-SIP with XY routing, and (**c**) NOC-SIP with source routing

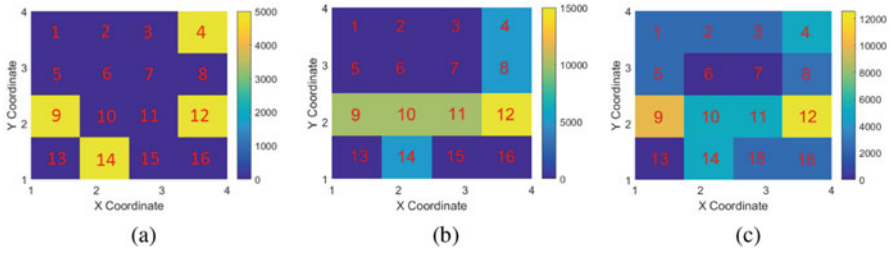

Fig. 7.19 Double pad-to-pad vertical communication. (**a**) Direct TSV, (**b**) NOC-SIP with XY routing, and (**c**) NOC-SIP with source routing

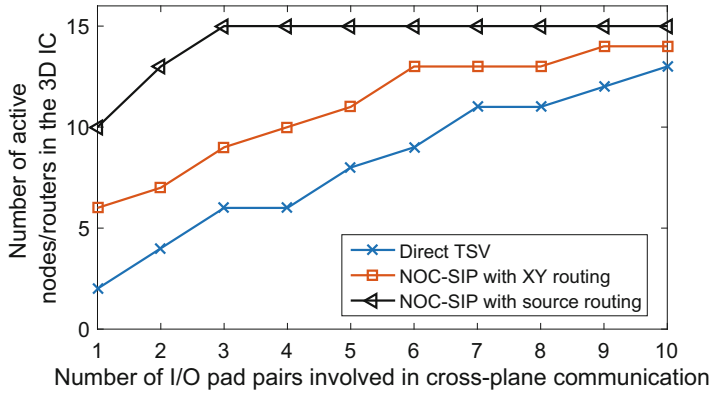

Fig. 7.20 The number of active routers versus the number of routers having incoming data

plane communication by increasing the reverse engineering time as compared to using direct TSV connections. The direct TSV is the most vulnerable method against reverse engineering on vertical connections.

7.5.4 Discussion

Existing works have demonstrated that 3D integration provides additional security defense to thwart hardware attacks in 2D ICs. The security threats that are inherent to true 3D ICs, however, have not received much attention. To thwart reverse engineering attacks on the cross-plane communication (i.e., vertical communication channel), an NOC-SIP layer is proposed. The proposed NOC-SIP obfuscates the vertical communication within a 3D stack. Simulation results demonstrate that the proposed NOC-SIP engages all of the routers as long as more than two pairs of I/O pads from different planes are in use. Alternatively, direct TSV connection method requires ten pairs of I/O pads from two planes to reach the same level of obfuscation as the proposed method. Thus, the proposed NOC-SIP makes it significantly more challenging for a reverse engineer to retrieve the vertical connectivity information.

7.6 Summary

Due to the trend of outsourcing designs to overseas foundries, IC designers and users need to re-evaluate the trust in hardware. The hardware threats such as reverse engineering, IP piracy, chip overproduction, and hardware Trojan insertion are major concerns for hardware security. Key-based obfuscation is one of the promising countermeasures to thwart the hardware attacks like reverse engineering and IP piracy. This chapter summarizes the state-of-the-art hardware hardening methods at layout, gate, and register transfer levels. Camouflaging, logic encryption, and state obfuscation are three representable categories for hardware hardening against malicious reverse engineering and IP piracy attacks. To address the potential security threats in 3D ICs, we also introduce an obfuscation method to protect the vertical communication channels in 3D ICs. As the attackers may obtain extra knowledge and tools to perform reverse engineering and IP piracy attacks, it is imperative to explore new hardware-efficient hardening methods to combat the existing hardware attacks.

References

1. S. Adee, The hunt for the kill switch. IEEE Spectr. **45**, 34–39 (2008)
2. Y. Alkabani, F. Koushanfar, M. Potkonjak, Remote activation of ICs for piracy prevention and digital right management, in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2007), pp. 674–677
3. S. Bansal, 3D IC Design. EETimes (2011). http://www.eetimes.com/document.asp?doc_id=1279081
4. C. Bao, A. Srivastava, 3D Integration: new opportunities in defense against cache-timing side-channel attacks, in *Proceedings of Computer Design (ICCD)* (2015), pp. 273–280

5. A. Baumgarten, A. Tyagi, J. Zambreno, Preventing IC piracy using reconfigurable logic barriers. IEEE Des. Test Comput. **27**(1), 66–75 (2010)
6. D.J. Bernstein, Cache-timing attacks on AES. Technical Report (2005)
7. S. Bhunia, M.S. Hsiao, M. Banga, S. Narasimhan, Hardware trojan attacks: threat analysis and countermeasures. Proc. IEEE **102**(8), 1229–1247 (2014)
8. E. Castillo, U. Meyer-Baese, A. Garcia, L. Parrilla, A. Lloris, IPP@HDL: efficient intellectual property protection scheme for IP cores. IEEE Trans. Very Large Scale Integr. VLSI Syst. **15**(5), 578–591 (2007)
9. R.S. Chakraborty, S. Bhunia, Hardware protection and authentication through netlist level obfuscation, in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2008), pp. 674–677
10. R. Chakraborty, S. Bhunia, HARPOON: an obfuscation-based soc design methodology for hardware protection. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **28**(10), 1493–1502 (2009)
11. R.S. Chakraborty, S. Bhunia, Security through obscurity: an approach for protecting register transfer level hardware IP, in *Proceedings of Hardware Oriented Security and Trust (HOST)* (2009), pp. 96–99
12. R. Chakraborty, S. Bhunia, RTL Hardware IP protection using key-based control and data flow obfuscation, in *Proceedings of International Conference on VLSI Design (VLSID)* (2010), pp. 405–410
13. R.S. Chakraborty, S. Bhunia, Security against hardware trojan attacks using key-based design obfuscation. J. Electron. Test. **27**(6), 767–785 (2011)
14. R.S. Chakraborty, S. Narasimhan, S. Bhunia, Hardware trojan: threats and emerging solutions, in *Proceedings of High Level Design Validation and Test Workshop (HLDVT'09)* (2009), pp. 166–171
15. Chipworks (2012). http://www.chipworks.com/en/technical-competitive-analysis/
16. Circuit camouflage technology (2012). http://www.smi.tv/SMI_SypherMedia_Library_Intro.pdf
17. R. Cocchi, J. Baukus, B. Wang, L. Chow, P. Ouyang, Building block for a secure CMOS logic cell library. US Patent App. 12/786,205 (2010)
18. R. Cocchi, L. Chow, J. Baukus, B. Wang, Method and apparatus for camouflaging a standard cell based integrated circuit with micro circuits and post processing. US Patent 8,510,700 (2013)
19. R. Cocchi, J.P. Baukus, L.W. Chow, B.J. Wang, Circuit camouflage integration for hardware IP protection, in *Proceedings of Design Automation Conference (DAC)* (2014), pp. 1–5
20. A.R. Desai, M.S. Hsiao, C. Wang, L. Nazhandali, S. Hall, Interlocking obfuscation for anti-tamper hardware, in *Proceedings of Cyber Security and Information Intelligence Research Workshop (CSIIRW)* (2013), pp. 8:1–8:4
21. J. Dofe, Y. Zhang, Q. Yu, DSD: a dynamic state-deflection method for gate-level netlist obfuscation, in *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (2016), pp. 565–570
22. J. Dofe, Q. Yu, H. Wang, E. Salman, Hardware security threats and potential countermeasures in emerging 3D ICS, in *Proceedings of Great Lakes Symposium on VLSI (GLSVLSI)* (ACM, New York, 2016), pp. 69–74
23. S. Dupuis, P.S. Ba, G.D. Natale, M.L. Flottes, B. Rouzeyre, A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans, in *Proceedings of International On-Line Testing Symposium (IOLTS)* (2014), pp. 49–54
24. ExtremeTech, iphone 5 A6 SoC reverse engineered, reveals rare hand-made custom CPU, and tri-core GPU (2012)
25. Federal statutory protection for mask works (1996). http://www.copyright.gov/circs/circ100.pdf
26. J. Frey, Q. Yu, Exploiting state obfuscation to detect hardware trojans in NoC network interfaces, in *Proceedings of Midwest Symposium on Circuits and Systems (MWSCAS)* (2015), pp. 1–4

27. U. Guin, K. Huang, D. DiMase, J.M. Carulli, M. Tehranipoor, Y. Makris, Counterfeit integrated circuits: a rising threat in the global semiconductor supply chain. Proc. IEEE **102**(8), 1207–1228 (2014)
28. F. Imeson, A. Emtenan, S. Garg, M.V. Tripunitara, Securing computer hardware using 3D integrated circuit (ic) technology and split manufacturing for obfuscation. USENIX Security, 13 (2013)
29. Intel's 22-nm tri-gate transistors exposed (2012). http://www.chipworks.com/blog/technologyblog/2012/04/23/intels-22-nm-tri-gate-transistors-exposed
30. P.C. Kocher, Timing attacks on implementations of diffie-Hellman, RSA, DSS, and Other Systems, in *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*. CRYPTO '96 (Springer, London, 1996), pp. 104–113. [Online]. Available: http://dl.acm.org/citation.cfm?id=646761.706156
31. P. Kocher, J. Jaffe, B. Jun, Differential power analysis, in *Advances in Cryptology—CRYPTO'99* (Springer, Berlin, 1999), pp. 388–397
32. F. Koushanfar, Provably secure active IC metering techniques for piracy avoidance and digital rights management. IEEE Trans. Inf. Forensics Secur. **7**(1), 51–63 (2012)
33. L. Frontier Economics Ltd, Estimating the global economic and social impacts of counterfeiting and piracy (2011)
34. H.K. Lee, D.S. Ha, HOPE: an efficient parallel fault simulator for synchronous sequential circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **15**(9), 1048–1058 (1996)
35. Y.W. Lee, N.A. Touba, Improving logic obfuscation via logic cone analysis, in *Proceedings of Latin-American Test Symposium (LATS)* (2015), pp. 1–6
36. B. Liu, B. Wang, Reconfiguration-based VLSI design for security. IEEE J. Emerging Sel. Top. Circuits Syst. **5**(1), 98–108 (2015)
37. T. Meade, S. Zhang, Y. Jin, Netlist reverse engineering for high-level functionality reconstruction, in *Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC)* (2016), pp. 655–660
38. A. Moradi, M.T.M. Shalmani, M. Salmasizadeh, A generalized method of differential fault attack against AES cryptosystem, *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems (CHES)* (Springer, Berlin, Heidelberg, 2006), pp. 91–100
39. S.M. Plaza, I.L. Markov, Solving the third-shift problem in ic piracy with test-aware logic locking. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **34**(6), 961–971 (2015)
40. J. Rajendran, Y. Pino, O. Sinanoglu, R. Karri, Logic encryption: a fault analysis perspective, in *Proceedings of Design, Automation and Test in Europe (DATE)*. EDA Consortium (2012), pp. 953–958
41. J. Rajendran, Y. Pino, O. Sinanoglu, R. Karri, Security analysis of logic obfuscation, in *Proceedings of Design Automation Conference (DAC), ACM/EDAC/IEEE* (2012), pp. 83–89
42. J. Rajendran, O. Sinanoglu, R. Karri, Is manufacturing secure? in *Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)* (2013), pp. 1259–1264
43. J. Rajendran, M. Sam, O. Sinanoglu, R. Karri, Security analysis of integrated circuit camouflaging, in *Proceedings of ACM SIGSAC Conference on Computer Communications Security*, CCS '13 (ACM, New York, 2013), pp. 709–720
44. J. Rajendran, O. Sinanoglu, R. Karri, VLSI testing based security metric for IC camouflaging, in *Proceedings of IEEE International Test Conference (ITC)* (2013), pp. 1–4
45. J. Rajendran, A. Ali, O. Sinanoglu, R. Karri, Belling the CAD: toward security-centric electronic system design. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **34**(11), 1756–1769 (2015)
46. J. Rajendran, H. Zhang, C. Zhang, G.S. Rose, Y. Pino, O. Sinanoglu, R. Karri, Fault analysis-based logic encryption. IEEE Trans. Comput. **64**(2), 410–424 (2015)
47. M. Rostami, F. Koushanfar, R. Karri, A primer on hardware security: models, methods, and metrics. Proc. IEEE **102**(8), 1283–1295 (2014)
48. J. Roy, F. Koushanfar, I. Markov, EPIC: ending piracy of integrated circuits, in *Proceedings of Design, Automation and Test in Europe (DATE)* (2008), pp. 1069–1074

49. B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, 2nd edn. (Wiley, New York, 1995)
50. SEMI, Innovation is at risk as semiconductor equipment and materials industry loses up to $4 billion annually due to ip infringement (2008). www.semi.org/en/Press/P043775/
51. O. Sinanoglu, Y. Pino, J. Rajendran, R. Karri, Systems, processes and computer-accessible medium for providing logic encryption utilizing fault analysis. US Patent App. 13/735,642 (2014)
52. P. Subramanyan, S. Ray, S. Malik, Evaluating the security of logic encryption algorithms, in *Proceedings of Hardware Oriented Security and Trust (HOST)* (2015), pp. 137–143
53. R. Torrance, D. James, The state-of-the-art in IC reverse engineering, in *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems* (CHES) (Springer, Berlin, Heidelberg, 2009), pp. 363–381
54. J. Valamehr et al., A qualitative security analysis of a new class of 3-D integrated crypto co-processors. in *Cryptography and Security*, ed. by D. Naccache (Springer, Berlin, Heidelberg, 2012), pp. 364–382
55. J. Valamehr et al., A 3-D split manufacturing approach to trustworthy system development. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **32**(4), 611–615 (2013)
56. J.B. Wendt, M. Potkonjak, Hardware obfuscation using puf-based logic, in *2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (2014), pp. 270–271
57. K. Xiao, D. Forte, M.M. Tehranipoor, Efficient and secure split manufacturing via obfuscated built-in self-authentication, in *Proceedings of Hardware Oriented Security and Trust (HOST)* (2015), pp. 14–19
58. Y. Xie, A. Srivastava, Mitigating sat attack on logic locking. Cryptology ePrint Archive, Report 2016/590 (2016). http://eprint.iacr.org/
59. Y. Xie, C. Bao, A. Srivastava, Security-aware design flow for 2.5 D IC technology, in *Proceedings of the 5th International Workshop on Trustworthy Embedded Devices(TrustED)*. (ACM, New York, 2015), pp. 31–38
60. M. Yasin, O. Sinanoglu, Transforming between logic locking and IC camouflaging, in *Proceedings of International Design Test Symposium (IDT)* (2015), pp. 1–4
61. M. Yasin, J. Rajendranand, O. Sinanoglu, R. Karri, On improving the security of logic locking. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **35**(9), 1411–1424 (2015)
62. M. Yasin, B. Mazumdar, J. Rajendranand, O. Sinanoglu, SARlock: SAT attack resistant logic locking, in *Proceedings of Hardware Oriented Security and Trust (HOST)* (2016), pp. 236–241
63. J. Zhang, H. Yu, Q. Xu, HTOutlier: hardware trojan detection with side-channel signature outlier identification, in *Proceedings of Hardware Oriented Security and Trust (HOST)* (2012), pp. 55–58