Chapter 5
Permutation-Based Obfuscation

Zimu Guo, Mark M. Tehranipoor and Domenic Forte

5.1 Introduction

As discussed in previous chapters, hardware obfuscation techniques can be categorized into chip level [1] and board level [2] according to the platform where these techniques are implemented. Chip-level obfuscation is performed on integrated circuits (ICs), while board-level obfuscation is performed on printed circuit boards (PCBs). The chip-level obfuscation techniques can be further classified into register-transfer (RT) level and gate level as per the design abstraction level. Several approaches can be exploited to accomplish design obfuscation. Based on the mechanism of these approaches, they can be classified as finite-state machine (FSM) obfuscation, logic encryption, and logic permutation. Figure 5.1 presents the relationship between these approaches and the design levels where they are applied. No single obfuscation approach can fit all the design levels. Additionally, some of them can be easily broken when implemented on a particular design level.

An obfuscation-protected system often consists of two operation modes: **functional mode** and **obfuscated mode** [3]. These two modes are controlled by one or more keys/configurations. The functional mode indicates that the system performs the designed functionalities, while obfuscated mode implies that the system presents no meaningful behavior.

Z. Guo (✉) · M.M. Tehranipoor · D. Forte
University of Florida, Gainesville, FL, USA
e-mail: zimuguo@ufl.edu

M.M. Tehranipoor
e-mail: tehranipoor@ece.ufl.edu

D. Forte
e-mail: dforte@ece.ufl.edu

D. Forte et al. (eds.), *Hardware Protection through Obfuscation*,
DOI 10.1007/978-3-319-49019-9_5

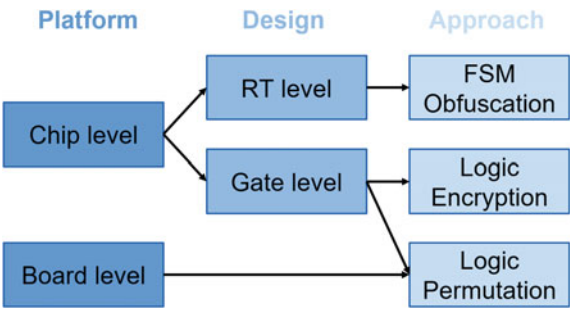

Fig. 5.1 Hardware obfuscation classification

During the rest of this section, a brief introduction about each of the obfuscation approaches is provided. Section 5.1.1 talks about chip-level obfuscation techniques, and Sect. 5.1.2 deals with board-level obfuscation. The rest of the chapter focuses on chip and board-level logic permutation.

5.1.1 Chip Level

For chip-level application, the following three obfuscation approaches can be applied.

- FSM obfuscation
- Logic encryption
- Logic permutation

As stated in Fig. 5.1, the **FSM obfuscation** can only be implemented by modifying the RTL design [4]. A general FSM obfuscation approach involves adding extra states into the original state transition graph (STG) of the design. These inserted states prevent the system from entering the functional mode without a correct key/configuration [4]. This key (which can either be fixed or generated based on the chip's ID) transforms the chip from the obfuscated mode to the functional mode. Some designs also contain another FSM called a black-hole FSM which permanently locks the chip if the applied key is incorrect [5]. The aforementioned FSM-based approach cannot be applied on gate-level nor board-level designs since neither of them provides the states which can be modified.

The **logic encryption** approach inserts locking blocks into the gate-level design [6]. These blocks can be as simple as a set of XOR gates which mask the internal signals with the keys or configuration bits. Since these encrypting units block the internal data/signal paths of the original design, they are also named as logic barriers [6]. These logic barriers can only be inserted in gate-level and board-level designs since there is no data/signal path abstraction at RTL level. However, logic encryption is much easier to be broken on board-level designs. A short answer for this is that it is much simpler to remove the encrypting blocks from board-level designs than chip-level ones. More detailed explanations will be given in Sect. 5.1.2.

The **logic permutation** approach permutes the data/signal paths instead of encrypting them. This method can be utilized to accomplish the obfuscation goal at both the gate level [7] and the board level [2]. By adding a *permutation block*, the correct orders of internal connections are concealed. Similar to other obfuscation approaches, a key/configuration is assigned to drive the system to the functional mode.

5.1.2 Board Level

Significantly, different constraints can be observed between chip and board-level obfuscation. These differences include the design modification opportunities, attack challenges, and design dimension differences. Limited by these differences, the number of obfuscation techniques which can be applied on PCBs is less than on ICs. The design modification feasibility indicates what levels of design can be manipulated to achieve obfuscation. For instance, the RTL and gate-level design can be obfuscated for a chip. For a board-level design, no RTL definition can be found. However, the chip-level intergate connections can be extended to board-level interchip connections. As a result, the gate-level obfuscation approaches, such as logic encryption and logic permutation, can also be applied at the board level. However, these obfuscation methods may be easily broken when they are applied directly on the board level under low-cost attacks. The designers should also be aware of the dimension difference between chips and boards. The onboard connections (e.g., between chips) are more straightforward to be discovered than the chip-level ones. The differences mentioned above enable the attackers to identify, bypass, or remove inserted obfuscation components from the board more quickly. An example is provided as follows to clarify how certain chip-level obfuscation techniques fail on the board level.

A simple bypass attack on board-level logic encryption is provided as follows. Shown in Fig. 5.2a, as logic barriers, XOR and XNOR gates are inserted in the middle of the in-chip paths [6]. These barriers are embedded within a fabricated chip and can only be identified and bypassed by applying costly techniques such as IC reverse engineering [8]. For board-level logic encryption, these logic barriers are implemented on a dedicated chip. The same attack on the boards can be simply accomplished by bypassing the logic barriers with jumper wires. As presented in Fig. 5.2b, the attacker only needs to connect corresponding inputs and outputs of the logic barrier chip. These jumper wires are presented in red dashed line in Fig. 5.2b. Even though these connections are not public, the attackers can always find the correct input/output pairs by matching waveforms. This matching process is named as the probing attack and will be elaborated on in Sect. 5.8.

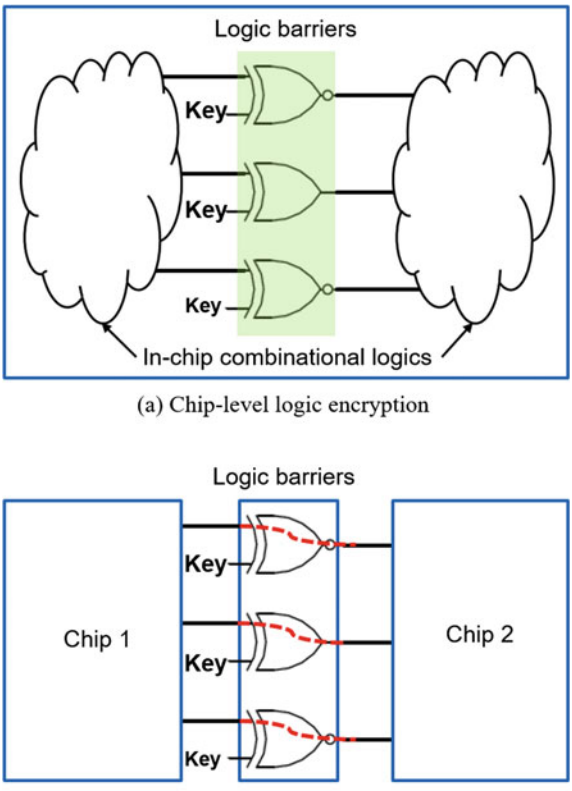

Fig. 5.2 Comparison of logic encryption on chip level and board level

5.1.3 Chapter Organization

In this chapter, the detailed framework for implementing logic permutation on both chip level and board level is provided. Section 5.2 presents the high-level permutation-based obfuscation framework. The goals of the attacker, as well as the designer, are discussed. In Sect. 5.3, the major differences between chip-level and board-level designs are presented. These differences should be carefully considered when obfuscating the design on different levels. Section 5.4 performs the analyses on implementing the permutation. These analyses help in determining which paths are good permutation candidates and why. This section provides a guideline for the designer to achieve the best obfuscation performance. Section 5.5 introduces the permutation networks which could be used for obfuscation. These permutation networks are studied by their capabilities and area utilizations. Capability indicates whether a permutation network can achieve the full permutation or not. Besides these analyses, two configuration scenarios of a popularly used permutation network are presented. According to these scenarios, the attacker's goal is reanalyzed. Section 5.6 discusses

the ways to generate and store the key/configurations used for obfuscation. The advantages and drawbacks of each key generation/storage approach are provided.

Section 5.7 provides a comprehensive flow for evaluating the performance of an obfuscated system against various attacks. This performance indicates how difficult an attacker can break the obfuscation. In Sect. 5.8, potential attacks and their corresponding countermeasures are provided. Both the permutation network and its key/configurations can be attacked. These attacks are discussed at the chip as well as board level. The attacks are organized into three classes based on the required information and resources to carry them out. Next, the countermeasures are grouped into three levels of security requirement. The coverage of the countermeasures is also discussed. Finally, the conclusions are summarized in Sect. 5.9.

5.2 Permutation-Based Obfuscation Overview

A general idea of hardware obfuscation is shown in Fig. 5.3. The solid-line rectangles imply the components which belong to the original design. For the chip level, these components can be either logic gates or registers. For the board level, these components can be chips on the board. In the original design, these components are connected directly by fixed wires or traces. Since all the intercomponent connections are established in the original designs, these products are also functional after being sold in the market. An attacker with the chip-level or board-level reverse engineering capabilities can obtain the design. The question mark in Fig. 5.3 denotes the obscurity introduced by the obfuscation which prevents simple reverse engineering. This mystery can be in the form of interconnection permutation, modification of signal values, or a combination of the two. In hardware obfuscation, typically a secret key is used to remove the obscurity and allow the design to behave as originally intended. In this chapter, we consider that the obscurity is provided by a permutation block. This permutation block consists of a key-controlled permutation network and permutes the selected intercomponent connections.

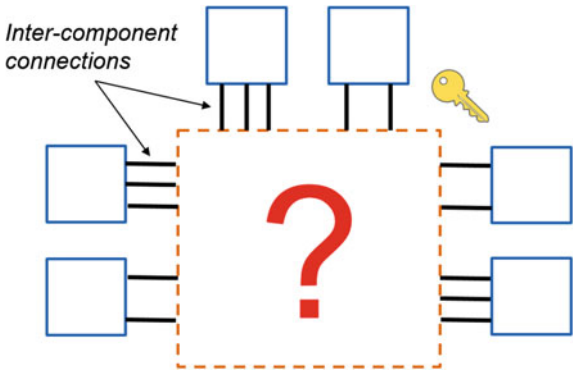

Fig. 5.3 General obfuscation scheme

Breaking the permutation-based obfuscation can be achieved by discovering the mystery above. The most straightforward way to accomplish this task is through brute force attacks. Two possible brute force objectives may be realized: **(i)** retrieving the original connections and **(ii)** key entries to the obfuscation chip. In the former, an attacker tests all the possible intercomponent connections to identify the one that results in the correct operation. This attack is usually achieved on board level since it is simple to examine the obfuscated connections by physically connecting them. On the other hand, the same procedure is nearly impossible to be accomplished at the chip level. The reason is that connecting the traces within the chip requires chip-level reverse engineering. Since such reverse engineering is usually destructive, the chip under test would no longer be functional. Compared with the former brute force objective, the latter one can be applied on both chip level and board level. Achieving this attacking goal, an attacker tests all possible key inputs through the permutation network's interface until the system functions correctly. Note that this objective differs from the first one because it depends entirely on the implementation of the permutation network (i.e., the relationship between keys and input/output combinations).

The term *breaking probability* denotes the likelihood of discovering the mystery through these two brute force methods.We use P_{com} to denote the breaking probability for examining the input/output combinations of the permutation network, and P_{key} denotes the breaking probability for reviewing the key entries. These breaking probabilities are used to evaluate the performance/strength of the permutation-based obfuscation. Besides the brute force attack, the feasibility analyses of other potential attacks are discussed in Sect. 5.8.

5.3 Obfuscation Considerations

Since significant design differences exist between ICs and PCBs, the obfuscation implemented on these platforms should be considered separately. In this section, these considerations are classified into two categories: **obfuscation-induced overheads** and the **cost of obtaining the design**.

The obfuscation-induced overheads indicate the amount of area, power, and cost overheads introduced by obfuscating the original design. Higher overheads make the obfuscation less practical. For the board-level obfuscation, the permutation block is usually implemented within an additional chip. The reason for adding another chip is that the functionalities of the chips in the original design are dedicated and cannot be used to permute the interconnections. However, this extra component induces overhead to the original design. Compared with obfuscating the board, accomplishing the same task within a chip presents fewer overheads. The area overhead is negligible when inserting hundreds of gates into millions of gates. For chip-level obfuscation, the obfuscation procedure should be achieved during the IC design phase. The insertion of the permutation block may require redesigning the entire IC.

Learning the layout or schematic of the obfuscated design benefits the attacker in breaking the obfuscation. Since this design information is usually not public, the attackers would exploit techniques such as hardware probing [9] and reverse engineering [10] to realize this goal. To learn the internal structure of an IC, an attacker can apply destructive [10] or nondestructive [11] reverse engineering. Destructive reverse engineering requires delayering of the chip and capturing high-resolution images for each layer. With these images, the layout of the chip can be recovered. However, reverse engineering a PCB is much easier. PCBs can be destructively reverse engineered by a similar process. However, automated nondestructive technique based on X-ray can be used to extract the PCB internal structure without delayering [11]. Many online resources provide low-cost PCB reverse engineering services. Besides reverse engineering, hardware probing aims to monitor the signals on a running board or chip. Similar to reverse engineering, probing the board is more straightforward than the chip. The interchip connections of a board are accessible simply by probing the traces exposed on the surface [12]. These low-cost board-level reverse engineering and hardware probing make it harder to hide secrets in a board than a chip.

5.4 Design Modification

As presented in Fig. 5.3, a mystery (i.e., permutation block) hides the actual intercomponent connections from the attackers. The designer needs to modify the design by inserting the permutation block and selecting the intercomponent connections to be obfuscated. For the board level, this permutation block can be implemented by a complex programmable logic device (CPLD), field-programmable gate array (FPGA), or application-specific integrated circuit (ASIC). For the chip level, this permutation block is inserted by modifying the gate-level design.

The intercomponent connections used for obfuscation should be carefully selected. A smart designer would not involve all the intercomponent connections into the permutation-based obfuscation. Even though performing this selection strategy increases the attack difficulty, it may cause significant timing, area, and power overheads on both chip and board levels. Further, many onboard chips may be commonly used parts, whose connections can be easily guessed by an attacker (e.g., clock signals and analog signal pins). This further constrains the type of interconnections that may be used for permutation-based obfuscation. Moreover, not all the types of signals can by permuted by the permutation block (e.g., the CPLD and FPGA can only take digital signals as input). In summary, three general requirements need to be met before involving an intercomponent connection into the obfuscation. These requirements are as follows:

(i) The obfuscated connection should not be easily guessed, matched, or bypassed;
(ii) The obfuscated connection should not be on a timing-critical path; and
(iii) The type of connection (digital or analog) should be manipulated by the permutation block.

Requirement (ii) should be considered when implementing both the chip-level and board-level obfuscations, while requirements (i) and (iii) are considered only during the board-level obfuscation. In this section, these design modification requirements are discussed on board level and chip level, respectively. Besides these three general rules, other more detailed design frameworks have been provided by researchers [2, 7, 13] to either minimize the overheads or improve the obfuscation performance.

5.4.1 Board Level

When implementing the board-level permutation-based obfuscation, all three requirements should be considered. First of all, it is crucial to obfuscate the connections which do not perform dedicated functionalities. The term functionality indicates what these ports can be utilized as (e.g., analog-to-digital converter and serial peripheral interface bus). This information is usually public and can be accessed by anyone including the attackers (e.g., by publicly available data sheets of the ICs).

As an example, a reference design from NXP is shown in Fig. 5.4. The reference design is based on Freescale ColdFire V1 MCF51MM256CLL microcontroller unit (MCU). This instrument is a noninvasive acquisition system incorporating a pedometer, ECG, food intake table, data storage, wireless communication, timer, and a chronometer. Biometric data can be saved and stored in the integrated Micro SD Card Reader. A touch-sensing interface allows the user to have control of the device through a capacitive touch-sensing film. As shown in Fig. 5.4, the digital accelerometer chip communicates with the MCU via interintegrated circuit (I^2C) bus. This chip only consists of 10 pins, and most of them are either reserved or required to be attached to the power/ground according to the datasheet. Thus, these reserved pins cannot be obfuscated. If the rest of the chip's pins are obfuscated, the attackers need to figure out the corresponding ports from MCU to break the obfuscation. Achieving this task, they may examine the MCU's ports which provide the I^2C function. According to the datasheet of this MCU, only two sets of port preform the I^2C function. As a result, the attacker could partially figure out the obfuscated connections. The breaking probability will be significantly increased if these I^2C pins are involved in the obfuscation.

The scenario above can be found in most of the modern embedded systems. Besides I^2C, the serial peripheral interface (*SPI*) is also widely utilized in interchip communications. These highly dedicated ports/connections are not good candidates for implementing obfuscation. On the other hand, examples of good obfuscation candidates can be the general-purpose input/output (GPIO) pins. They are generic pins on an integrated circuit whose behaviors are controllable by the user at run-time. Since GPIO pins have no predefined purpose, it is exceedingly difficult to discover the actual connections by their functionalities. Thus, the components which provide a significant number of these pins are ideal candidates for obfuscation. A system/PCB can be protected by the permutation-based obfuscation if it consists of one or more of these components. CPLDs, FPGAs, and MCUs are examples of the components mentioned

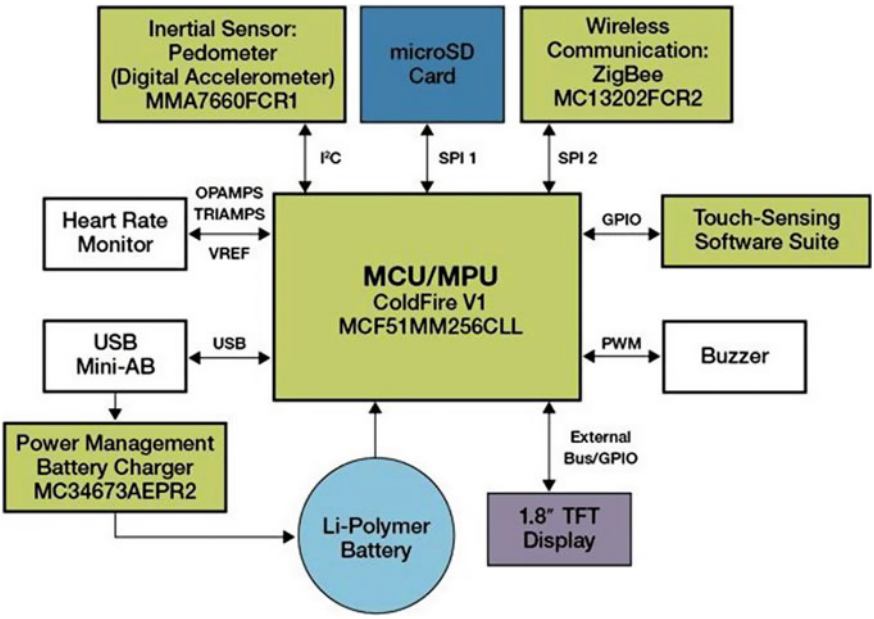

Fig. 5.4 Block diagram of NXPs activity monitor reference design

above. These components are named as **programmable components** since they can be programmed by the designer [2]. The components connected and controlled by the programmable components are named as **non-programmable components**. For instance, the heart rate monitor and display block in Fig. 5.4 are examples of the non-programmable components.

The second requirement is related to the timing constraints. Since the permutation block introduces additional propagation delays, the delay-sensitive interchip connections should be avoided in implementing the obfuscation. These connections consist of clock inputs, other control signals with strict timing requirements, etc. The last requirement specifies the types of the signal which can be permuted. Since the permutation block can only take digital signal as inputs, only the digital signals in the original design can be permuted.

Besides the connected ports in the original design, the system may not utilize all the ports of the programmable component. These unconnected ports can also be used in the obfuscation if they satisfy the requirements mentioned earlier. These unconnected ports are referred to as *dummy ports*, while the connected ones are referred to as *real ports* [2]. Involving dummy ports increases the total number of combinations which an attacker needs to examine. All the real ports and dummy ports are the inputs of the permutation block.

The schematic view of the board-level design modification is shown in Fig. 5.5. In this figure, the original connections are permuted by the permutation block. Moreover, only part of the permutation block's inputs are real inputs.

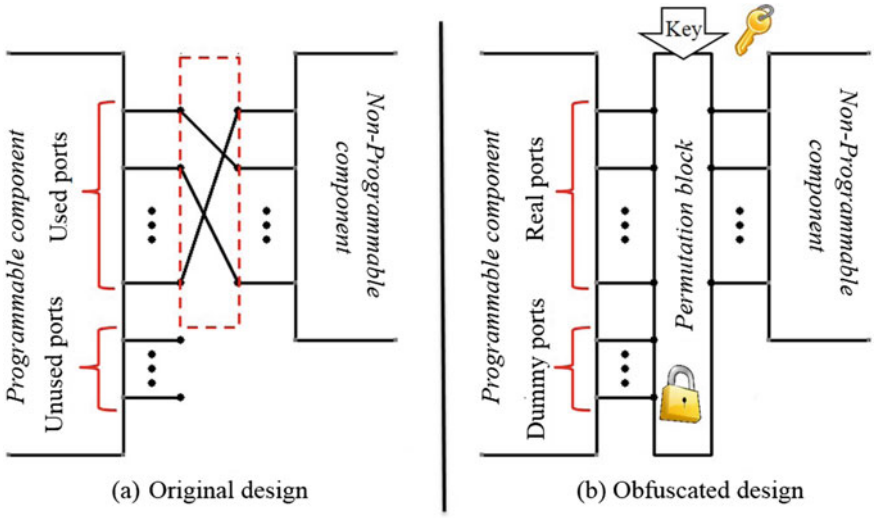

Fig. 5.5 Board-level design modification

5.4.2 Chip Level

For the chip-level application, the internal connections are the paths between gates. The functionalities of the intergate connections are difficult to be discovered without knowing the full design. Additionally, since the permutation network is merged into the gate-level design, all the intergate signals are digital. Thus, all the chip-level interconnections besides the timing-critical paths can be permuted since they are tough to be bypassed physically. As a result, the requirements (i) and (iii) need not be considered in the chip-level permutation-based obfuscation. However, the timing constraint, which is the requirement (ii), is crucial to be studied when inserting the permutation network.

Researches in [13] proposed a bus-based hardware IP protection scheme. This scheme is applicable to a broad category of electronic systems with a primary bus. Such designs include (1) numerous IP offerings for USB, PCI, PCI-E, AMBA, and other bus standards typically used in system-on-a-chip designs and computer peripherals, (2) SRAM-based FPGAs that are programmed through an input bus, (3) general-purpose and embedded microprocessors, including soft cores, (4) DSPs, (5) network processors, and (6) game consoles. This requirement is illustrated in Fig. 5.6 using a SoC architecture as an example.

The bus is equipped with additional bus-key inputs such that only a certain key combination activates the bus, while other combinations scramble it. In other words, a lock is merged with the bus and only the correct can remove the lock. According to Fig. 5.6, this lock can be allocated at the region where the bus is implemented. Several techniques can be utilized to accomplish this locking, such as bit-wise XOR, arithmetic transformations, and bit permutations.

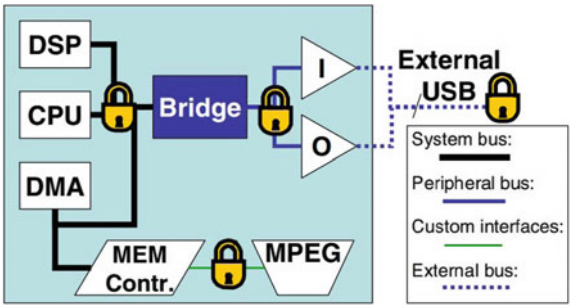

Fig. 5.6 Bus-based IP protection [13]

Comparing the requirements which need to be met in the chip-level and board-level obfuscation, obfuscating chip-level designs is more flexible than the board-level ones.

5.5 Permutation Network

Choosing a proper permutation network and implementing it in this permutation block play an essential role. The inputs of this inserted permutation block come from the selected components in the original design. The outputs are connected with the components in which these obfuscated connections are originally attached to.

In this section, the background of various permutation networks is introduced, and the best candidate for implementing the permutation-based obfuscation scheme is selected. As an example, according to the selected permutation network, this section provides the configuration approaches under different scenarios. Based on the capability of achieving full input/output permutations, permutation networks can be classified into *blocking* and *non-blocking* networks [14].

Blocking permutation network indicates that the network can only realize partial input/output combinations. Butterfly network [15] and basic Omega network [16] are two examples of blocking permutation network. These blocking networks are presented in Fig. 5.7. In this figure, the rectangles represent 2-to-2 switches. Each of these switches is controlled by a binary number. The key is formed by concatenating all these binary numbers. The total number of different keys is directly related to the number of switches. For instance, either of the permutation networks shown in Fig. 5.7 consists of 12 switches. The total number of different keys is $2^{12} = 4096$ for either butterfly network or Clos network. On the other hand, the total number of input/output combinations needed for any 8-input permutation network is $8! = 40320$. This number is greater than the total number of the keys. This observation indicates that a large percentage of input/output combinations cannot be accomplished, no matter how one configures the switches.

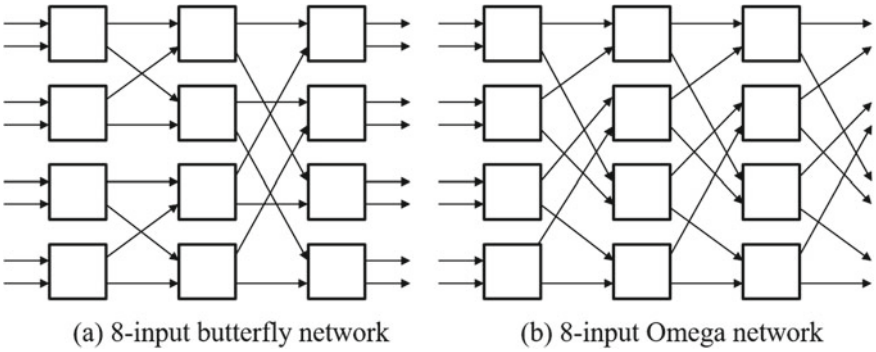

Fig. 5.7 Examples of blocking networks

Non-blocking permutation network indicates that the network can realize all input/output combinations with or without constraints. This permutation network category consists of three subcategories.

(i) *Strict-sense non-blocking network* [17] can construct a new path connecting unconnected inputs and outputs regardless of any pre-established paths. Designers can switch any two paths without adjusting the rest of the network settings. This path independence property makes strict-sense non-blocking network ideal for a telephone router. The multiplexer (MUX) array network and Clos network with high-order switches [18] are two examples of this type of network. A general design of Clos network is presented in Fig. 5.8. Clos networks have three stages: the ingress stage (Stage1), middle stage (Stage2), and the egress stage (Stage3). Each stage is made up of a number of crossbar switches, often just called crossbars. Clos networks are defined by three parameters n, m, and r. n represents the number of sources which feed into each of r ingress stage crossbar switches. Each ingress stage crossbar switch has m outlets, and there are m middle stage crossbar switches. These parameters should follow the relationship $m \geq 2n - 1$, in order to achieve the non-blocking property [19].

An example of 4-bit MUX array network is provided in [7]. This network is named as wire scrambling (WS) cells which shuffle the intergate connections. A generic WS-cell can be implemented by using multiplexers or pass transistors as shown in Fig. 5.9. The full shuffler (Fig. 5.9b) meets the definition of strict-sense non-blocking network, while the partial shuffler (Fig. 5.9c) does not. The permutations which can be achieved by this partial shuffler are a subset of the permutations that the full shuffler can achieve. For the MUX array network, to preserve the strict-sense non-blocking property, the number of MUXs should be equal to the number of the inputs of the permutation network. Additionally, each MUX should have the capability to select any input of the permutation network.

(ii) *Wide-sense non-blocking network* [20] does not provide the strict independence guarantee as a strict-sense non-blocking network does. It is still possible to connect any unused input to any unused output with certain algorithms.

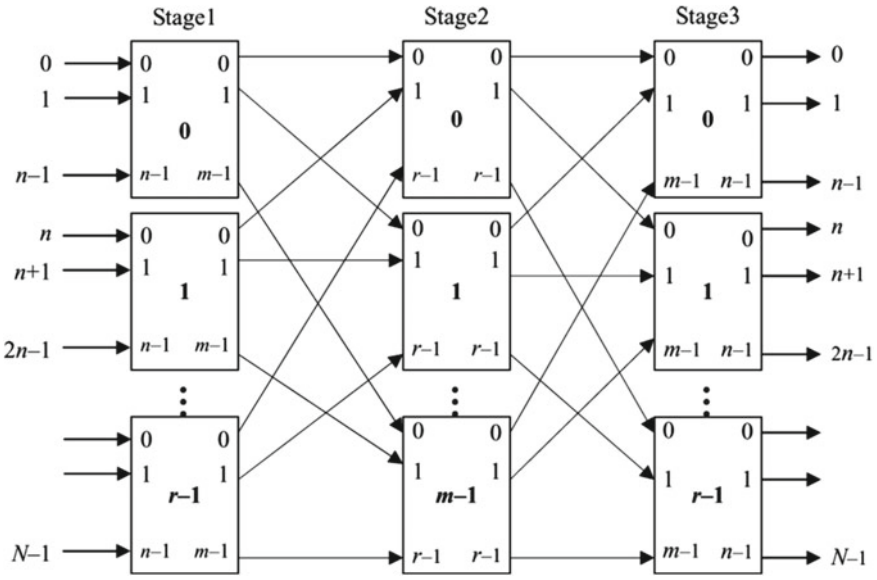

Fig. 5.8 $N \times N$ three-stage non-blocking Clos network [19]

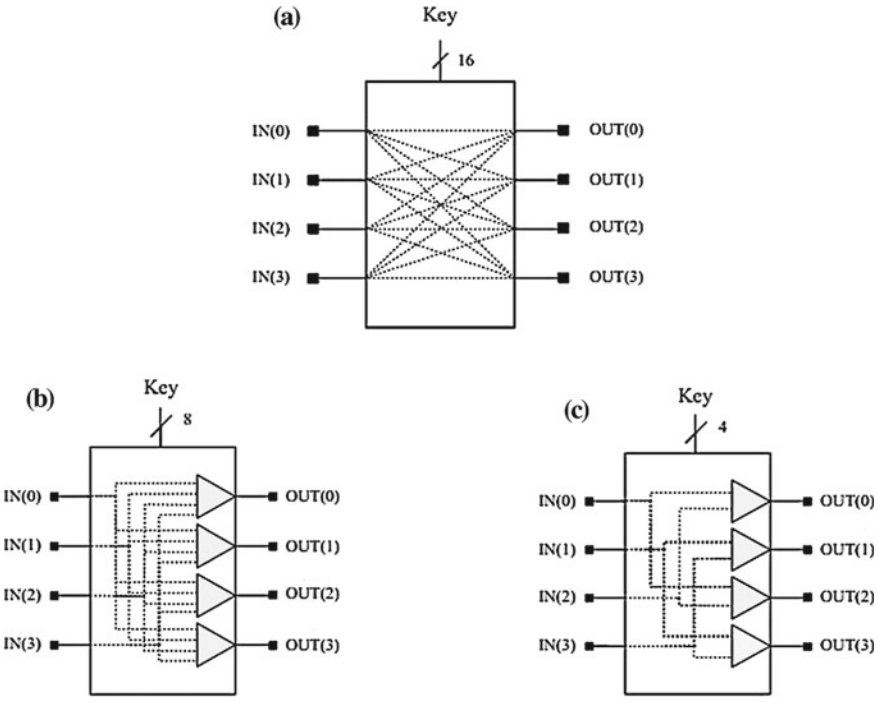

Fig. 5.9 **a** Structure of a generic WS-cell. **b** MUX base WS-cell structure, Full shuffler. **c** Partial shuffler. The multiplexers are shown as the triangles [7]

(iii) The weakest notion of non-blocking permutation network is *rearrangeable non-blocking network* [21]. This kind of network is not capable of fully realizing network configurations without the prior knowledge of inputs' and outputs' order. Benes network [22] is one example of this kind of network.

5.5.1 Area Utilization

Comparisons of implementation area utilization are performed among three permutation networks: basic Omega network (blocking), MUX array-based network (strict-sense non-blocking), and Benes network (rearrangeable non-blocking). The Omega network and Benes network consist of 2-to-2 switches, while the multiplexer-based network is composed by n-to-1 multiplexers. The parameter n depends on the number of inputs of the permutation network.

These networks are synthesized using Quartus II software and implemented in Altera MAX V CPLD. Target CPLD consists of 570 logic elements (LEs), which is the basic logic unit in CPLD. The area utilizations are presented by the percentage of exploited LEs. The results are summarized in Table 5.1. It can be observed that the Benes network utilizes about half of the LEs (60%) compared with the multiplexer-based network (115%). The number of gates needed to achieve each network increases exponentially as the number of inputs/outputs increases.

Benes network is a good candidate in implementing the permutation block for the following two reasons: (i) It is apparent that blocking permutation networks are inappropriate since they produce limited inputs/outputs combinations. For our case, designers should choose a candidate from non-blocking networks. (ii) Benes network is much better than multiplexed based network considering the hardware area utilization. For this application, the strict-sense non-blocking networks have significant overhead and their benefits for the permutation-based obfuscation are not needed.

Even though the strict-sense non-blocking networks present much larger area utilizations, they have a unique advantage, which is presented in Fig. 5.10. According to this figure, this type of network can route any input to one or more outputs. This unique benefit provides the designer more choices in intercomponent connection selection. For certain applications, one signal may be routed to multiple chips/gates (e.g., the chip selection signal which controls multiple memory chips). In these applications, the above capability of the permutation network is required.

Table 5.1 Permutation network CPLD area utilization

I/O	4/4	8/8	16/16	32/32
Omega network (%)	2	6	20	23
Benes network (%)	2	7	22	60
Multiplex router (%)	2	10	31	115

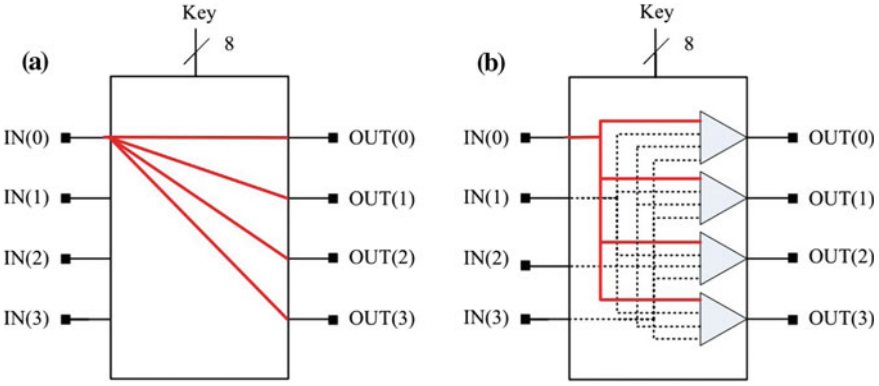

Fig. 5.10 The Multiterminal net connection in WS-cells. **a** IN(0) should be routed to all the outputs. **b** The MUX-based hardware implementation of the routing requirement shown in **a** [7]

5.5.2 Network Configuration

After selecting a permutation network, it should be configured properly. Since the Benes network presents several advantages mentioned above, its configuration situations are discussed in this section as an example. The most basic Benes network unit is a 1-bit controlled 2-to-2 switch as shown in Fig. 5.11a. The switches operate in two modes: straight mode when the control bit is 0 and exchange mode when the control bit is 1. Bits from the obfuscation *key* are used as the control bits for each switch. We define each column of switches as one stage. Two Benes network properties play a crucial role in the configuration process: (i) *Recursion* indicates that a Benes network can be split into two identical lower-dimension Benes networks. For example, in Fig. 5.11a, the 8-input Benes network can be split into two 4-input Benes networks (shown in red dashed boxes). This splitting can be recursively repeated on any lower-dimension Benes networks till reaching the basic unit (2-to-2 switch). (ii) *Symmetry* indicates that the Benes network possesses rotational symmetry across the center stage. According to this property, the stages on the left of center stage are named as *Forward stages* and the stages on the right as *Backward stages*. These stage categories are shown in Fig. 5.11b. A designer can take advantage of these properties for simplifying the configuration.

Network configurations can be classified into two situations based on the prior knowledge and the goals.

Situation 1: Designers aim to find the order of outputs with the knowledge of input order and a predefined key. Since Benes network possesses rotational symmetry, it always consists of an odd number of stages. The number of stages (S) can be found by $S = 2^* \log_2 N - 1$, where N is network dimension (i.e., the number of inputs to the Benes network). Each stage can be mathematically represented by a square permutation matrix (PM) derived from the key. A permutation matrix has exactly one '1' in each row/column and '0' elsewhere. Each permutation matrix represents the

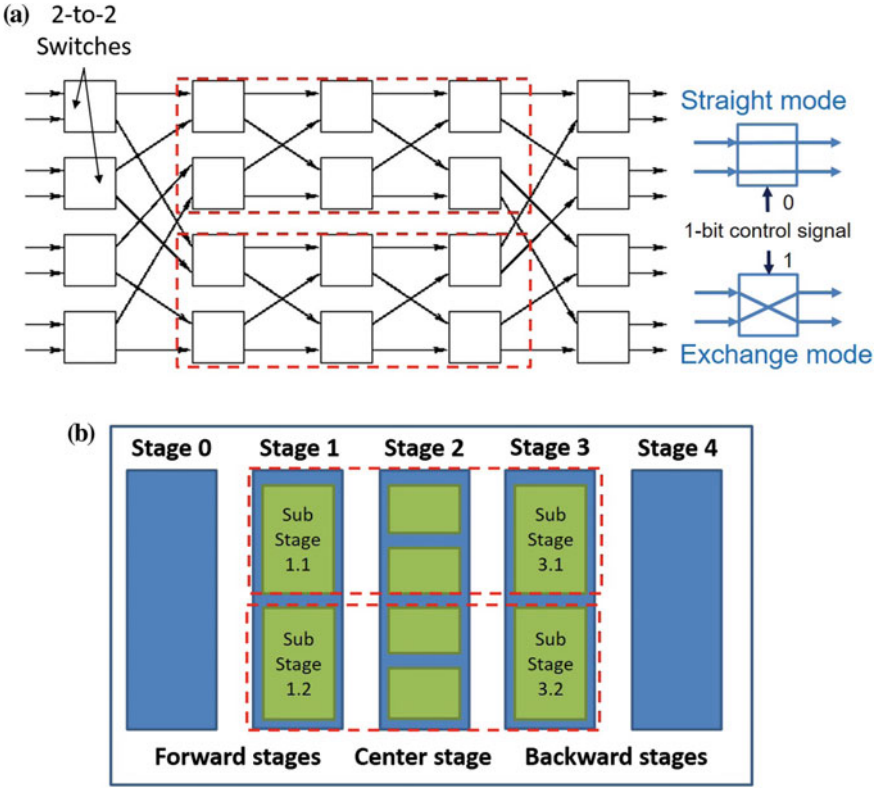

Fig. 5.11 **a** 8-inputs Benes network and **b** stage partition

permuted inputs passing the corresponding stage. Multiplying permutation matrices represents the inputs passing multiple stages coherently. The property of symmetry benefits designers by simplifying the permutation matrix calculation. With PMs computed and the input order (I), we can formalize the output order (O) as follows

$$O = I \times \prod_{i=1}^{\log_2 N - 1} PM_i \tag{5.1}$$

where N represents the number of inputs. The same procedure can be repeated for computing input order with the knowledge of output order and the key.

Situation 2: The designer's objective is to find one or more keys achieving a required input/output permutation. This procedure is also known as the network routing. Prior work such as [23] proposed an efficient routing algorithm based on searching *loops* (defined later) of *outer stages*. Outer stages (OS) are defined as mirrored stage pairs such as Stage 0 and Stage 4 in Fig. 5.11b. The number of outer stages K is defined according to the number of inputs N as follows,

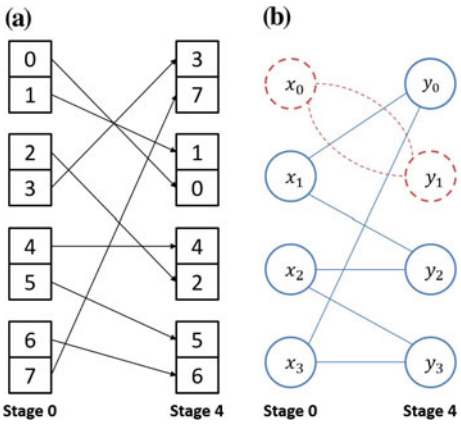

Fig. 5.12 Outer stage schematic [23] **a** switches connections and **b** groups illustration

$$K = \log_2 N - 1 \tag{5.2}$$

The *loop* structure is shown in Fig. 5.12. As an example, assume the network input as [0 1 2 3 4 5 6 7] and the output as [3 7 1 0 4 2 5 6]. To determine the loops, the proposed approach illustrates each switch as a node labeled x_0, y_0, etc., in Fig. 5.12b. The connections between nodes are constructed based on the input/output orders. A loop collects the interconnected nodes. For example, the dashed-line loop consists of nodes x_0 and y_1. Nodes within one loop should not connect the nodes outside this loop. Besides the outer stage formalized by Stage 0 and Stage 4, similar loop structures can be found in other outer stages such as Sub Stage 1.1 and Sub Stage 3.1 in Fig. 5.11b. A configuration can be accomplished by assigning '0' or '1' to each node (switch). As mentioned in [23], two equivalent key configurations can be assigned to each loop. These two configurations are complementary binary chains, such as '1011' and '0100'.

This multiple-configuration phenomenon can lead to more than one key achieving the same input/output combination. In this chapter, we refer to this phenomena as the *multiple-key effect*. This effect could diminish the protection strength by increasing the breaking probability. At the beginning of Sect. 5.2, the breaking probabilities are classified into two categories depending on what information the attacker attempts to obtain. This breaking probability refers to the probability of the attacker receiving the correct key by exhaustive search (P_{key}).

5.5.3 Multiple-Key Effect

For an 8-bit Benes network with a 20-bit key, the breaking probability is $P_{com} = 1/8! = 2.5e-5$ when the attacker examines the input/output pairs. If the attacker examines the keys instead of input/output pairs, the size of the key space to examine

is $P_{key} = 2^{20} = 1048576$. Besides the Benes network, this effect may be observed in other permutation networks where the key space is larger than the input/output combinations. Let the number of multiple keys be m for a given input/output combination. The following scenarios may happen: (i) $m/2^{20} < 1/8!$, multiple-key effect decreases the breaking probability. (ii) $m/2^{20} > 1/8!$, multiple-key effect increases the breaking probability. (iii) $m/2^{20} = 1/8!$, the breaking probability remains unchanged. The last situation only holds when a one-to-one correspondence between the key and the input/output combination holds. However, this situation does not hold when certain permutation networks, such as the Benes network, are implemented. Discovering minimal, maximal, mean, and medium values of m is invaluable for precisely evaluating obfuscation performance of the overall approach.

The breaking probability can be computed as Eq. (5.3) when the attacker chooses to examine the keys.

$$P_{key} = \frac{\text{Number of correct keys}}{\text{Total number of keys}} \tag{5.3}$$

Since the total number of keys is more than the total number of input/output combinations, the multiple-key effect may or may not increase the breaking probability.

The second configuration situation in Sect. 5.5.2 indicates that the switches within each loop can be configured in two ways, such as '1101' and '0010'. These switches' configurations are connected formalizing the key. In general, a relationship between the number of loops and multiple keys (*Mkey*) can be described in Eq. (5.4).

$$Mkey = \prod_{k=1}^{K} 2^{l_i} = 2^{\sum_{i=1}^{K} l_k} \tag{5.4}$$

where K is the number of outer stages (OS), and each of them consists of l_k loops. The number of loops (l_k) depends on the input/output combinations.

In order to break the proposed board-level obfuscation by brute force, the attacker would choose from the following two strategies: **Strategy (i)** examines the input/output order combinations and **Strategy (ii)** examines the keys. These two strategies correspond to the two types of breaking probabilities: P_{com} and P_{key}. Affected by multiple keys with the same permutated outcome, the attacker only needs to figure out one of the keys when choosing the strategy (ii).

Effects of multiple keys on obfuscation strength are studied by comparing the breaking probabilities of applying strategy (i) and strategy (ii). Applying **Strategy (i)**, the breaking probability is $1/32! = 3.8004E - 36$ for a 32-bit Benes network. For **Strategy (ii)**, the breaking probabilities can be computed through dividing the numbers of multiple keys by the total number of keys. The breaking probabilities under strategy (i) and strategy (ii) are summarized in Table 5.2. Row **# of multiple keys** represents the minimal, maximal, and mean number of multiple keys under strategy (ii). These values can be obtained from the number of loops distribution. Row **Probability** shows the corresponding breaking probabilities. The mean breaking probability is computed based on uniform distribution assumption of the Benes network input/output combinations.

Table 5.2 Effects of multiple keys

	Strategy (i)	Strategy (ii)		
		Min.	Max.	Mean
# Multiple keys	N/A	32768	1.8447E+19	8.1859e+07
Probability	3.8004E−36	1.4694E−39	8.2719E−25	3.6706e−36

Since breaking probabilities vary with different input/output combinations when applying strategy (ii), we compare the mean breaking probability with the breaking probability under strategy (i). According to Table 5.2, *exploiting the multiple-key effect of the Benes network decreases the expected (average) breaking probability. This means that it is even harder to execute strategy (ii) compared to strategy (i).*

5.6 Key Management

The permutation obfuscation is controlled by a key. This key configures the system (either a board or chip) and removes the mystery (the question mark in Fig. 5.3). Several options can be exploited to generate and store the key/configuration. Each of these options has advantages and drawbacks.

The key/configuration can be input into the permutation/obfuscated chip either internally or externally as shown in Fig. 5.13. The internal mode (Fig. 5.13a) implies that this key is permanently stored in the on-chip nonvolatile memory such as electrically erasable programmable read-only memory (EEPROM) or Flash. The booting unit automatically loads the saved configuration into the permutation network during each power-up. As discussed in Sect. 5.5.3, the stored key can be either the same or different among the various chips. Reference [13] proposed an IC activation protocol which utilizes the Diffie–Hellman (D-H) key exchange scheme. This protocol is presented in Fig. 5.14.

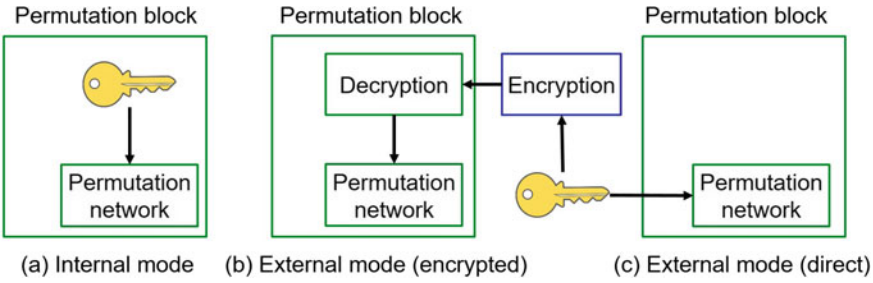

Fig. 5.13 Key loading modes

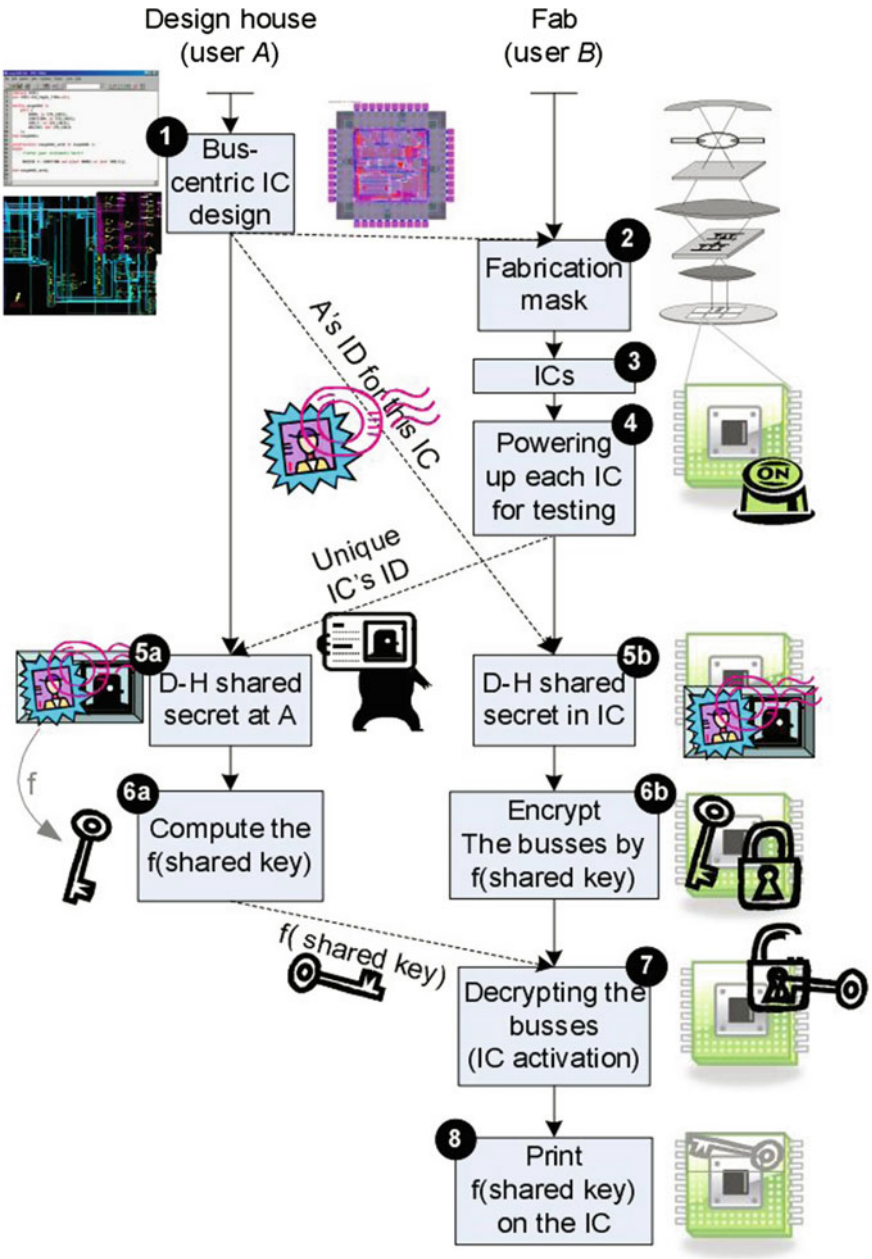

Fig. 5.14 IC activation protocol [13]

In this protocol, the obfuscated chip generates a unique identity number exploiting its embedded physical unclonable function (PUF) structure. The unique ID for the pertinent chip under test would be used as the D-H component b. The value $g^b \bmod p$ is then communicated to the design house (user A), who has also generated a unique ID a corresponding to the chip under test. In turn, A sends $g^a \bmod p$ to B. Now, the design house and the chip under test share the same secret. The key to unlock the chip can be computed by this shared secret by a secret one-way function f. This function is implemented in the hardware of the chip and only known by the designer. If the chip can be successfully activated, the key will be printed in this chip. The internal key storage mode is a convenient but unsafe scheme since these memories represent serious security concerns if they are compromised, and the keys are stolen [10, 24]. Besides the storage vulnerability, the D-H key exchange scheme encounters several known attacks. These vulnerabilities will be elaborated in Sect. 5.8.

Alternatively, the key can be loaded into the permutation network from outside the chip during powering up (Fig. 5.13b, c). The configuration information is stored in the embedded volatile memory and erased after the system loses power. As shown in Fig. 5.13b, the key is encrypted outside the chip and decrypted within the chip. Constrained by the power consumption, area overhead, etc., not all the permutation/obfuscated chips have this decryption capability. A more general case is provided in Fig. 5.13c. In this case, the key is loaded into the permutation block directly. For example, this could be done by the owner of the system through a USB drive or smart card.

5.6.1 Biometric-Based Key Generation

Besides the binary keys stored in the tokens, this key can be generated from a person's biometrics in the external key loading modes [25]. Biometrics have been investigated for identification, authentication, and key generation for many years [26]. Most of the biometric modalities, such iris and electrocardiogram (ECG), present strong capability against being duplicated by the attackers. A general ECG-based key generation flow is shown in Fig. 5.15. In this figure, the upper block illustrates the enrollment phase. During this phase, the user's biometric signal is collected and processed. The features of the processed signal are extracted and quantized to generate the enrolled key. The helper data shown in the enrollment phase consist of the information related to the key regeneration process. It is unnecessary to keep these helper data confidential since the enrolled key cannot be recovered from it. This enrolled key can be utilized to design the permutation network following the configuration situation 1 provided in Sect. 5.5.2. The key regeneration phase is presented in the lower block. This phase processes the following steps: preprocessing, feature extraction, and quantization with the support of helper data. These steps are the same as the enrollment phase and can be executed by the hardware integrated into the obfuscated system.

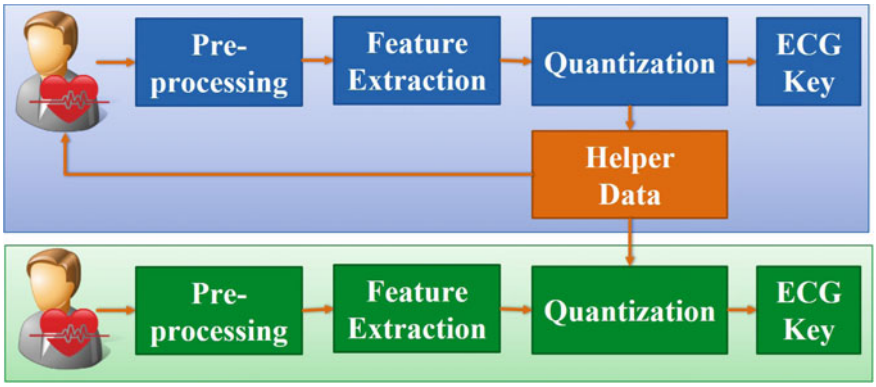

Fig. 5.15 ECG-based binary key generation

Bonding the biometrics and the obfuscated system provides a unique benefit. Combining the biometric-based key and the obfuscation protection enables a one-to-one relationship between the device and its operator. Based on the advantages of utilizing biometric-based keys, they are ideal candidates for some critical applications such as military devices.

5.7 Obfuscation Performance Evaluation

The obfuscation performance indicates how much work/time an attacker needs to devote in breaking the obfuscation. A comprehensive evaluation approach should be provided with the obfuscation framework. This evaluation can be split into two parts: (i) the feasibility of breaking the obfuscation by brute force and (ii) the robustness against other attacks.

The part (i) evaluation can be accomplished by providing expected time to break the obfuscation by brute force. This validation (T) can be calculated in Equation

$$T = \frac{t}{P} \tag{5.5}$$

where t refers to the time needed to verify one brute force guess. This guess can be either a key or an input/output combination of the permutation block. P refers the breaking probability defined at the beginning of Sect. 5.2. This breaking probability can be P_{com} or P_{key} depending which attacking strategy is applied. P_{com} and P_{key} are equal if and only if a one-to-one correspondence exists between the input/output combination and the key. As proved in Sect. 5.5.3, the expected P_{key} is smaller than P_{com} when the Benes network is attacked in a brute force manner. Thus, the breaking probability P_{com} evaluates the obfuscation performance better when the Benes

network is engaged. For other types of permutation networks, both of P_{key} and P_{com} should be computed, and the larger one indicates the obfuscation performance.

The calculation of P_{key} only depends on the total number of keys and the number of correct keys. For calculating the probability P_{com}, the result not only depends on the number of correct input/output combinations and the total combinations. For the board-level obfuscation, other knowledge from the chips' datasheets may help to increase P_{com}. An example [2] is provided in this section to show how this knowledge benefits in attacking the obfuscation. In the reference design presented in Fig. 5.4, the touch-sensing block will communicate with MCU via Rapid GPIO (RGPIO). Not all ports on the MCU have the capability to work as RGPIO (in fact, only 16 ports). In this sense, the attacker can shrink the scope by searching the correct connection for the touch-sensing device from 16 ports instead of all the obfuscated ports on MCU. In computing the breaking probability P_{com}, the functionalities of each obfuscated ports should be considered.

Part of the board-level obfuscation performance evaluations provided in [2] is presented in Table 5.3. The reference designs are listed below:

- Driving a Stepper Motor Reference Design with High-Performance MCU
- Ultralow Power Multi-sensor Data Logger with NFC Interface Reference Design
- Single-axis Motor Control Reference Design with Integrated Power Factor Correction
- SimpleLink Multi-Standard CC2650 SensorTag Reference Design

In the table, the column 'Programmable Component' provides the models of MCUs utilized in the reference. The column 'Break Probability' refers to the probability of breaking the obfuscation by examining the input/output combinations. The column 'Clock frequency' indicates the maximal operating clock frequencies of the programmable components. The parameter t in Eq. (5.5) is assumed as one clock cycle. Then, the column 'Validation Time' is computed by Eq. (5.5). The real validation time would be much longer than the value in Table 5.3 since it is extremely difficult for the attacker to validate each input/output combination in single clock cycle. In real world, he needs to observe the behavior of the board to tell whether this combination under test is correct. This process will be much slower than one clock cycle. Hence, these results should be considered as pessimistic. According to this table, besides the Design No.4, the validation times for the rest are longer than thousands of years. It is impossible to break the obfuscation by brute force in a reasonable time period. The validation time for Design No.4 is extremely short since this design only consists of a small programmable component with less than 32 pins. Among these pins, few of them can be obfuscated. Thus, in order to achieve a good board-level obfuscation performance, the original design is required to consist of a programmable component with more than 32 pins.

Besides the brute force attack feasibility analysis, the robustness against other attacks such as the hardware probing and reverse engineering should also be evaluated. The part (ii) evaluation studies the obfuscation's resistance of other types of attacks. This part of evaluation is presented in Sect. 5.8. In the same section, the corresponding countermeasures are also provided.

Table 5.3 Board-level obfuscation performance evaluation

Design no.	Programmable component	Break probability (P_{com})	Clock frequency (f) (Mhz)	Validation time (T)
1	TM4C123GH6PM	9.70E−41	60	5.4E+24 yrs
2	MSP430FR5969	1.34E−18	16	1.5E+03 yrs
3	TMS320F28050	1.23E−32	60	2.6E+24 yrs
4	CC2650	2.81E−15	48	85.8 h

5.8 Attack Analyses and Countermeasures

In the section, a list of potential attacks is provided. These attacks are grouped into three categories based on the attackers' capabilities. Against these attacks, the countermeasures are organized by the levels of security requirement.

5.8.1 Potential Attacks

To break the permutation-based obfuscation, the attacker needs to discover the following information: true intercomponent connections or the correct keys as discussed in Sect. 5.2. The following attacks can be carried out by the attacker to accomplish the attacking goal. These attacks cover the ones that can be either applied on chip level or board level. All the following attacks can be implemented on board level, while only a few of them are applicable on chips. Some of these attacks directly provide the attackers the information they desired, while some of them reinforce others. Since different attacks demand various equipment and knowledge, the capabilities of the attackers are studied.

Brute force attack is the most straightforward one and can be applied on both board level and chip level. As discussed earlier in Sect. 5.2, this attack entails attackers trying all the keys or the input/output permutations. The success of such attacks is highly dependent on the breaking probabilities (either P_{com} or P_{key}) and the time required to validate the system for correct behavior. It is extremely difficult for the attackers to break our obfuscation method by brute force as reported in [2].

Surface trace probing attack: The waveform between permutation block inputs and the corresponding outputs are the same for every system even though the key may be different. Attackers with a working system can probe all the I/Os of the permutation block with a multichannel logic analyzer. By simply matching the waveforms of the probed signals, attackers would find out the true connections between the programmable and non-programmable components. This attack can only be applied on board level since it applies to traces on the surface.

Storage compromise attack: The attacker may attempt to extract the keys/configurations if they are stored in the system. Several techniques can be utilized to retrieve the data from the nonvolatile memory [10, 24]. This attack is applicable on both board level and chip level when the internal key storage mode is applied. If the keys are the same for different chips/boards, this attack becomes powerful. Extracting one key from one chip enables the attacker to unlock other chips/boards.

Man-in-the-middle attack [13]: This is a well-known attack on the D-H key exchange scheme. Differing from the storage compromise attack, this attack provides an attacker the ability to compute the key for each chip/board. In this attack, the adversary intercepts designers public value and transmits its public value to the IC. When the IC sends its public value, the attacker substitutes it with it own and transmits it to the designer. Therefore, the adversary and designer agree on one shared key and IC and attacker agree on another shared key. Now, the attacker can simply decrypt messages transmitted by the designer and the IC and can read and potentially modify them before re-encrypting with the proper key and sending them to the other party. This vulnerability is possible because D-H key exchange protocol does not authenticate the users.

Permutation block reinstallation attack: This attack can only be applied on the board level. Removing the permutation block from the fabricated chip damages both the permutation block and the chip. Applying this attack, attackers unmount the permutation block (typically a CPLD) off the board and reinstall onto other platforms without damaging the package. Reinstalling the unmounted CPLD makes the surface trace probing applicable. This attack does not provide the attacker secret information directly. However, combining this attack makes surface trace probing attack more powerful.

Middle-layer probing attack: This is another attack which aims to empower the surface trace probing attack. Thus, this attack can be only applied on the board level. Attackers can discover full PCB layout by nondestructive reverse engineering approaches such as X-ray-based techniques [11, 27]. Guided by the layout information, attackers can mill holes for probing every CPLD port. These holes enable the attacker to access the CPLD ports without damaging other traces on the board. For the chip-level attack, microprobing could be used to read buses on the chip. This attack enables the attacker to sniff the data transmitted on the buses. However, this attack is much more expensive than the board-level probing.

IC Reverse engineering attack: The attacker can fully reverse engineer any chip on the board and learn completely secret information stored in these chips. These chips can be either the permutation block in board-level application or obfuscated chip in chip level. This secret information includes the firmware of the programmable component, internal or external memory content, and the permutation network structure. This attack can be applied on both chip level and board level.

In a well-known article from IBM [28], the authors suggest that the attackers can be grouped into three classes, depending on their expected abilities and attack strength:

- **Class I (clever outsiders)**:
  They are often very intelligent but may have insufficient knowledge of the system. They may have access to only moderately sophisticated equipment. They often try to take advantage of an existing weakness in the system, rather than try to create one.
- **Class II (knowledgeable insiders)**:
  They have solid specialized technical education and experience. They have varying degrees of understanding of parts of the system but potential access to most of it. They often have access to highly sophisticated tools and instruments for analysis.
- **Class III (funded organizations)**:
  They can assemble teams of specialists with related and complementary skills backed by excellent funding resources. They are capable of in-depth analysis of the system, designing sophisticated attacks, and using the most advanced analysis tools. They may use Class II adversaries as part of the attacking team.

The brute force attack, the surface probing attack, and the man-in-the-middle attack can be achieved by **Class I attackers**. Since these attackers only have insufficient knowledge of the system, brute force attack is the most straightforward one. Equipped with oscilloscopes, this class of attackers can accomplish the surface probing attacks. Compared with Class I attackers, **Class II attackers** can apply the storage compromise attack, permutation block reinstalling attack, and middle-layer probing attack with more sophisticated instruments. The tools required for these attacks should have the capabilities of uninstalling CPLDs without damaging the package, applying nondestructive reverse engineering, and drilling holes on a multilayer PCB. **Class III attackers** are the only ones that can apply IC reverse engineering attack since they have access to most advanced analysis tools and greatly funded.

5.8.2 Countermeasures and Attack Coverage

Considering the various classes of attackers and their capabilities, the countermeasures are organized into three security requirement levels.

Low-level security requirement: Devices satisfying this security requirement level directly store the same key in nonvolatile memories. To achieve low-level security requirement, we suggest that ball grid array (BGA) packages should be exploited for both the programmable component and permutation block and at least one middle layer in the PCB to route connections. The requirement of multiple layers is easy to meet due to the complexity of current PCBs. For the requirement of BGA package, a statistical result in Table 5.4 shows a significant portion of obfuscation candidates have BGA/VTLA (a package standard used by Microchip very similar with BGA package) package among the programmable components with equal to or more than 64 pins. As discussed in Sect. 5.7, a programmable component with more than 32 pins is required to guarantee a good obfuscation performance. Using BGA package introduces no additional area and power overhead. The cost is the same

Table 5.4 Percentage of obfuscation candidates with BGA/VTLA package

Manufacturer	⩾64 pins	BGA/VTLA	Percentage (%)
Microchip	241	107	44.40
Freescale	85	80	94.12
NXP	93	93	100
Total	419	280	66.82

between BGA package and other packages for the same model of the chip as well. When routing the PCB, the designer should identify all the connections between the programmable component and the permutation block and route them in the middle layers. The routing requirement is named as *middle-layer routing* for the remainder of the paper. This step is meant to reduce the attackers' chance of probing a working device to identify connections between chips. Low-cost tamper resistance techniques [29] should also be applied.

Medium-level security requirement: System rebooting is unavoidable for most industry systems and consumer electronics. Since reapplying keys after every reboot is often impracticable in this case, these systems need to store the keys in nonvolatile memory and reload them prior to rebooting automatically. This behavior provides attackers opportunities of breaking the obfuscation by certain techniques such as the permutation block reinstalling attack and storage compromise attack.

To obtain medium-level security requirements, the BGA packages and middle-layer routing need to be engaged. Moreover, instead of storing the same key for all the chips/boards, each of these systems should have its unique key. Since the internal key storage mode is applied, we can take advantage of the D-H key exchange scheme and the one-way function as provided in Sect. 5.6. Combining D-H key exchange scheme and a unique ID, each chip/board can be assigned a unique key. This unique ID can be generated by incorporating a chip-level or board-level PUF.

High-level security requirement: In this case, the key is not stored in system's nonvolatile memories and needs to be reapplied into the system after rebooting. Devices satisfying this security requirement should be capable of preventing all known attacks. Critical applications such as military installations and commercial devices storing sensitive data require this level of protection. The biometric-based keys (discussed in Sect. 5.6.1) can also contribute to this level of security requirement.

Since the key is not stored in nonvolatile memory, a D flip-flop chain is utilized to form a shift register in permutation block or obfuscated chip. One dedicated port on the chip is designed as the serial key input. When the system is powered off, D flip-flops lose their values, and the key is destroyed. The user needs to input the key to the system whenever it reboots. Additionally, BGA packages and middle-layer routing should also be involved in this level of security requirement.

Considering various attackers' capabilities with aforementioned security requirement levels, the protection coverage is analyzed in Fig. 5.16.

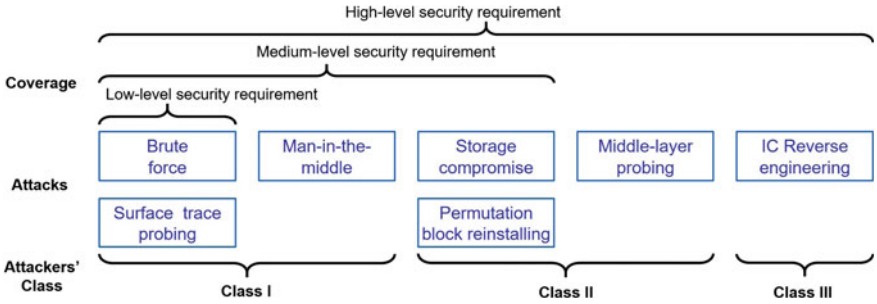

Fig. 5.16 Attacks coverage analysis

Devices satisfying low-level security requirement eliminate the brute force and surface trace probing attacks applied by Class I attackers. The surface trace probing attack is unavailable since the obfuscated connections are hidden in the middle layers. However, since the key is stored in the onboard nonvolatile memory, Class II attackers can unmount and reinstall the permutation block onto their platforms (permutation block reinstalling attack) for applying hardware probing attack. The techniques for extracting the content from the nonvolatile memory (storage compromise attack) also directly enable an attacker to learn the key. Since this key is the same among different boards, this compromised key can be used to active other boards.

The permutation block reinstalling attack, storage compromise attack, and man-in-the-middle attack can be eliminated if the device is compatible with the medium-level security requirement. Devices under this level of security requirement incorporate the obfuscation scheme with the following techniques: chip-level/board-level unique identifier; the D-H key exchange scheme; and the one-way function. This combination guarantees the following: (i) The permutation block can unlock the system using the prestored key only when it is attached to its original PCB; (ii) each chip/board can be only activated by a unique key; (iii) the man-in-the-middle attack is prevented. The permutation block reinstalling attack can be eliminated by the first guarantee, and the second guarantee prevents the storage compromise attack. Even if a man-in-the-middle attack establishes two different channels (one with the IC and one with the designer), he will not be able to compute the key specific to the IC or to use the key given for another chip.

Unfortunately, automatically loading the key after system rebooting again makes middle-layer probing attack available. Instead of reallocating the permutation block, this attack extracts the board layout nondestructively through techniques such as X-ray [11]. With this knowledge, Class II attackers can design and create holes to probe the permutation block pins. This attack can be accomplished during run-time without delayering the PCB. Even if the middle-layer probing attack cannot be achieved due to extremely complex middle layers, Class III attackers can always learn the key by reverse engineering the chips.

For completely preventing all known attacks, the key should be destroyed when the system loses power. High-level security requirement enables devices achieving this objective through storing the key in volatile memory (e.g., in the embedded D flip-flop chain). Since Class II and Class III attackers require a working device after rebooting, it is impossible for them to apply attacks on the devices which the keys are destroyed after powering off.

5.9 Conclusions

Among various obfuscation-based hardware protection approaches, the permutation-based technique presents certain advantages. For instance, this technique is the only one that can be exploited in protecting PCBs. Other obfuscation techniques such as logic encryption can be easily broken if applied at board level.

The permutation-based obfuscation permutes the intercomponent connections of either an IC or PCB. The designer needs to be aware of selecting these connections. Several requirements should be met when determining whether a connection is suitable for permutation. These requirements consist of the timing, functionalities, and signal types. Note that the requirements needed for the chip-level application are fewer than the ones for the board level. After appropriate connections are chosen, they will be permuted by a permutation network. The permutation network capability and its area utilization should be carefully balanced. For certain permutation networks (i.e., Benes network), the multiple-key effect can be observed. This effect causes more than one kind of the network's configuration result to end up with the same permutation outcome. This effect entails the designer to evaluate the brute force breaking probabilities in two ways (i.e., the probability when the attacker examines the keys or the input/output combinations). The larger breaking probability is used to determine the robustness against the brute force attack. The time required to break the obfuscation has a direct relationship with this breaking probability. As reported by the existing work, it may take longer than thousands of years to break a properly obfuscated system.

Besides the brute force attack, the designer should also evaluate the robustness against other attacks such as the hardware probing and reverse engineering. Since these attacks require various equipment and resources, they are classified into three levels based on the difficulty to execute them. Although some attacks are powerful in breaking the obfuscation, they can be mitigated by certain countermeasures. Similar to the classification of the attacks, the countermeasures are grouped into three levels based on the attacks they cover.

Since the permutation-based obfuscation is controlled by a key/configuration, various schemes can be exploited to manage the key either internally or externally. Each of these schemes has its advantages and drawbacks. The more convenient the key management scheme is, the less secure the system will be.

References

1. Roy JA, Koushanfar F, Markov IL (2008) Epic: Ending piracy of integrated circuits. In: Proceedings of the conference on Design, automation and test in Europe. ACM, pp 1069–1074
2. Guo Z, Tehranipoor M, Forte D, Di J (2015) Investigation of obfuscation-based anti-reverse engineering for printed circuit boards. In: Proceedings of the 52nd annual design automation conference. ACM, p 114
3. Chakraborty R, Bhunia S (2009) Harpoon: An obfuscation-based soc design methodology for hardware protection. IEEE Trans Comput-Aided Design Integr Circuits Syst 28:1493
4. Chakraborty R, Bhunia S (2010) Rtl hardware ip protection using key-based control and data flow obfuscation. In: 23rd international conference on VLSI design, VLSID'10. IEEE, pp 405–410
5. Koushanfar F (2012) Provably secure active ic metering techniques for piracy avoidance and digital rights management. IEEE Trans Inf Forensics Secur 7(1):51–63
6. Baumgarten AC (2009) Preventing integrated circuit piracy using reconfigurable logic barriers
7. Zamanzadeh S, Jahanian A (2016) Higher security of asic fabrication process against reverse engineering attack using automatic netlist encryption methodology. Microprocess Microsyst 42:1–9
8. Tehranipoor M, Wang C (2011) Introduction to hardware security and trust. Springer Science & Business Media, New York
9. Handschuh H, Paillier P, Stern J (1999) Probing attacks on tamper-resistant devices. Cryptographic hardware and embedded systems. Springer, Berlin, pp 303–315
10. Quadir SE, Chen J, Forte D, Asadizanjani N, Shahbazmohamadi S, Wang L, Chandy J, Tehranipoor M (2016) A survey on chip to system reverse engineering. ACM J Emerg Technol Comput Syst (JETC) 13(1):6
11. Asadizanjani N (2015) Non-destructive pcb reverse engineering using x-ray micro computed tomography. In: ISTFA
12. Zhang F, Hennessy A, Bhunia S (2015) Robust counterfeit pcb detection exploiting intrinsic trace impedance variations. In: IEEE 33rd VLSI test symposium (VTS). IEEE, pp 1–6
13. Roy JA, Koushanfar F, Markov IL (2008) Protecting bus-based hardware ip by secret sharing. In: Proceedings of the 45th annual design automation conference. ACM, pp 846–851
14. Waksman A (1968) A permutation network. In: JACM
15. Thamarakuzhi A, Chandy JA (2010) 2-dilated flattened butterfly: A nonblocking switching network. In: HPSR
16. Mitra D, Cieslak RA (1987) Randomized parallel communications on an extension of the omega network. In: JACM
17. Giacomazzi P, Trecordi V (1995) A study of non blocking multicast switching networks. IEEE Trans Commun 43:1163
18. Jajszczyk A (2003) Nonblocking, repackable, and rearrangeable clos networks: fifty years of the theory evolution. IEEE Commun Mag 41(10):28–33
19. Yang J, Yang J, Li X, Chang S, Su S, Ping X (2011) Optical implementation of polarization-independent, bidirectional, nonblocking clos network using polarization control technique in free space. In: Optic Eng 50(4):045 003–045 003
20. Feldman P, Friedman J, Pippenger N (1988) Wide-sense nonblocking networks. SIAM J Discrete Math 1(2):158–173
21. Pippenger N (1978) On rearrangeable and non-blocking switching networks. J Comput Syst Sci 17(2):145–162
22. Chang C, Melhem R (1997) Arbitrary size benes networks. Parallel Process Lett 7(3):279–284
23. Nassimi D, Sahni S (1982) Parallel algorithms to set up the benes permutation network. IEEE Trans Comput vC-31(2):148–154
24. Jeong H, Choi Y, Jeon W, Yang F, Lee Y, Kim S, Won D (2007) Vulnerability analysis of secure usb flash drives. In: IEEE international workshop on memory technology, design and testing, MTDT. IEEE, pp 61–64

25. Guo Z, Karimian N, Tehranipoor MM, Forte D (2016) Hardware security meets biometrics for the age of iot. In: IEEE International Symposium on Circuits and Systems (ISCAS)
26. Wayman J, Jain A, Maltoni D, Maio D (2005) An introduction to biometric authentication systems. Springer, London
27. Ahi K, Asadizanjani N, Shahbazmohamadi S, Tehranipoor M, Anwar M (2015) Terahertz characterization of electronic components and comparison of terahertz imaging with x-ray imaging techniques. In: SPIE sensing technology + applications
28. Abraham DG, Dolan GM, Double GP, Stevens JV (1991) Transaction security system. In: IBM Syst J
29. Karri R, Rajendran J, Rosenfeld K, Tehranipoor M (2010) Trustworthy hardware: Identifying and classifying hardware trojans. IEEE Trans Comput 43(10):39–46