CHAPTER

TEST-ORIENTED ATTACKS

9

CONTENTS

9.1 INTRODUCTION

Testability and security inherently contradict each other [1]. The testability of a chip can be defined by the amount of controllability and observability that the test engineer is granted. The higher degree of controllability and observability allowed, the easier it is to test the circuit under test (CUT). The test is not only easier to perform, but the result of the test becomes more reliable due to the higher fault coverage.

On the other hand, security ensures that anything in a circuit is safely stored within itself. The most common manner of providing security is to hide the information behind some form of recognition that would be able to tell an authorized user from an attacker. Modern-day security in all realms uses this method to protect vital assets, whether it is a security code for a home, retinal scanner for a lab, or encryption key for information. Simply put, security relies on making information obscure, and difficult, to figure out.

When trying to relate testability and security together in a chip, security is clearly contradicted by testability. By designing for testability, a designer is essentially revealing vital information about the chip through the use of scan test. If the aim of designing a chip is security, it is very difficult to justify the amount of controllability and observability that testability aims to provide because of test-related leaks. It is also necessary, however, to ensure that the chip will function properly through testing in a fast and reliable manner. The only system secure from any leaks is one without any controllable inputs and observable outputs, but this is absurd from both a testability and usability standpoint.

Chip security has grown concern mostly to protect IP from malicious users and hackers. There are many hackers in the world with many different motivations. They range from the noble (attempting to make their fellow developers aware of their pitfalls), to the malicious (stealing information), and to simply the curious [2]. The skill-set of hackers vary as much as their intentions.

Testability and security have what appears to be a mutually exclusive relationship. It is very difficult to satisfactorily meet the needs of both specifications. A middle ground must be met between the fully controllable and observable CUT and a black box. If one considers the hacker during design, a clearer relationship between testability and security can more easily be concluded. If the designer can target specifically, which features he/she would like to prevent access to, it may be easier to make design compromises between testability and security.

9.2 SCAN-BASED ATTACKS

Controllability and observability of a circuit under test have significantly reduced as chip design complexity continues to increase. This problem greatly affects test engineers' ability to perform fast and reliable tests using the primary input and primary output alone, which negatively affect time to market and the number of defective parts delivered to the customer. Design-for-testability (DFT) addresses this issue by considering manufacturing test during design.

Scan-based DFT is one commonly practiced technique that greatly improves controllability and observability by modifying flip-flops into a long chain, essentially creating a shift register. This allows test engineers the ability to treat each flip-flop in the scan chain as a controllable input and observable output.

Unfortunately, the same properties that scan improves upon for testing also creates a severe security hazard. Scan chains become an easily exploitable side channel for cryptanalysis [3,4] due to the possibility of placing the CUT into any state (controllability), and stop the chip in any intermediate state for analysis (observability). Because of the widespread use of scan-based testing, this side channel has become a major concern in the industry [5].

These concerns add to an already mounting problem of hardware security. Other side-channel attacks, such as differential power analysis [6], timing analysis [7], and fault-injection [8,9] attacks have also been shown to be potentially serious sources of security failures. Tamper-resistant designs [10,11] propose to mend these leaks. However, scan chains are necessary to expose any defects in the chip that may exist. Whereas disabling the scan chains after manufacturing test (for example, by blowing fuses) has become a common practice for applications, such as smartcards [12], there are also applications that require in-field testing, which make it impossible to deliberately destroy access to the test ports.

As scan-based attacks require minimally invasive techniques to perform the attack [12], this exploit becomes accessible to attackers with a wide-variety of knowledge and resources [1]. A security measure that prevents scan-based attacks requires the ability to scale the desired level of security on the application. Such a measure must also minimally affect the ability of a test engineer to efficiently test the chip after fabrication.

The latter point must be keenly considered, as the entire purpose of using scan is for testing. Although the goals of security and testing appear to be contradictory, the security of the chip can easily fail if it is not properly tested.

9.2.1 SCAN-BASED ATTACK CATEGORIZATION

Developing a secure scan design is dependent on targeting both the type of attacker [1], and how they can potentially make the attack. The authors categorize the scan-based attacks into two types: scan-based observability and scan-based controllability/observability attacks. Each requires that a hacker has access to the test control (TC) pin. The type of attack depends on how a hacker decides to apply stimuli. The proposed low-cost secure scan design removes the hacker's ability to correlate test response data by creating a random response when an unauthorized user attempts access, which prevents hackers from exploiting the two attacks described in the remainder of this section.

9.2.1.1 Scan-Based Observability Attack

A scan-based observability attack relies on a hacker's ability to use the scan chain to take snapshots of the system at any time, which is a result of the observability from scan-based testing. Figure 9.1A shows the necessary steps to perform a scan-based observability attack.

The hacker begins this attack by observing the position of critical registers in the scan chain. First, a known vector is placed on the primary input (PI) of the chip, and the chip is allowed to run in functional mode until the targeted register is supposed to have data in it. At this point, the chip is placed into test mode using TC, and the response in the scan chain is scanned-out. The chip is reset, and a new vector that will cause a new response only in the targeted register is placed on the PI. The chip again is run in functional mode for the specific number of cycles, and then set into test mode. The new response is scanned-out and analyzed with the previous response. This process continues until there are enough responses to analyze where in the scan chain the targeted register is positioned.

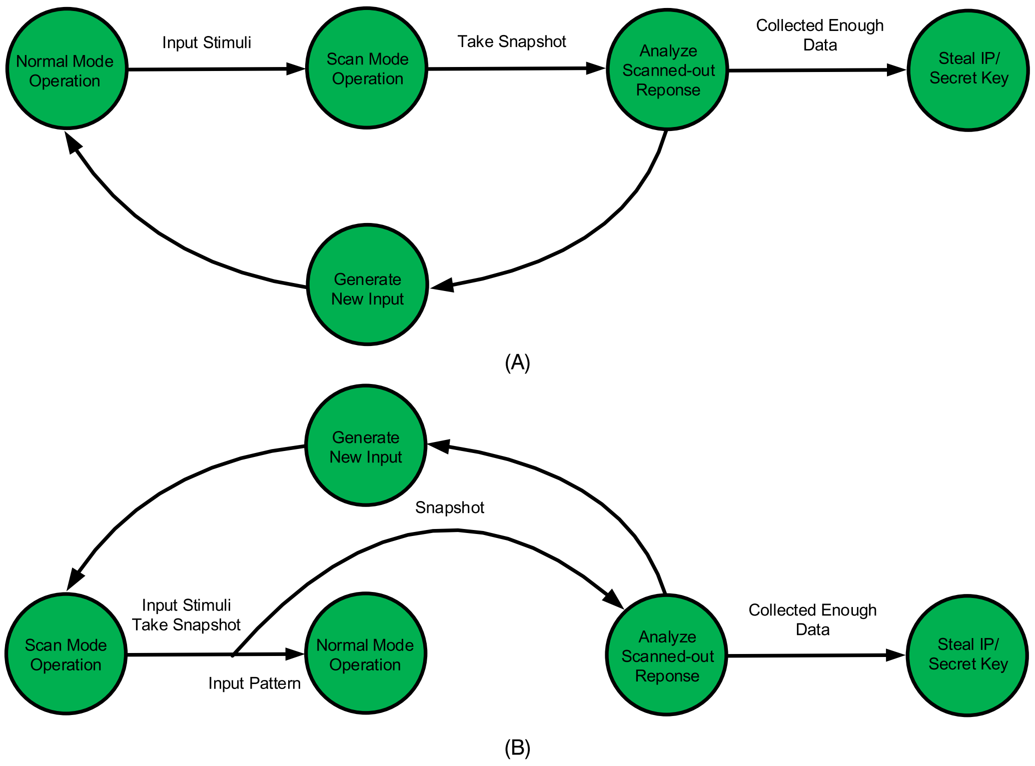

FIGURE 9.1

Summary of the steps necessary to perform successful scan-based attacks: (A) scan-based observability attack and (B) scan-based controllability/observability attack.

Once the targeted register is determined, a similar process can be used to either determine a secret key in the case of cryptochips, or determine design secrets (or IP) for a particularly innovative chip.

9.2.1.2 Scan-Based Controllability/Observability Attack

Scan-based controllability/observability attacks take a different approach to applying stimuli to the CUT, which is shown in Fig. 9.1B. Scan-based controllability/observability attacks begin by applying the stimuli directly into the scan chain as opposed to the PI. In order to mount an effective attack, the hacker must first determine the position of any critical registers, as was done for the scan-based observability attack. Once located, the hacker can load the registers with any desired data during test mode. Next, the chip can be switched to functional mode using the vector the hacker scanned-in, potentially bypassing any information security measures. Finally, the chip can be switched back to test mode to allow the hacker a level of observability, which the system's primary output (PO) would not provide otherwise.

As opposed to using a known vector to scan-in to the chain, hackers also have the opportunity to choose a random vector to induce a fault in the system. Based on the fault-injection side-channel attack

[8,9], by inducing a fault, the chip may malfunction, potentially revealing critical data. The scan chain becomes an easily accessible entry point for inducing a fault, and makes the attack easily repeatable. In order to protect from such side-channel attacks, additional hardware security measures must be included in the design.

9.2.2 THREAT MODELS

An attacker in the supply chain tries to use the scan chain (sometimes through JTAG [13]) to [14]:

- Steal critical information from an SoC (for example, crypto IP) [15,16].
- Violate confidentiality and integrity policies [17].
- Pirate IP design by unlocking an obfuscated IP [9,18].
- Illegally take control of the chip [19].

The scan-based noninvasive attack methods making such malicious acts possible are [14]:
Scan-facilitated differential attack: Differential attack has been proposed in [9,20]. By applying challenge pairs, running the crypto-algorithm, and comparing the responses, the key can be obtained. This attack has been facilitated by scan, due to added controllability and observability. Through switching from functional mode to test mode, the attacker can identify key flip-flops from the scan chain; then the key can be recovered through the already constructed correlation among input pairs, key flip-flops, and key [21]. Although some test mode protection techniques attempt to reset data registers when the chip is switched to test mode, test-mode-only differential attacks have recently been discussed in [22]. Furthermore, differential attacks are reported even in the presence of advanced DFT structures, that is, on-chip compression, Xmasking, and X-tolerance [22,23].
Attacks designed for specific countermeasures: In addition to the on-chip compression used in DFT structures, scan chain reordering and obfuscation have been developed as countermeasures, which can be defeated by the following attacks:

- Resetting and flushing attacks: By resetting the scan cells or flushing the scan chain with the known patterns, the fixed inverted bits [24], and modified bits [25] in the obfuscated scan chain, can be identified so that the plain text can be deciphered.
- Bit-role identification attack: For countermeasures using the key & lock scheme [1,26–29], the scanned-out responses are determined by the test authentication status. The authentication key bit-flipping would make scan out vectors differ, whereas a nonkey bit would not. This would significantly reduce the difficulty of identifying the key bits (especially for malicious users in fab or assembly).
- Combinational function recovery attack: Since the scan chains unfold the sequential logic as combinational, and directly reveal the internal state of the circuit, extracting design information from them has become easier. Thus, the device's functionality can be reverse engineered [18].

9.2.3 TEST-ORIENTED THREAT MODELS APPLICABLE TO DIFFERENT STAGES OF SUPPLY CHAIN

In the supply chain, a design goes through SoC integrator (or SoC design house), foundry, assembly/test facilities, original equipment manufacturer (OEM), electronics manufacturing service (EMS) provider, distributor, and end customer [30]. Thus, it is worthy to analyze the security risks due to scan-based attacks at each stage [31].

- IC Integrator: Here, the IC or SoC integrator refers to the members belonging to IC design house (or IP owner), who integrates custom logic, 3PIPs, and peripherals macros to form the whole IC. In other words, an IC integrator can be either a design, verification, DFT, or even a firmware engineer within the IC design house. Hence, the threat model is that, during integration, the confidentiality and integrity policies can be violated by a malicious IC integrator. For example, a malicious frontend RTL Designer, who is eligible to access the RTL code, can leak function of IP cores.
- Foundry: Before wafer slicing, all individual dies are tested on wafer by applying scan-based test patterns, and scanning out test responses to ATE. Likewise, a malicious foundry can pirate IP design utilizing the unobfuscated full-scan. Furthermore, some of the existing secure scan solutions have been proven to be insecure. Thus, sensitive information (such as keys and seeds) for the secure scan can be the target of a malicious foundry.
- Assembly/Test Facilities: As described in [32], for many cases, the structural test carried by foundry on wafer is enough for quality assurance. However, for some IC design houses making chip for industrial (that is, automotive) and military applications, the packaged ICs are required to be thoroughly tested after packaging, which extends the scan accessibility to assembly/test, and even OEM/EMS. Hence, the risks for scan-based attacks at assembly/test facilities are similar to those at foundry.
- OEM/EMS: At OEM or EMS, the printed circuit board (PCB) (equipment carrying IC) is developed; at the same time, the IC is programmed/configured to work with the system. To keep in-field failure analysis ability, usually scan chains are accessed through JTAG interface [3]. The keys and seeds for crypto IP (such as AES, DES, or RSA) loaded into IC in these stages can be leaked. Moreover, programs illegally utilizing scan chain to control the IC can also be loaded into the device at this stage.
- Distributor: The distributor loads customized programs into IC for different customers, as required. The risks in this stage are similar to OEM/EMS.
- End Customer: Malicious end customers (or hackers) may be interested in the sensitive information stored in the IC (that is, device configuration bits). The scan chain accessibility makes crypto IP (such as AES, DES, or RSA) vulnerable. Scan chains can also be used by attackers to illegally control the IC.

In summary, there are many opportunities for malicious entities within the supply chain to exploit the scan chains. Figure 9.2 shows the attacks each malicious entity in the supply chain can potentially carry out. Therefore, protecting IPs against scan-based attacks throughout the supply chain is necessary.

9.2.4 DYNAMICALLY OBFUSCATED SCAN (DOS)

The authors in [31] propose a design and test methodology against scan-based attacks throughout the supply chain, which includes a dynamically obfuscated scan for protecting IP/ICs. By perturbing test patterns/responses, and protecting the obfuscation key, the proposed architecture is proven to be robust

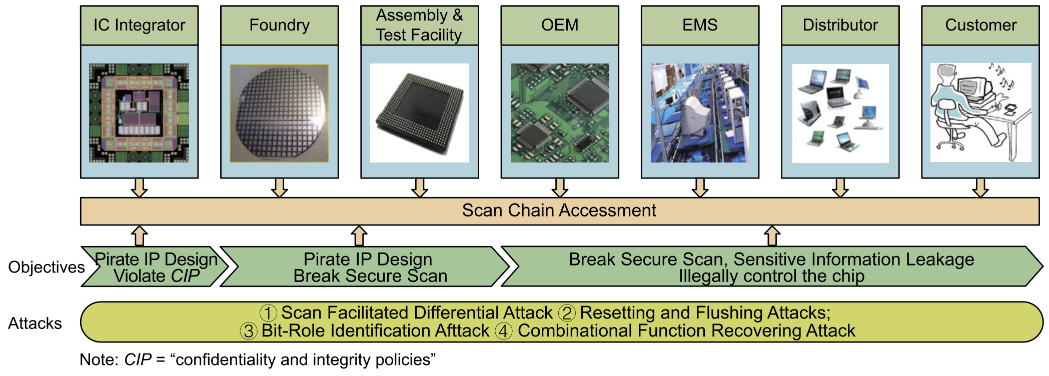
IC Integrator
Foundry
Assembly & Test Facility
OEM
EMS
Distributor
Customer
Scan Chain Accessment
Objectives
Pirate IP Design Violate CIP
Pirate IP Design Break Secure Scan
Break Secure Scan, Sensitive Information Leakage Illegally control the chip
Attacks
① Scan Facilitated Differential Attack ② Resetting and Flushing Attacks;
③ Bit-Role Identification Afttack ④ Combinational Function Recovering Attack
Note: CIP = "confidentiality and integrity policies"

FIGURE 9.2

Attacker's objectives throughout IC supply chain.

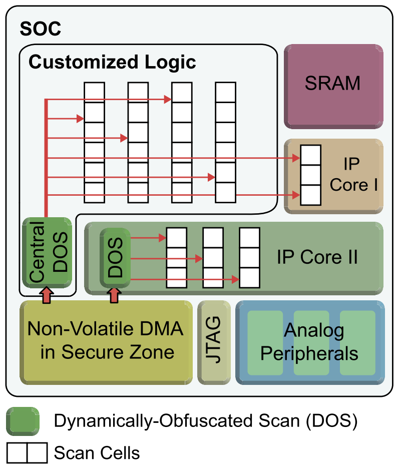

FIGURE 9.3

Overview of an SoC protected by DOS architecture.

against existing noninvasive scan-based attacks, and can protect all scan data from attackers in foundry, assembly, and system development, without compromising the testability.

An overview of the proposed secure scan in a SoC is shown in Fig. 9.3. The DOS architecture reads Control Vector from nonvolatile direct memory access (DMA) in secure zone, and provides protection to scan chains. The DOS architecture has capacity and flexibility to provide protection for IP owner, and IC integrator. According to Fig. 9.3, IP owner can either integrate one DOS into IP, as IP core II, or share the central DOS belonging to the customized logic, as IP core I.

9.2.4.1 DOS Architecture

As illustrated in Fig. 9.4, the DOS architecture is composed of a linear feedback shift register (LFSR), a Shadow chain with XOR gates, and a Control Unit.

LFSR: The LFSR is adopted to generate a λ-bit obfuscation key (λ is the length of scan chains), which is used to scramble scan in/out vectors as shown in Fig. 9.4. The obfuscation key is protected through the AND gates of the Shadow chain. The LFSR is driven by the control unit, and changes its output only when the obfuscation key update is required. It should be noted that for LFSR, a seed with all zeros is illegal when using an XOR feedback; the LFSR would remain in locked-up state and continue providing all zero obfuscation Key. Therefore, the scan chains cannot be obfuscated. To avoid the above scenario, it is suggested that some of XOR gates in LFSR would be replaced with XNOR gates.

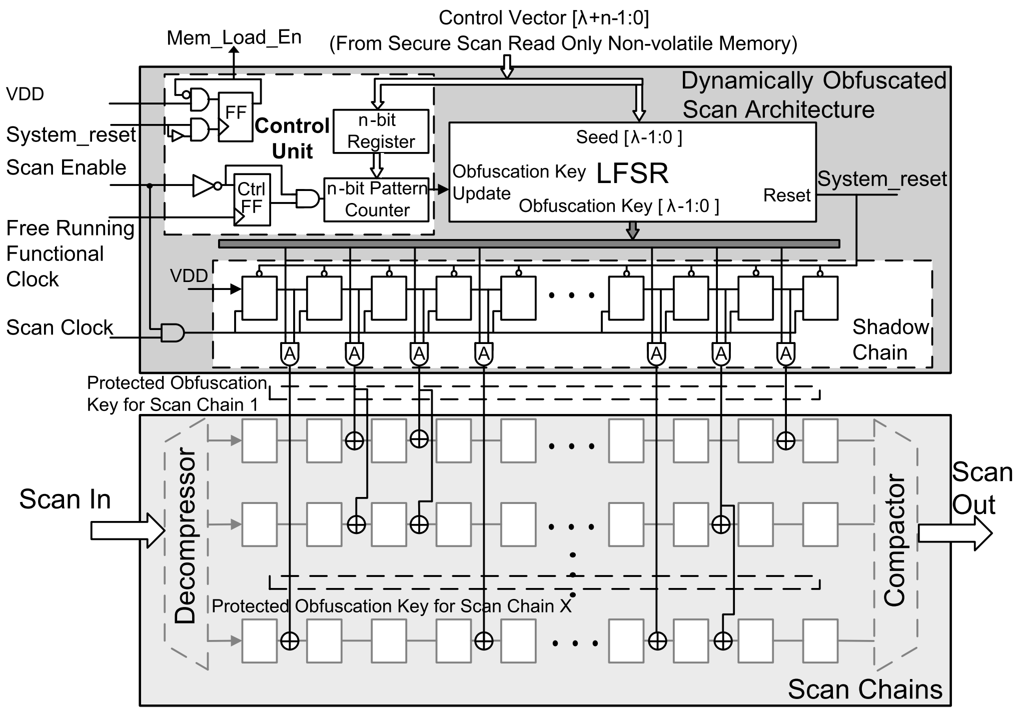

FIGURE 9.4

Detailed architecture of the DOS.

Shadow chain and XOR gates: As shown in Fig. 9.4, the input of the Shadow chain is the λ-bit obfuscation key generated by the LFSR, whereas the outputs are $k[\lambda \times \alpha]$ bit protected obfuscation keys, where α is the permutation rate (the percentage of bits permuted inside each DFT scan chain), and k is the number of scan chains [31]. The Shadow chain is designed for propagating the obfuscation key at the i_{th} scan cell along the scan chain, when the i_{th} scan clock comes. Therefore, the Shadow chain is able to – (i) protect the obfuscation key from being leaked through resetting attack, (ii) prevent any unscrambled data from being scanned out, and (iii) prevent adversaries from scanning in values intentionally, and at the same time, make no impact on structural and chain tests.

It can be seen that the Shadow chain is designed as a cascade of λ flip-flops, which is driven by the scan clock gated by scan enable signal. As shown in Fig. 9.4, the data input of its first flip-flop is connected to Vdd. The XOR gate inserted after the i_{th} scan cell of scan chain X is controlled by the output of the i_{th} flip-flop of the Shadow chain through a type A AND gate. As shown in Fig. 9.4, the type A AND gates of DOS are the AND gates connecting the scan cells within Shadow chain, the obfuscation key bits generated by the LFSR, and the XOR gates inserted into the scan chain, which actually are used to gate the individual obfuscation key bits by the scan cells of Shadow chain.

After reset, as the scan clock forces the flip-flop along the Shadow chain to logic “1”, one by one, only when the last flip-flop in the Shadow chain becomes logic “1” at the λ_{th} scan clock, the scrambled response starts to show up at the scan output. At the same time, the Shadow chain’s i_{th} flip-flop starts to obfuscate the i_{th} flip-flop of Scan chain X at the i_{th} scan clock, which prevents the attacker from scanning in any intended values. Therefore, if the attacker keeps flushing the scan chain, an original or inverted scan in sequence shows up at the scan output after λ bits of zeros. Furthermore, as the protected obfuscation key has settled down after the whole chain is scanned, the Shadow chain does not impact the DFT launching or capturing process, such as when applying stuck-at or transition delay faults. The scrambled test responses are then scanned out. The Shadow chain should be synchronously reset with the LFSR at any reset event. As all of the DFT scan chains are scanned synchronously, and the length of the scan chain is usually short with on-chip compression, the architecture only needs a single short Shadow chain, which has low area penalty. Furthermore, as the Shadow chain is plugged into the scan chains, it is not bypassable.

Control unit: The control unit, as shown in Fig. 9.4, is designed to control memory loading and LFSR activities. It is composed of a small n-bit register, a n-bit pattern counter, and a control flip-flop. During system initialization, a control vector is loaded from the secure scan read-only nonvolatile DMA, which includes a λ-bit seed for the LFSR, an n-bit value p (determining the obfuscation key update frequency), and the maximum obfuscation key update count. The control unit of DOS generates the Mem_Load_En signal. This signal allows the control vector of DOS to be loaded from DMA, once the system resets. The control vector is determined by the IC designer. As a part of system firmware, the control vector is stored into read only nonvolatile memory located in secure zone with DMA, which satisfies: 1) immediate control vector accessing: the control vector is automatically loaded into DOS at powering up, which can be guaranteed by hard coding the control vector address in DMA; 2) limited readability: the control vector can only be read by DOS, which can be satisfied by using the handshaking signal Mem_Load_En (in Fig. 9.4) generated by DOS, as an input of the DMA address accessing authorization. Additionally, as shown in Fig. 9.4, during scan, Mem_Load_En also ensures that the control vector can only be read after the reset event. Furthermore, the memory encryption technique such as [45], which allows the control vector to be stored into the nonvolatile memory in an encrypted manner, is recommended, but not required. When the pattern counter value reaches p, the obfuscation key is updated. Otherwise, the obfuscation key is locked. As sometimes the set of test patterns cannot be delivered at once, this feature offers the IP owner flexibility to dynamically add new patterns with updated obfuscation key.

9.2.4.2 The Obfuscation Flow of DOS Architecture

Based on the three major components discussed above, the obfuscation flow of the proposed design is summarized below. In Step 1, during system initialization, a control vector is loaded into the LFSR and the control unit, which is composed of a seed for the LFSR and a vector to determine the obfuscation key update frequency. In Step 2, the obfuscation key is generated at the output of the LFSR, which is driven by the control unit. In Step 3, during the first λ scan clocks after reset, the protected obfuscation key is generated bit by bit based on the Shadow chain and the obfuscation key. In Step 4, at the λ_{th} scan clock, the protected obfuscation key settles down. Then, all the test patterns and responses scrambles as a result of the protected obfuscation key.

Figure 9.5 shows the timing diagram of DOS architecture. It can be seen that the obfuscation key is generated at the output of the LFSR in waveform (C), and is dynamically changed every p pattern (p is configurable by the IP owner), when the obfuscation key update is enabled and generated by the control unit (waveforms (C) and (F)). As presented before, after reset, the protected obfuscation key for scan chain X, generated by the Shadow chain, is updated bit by bit with the scan clock, and settles down at the λ_{th} scan clock (waveform (G)). During the period of the first λ scan clocks, the scan out is locked to “0”. Once the λ_{th} scan clock comes, the scan out starts to output obfuscated responses (waveform (H)).

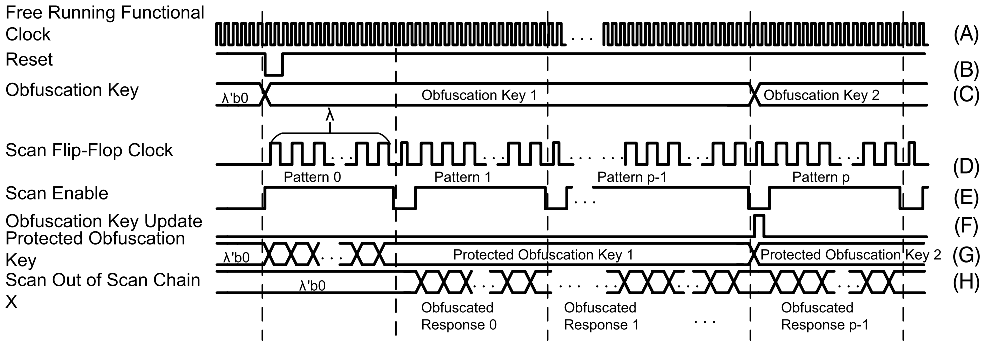

FIGURE 9.5

The timing diagrams for DOS architecture.

9.2.5 LOW-COST SECURE SCAN (LCSS)

Figure 9.6 presents the LCSS solution. LCSS is implemented by inserting dummy flip-flops into the scan chains; it inserts the key into the test patterns with respect to the position of the dummy flip-flops in the chains. By doing so, it verifies that all vectors scanned-in come from an authorized user, and the correct response can be safely scanned-out after functional mode operation. If the correct key is not integrated into the vector, an unpredictable response is scanned-out, making analysis very difficult for an attacker. By using an unpredictable response, a hacker would not be able to immediately realize that their intrusion has been detected, as could be discerned if the CUT were to immediately reset [33].

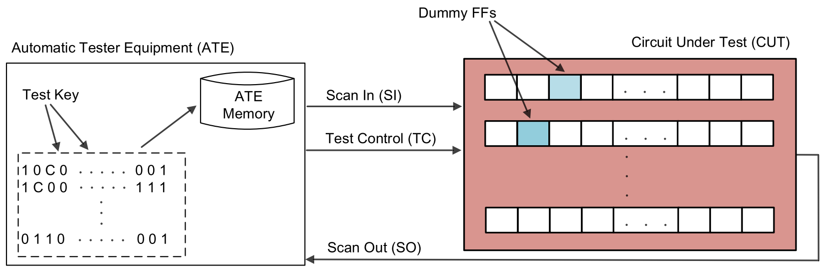

FIGURE 9.6

Sample test pattern stored in ATE with test key bits located in pattern with respect to the location of the dummy flip-flops in the CUT.

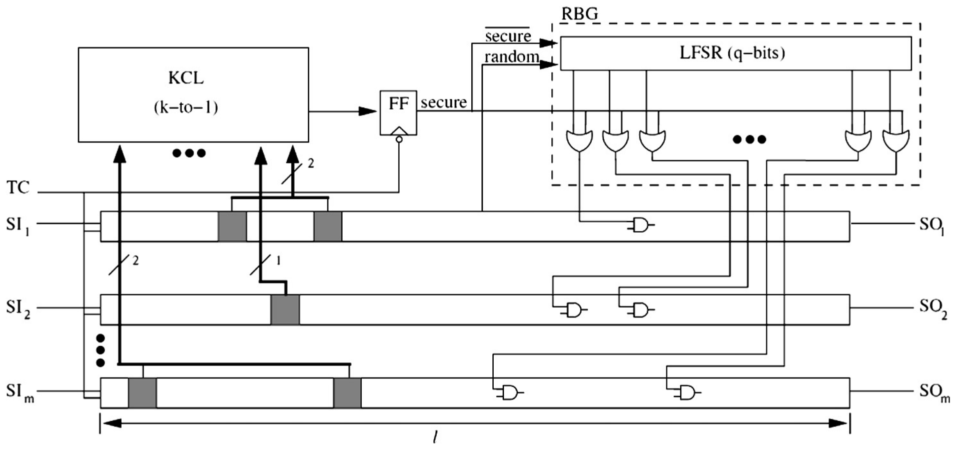

FIGURE 9.7

LCSS architecture.

9.2.5.1 LCSS Architecture

The state of the scan chain is dependent on the test key that is integrated into all test vectors. There are two possible states for the chain: secure and insecure. By integrating the key, all vectors scanned-in can be verified to be from a trustworthy source (secure). Without a correct key integrated into the test vector, when scanning in a new vector and scanning out the response, the response will be randomly altered to prevent reverse engineering of sensitive data that is being stored in registers (insecure). By altering the response scanned-out of the chain, both the scan-based observability and scan-based controllability/observability attacks are prevented, as any attempt to correlate the responses from various inputs will prove unsuccessful due to the random altering of the data.

The LCSS architecture is shown in Fig. 9.7, and a more detailed look at a secure scan chain is provided in Fig. 9.8. In order to use the same key for every test vector, dummy flip-flops (dFFs) are inserted and used as test key registers. Each dFF is designed similar to a scan cell, except that there

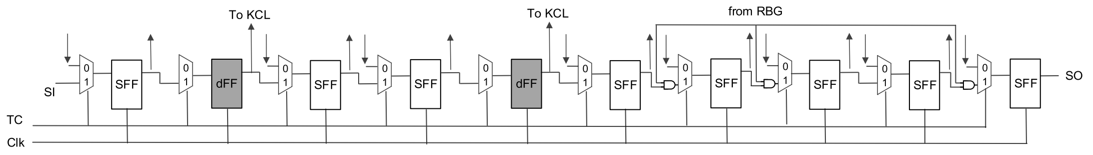

FIGURE 9.8

An example of the LCSS with integrated dummy flip-flops and random response network.

is no connection to a combinational block. The number of dFFs included in the scan chain depends on the level of security the designer would like to include, since the number of dFFs determines the size of the test key. When implementing LCSS for multiple-scan design, the test key is inserted into the scan chain before it is broken into multiple scan chains. This ensures that the key can be randomly distributed throughout the many scan chains without needing to have a constant number of key registers in each chain.

All dFFs are concurrently checked by the Key Checking Logic (KCL), which is made of a block of combinational logic. The k-input block, where k is the total number of dFFs in the scan design (length of the test key), has a fan-out of a single FF (KCL-FF), which is negative edge-sensitive to test control (TC). TC is sometime called test enable (TE); it enables test mode (TC $=$ 1 enables test mode, whereas TC $=$ 0 switches scan flip-flops to functional mode). As the CUT switches from test mode to functional mode (TC falls), the FF clocks in the output of the key checking logic. The KCL-FF is then used to inform the remainder of the secure design of the current secure or insecure state of the vector in the scan chain.

There is potential for the KCL to be implemented using a variety of more secure options, since the KCL will, essentially, be the same for all chips fabricated using the same design. One such option would be to implement a post-fabrication configurable KCL. This KCL implementation would allow different test keys from chip to chip, and would prevent the possibility of determining a single key to compromise all chips of the same design. However, as each of the devices has a different test key, essentially a new test pattern set would need to be generated for each individual chip. This would either create a significant increase in test time, or require a new secure test protocol that would need the tester to insert the test key into the pattern dynamically.

The third component of the LCSS architecture ensures the random response in the scan chain, when the test key fails to be verified by the KCL. The output of the KCL-FF fans out to an array of q 2-input OR gates. The second input of each OR gate comes from a q-bit LFSR that has been randomly seeded using one of a variety of options including, but not limited to, the value already present at reset, a random signal from an FF in the scan chain as shown in Fig. 9.7, or a random signal from the output of a separate random number generator [34]. The former option provides the least amount of overhead, but potentially the least secure, whereas the latter has the most security, but also the most overhead. By also using a secure signal to the LFSR, the LFSR seed can continually be changed by an additional random source. Together, the LFSR and OR gate array make up the random bit generator (RBG). The RBG output is used as input to the random response network (RRN) that has also been inserted into the scan chain. The RRN can be made of both AND and OR gates to equalize the random transitions, and prevent the random response from being all zeros, or all ones. The optimal choice for randomness would be to use XOR gates, but as XORs add more delay, the design choice was to use AND, and OR gates. As the dFFs are used to check the test key, dFFs must be placed before any gates of the RRN in the scan chain, as shown in Fig. 9.8. If this principle is not applied, any key information that is trying to pass a gate of the RRN in the scan chain may potentially get altered, either preventing the test key from ever being verified, or even randomly changing a value to the correct key.

Normal mode operation of the CUT is unaffected by the addition of the LCSS design, since the dFFs are only used for testing, and security purposes and are not connected to the original design.

9.2.5.2 LCSS Test Flow

The low-cost secure scan design deviates very little from current scan test flow. As the security of the scan chain is ensured by integrating a test key into the test vectors themselves, no additional pins are necessary to use LCSS.

After a system reset, and TC has been enabled for the first time, the secure scan design begins in an insecure state, causing any data in the scan chain to be modified as it passes through each RRN gate in the chain. In order to begin the testing process, the secure scan chain(s) must be initialized with the test key in order to set the output of the KCL-FF to 1. Only the test key is required to be in this initialization vector, since any other data beyond the first RRN gate will most likely be modified. During this time the KCL will constantly check the dFFs for a correct key. After the initialization vector has been scanned-in, the CUT must be switched to functional mode for one clock in order to allow the KCL-FF to capture the result from the KCL. If the KCL verifies the key stored in the dFFs, the KCL-FF is set to 1 and propagates the signal to the RRN, which becomes transparent for the next round of testing, allowing the new vector to be scanned-in without alteration.

Testing can continue as normal, once the initialization process has been finished. However, the chain can return to insecure mode at any time during scan testing, if the correct test key is not present in all subsequent test vectors, requiring the k-bit key to be in all test patterns. Should that occur, the RRN will again affect the response in the scan chain, and the initialization process must again be performed in order to resume a predictable testing process.

9.2.6 LOCK & KEY

Lock & Key solution was developed to neutralize the potential for scan-based side-channel attacks [1]. The Lock & Key technique provides a flexible security strategy to modern designs, without significant changes to scan structure used in practice. Using this technique, the scan chains in a system on chip (SoC) are divided into smaller subchains. With the inclusion of a test security controller, access to subchains are randomized when being accessed by an unauthorized user. Random access reduces repeatability and predictability, making reverse engineering more difficult. Without proper authorization, an attacker would need to unveil several layers of security before gaining proper access to the scan chain in order to exploit it. The proposed Lock & Key technique is design-independent, while maintaining a relatively low area overhead.

9.2.6.1 Lock & Key Architecture

The Lock & Key technique can be used to secure both single- and multiple-scan designs. For either case, the scan chain is divided into smaller subchains of equal length. Test vectors are not sequentially shifted into each subchain, but rather an LFSR performs a pseudorandom selection of a subchain to be filled. Figure 9.9 shows the architecture for the Lock & Key technique for a single-scan design. This technique provides a trade-off between testability and security, since the LFSR, during insecure mode, will protect the scan chain but also requires a nonsequential scan chain access when the user has been verified.

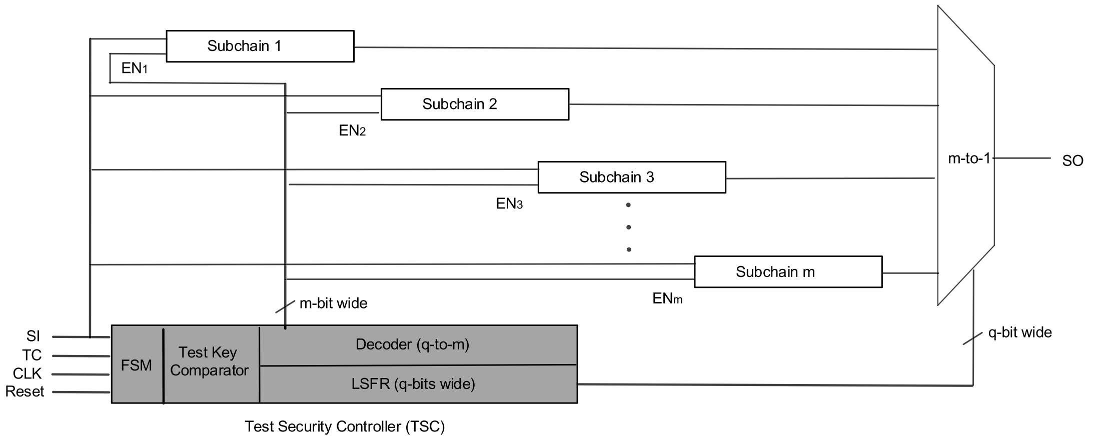

FIGURE 9.9

Architecture of Lock & Key security measure.

This method prevents scan chain manipulation without the presence of a valid test key. This is ensured by the test security controller (TSC), which consists of four main components: a finite state machine (FSM), test key comparator, LFSR, and decoder. There are two states the TSC can be in, namely, secure and insecure modes. The secure mode signifies that a trusted user is accessing the scan chain, so the TSC will select subchains in a predictable nonsequential order. The insecure mode signifies a state, where the user attempting to access the scan chain is considered untrustworthy until deemed otherwise with a correct test key. Unless the test key is entered and confirmed to be correct, the TSC will unpredictably select subchains using the LFSR to scan in (SI) and scan out (SO), presenting the user with false information about the scan chain.

A test engineer must perform two steps before sending in a test vector into the scan chain for the first time. After enabling test control for the first time after a system reset occurs, the TSC controls all functions of the subchains until an authorized or unauthorized party is detected. A test key must be the first pattern fed into the TSC. During the first k cycles after TC has been enabled, the first k-bits applied to the SI will be serially passed to the test key comparator, and checked. After the k cycles, the FSM will receive the result. If the key matches the test key stored in a tamper proof nonvolatile memory, the secure signal will be raised, allowing the TSC to begin operation in a secure mode, in which it will remain until the CUT is reset. If the secure signal remains low, operation in the insecure mode will resume. If the test key passes and the TSC enters secure mode, the test engineer then has the ability to seed the LFSR with a known seed in order to predict the order of the LFSR's selection of subchains. Otherwise, the LFSR will work with the unpredictable random seed created in the LFSR right after a system reset.

With the LFSR seeded, formation of the scan chain can begin. Using a decoder to interface between the LFSR and the subchains, the TSC uses a one-hot output method to enable one subchain at a time to read from SI. The output of the LFSR is also directly connected to the multiplexer selector bits' allowance of the data from the subchain to pass to SO. Assuming the length of each subchain is l-bit long, after l clock cycle, the LFSR will shift to a new value, and the decoder will disable the currently active subchain, and select a new subchain to read from SI. After $n_{\text{dff}} = l \times m$ cycles, where m is the number of subchains, l is the length of subchain, and n_{dff} is the length of the scan chain, the full length of the scan chain is initialized with the first test vector. TC can again be set to zero to place the CUT into normal mode for one cycle to allow the pattern to propagate and capture the response back into the scan chain. When the CUT has returned to test mode, a new test vector is scanned into the subchains, while scanning out the response.

Since test key verification is a one time startup check, a failed test key causes the TSC to remain in an insecure mode until the CUT is reset. This essentially locks the scan chain from being used correctly for the duration of the testing process. This locking mechanism is also fairly transparent to a hacker, since without prior knowledge of the security scheme, the chip would appear to be working as it should, while still giving the hacker false data.

9.2.6.2 Design of Lock & Key

The Lock & Key technique depends on the design of the TSC, which is composed of four components. A finite state machine (FSM) controls the current mode of the TSC; the test key comparator is only used when TC is enabled for the first time, returning a secure or insecure result; the LFSR selects a single subchain during scan operation, and controls the output multiplexer; and the decoder translates the output of the LFSR into a one-hot enable scheme. Figure 9.10 shows the signals passed between each of the components of the TSC. Communication between each of the components is kept to a minimum to reduce routing, and the overall size of the TSC.

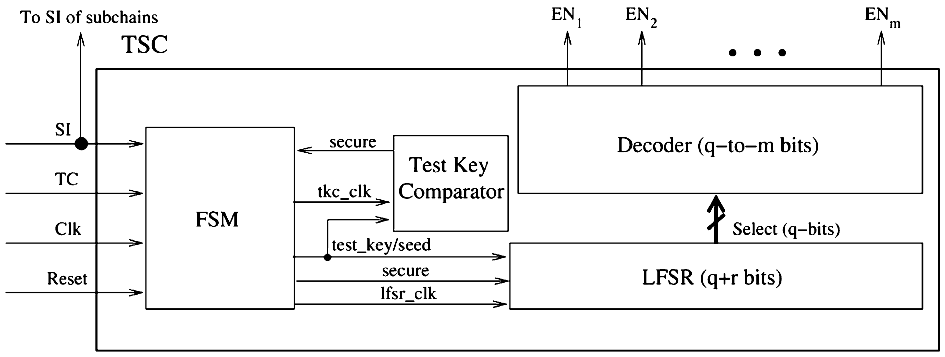

FIGURE 9.10

TSC design scheme.

The FSM block consists of simple state logic and two counters. The state logic controls the test key comparator and LFSR. The FSM also determines, according to the response of the test key comparator, whether to seed the LFSR with a vector from SI, or to use the random seed created in the LFSR by the system reset. The random seed can be created many different ways, including using a true random number generator (TRNG). The first counter used in FSM block was a $\log_2(q)$ counter, which is used only for seeding the LFSR, where q is the length of the LFSR. The second counter is a $\log_2(l)$ counter used for clocking the LFSR after l cycles, shifting the contents of the LFSR to enable a new subchain.

The test key comparator is used once, only after the system has been reset and put into test mode for the first time. In order to keep the comparator small, and since the test key from SI is read serially, each bit is serially checked against the key being stored on the chip in a secure memory. As each bit is compared, an FF stores the running result, which is eventually read by the FSM. After k cycles, the final result is read by the FSM determining whether the TSC will run in a secure mode, or continue in an insecure mode.

When designing the Lock & Key technique, the goal is to have the ability to ensure security of the scan chains, while maintaining simplicity and design independence. To prevent the decoder from becoming too complex, an LFSR with a primitive polynomial configuration allows the selection of $m = 2^q - 1$ subchains, where q is the size of the LFSR in secure mode. Using a primitive polynomial allows the selection of all subchains once, and only once, during a test round. If a nonprimitive polynomial configuration is used, unless additional logic is included, some subchains may be selected more than once, or never selected at all. Using the q bits from the LFSR, the decoder enables one of m outputs leaving the others at zero. Since there is at least one primitive polynomial for all values of q, the LFSR is guaranteed to choose each subchain once, before repeating for any length of the LFSR [35].

The number of FFs in the design before scan insertion does not necessarily need to be evenly divisible by m. There are two possibilities to resolve this issue. The first is the inclusion of dummy FFs (in form of test points: control test point and observe test point), which has become a common practice when dealing with delay testing [35]. The total number of FFs, n, and the total number of dummy FFs, n_{dFF}, needed is noted as follows:

$$n_{dFF} = \begin{cases} 0 & \text{when } (n \mod m) = 0, \\ m - (n \mod m) & \text{otherwise.} \end{cases} \tag{9.1}$$

The second option would be to pad portions of the test pattern that are related to the shorter subchains. This would immediately shift out any dummy values at the beginning of the pattern, and would have no effect on the functional operation of the CUT. This option requires less design effort, since it does not use additional logic, but does add overhead to the test pattern. However, due to test compression techniques, the overhead would be minimal, since the dummy values can be set to values that maximize compression.

The choice of a primitive polynomial significantly simplifies the design of the decoder. The decoder can directly translate the output of the LFSR into a run of zeros, and choose one to directly control each subchain. This method not only shortens design time, but also reduces the area overhead of the TSC as a whole, since additional logic is not needed to ensure all subchains are selected once during a test round.

The problem with using a primitive polynomial configured LFSR is the predictability of its behavior. If the LFSR were to remain unchanged for insecure mode operation, determining the order would not take long, since the order is always the same, only the start and end points would differ. To avoid this predictability, the LFSR configuration must be altered when set to insecure mode. By modifying the LFSR to incorporate an additional r-bits for insecure mode operation, the primitive polynomial LFSR becomes a nonprimitive polynomial LFSR. As can be seen in Fig. 9.11, the additional bits are hidden behind a multiplexer, and only become active for insecure mode operation. The interface between the LFSR and the decoder is not affected. Since the original LFSR only makes up a smaller part

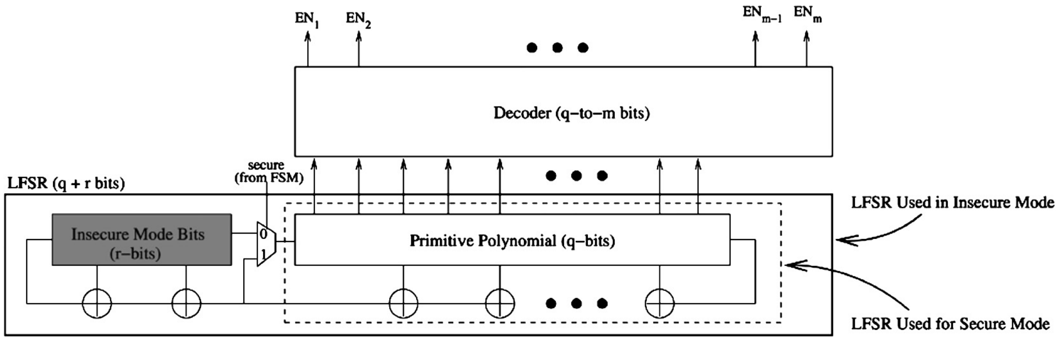
EN1
EN2
ENm−1
ENm
Decoder (q–to–m bits)
LFSR (q + r bits)
secure (from FSM)
Insecure Mode Bits (r–bits)
0
1
Primitive Polynomial (q–bits)
LFSR Used in Insecure Mode
LFSR Used for Secure Mode

FIGURE 9.11

Modifiable LFSR determined by security mode of TSC.

of the insecure mode LFSR, repetitively selecting the same subchain during one test cycle becomes possible, which results in a more complex output. A shorter periodicity is not a concern, as it was in secure mode, since all subchains do not need to be accessed, but the facade of a fully functional scan chain still exists.

9.2.7 SCAN INTERFACE ENCRYPTION

A countermeasure against scan-based side channel attacks could be done through the encryption of the scan chain content [36]. These attacks use an efficient and secure block cipher placed at each scan port to decrypt/encrypt scan patterns/responses at each scan input/output, respectively.

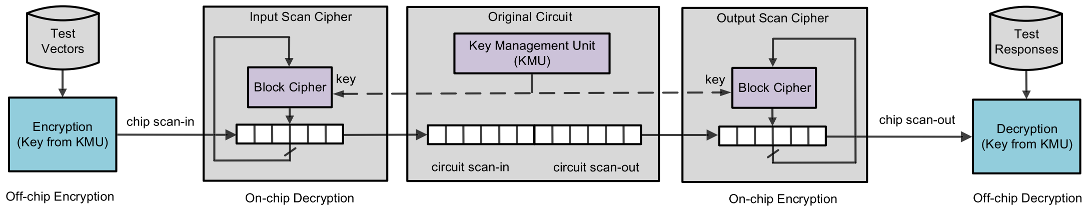

FIGURE 9.12

Scan interface encryption structure.

As illustrated in Fig. 9.12, two block ciphers are inserted into the circuit. Whereas the input scan cipher decrypts test patterns provided by ATE, the output scan cipher encrypts test response before sending back to ATE. Based on this scheme, the test flow is as follows [36]:

- Generate test patterns for CUT and calculate expected test responses;
- Off-chip encryption of the test patterns based on pre-selected (e.g., AES) encryption algorithm and secret key;
- On-the-fly decryption of the test patterns with input block cipher, then scan in patterns for CUT;

- On-the-fly encryption of the test responses with output block cipher before response extraction;
- Off-chip decryption of test responses to get the original responses and compare them with the expected ones.

9.2.8 OBFUSCATED SCAN

Secure scan architecture using test key randomization (SSTKR) was developed to address security and testability issues [37]. Specifically, SSTKR is a key-based technique to prevent an attacker from illegally obtaining critical information while using scan infrastructure. The authentication keys are generated through linear feedback shift register and inserted into test vectors.

Furthermore, test keys are embedded into test vectors in two different ways, that is, with dummy flip-flops and without dummy flip-flops. In the first case, dummy flip-flops holding the key are inserted into the scan chain to randomize scan outputs. It should be noted that all dummy flip-flops should not be connected to the combinational logic. In the second case, authentication keys are inserted into the positions of don't-care bits, generated by ATPG to reduce area overhead and test time. The structure of SSTKR for the above two cases are illustrated in Fig. 9.13 and Fig. 9.14, respectively.

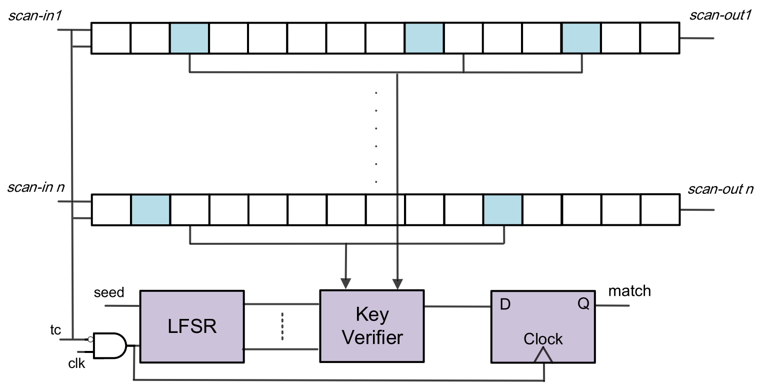

FIGURE 9.13

SSTKR architecture.

9.2.9 SCAN-CHAIN REORDERING

A secure scan tree architecture was developed to protect cryptosystem against scan-based attacks [38]. This architecture offers low area overhead compared with the traditional scan tree architecture followed by a compactor, locking, and test access port (TAP) architecture. In contrast to the normal scan tree architecture, as shown in Fig. 9.15, this architecture is based on the flipped scan tree (F-scan tree). To be exact, they adopt special flip-flops (that is, flipped FFs), in which inverter gates are added at the scan-in pin of scan flip-flop. The flipped scan tree architecture is built through normal SDFFs and flipped FFs. Since the attacker cannot identify the position of inverters, he/she is neither able to control the inputs, nor observe the outputs of the flip-flops.

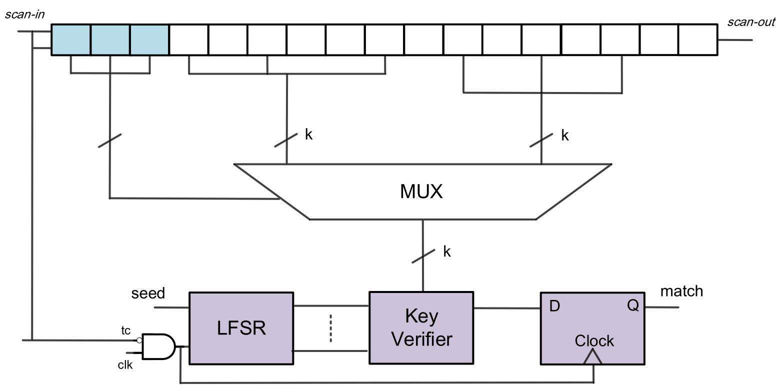

FIGURE 9.14

SSTKR architecture without using dummy FFs.

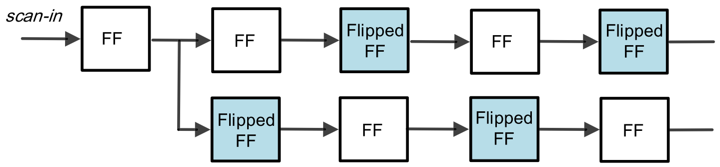

FIGURE 9.15

Flipped scan tree.

9.3 JTAG-BASED ATTACKS

Initially, the IEEE Standard 1149.1, also known as JTAG or boundary scan, was introduced in 1990 [39] to address the need for a standardized interconnect test that could be performed on a printed circuit board (PCB), or other substrate. In recent years, the accessibility of the 1149.1 standard has been extended from the chip periphery to on-chip debugging [40,41]. Based on JTAG protocol, test signals include test clock input (TCK), test mode select input (TMS), test data input (TDI), test data output (TDO), and test reset input (TRST).

The JTAG architecture, shown in Fig. 9.16, consists of the following parts: (a) the TAP controller, a 16-state finite state machine driven by TCK, TMS, and TRST signals, which generates the internal clock and control signals for the instruction register (IR); (b) the user-defined registers (UDRs), and (c) the obligatory registers (e.g., bypass register (BR), boundary scan register (BSR), and IDCODE Register). The IR is used to load and update the instructions shifted from the TDI terminal, which determines the action to be performed, and the TDR to be accessed. The instruction decoder (IDEC) is responsible for decoding the instructions as the selection signal to enable TDR between TDI and TDO. User-defined registers are introduced to access internal logic of the chip.

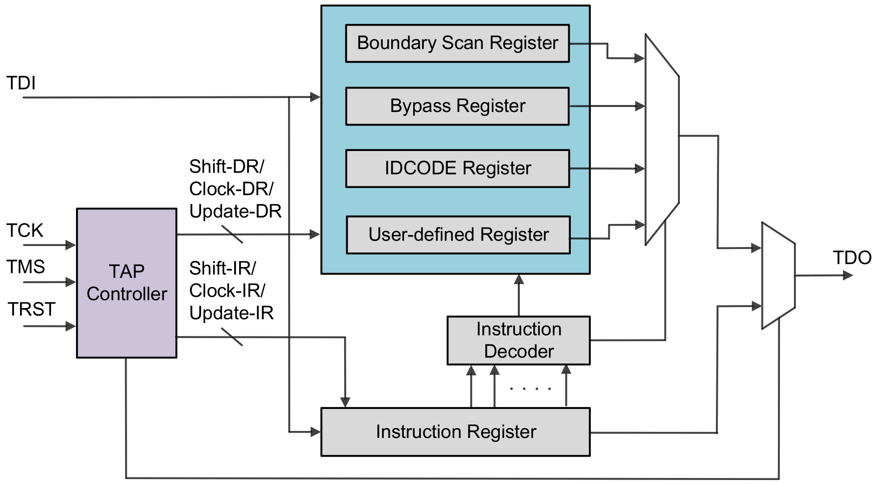

FIGURE 9.16

The JTAG architecture.

Several user defined instructions corresponding to the UDRs are introduced in the private instruction set. Each time only one public or private instruction is loaded into the IR, the corresponding data register is enabled and placed between TDI and TDO.

As can be seen from the state diagram, shown in Fig. 9.17, there are two similar branches: the instruction register (IR) scan and the data register (DR) scan. The IR branch is used to operate on the IR, whereas the DR branch is used for operations on the current TDR.

9.3.1 JTAG HACKS

IEEE 1149.1 standard was initially developed without considering security. Specifically, JTAG is designed to use scan-based testing to access internal logic of the chip and inter-chip wiring. Since JTAG is not aware of the chip's reaction to external commands, hacking a device using JTAG is technically possible. As a result, over the past decade, JTAG-based attacks have become feasible. For example, when a debugging software, such as open on-chip debugger (OpenOCD) is given control over JTAG interface, it can manipulate JTAG on the target device, and send vectors to it, which the chip interprets as valid commands [43]. Also, JTAG has been used to hack Xbox 360 to bypass DRM policies of the device [44]. In addition, JTAG is initially used in ARM11 processor to provide extensive test and debug capabilities. However, JTAG has been exploited to unlock services of cell phones (for example, to hack iphones [45]).

In [13], the authors analyze various attacks based on JTAG. The vulnerabilities are coming from the daisy-chaining topology of JTAG wiring, as shown in Fig. 9.18. They examine the potential threats in the following scenarios: 1) acquire secret data through overhearing JTAG data path; 2) acquire an embedded asset through placing test patterns to JTAG lines; 3) acquire test patterns and test responses in the daisy-chain; 4) intercept test patterns sent to other chips, and send bogus responses to the tester.

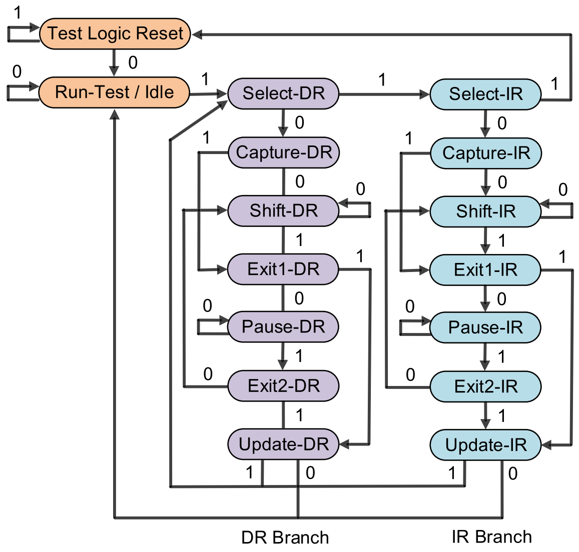

FIGURE 9.17

JTAG TAP controller state diagram.

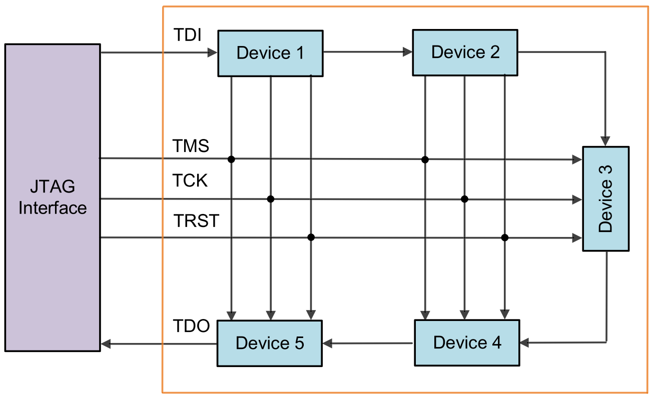

FIGURE 9.18

A JTAG system with daisy-chain topology.

9.3.2 JTAG DEFENSES

Over the past two decades, researchers in industry and academia have focused on developing countermeasures against JTAG hacks. The methodologies developed include destruction of JTAG after use, password protection of JTAG, hiding JTAG behind a system controller, and crypto JTAG with embedded keys. These methods are explained in detail in the following.

9.3.2.1 Destruction of JTAG After Use

In some cases, JTAG is only needed during debug process and manufacturing test. Once the chip is manufactured, tested and shipped to the customers, the JTAG component becomes a source of vulnerabilities. Thus, in these circumstances, engineers can disable the JTAG component before volume shipment. Typically, the JTAG fuses can be blown for security consideration. However, since the process of blowing the physical fuse is not reversible, some techniques [46] are developed to achieve finer control over the test infrastructure; hence, some test capabilities can be secured. Such designs are usually developed with more than one fuse.

9.3.2.2 Password Protection of JTAG

In [47], the authors proposed a methodology, named as "Protected JTAG", to prevent unauthenticated and unauthorized users from accessing private and confidential information. However, it still allows debugging, and testing functions to be performed by authorised users. To be specific, this scheme provides different protection levels and access modes. Protection levels define the actual protection of device, whereas access modes are the configured attributes defining default protection level, and availability of protection feature. This method uses a secure server that utilizes an ephemeral elliptic curve key pair to authenticate and authorize the user's JTAG access request. Once the authentication and authorization are successful, the device holds the authenticated and authorized state during the debugging and testing process.

9.3.2.3 Hiding JTAG Behind a System Controller

Another approach to address JTAG security issue is to hide JTAG behind a system controller [48]. To be exact, the system controller is usually implemented on a PCB, and acts as an agent to communicate with the chip to be tested. This method improves system security without modifying design. Security protocol implemented through the system controller authenticates all accesses. In addition, the authenticated users can only access the authorized parts. Not only can the system controller act as an agent between the tester and chips, it can also store test patterns, and automatically test chips when a test routine is invoked [48].

9.3.2.4 Crypto JTAG With Embedded Keys

An approach to protect JTAG relies on embedding keys with cryptoengine. There are three security components included in this approach, namely a hash function, a stream cipher, and a message authentication code [13]. They are used to authenticate the devices to be tested, encrypt test vectors and responses, and prevent unauthentic JTAG messages. Furthermore, protocols are constructed based on these security components to address the potential security threat introduced by JTAG.

9.4 HANDS-ON EXPERIMENT: JTAG ATTACK

9.4.1 OBJECTIVE

This experiment is designed to introduce the JTAG attack to the students. The experiment is designed on the HaHa platform and utilizes the JTAG infrastructure (that is, the ports and connection of the FPGA and microcontroller chip in a JTAG chain), which connects the JTAG-compliant chips in a PCB *in a chain for the purpose of test, debug, and probing after manufacture. Since there is a growing trend of using JTAG for programming FPGA and CPLD devices, their security is of utmost importance.*

9.4.2 METHOD

The first part of the experiment will allow the students to build a hacking tool that acts as a JTAG programmer. Students need to first learn how to locate the chip's ID, using instructions sent by the hacked JTAG programmer. Next, the students will use the hacked module to attack other modules in the HaHa platform.

9.4.3 LEARNING OUTCOME

By performing the specific steps of the experiments, the students will learn the level of accessibility that a hacked JTAG can provide, and the best techniques to modify a target chip and maliciously control it. They will also gain experience about the challenges and opportunities with respect to protecting hardware against JTAG attacks.

9.4.4 ADVANCED OPTIONS

Additional exploration on this topic can be done through configuring the JTAG to get the delay of the JTAG paths to create a signature.

More details about the experiment is available in the supplementary document. Please visit http://hwsecuritybook.org.

9.5 EXERCISES

9.5.1 TRUE/FALSE QUESTIONS

1. Malicious entities within the whole supply chain can perform attacks through exploiting the scan chains.
2. Testability and security do not contradict each other.
3. Scan-based DFT used by test engineer have no impact on security.
4. Destruction of JTAG after use is developed to improve security and has no impact on testability.
5. The TAP controller is a finite state machine driven by TCK, TMS, and TRST signals.

9.5.2 SHORT-ANSWER TYPE QUESTIONS

1. List the potential threats when an attacker in the supply chain performs scan-based attacks.
2. List three scan-based noninvasive attacks.
3. Explain the purpose of placing the Shadow chain into the dynamically obfuscated scan (DOS) architecture.
4. How to choose the number of dFFs inserted in the scan chain? Explain.
5. Introduce a few more scan-based attacks countermeasures besides which are discussed in this chapter.

9.5.3 LONG-ANSWER TYPE QUESTIONS

1. Explain the relationship between testability and security.
2. List several countermeasures against JTAG hacks and explain any one of them in detail.

REFERENCES

[1] J. Lee, M. Tehranipoor, C. Patel, J. Plusquellic, Securing scan design using lock and key technique, in: Defect and Fault Tolerance in VLSI Systems, 2005. DFT 2005. 20th IEEE International Symposium on, IEEE, pp. 51–62.
[2] P. Ludlow, High Noon on the Electronic Frontier: Conceptual Issues in Cyberspace, MIT Press, 1996.
[3] B. Yang, K. Wu, R. Karri, Scan-based side channel attack on dedicated hardware implementations of data encryption standard, in: Test Conference, 2004. Proceedings. ITC 2004, International, IEEE, pp. 339–344.
[4] B. Yang, K. Wu, R. Karri, Secure scan: a design-for-test architecture for crypto chips, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25 (2006) 2287–2293.
[5] R. Goering, Scan design called portal for hackers, EE Times (Oct 2004), https://www.eetimes.com/document.asp?doc_id=1151658.
[6] P. Kocher, J. Jaffe, B. Jun, Differential power analysis, in: Annual International Cryptology Conference, Springer, 1999, pp. 388–397.
[7] P.C. Kocher, Timing attacks on implementations of Diffie–Hellman, RSA, DSS, and other systems, in: Annual International Cryptology Conference, Springer, 1996, pp. 104–113.
[8] D. Boneh, R.A. DeMillo, R.J. Lipton, On the importance of checking cryptographic protocols for faults, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 1997, pp. 37–51.
[9] E. Biham, A. Shamir, Differential fault analysis of secret key cryptosystems, in: Annual International Cryptology Conference, Springer, 1997, pp. 513–525.
[10] O. Kömmerling, M.G. Kuhn, Design principles for tamper-resistant smartcard processors, Smartcard 99 (1999) 9–20.
[11] M. Renaudin, F. Bouesse, P. Proust, J. Tual, L. Sourgen, F. Germain, High security smartcards, in: Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings, vol. 1, IEEE, pp. 228–232.
[12] S.P. Skorobogatov, Semi-invasive attacks: a new approach to hardware security analysis, Technical Report UCAM-CL-TR-630, University of Cambridge Computer Laboratory, 2005.
[13] K. Rosenfeld, R. Karri, Attacks and defenses for JTAG, IEEE Design & Test of Computers 27 (2010).
[14] D. Zhang, M. He, X. Wang, M. Tehranipoor, Dynamically obfuscated scan for protecting IPs against scan-based attacks throughout supply chain, in: VLSI Test Symposium (VTS), 2017 IEEE 35th, IEEE, pp. 1–6.
[15] D. Mukhopadhyay, S. Banerjee, D. RoyChowdhury, B.B. Bhattacharya, Cryptoscan: a secured scan chain architecture, in: Test Symposium, 2005. Proceedings. 14th Asian, IEEE, pp. 348–353.
[16] R. Nara, K. Satoh, M. Yanagisawa, T. Ohtsuki, N. Togawa, Scan-based side-channel attack against RSA cryptosystems using scan signatures, IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences 93 (2010) 2481–2489.
[17] G.K. Contreras, A. Nahiyan, S. Bhunia, D. Forte, M. Tehranipoor, Security vulnerability analysis of design-for-test exploits for asset protection in SoCs, in: Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific, IEEE, pp. 617–622.
[18] L. Azriel, R. Ginosar, A. Mendelson, Exploiting the scan side channel for reverse engineering of a VLSI device, Technion, Israel Institute of Technology, 2016, Tech. Rep. CCIT Report 897.
[19] D. Hely, M.-L. Flottes, F. Bancel, B. Rouzeyre, N. Berard, M. Renovell, Scan design and secure chip, in: IOLTS, vol. 4, pp. 219–224.
[20] S.P. Skorobogatov, R.J. Anderson, Optical fault induction attacks, in: International Workshop on Cryptographic Hardware and Embedded Systems, Springer, 2002, pp. 2–12.
[21] J.D. Rolt, G.D. Natale, M.-L. Flottes, B. Rouzeyre, A novel differential scan attack on advanced DFT structures, ACM Transactions on Design Automation of Electronic Systems (TODAES) 18 (2013) 58.
[22] S.M. Saeed, S.S. Ali, O. Sinanoglu, R. Karri, Test-mode-only scan attack and countermeasure for contemporary scan architectures, in: Test Conference (ITC), 2014 IEEE International, IEEE, pp. 1–8.

[23] A. Das, B. Ege, S. Ghosh, L. Batina, I. Verbauwhede, Security analysis of industrial test compression schemes, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 32 (2013) 1966–1977.

[24] G. Sengar, D. Mukhopadhyay, D.R. Chowdhury, Secured flipped scan-chain model for crypto-architecture, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 26 (11) (2007) 2080–2084.

[25] Y. Atobe, Y. Shi, M. Yanagisawa, N. Togawa, Dynamically changeable secure scan architecture against scan-based side channel attack, in: SoC Des. Conf. ISOCC Int., 2012, pp. 155–158.

[26] J. Lee, M. Tebranipoor, J. Plusquellic, A low-cost solution for protecting IPs against scan-based side-channel attacks, in: Proc. VLSI Test Symposium (VTS), 2006, pp. 42–47.

[27] M.A. Razzaq, V. Singh, A. Singh, SSTKR: secure and testable scan design through test key randomization, in: Proc. of Asian Test Symposium (ATS), 2011, pp. 60–65.

[28] S. Paul, R.S. Chakraborty, S. Bhunia, Vim-scan: a low overhead scan design approach for protection of secret key in scan-based secure chips, in: Proc. VLSI Test Symposium (VTS), 2007.

[29] J. Lee, M. Tehranipoor, C. Patel, J. Plusquellic, Securing designs against scan-based side-channel attacks, IEEE transactions on dependable and secure computing 4 (4) (2007) 325–336.

[30] J.P. Skudlarek, T. Katsioulas, M. Chen, A platform solution for secure supply-chain and chip life-cycle management, Computer 49 (2016) 28–34.

[31] X. Wang, D. Zhang, M. He, D. Su, M. Tehranipoor, Secure scan and test using obfuscation throughout supply chain, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 37 (9) (2017) 1867–1880.

[32] M. Tehranipoor, C. Wang, Introduction to Hardware Security and Trust, Springer Science & Business Media, 2011.

[33] D. Hely, F. Bancel, M.-L. Flottes, B. Rouzeyre, Test control for secure scan designs, in: Test Symposium, 2005. European, IEEE, pp. 190–195.

[34] B. Jun, P. Kocher, The Intel random number generator, Cryptography Research Inc., 1999, white paper.

[35] M. Bushnell, V. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits, vol. 17, Springer Science & Business Media, 2004.

[36] M. Da Silva, M.-l. Flottes, G. Di Natale, B. Rouzeyre, P. Prinetto, M. Restifo, Scan chain encryption for the test, diagnosis and debug of secure circuits, in: Test Symposium (ETS), 2017 22nd IEEE, IEEE, pp. 1–6.

[37] M.A. Razzaq, V. Singh, A. Singh, SSTKR: secure and testable scan design through test key randomization, in: Test Symposium (ATS), 2011 20th Asian, IEEE, pp. 60–65.

[38] G. Sengar, D. Mukhopadhyay, D.R. Chowdhury, An efficient approach to develop secure scan tree for crypto-hardware, in: Advanced Computing and Communications, 2007. ADCOM 2007. International Conference on, IEEE, pp. 21–26.

[39] C. Maunder, Standard test access port and boundary-scan architecture, IEEE Std 1149.1-1993a, 1993.

[40] J. Rearick, B. Eklow, K. Posse, A. Crouch, B. Bennetts, IJTAG (Internal JTAG): A step toward a DFT standard, in: Test Conference, 2005. Proceedings. ITC 2005, IEEE International, IEEE, 8 pp.

[41] M.T. He, M. Tehranipoor, An access mechanism for embedded sensors in modern SoCs, Journal of Electronic Testing 33 (2017) 397–413.

[42] IEEE standard test access port and boundary-scan architecture: Approved February 15, 1990, IEEE Standards Board; Approved June 17, 1990, American National Standards Institute, IEEE, 1990.

[43] JTAG explained (finally!): Why "IoT", software security engineers, and manufacturers should care, http://blog.senr.io/blog/jtag-explained, Sept. 2016.

[44] Free60 SMC Hack, http://www.free60.org/SMC_Hack, Jan. 2014.

[45] L. Greenemeier, iPhone hacks annoy AT&T but are unlikely to bruise apple, Scientific American (2007).

[46] L. Sourgen, Security locks for integrated circuit, US Patent 5,101,121, 1992.

[47] R.F. Buskey, B.B. Frosik, Protected JTAG, in: Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on, IEEE, 8 pp.

[48] C. Clark, M. Ricchetti, A code-less BIST processor for embedded test and in-system configuration of boards and systems, in: Test Conference, 2004. Proceedings. ITC 2004. International, IEEE, pp. 857–866.

[49] D. Hely, F. Bancel, M.-L. Flottes, B. Rouzeyre, Secure scan techniques: a comparison, in: On-Line Testing Symposium, 2006. IOLTS 2006. 12th IEEE International, IEEE, 6 pp.

[50] J. Da Rolt, G. Di Natale, M.-L. Flottes, B. Rouzeyre, A smart test controller for scan chains in secure circuits, in: On-Line Testing Symposium (IOLTS), 2013 IEEE 19th International, IEEE, pp. 228–229.