A Dynamic-Key Secure Scan Structure Against Scan-Based Side Channel and Memory Cold Boot Attacks

Chia-Chi Wu, Man-Hsuan Kuo and Kuen-Jong Lee
Dept. EE, National Cheng Kung University, Tainan, Taiwan

***Abstract*— Scan design is a universal *design for test* (DFT) technology to increase the observability and controllability of the circuits under test by using scan chains. However, it also leads to a potential security problem that attackers can use scan design as a backdoor to extract confidential information. Researchers have tried to address this problem by using secure scan structures that usually have some keys to confirm the identities of users. However, the traditional methods to store intermediate data or keys in memory are also under high risk of being attacked. In this paper, we propose a dynamic-key secure DFT structure that can defend scan-based and memory attacks without decreasing the system performance and the testability. The main idea is to build a scan design key generator that can generate the keys dynamically instead of storing and using keys in the circuit statically. Only specific patterns derived from the original test patterns are valid to construct the keys and hence the attackers cannot shift in any other patterns to extract correct internal response from the scan chains or retrieve the keys from memory. Analysis results show that the proposed method can achieve a very high security level and the security level will not decrease no matter how many guess rounds the attackers have tried due to the dynamic nature of our method.**

Keywords—hardware security, side-channel attack, memory attack, dynamic key generation, secure scan architecture.

I. Introduction

Security has become one of the most important issues for both software and hardware design. Recently, *scan-based side channel attacks* that use the observability and controllability of scan design to extract confidential information have been reported [1][2]. Such attacks may cause the following threats. First, attackers may extract sensitive data such as secret encryption keys, operation and process parameters in control systems, and firmware for reverse engineering. Second, attackers may manipulate the system, e.g., changing the behavior of the chip, resulting in the violation of functional safety. In addition, some high level attackers may steal some IPs illegally, as often referred to as "commercial spies" [3].

Another type of attacks, called *memory attacks*, target memory devices in a system, which include RAM attacks and register attacks. "Cold boot attack" [4]-[8] is a common way of RAM attack which tries to access confidential information stored in RAM by putting the RAM in an environment that is fully controlled by the attacker. Thus storing some secret keys in RAM is extremely vulnerable if the RAM is not secure-protected. Also, for test coverage consideration, a key register may be a part of a scan chain and hence its contents may be scanned out if not well protected. Above attacking methods are difficult to prevent using software-based countermeasures, and hence hardware-based defending methods are desired [7].

In this paper, we propose a dynamic-key secure DFT structure that can prevent scan-based and memory cold boot attacks without decreasing system performance and testability. With this DFT structure, the user can shift out the data in the scan chains only when the data are the test response of a valid (authorized) test pattern that is just shifted into the scan chains and applied to the circuit, and the correct key for the secure DFT changes with every test pattern. Therefore, the scan chains can be correctly accessed only during the test mode, and only valid test patterns can be used to access the scan chains. Since no static key for accessing the scan chains is stored in memory, our method can also defend the memory cold boot attack. Furthermore, a fake response generator is developed to generate fake responses corresponding to incorrect input patterns. Moreover, our design hides the secure design by embedding the key pin into the scan input pins. As a result, it is almost impossible for the attackers who do not know the details of the DFT structure to retrieve the keys.

We organize the paper as follows. In Section II we survey some countermeasures for scan attacks and memory attacks. Section III describes the details of the proposed architecture. In Section IV we present the test procedure of our method. Section V explains the implementation flow. In Section VI, we analyze the security of our method against different kinds of attacks, and compare the security level of our design with those of previous methods. We also analyze the area and timing overhead as well as the testability and diagnosability of our method. Finally, we conclude this paper in Section VII.

II. Previous Work

Several approaches have been proposed to defend the side-channel and memory attacks as described in this section.

A. Countermeasures of Scan-based side channel attack

Some countermeasures have been proposed to disable the scan access after manufacturing test such that no backdoor exists for attackers [1]. This method may compromise the in-field test and debug capability if the scan chains cannot be accessed [9]. Most methods on defending the scan attack try to lock the scan mode (such as using a test key to decide whether to enable or disable the scan mode) or alter the scan structure according to whether the key is correct or not. These methods include Lock and Key [10][11][12], Scrambling [13][14], and Dummy flip-flops (DFFs) [15]. The security level of Lock and Key method depends on the length of the key, hence a long key is essential in order to defend brute force (trial-and-error) key attacks. The basic concept of Scrambling is to permute the data shifted into the scan chain by dividing each scan chain into a number of segments such that the attackers cannot figure out where the confidential information is stored. To achieve a higher security level, more scan segments are needed, which may result in high routing area overhead. The DFFs method inserts some extra dummy flip-flops in multiple scan chains, and stores the golden key in a key checking logic (KCL) in order to check whether the key is right or not. This method requires the same key to be used for every pattern, and hence all input patterns must have the same values on the locations of the dummy flip-flops, which makes it vulnerable. The above methods also suffer from the memory cold boot or key register

2377-5386/18/$31.00 ©2018 IEEE
DOI 10.1109/ATS.2018.00020

attacks since the keys to the security structures are stored in the chip statically. In addition, most of these techniques require changing the original DFT in the circuit, which is infeasible for legacy cores or hard cores [11].

B. Countermeasures of Memory attack

Memory attacks are difficult to prevent if keys in active use must be stored somewhere in the memory. Attackers do not need to address the software protection methods if they can access the hardware directly. Previous countermeasure methods include obscuring encryption keys [16], physically protecting the DRAM chips and architectural changing [17]. Using obscuring encryption keys to protect the original key makes the attack harder, but it is simply a matter of time to crack a system. Physically protecting DRAM from cold boot attack often requires soldering the memory chip to the motherboard and thus may restrict the flexibility of the system. Architectural changing tries to make the contents of memory decay rapidly, but it is difficult for today's technology since the faster DRAM decays, the harder to refresh it to maintain the correct data.

In this paper, we can avoid the memory cold boot attack since no key for the secure test architecture is stored in the memory.

III. Dynamic-Key Secure Scan Structure

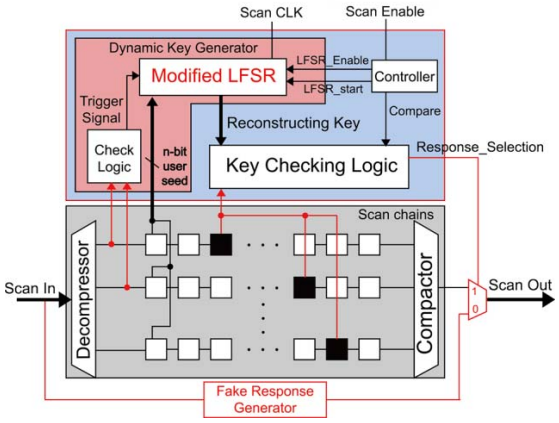

Fig. 1. The proposed dynamic-key secure scan structure

As shown in Fig. 1, our secure scan architecture is based on three techniques. First, we choose several bits of the original scan chains as the "key flip-flops (KFFs)," as the black squares show. We use these bits of each test pattern as its key. This implies that each test pattern can have its own and unique key. Note that we do not use extra bits for these KFFs and thus no overhead on the scan chains is needed, which is different from the extra dummy flip-flops in [15]. Second, based on a seed associated with each pattern, a dynamic key generator is used to regenerate the key for each pattern, and then compared to the key in the KFFs using the key checking logic (KCL). Third, a fake response generator is used to shift out correct or fake responses depending on the comparison result in KCL. This makes it difficult for the attacker to determine whether the shifted out data are correct or faked. We describe the details of the main components implementing the three techniques as follows.

A. Key Flip-Flops (KFFs) and Key Checking Logic (KCL)

The KFFs are the flip-flops that will contain the embedded key when a test pattern is shifted into the scan chains. These KFFs are connected to the KCL so as to check whether the key regenerated by the dynamic key generator matches that in the KFFs. The keys shifted to the KFFs are different for different patterns. Thus in every round, i.e., the processing of each pattern, a new key is shifted in. This key is then compared with the key regenerated by the dynamic key generator via the KCL to determine whether the pattern shifted into scan chains is valid or not, which means it can prevent the scan based controllability attack, i.e., the attackers cannot use the patterns they designate at will.

Note that whenever the designers want to use a new set of test patterns, they can easily calculate the required seeds based on the configuration of the dynamic key generator as will be described in Section V.

B. Dynamic Key Generator

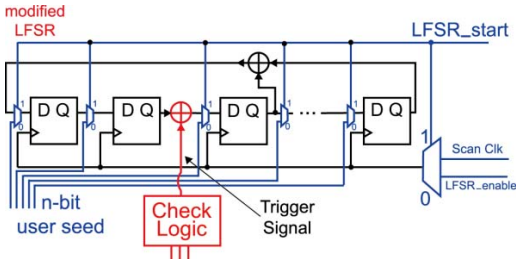

Fig. 2. The n-bit modified LFSR with the Check Logic

The Dynamic Key Generator consists of a *Modified LFSR* and a *Check Logic* as shown in Fig. 2. For each test pattern (round), the initial contents of the LFSR are determined by a seed derived from the test pattern. We embed the seed into the test pattern by adding it to the front of the original pattern. For example, if the seed is 32-bit and there are 8 scan chains in the scan design, we may load the seed into the scan chains in the first 4 cycles. Thus in the beginning, the LFSR_start signal is set to 0 in order to gate the scan clock and load the user seed from the scan chains. After the four seed shift-in cycles, the LFSR_enable signal is set from 0 to 1 to load the seed into the LFSR. The LFSR_start signal is then set to 1 to connect the clock of the LFSR to the scan clock to start running the LFSR. At the same time, the output of the Check Logic starts to alternate the contents of the LFSR as follows.

The inputs and the trigger data of the Check Logic are designer-defined, which are used to make the key generation algorithm irregular and difficult to predict. Refer to Fig. 1 again. The inputs of the Check Logic are connected to some outputs of the decompressor. When the exact trigger data appear on these outputs, i.e., the inputs to the Check Logic, the Check Logic will send a trigger signal to the LFSR to change the contents of the LFSR. For example, Fig. 2 shows a 3-input/1-output Check Logic. The three inputs of the Check Logic are connected to three outputs of the test decompressor. Assume the trigger data is 110. Then whenever 110 appear on the three outputs of the decompressor, the Check Logic will send a trigger signal to the LFSR to change its contents. Thus each pattern can

result in a different key in the LFSR. Even if it is possible that different patterns may have the same number of appearances of trigger data, one can still get different keys in the LFSR because these trigger data are likely to appear at different times.

Note that there is no extra user-seed port in our design. Thus the secure scan architecture looks just like a normal scan design from outside and thus the attacker may never know there are some secure designs. Also it is very difficult for the attacker to figure out what keys are used. However, this method may lead to a slight increase in test time due to the extra seed embedded in the test pattern, which will be discussed in Section VI.

C. Fake Response Generator

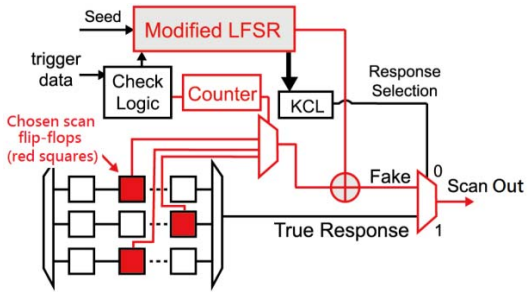

Fig. 3. Modules employed in the fake response generator.

Refer to Fig.3. The fake response generator is used to generate the true or fake response depending on whether the comparison of the keys is correct or not. We reuse the modified LFSR, KCL and some scan flip-flops, and add a counter, a multiplexer and an XOR to generate the fake responses. We use the counter to count how many times the Check Logic has received the trigger data and use this number to choose different scan flip-flops at different locations for each fake response. The fake response is obtained by XORing the output of the modified LFSR and the chosen scan flip-flop. At the beginning of a round, the response selection signal chooses the fake response. After the whole pattern is shifted in, the true response is chosen only if the comparison is correct. For example, if the scan length is 100, the counter needs to be 7 bits because the possible number of occurrences of the trigger data in any pattern is 100. For a test pattern that contains 3 times of the trigger data, the counter will output 3 and thus the output of the third chosen scan flip-flop will be XORed with the output of the LFSR. No matter the pattern is valid or not, each pattern can result in a fixed response that is different from the responses of other patterns. Therefore, it is difficult for attackers to determine whether the response is true or not.

The above three techniques are closely linked. Therefore, attackers cannot implement the attacks on our architecture unless they know all the details of these techniques.

IV. Test Procedure of The Proposed Method

With our design, each test pattern is processed in one round, which includes shifting in the pattern to scan chains, comparing the keys in KCL, and shifting out the response of the pattern.

The following describes the operation of the whole test procedure. At the beginning of each round, we need to shift in a modified pattern that contains the initial seed for the LFSR and the original test pattern, with the key embedded in the pattern (note that the key is different from the seed). The number of bits of the LFSR is equal to the number of bits of the seed. The clock of the modified LFSR is gated until the seed is shifted into the scan chains completely. Then the seed is loaded to the LFSR and the LFSR starts to run by the scan clock. The Check Logic will send the trigger signal to change a bit data of the LFSR from time to time when shifting in the original pattern.

After the pattern is shifted into the scan chains completely, the controller will issue a compare signal to enable the KCL to compare the regenerated key in the LFSR and the data of KFFs. Finally, the true or fake response will be shifted out depending on the result of the key comparison. Note that the next pattern will be shifted in when the current response is shifted out. This completes one test round, i.e., the process of one test pattern. Therefore, the keys for different rounds are different, and each key appears in the chip only for a short time. It is also worth to point out that no matter how many times the attacker has tried, the security level will not degrade since the key in the next round is totally independent of the current key.

V. Implementation Flow

The implementation flow of the proposed method is described as follows. First, we need to add the normal multiple scan chain structure to the original circuit. Second, we run ATPG to generate the test patterns. Third, we add the proposed security architectures, including the dynamic key generator and the fake response generator into the normal multiple scan chain structure, and select some scan flip-flops as the KFFs. We allow the KFFs to be selected randomly, which determines the correct key for the pattern and is also the key to be regenerated by the Dynamic key generator.

Next is an important step to implement the proposed architecture, that is, to determine the seed of each round based on each given test pattern. We determine all the cycles at which the Check Logic is triggered, and then do the reverse derivation for the LFSR with the initial values being the logic values of the key on the KFFs. This can be done by simulating an LFSR with a reciprocal characteristic polynomial [18], which we call R-LFSR here. The triggered signals from the Check Logic can be inserted whenever needed. We formally describe this step as follows. If the modified LFSR has n bits and the trigger signal is connected to the output of the m^{th} flip-flop of the LFSR, the trigger signal will be connected to the output of the $(n-m-1)^{th}$ flip-flop of the R-LFSR. If the length of each scan chain is L and the trigger data for the LFSR appears at the k cycle, the trigger data for the R-LFSR will appear at the (L-k) cycle.

For example, consider the 4-bit modified LFSR shown in Fig. 4(a). Assume that there are 4 scan chains and the length of each scan chain is 6. Also assume that a correct trigger data appears at the second shifting cycle and there is only one trigger signal input to the LFSR. If the correct key is 1101 and the trigger signal is connected to the output of the second flip-flop of the modified LFSR, we can use the

R-LFSR as shown in Fig. 4(b) and give the reverse key "1011" as the initial value of the R-LFSR. We run (L-k) = 6-2 = 4 cycles first and then change the output of the first flip-flop (because (n-m-1) = (4-2-1) =1) of the R-LFSR to reflect the effect of the trigger data, which will give a new value of 0010. We then run the remaining 2 cycles and can get the reverse seed 1000, so the desired seed is "0001". With the above procedure, we can get the seed of each test pattern for the circuit with the proposed secure DFT. Notice that the R-LFSR is used only for the simulation to obtain the required seeds. It does not physically exist in the proposed architecture. Finally we can combine the user seeds and the test patterns into the modified patterns.

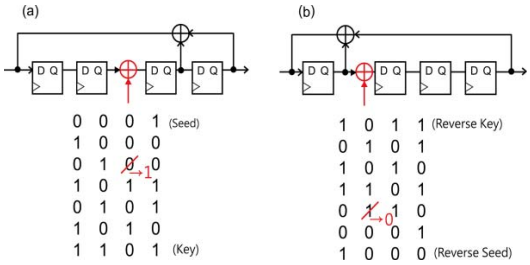

Fig. 4. (a) The 4-bit modified LFSR. (b) The LFSR with a reciprocal characteristic polynomial

VI. Analysis and Results

To examine the security of the secure DFT, most papers use security level as the criteria. For static secure structures, the security level usually refers to the possibilities of successfully figuring out the key or configuring the secure architecture using a brute-force method. Recently, several "smarter" attacking methods specifically targeting various secure designs have been disclosed. In this section, we first describe the effectiveness of our secure DFT against different kinds of existing attacks. We then compare the security level of our design with those of recent secure designs. In addition we also analyze the area overhead of the proposed method.

A. Security Analysis

We analyze the effectiveness of our proposed methods against the various types of attacks as follows.

Scan-based Side Channel Attack: The main approach of this attack is first running a system in its functional mode, and then switching to the test mode when it is determined that some confidential information such as the key(s) to an encryption system appears in some scan chains [1]. The confidential information can then be shifted out through the scan chains and thus attackers can retrieve the information they desire. With our proposed method, the fake response is output unless the comparison of the constructing key is correct. Hence, the attacker cannot collect the true responses to implement this attack if (s)he does not know the correct keys of all patterns.

Test Mode Only Scan-based Attack: One way to defend the scan-based side channel attacks is to use the resetting countermeasure method which resets scan chains whenever switching between test and other modes occurs. However, a new attack method using only the test mode has been proposed in [19]. The first step of this attack relies on the comparison of the responses of the all-zero pattern and the walking-1 patterns to determine the corresponding positions between the scan cells and the bits of the target register. With our proposed method, the attacker cannot use the pattern they desire at will due to the KFFs and the fake response generator. Therefore, the attacker cannot get the correct responses from our secure scan design if they do not know the seeds for their malicious patterns.

Cold Boot Attack: The secure key for DFT and the intermediate values can be retrieved by cold boot attack if they are stored in the memory [7]. With our architecture, the secure key for DFT is not stored in the chip but constructed by the dynamic key generator. Thus there is no key when the secure scan design is not used. Also, in each round the key is correct only when a corresponding pattern is shifted in completely. In other words, the existence time of the real round key is very short. The data that collected by the cold boot attack is difficult to analyze because the secure key is changing irregularly during the process. Therefore, the proposed method can significantly reduce the probability of being attacked successfully.

Combinational Functional Reverse Engineering: In this attack [20], the attacker will first use the scan-based side channel attack to obtain the intermediate data of the target circuit by analyzing the output responses of the scan chains, and then use the information to implement the Boolean Learning method in order to rebuild a structure whose function is the same as the crypto circuit. Therefore, the using of normal scan design is needed in this attack. With the proposed method, we can use the design of fake response generator to protect the scan chains. The attacker can only retrieve the data generated by the fake response generator, but not the secret information, intermediate data, or the true responses of our secure architecture. As the result, the attacker cannot use our secure scan design at will to find out the function of the target circuit.

Security Levels under Brute Force Obfuscation Key Attack: In this attack, which is also called trial-and-error method, the security level represents the reciprocal of the probability to hit the correct configuration of the secure DFT in one guess. Previous methods have defined the security level in different ways. In this paper, we define the security level as the product of the number of combinations of the secure structures and the number of possible values of the keys to reflect the difficulty of figuring out the secure architecture as well as the keys. For our method, there are several obfuscating places to confuse the attackers, including 1) the number of possible positions of KFFs, 2) the number of possible key values, 3) the number of possible input connections from the decompressor to the Check Logic together with the number of data on these connections, 4) the number of possible appearances of the trigger data for each pattern, and 5) the number of possible connections from the Check Logic to the modified LFSR. We will use the following notation in the following analysis: K_b denotes the number of bits of the key, S_L denotes the scan chain length, $\#SC$ denotes the number of scan chains, $\#TP$ denotes the number of test patterns, and $\#S_G$ denotes the number of scan segments when the scrambling method is used.

TABLE I. THE POSSIBILITIES OF BRUTE FORCE GUESSING THE CORRECT CONFIGURATION OF OUR METHOD

Obfuscating places	Number of possibilities
The number of possible positions of KFFs	$C_{K_b}^{S_L*\#SC}$
The number of possible keys	2^{K_b}
The possible input combinations of the Check Logic	$\sum_{k=1}^{\#SC}(C_{\mathrm{k}}^{\#SC} * 2^k)$
The possible number of trigger data for each pattern	S_L
The possible position of the trigger signal	K_b

TABLE II. COMPARISON OF SECURITY LEVEL WITH PREVIOUS WORK (*#SC* SCAN CHAINS, S_L SCAN LENGTH, K_B-BIT KEY STORAGE FOR SCRAMBLING, $\#S_G$ SCAN SEGMENTS IN SCAN CHAIN, FOR VIM-SCAN, *M* DIFFERENT K_B-BIT KEYS)

Secure scan design	Secure test wrapper [11]	Vim-scan [12]	Scrambling [13][14]	DFFs [15]	Our method
Security level	2^{K_b}	2^{K_b*M}	$(\#S_G)!* 2^{K_b}$	$C_{K_b}^{S_L*\#SC}*2^{K_b}$	$(C_{K_b}^{S_L*\#SC})* 2^{K_b} * \sum_{k=1}^{\#SC}(C_k^{\#SC} * 2^k) * S_L*K_b$
For *#SC* =8, S_L=67, K_b=9, $\#S_G$=8, *M*=4	2^9	2^{36}	$2.1*10^7$	$4.3*10^{21}$	$1.72*10^{28}$ (For each round!)

Table I shows the number of possibilities of the 5 obfuscating places. The first and second items are easy to derive. The third item, i.e., the possible input combinations of the Check Logic, include the possibilities of the connections from the outputs of the decompressor to the inputs of the Check Logic and the possible values of trigger data designated by the designer. Since the number of inputs to the Check Logic can be from 1 to *#SC* (the number of scan chains), and there are 2^k possible values if the number of the inputs is k, the total number of possibilities for this item is $(C_1^{\#SC} *2^1) + (C_2^{\#SC} *2^2) + \ldots + (C_{\#SC}^{\#SC} *2^{\#SC}) = \sum_{k=1}^{\#SC}(C_{\mathrm{k}}^{\#SC} * 2^k)$. The 4th item, i.e., the possible number of trigger data for each pattern is equal to the scan length (S_L) because it is possible for the trigger data to appear at each shifting cycle. The possible position of the trigger signal, which is assumed to be only 1 bit in this paper, depends on the number of bits of the LFSR, which is in turn equal to the number of bits of the key. The security level of our secure architecture can be expressed as the product of these possibilities.

In Table II, we compare the security level of our approach with those of previous methods [11]-[15]. The second row gives the equation to calculate the security level of each method. In the third row we give the numbers with 8 scan chains, 67 scan cells in each chain, 8 scan segments in each scan chain for Scrambling, and a 9-bit key storage for all methods. We set these model parameters according to the 128-bit AES circuit since it is the target most likely to be attacked. The number of the scan segments is set to be the same as the number of scan chains so as to make the comparison fair. The key length is set to 9 to provide sufficient possibilities for all keys since the number of test patterns of this circuit is 256. The security level of the Secure Test Wrapper [11] only depends on the key size. The Vim-scan [12] uses multiple keys for authentication, and here we assume there are M=4 keys since the paper shows that 4 is the best number to balance security level and overhead. The security level of the Scrambling method [13][14] depends on the number of bits of the key, and the number of the scan segments, which can be expressed as $(\#S_G)!*2^{K_b}$. The security level of DFFs [15] depends on the positions of DFFs and the number of bits of the key, which can be expressed as $C_{K_b}^{S_L*\#SC}* 2^{K_b}$. It can be seen that our proposed architecture has the best security level among all methods even if only one run is considered. In fact, our proposed architecture can have an even better security level because the techniques in [11]-[15] are static, which means the key does not change during the whole process, and thus the security level will decrease when the attacker keeps trying. Differently, in our method the keys are dynamically generated in each round. Thus no matter how many trials the attackers have made, the "experiences" obtained from previous trials cannot be accumulated due to the following two reasons: 1) the key in each round in our design is independent of that in other rounds, and 2) the attacker cannot figure out whether the scanned out response is the true one or the faked one.

B. Overhead Analysis

Area Overhead: Table III shows the area overhead of our proposed architecture with different benchmarks. The first one is a 128-bit AES circuit. The other three circuits, one ISCAS benchmark circuit (s9234) and two IWLS circuits (wb_conmax and Leon3mp), have sizes from small to large. These circuits are synthesized by a commercial synthesis tool with a TSMC90nm technology file. We also use a commercial ATPG tool to generate the test patterns, and the experiments target 100% fault coverage for detectable stuck-at faults. The area of our secure DFT is dependent on both the modified LFSR whose number of bits depends on the number of test patterns and the counter used to choose the scan flip-flop for fake response generation. We compare the areas of the original circuit with normal DFT and our secure structure and calculate the area overhead. As we can see, the area overhead of the proposed architecture which contains the controller, KCL, dynamic key generator and fake response generator is only about 8% even on the small circuit ISCAS89_s9234 whose area (gate counts) is 6741 with normal DFT. For Leon3mp which contains 1699028 gates, the area overhead is only 0.053%.

TABLE III. AREA OVERHEAD OF THE PROPOSED METHOD WITH DIFFERENT BENCHMARKS

Benchmarks	128-bit AES	ISCAS89 s9234	wb_conmax	Leon3mp
#SC	8	32	19	256
S_L	67	8	41	426
#TP	256	157	133	3648
K_b	9	8	8	12
Area (gate counts) with normal DFT	49283	6741	80619	1699028
Area overhead	+1.24%	+8.12%	+0.76%	+0.053%

Test Time Overhead: For the proposed design, the modified test pattern contains the user seed and the original test pattern. The seed is at the front of the original pattern, so the seed will be first shifted into the scan chains and loaded into the modified LFSR. Let S_b denote the number of bits of the seed

and *#SC* denote the number of scan chains. The test time will increase by $\lceil S_b/\#SC \rceil$ clock cycles. For example, assume there are 100 scan chains. If the bit number of the user seed is smaller than 100, we only need one clock cycle to load the seed to the dynamic key generator. Therefore, the time overhead is quite small.

C. Testability and Diagnosability Analysis

Testability: For the proposed secure design, as long as the user applies the correct modified test patterns, the true CUT responses will be scanned out by the protected scan design. Since the modified test patterns contain all information of the original test patterns, the testability of the circuit is not changed; all faults detected by the original patterns are also detected by the modified patterns.

As for the faults occurred in the extra circuitry introduced by the secure scan design, since only simple primitive logic gates, counters and registers are used to secure the scan design in our dynamic obfuscation structure, they can be easily tested by build-in self-test with very high coverage [21]. Hence, for our proposed design, there is no impact on its testability.

One problem that needs to be addressed is the scan chain test which is usually conducted at the beginning of a test procedure. Suppose there is a defect in one of the "KFF" flops (black squares in Fig. 1) or in one of the scan flip-flops in front of the KFF, the key value will be changed due to the defect. Thus, a fake response will potentially be indistinguishable from a defective response. To address this problem, we may again use build-in self-test to test the scan chains such that whenever needed (such as power on time), the scan chains will first be tested by the build-in self-test. At this moment, the data in the flip-flop used to store the comparison result of the key is locked to 1. Consequently, the scan chain test can be implemented successfully. Therefore, the wrong key will not make the fake response indistinguishable from the defective response.

Diagnosability: As we mentioned in the last paragraph, for the proposed secure design, as long as the user seeds for the test patterns are correct, the test process can be implemented successfully without sacrificing testability. The diagnosis process can be implemented in a similar way, just changing the modified test patterns to the modified diagnosis patterns.

VII. Conclusions

In this paper, we propose a dynamic secure multiple scan chain architecture, which allows the user to access scan chains only when valid test patterns are shifted into the scan chain. Therefore, the test procedure can be carried out normally if the authorized test patterns are provided. If the user shifts in any non-authorized patterns, then only faked responses can be obtained. Thus it becomes almost impossible to make side-channel scan attacks.

In our method, the test key may change every round and hence it is difficult for the attackers to obtain the key of each round in time. The correct key of each round is not stored in the memory, so memory cold boot attack also cannot threat our test architecture. Moreover, the algorithm of the dynamic key generator is irregular because it generates the keys depending on the test patterns. In addition, we embed the seeds into the test patterns and thus no extra pin is needed for seed provision. Furthermore, we design a fake response generator to mislead the attacker. The last two techniques can confuse the attacker that the secure scan design is just a normal DFT design. Comparing with the previous work, our method can achieve a much better security level. We also show that the area overhead of our method is about 8% for a small circuit containing 6741 gate counts, and it requires only 0.053% for a circuit with about 1.7 million gate counts. It is also shown that the test time overhead is quite small.

Acknowledgement

This work was partially supported by the Ministry of Science and Technology of Taiwan under Contracts 105-2221-E-006-242 and 107-2218-E-006-025.

References

[1] J. Da Rolt et al., "Test versus security: Past and present," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 1, pp. 50-62, Mar. 2014.

[2] M. Tehranipoor and C. Wang, *Introduction to Hardware Security and Trust.* New York: Springer, 2011.

[3] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard," in *Proc. ITC*, Oct. 2004, pp. 339-344.

[4] J. G. Ooi and K. H. Kam, "A proof of concept on defending cold boot attack", in *Proc. Asia Symposium on Quality Electronic Design. ASQED*, Jul. 2009, pp. 330-335.

[5] J. Bauer, M. Gruhn, and F. C. Freiling, "Lest We Forget: Cold-Boot Attacks on Scrambled DDR3 Memory," *Digital Investigation*, vol. 16, pp.65-74, March 2016.

[6] S. F. Yitbarek, M. T. Aga, R. Das, and T. Austin, "Cold Boot Attacks are Still Hot: Security Analysis of Memory Scramblers in Modern Processors," in *Proc. IEEE International Symposium on High Performance Computer Architecture. HPCA*, Feb. 2017, pp. 313-324.

[7] J. Halderman, S. Schoen, N. Heninger, W. Clarkson, W. Paul, J. Calandrino, A. Feldman, J.Appelbaum, and E. Felten, "Lest We Remember:Cold Boot Attacks on Encryption Keys," *Communications of the ACM*, vol. 52, no. 5, pp.91-98, May. 2009.

[8] M. Gruhn and T. Muller, "On the Practicability of Cold Boot Attacks," in *Proc. International Conference on Availibility, Reliability and Security.* Sep. 2013, pp. 390-397.

[9] Sourgen, *Security locks for integrated circuits*, US Patent 638459, 1993

[10] B. Yang, R. Karri, and K. Wu, "Secure Scan: A Design for-Test Architecture for Crypto Chips," in *Proc. Design Automation Conf.,* June 2005, pp. 135-140.

[11] G.-M. Chiu and J. C.-M. Li, "A secure test wrapper design against internal and boundary scan attacks for embedded cores," *IEEE Transactions on VLSI Systems*, vol. 20, no. 1, pp. 126-134, Jan. 2012.

[12] S. Paul, R. S. Chakraborty, and S. Bhunia, "Vim-scan: A low overhead scan design approach for protection of secret key in scan-based secure chips," in *IEEE VTS*, 2007.

[13] D. Hely, F. Bancel, M. L. Flottes, B. Rouzeyre, M. Renovell, and N. Brard, "Scan design and secure chip," in *Proc. IEEE Int. On-Line Testing Symp.,* July 2004, pp. 219-224.

[14] M. Oya, Y. Atobe, Y. Shi, M. Yanagisawa and N. Togawa, "Secure scan design using improved random order and its evaluations," in *Proc. IEEE Asia Pacific Conference on Circuits and Systems.* Nov. 2014, pp. 555-558.

[15] J. Lee, M. Tehranipoor and J. Plusquellic, "A Low-Cost Solution for Protecting IPs Against Side-Channel Scan-Based Attacks," in *Proc. VLSI Test Symposium*, May 2006, pp. 6-9.

[16] L. Guan, J. Lin, B. Luo, J. Jing, and J. Wang, "Protecting private keys against memory disclosure attacks using hardware transactional memory," in *Proc. IEEE Symposium on Security and Privacy. SP'15*, May 2015, pp. 3-19.

[17] José L. Ayala, *Communication Architectures for Systems-on-Chip*. USA: CRC Press , 2017, ch.8.

[18] L-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles And Architectures: Design for Testability.* USA: Morgan Kaufmann Pub, 2006, ch.5.

[19] S. S. Ali, S. M. Saeed, O. Sinanoglu, and R. Karri, "New scan-based attack using only the test mode," in *Proc. IEEE VLSI-SoC*, Oct. 2013, pp. 234-239.

[20] L. Azriel, R. Ginosar, and A. Mendelson, "Exploiting the scan side channel for reverse engineering of a VLSI device," *Techn., Israel Inst. Technol.*, Haifa, Israel, Tech. Rep. CCIT #897, 2016.

[21] A. Cui, Y. Luo, and C. H. Chang, "Static and Dynamic Obfuscations of Scan Data Against Scan-based Side-channel Attacks." in *Proc. IEEE Transactions on Information Forensics and Security.* Feb 2017, pp. 363-376.