Test Generation for Hardware Trojan Detection Using Correlation Analysis and Genetic Algorithm

ZHENDONG SHI, HAOCHENG MA, QIZHI ZHANG, YANJIANG LIU, and YIQIANG ZHAO, School of Microelectronics, Tianjin University, China
JIAJI HE, Institute of Microelectronics, Tsinghua University, China

Hardware Trojan (HT) is a major threat to the security of integrated circuits (ICs). Among various HT detection approaches, side channel analysis (SCA)-based methods have been extensively studied. SCA-based methods try to detect HTs by comparing side channel signatures from circuits under test with those from trusted golden references. The pre-condition for SCA-based HT detection to work is that the testers can collect extra signatures/anomalies introduced by activated HTs. Thus, activation of HTs and amplification of the differences between circuits under test and golden references are the keys to SCA-based HT detection methods. Test vectors are of great importance to the activation of HTs, but existing test generation methods have two major limitations. First, the number of test vectors required to trigger HTs is quite large. Second, the HT circuit's activities are marginal compared with the whole circuit's activities. In this article, we propose an optimized test generation methodology to assist SCA-based HT detection. Considering the HTs' inherent surreptitious nature, inactive nodes with low transition probability are more likely to be selected as HT trigger nodes. Therefore, the correlations between circuit inputs and inactive nodes are first exploited to activate HTs. Then a test reordering process based on the genetic algorithm (GA) is implemented to increase the proportion of the HT circuit's activities to the whole circuit's activities. Experiments on 10 selected ISCAS benchmarks, wb_conmax benchmark, and b17 benchmark demonstrate that the number of test vectors required to trigger HTs reduces 28.8% on average compared with the result of MERO and MERS methods. After the test vector reordering process, the proportion of the HT circuit's activities to the whole circuit's activities is improved by 95% on average, compared with the result of MERS method.

CCS Concepts: • **Security and privacy → Malicious design modifications**;

Additional Key Words and Phrases: Hardware trojan detection, side channel analysis, test generation, correlation analysis, test reordering

This work was supported in part by National Natural Science Foundation of China (Grant No. 61832018 and No. 62004112) and China Postdoctoral Science Foundation (Grant No. 2019TQ0167).

Authors' addresses: Z. Shi, H. Ma, Q. Zhang, Y. Liu, and Y. Zhao (corresponding author), School of Microelectronics, Tianjin University, 92 Weijin Rd, Nankai Qu, Tianjin Shi, China; email: yq_zhao@tju.edu.cn; J. He (corresponding author), Institute of Microelectronics, Tsinghua University, 30 Shuangqing Rd, Haidian Qu, Beijing Shi, China; email: jiaji_he@mail.tsinghua.edu.cn.

1539-9087/2021/03-ART28 $15.00
https://doi.org/10.1145/3446837

ACM Reference format:
Zhendong Shi, Haocheng Ma, Qizhi Zhang, Yanjiang Liu, Yiqiang Zhao, and Jiaji He. 2021. Test Generation for Hardware Trojan Detection Using Correlation Analysis and Genetic Algorithm. *ACM Trans. Embed. Comput. Syst.* 20, 4, Article 28 (March 2021), 20 pages.
https://doi.org/10.1145/3446837

1 INTRODUCTION

Design and manufacture globalization has become the main trend in the IC supply chain. However, the uncontrollable participation of third-party services provides the opportunity for adversaries to maliciously modify the original design. This kind of hardware threat, namely, hardware Trojan, can cause malfunction of the IC, information leakage, privilege escalation, and even system failure. These devastating influences have motivated the study of HT detection methods to guarantee the security of ICs. Numerous HT detection techniques have been proposed over the past decades. Among them, logic testing and side channel analysis are two mainstream methods during the test phase of the IC's lifecycle. Logic testing methods identify logic changes on outputs after applying selected test vectors [11]. Nevertheless, it is extremely difficult to fully activate the HTs and propagate HTs' malicious effects to circuit outputs [24]. Besides, the inordinately large number of possible HTs increase the number of required test patterns exponentially [8]. Especially for large IC designs, the HT detection methods using logic testing are infeasible.

SCA-based HT detection methods rely on the fact that HTs will distort the side channel signatures of ICs [21], these side channel signatures include power [22, 37], electromagnetic (EM) radiation [14, 15, 41], temperature [2], and multiple parameters' combination [23, 25]. Since SCA methods analyze the differences of side channel signatures between the chip under test and golden references. It is of great significance to activate HTs efficiently and increase the proportion of the HT circuit's activities to the whole circuit's activities. The challenges for applying the test generation approach in SCA-based HT detection methods are from two aspects. First, the stealthiness of HTs determines that HTs will only be activated under rare conditions. Second, the differences caused by HTs on side channel signatures are marginal compared to the whole circuit's activities, i.e., the side channel anomalies introduced by HTs can be easily submerged. To overcome the limitations mentioned above, multiple test generation methods are proposed. Although some of them showed effectiveness in increasing HT trigger coverage and highlight the HT circuit's influences. However, those methods also have disadvantages such as low HT trigger efficiency and poor scalability for a large number of test vectors. Take MERS method proposed in Reference [18] for example, which includes two reordering approaches based on Hamming distance and simulation, respectively. MERS leverages an exhaustive technique to generate the initial test set, thus the time cost and the number of test vectors required to trigger HTs are quite large [26, 29]. Moreover, each test vector needs to be considered in the calculation process for reordering, so MERS also has huge computational complexity when the number of test vectors is large.

In this article, we propose a test generation method for SCA-based HT detection using correlation analysis and genetic algorithm. HTs' stealthiness is analyzed from the structural perspective first. Inactive nodes with low transition probability are considered to be the most potential HT trigger nodes. By performing correlation analysis, the circuit inputs that can induce the activation of inactive nodes are identified, which are referred to as *relevant inputs*. Test vectors are generated based on the *relevant inputs,* subsequently, the number of test vectors required to trigger HTs significantly reduces. After that, the HT payloads are considered to highlight the HT circuit's influences. To increase the proportion of the HT circuit's activities to the whole circuit's activities, GA is utilized to perform test vectors reordering. To the best of our knowledge, our work is the

first attempt in utilizing the genetic algorithm to reorder test vectors used for SCA methods. The contributions of our work are as follows:

- An efficient test generation approach is proposed to support SCA-based HT detection methods, and the proposed approach focuses on improving the switching activities in HT circuits.
- The correlations between different circuit inputs and the activation of inactive nodes are exploited to trigger HTs efficiently; the number of test vectors required to trigger HTs under the same condition is significantly reduced compared with previous methods.
- A test vector reordering method based on the genetic algorithm is proposed to improve the proportion of the HT circuit's activities to the whole circuit's activities.

The rest of the article is organized as follows: Section 2 presents our attack model and related work. Section 3 gives a detailed description of our proposed test generation and optimization methodology. In Section 4, we evaluate the effectiveness of generated test vectors on 10 selected ISCAS benchmarks, wb_conmax benchmark, and b17 benchmark. Section 5 concludes the article.

2 MOTIVATION

2.1 Attack Model

The structure of an HT can be divided into trigger part and payload part [5]. Based on the different trigger and payload schemes, HTs can be classified into various types. In this article, we primarily focus on combinational HTs whose trigger condition is the combinations of input signals or internal signals, which means the HTs will be activated when the logic value of each trigger node meets the value set by the attacker.

For HT-inserted circuits, the transition probability directly indicates the stealthy nature of potential trigger nodes [42], and the trigger part will be typically connected to inactive nodes with low transition probabilities [39, 43]. By using inactive nodes to form up trigger structure, HTs can have very low trigger probabilities without large area overhead. At the same time, the side channel impacts caused by HTs are marginal due to lower switching activities. Thus, the prerequisite for activating HTs is to increase the transition probability of the inactive nodes. For digital HTs, malicious functionalities are usually accomplished by modifying the logic values of critical nodes in circuits. Before the HTs are triggered, the malicious functionalities will be hidden from circuit testers. The observability characteristic of circuit nodes is proposed as an effective evaluation criterion for HT payload identification [19]. The observability value of circuit nodes reflects the effort required to observe the logic value of target nodes at the primary outputs [40]. The Sandia Controllability and Observability Analysis Program (SCOAP) is often used for observability analysis [13]. In SCOAP, each gate is assigned to corresponding level orders first from primary inputs to primary outputs. Then the observability value of each node is calculated in a breadth-first manner from primary outputs toward primary inputs. The range of the observability value is from 0 to infinity. A higher observability value means the logic change on that node is more difficult to be observed at circuit outputs, so the nodes with higher observability value are more likely to be chosen to deliver malicious functionalities by adversaries [30].

Based on the analysis above, internal nodes that have low transition probability and high observability value are the most potential implant locations for HTs, which must be taken into account by the test generation method used for SCA-based HT detection.

2.2 Related Work

ICs will go through a testing process to verify whether the designed functionalities meet the requirements. During the testing process, test pattern generation methods like Automatic Test

Table 1. Comparison of the Functioning Stage, Target HT Detection Methods, and Time Complexity between Our Proposed Method and Existing Test Generation Schemes

Test Generation Methods	Our Method	MERO [7]	MERS [18]	GA+SAT [29]	TRIAGE [26]	PMTP [34]	[33]	PODEM [35]
Functioning Stage	post-silicon	post-silicon	post-silicon	post-silicon	post-silicon	design	design	design
Target HT Detection Methods	SCA	logic test	SCA	logic test	logic test	SCA+logic test	SCA+logic test	SCA+logic test
Time Complexity	low	high	high	high	low	low	low	low

Pattern Generation (ATPG) are utilized [9]. However, these approaches focus on functional correctness and are not suitable for activating potential HTs [6]. A method named MERO is proposed in Reference [7] to trigger inactive nodes for N times, and the test vectors are continually updated until the switching requirement is satisfied. When N is sufficiently large, HTs that adopt inactive nodes' combinations as trigger part will be activated most possibly [1, 17, 27]. However, the number of test vectors required to trigger inactive nodes for N times for MERO is quite large. Besides, the time cost of MERO method is large when applied to even medium scale circuits [26]. Later, a combined GA and Boolean Satisfiability (SAT)-based approach is proposed in Reference [29] to generate and optimize test vectors, but this approach can not propagate logic modifications caused by HT's payload to circuit outputs. The TRIAGE approach combines both GA and SCOAP [26], where the controllability and observability values are considered in the fitness function of GA to quantify the inactive nodes. However, the TRIAGE approach only aims at promoting the HT trigger coverage, but the only activation of HTs without considering their malicious effects will not help HT detection.

Inserted HTs can be detected using SCA when side channel anomalies caused by HTs are beyond the sensitivity margin, thus HTs need to be triggered first. More importantly, the malicious effects introduced by HTs should be highlighted compared to the whole circuit's activities. A method named MERS is proposed [18] to increase the proportion of the HT circuit's activities to the whole circuit's activities. However, MERS method has low HT trigger efficiency because MERO method is utilized to generate the initial test set. The computational complexity of reordering methods in MERS is huge for a large number of test vectors, which limits its further application. Later, GA is used to search for the best succeeding test patterns for the initial test set to form test pairs [24], but still, this approach adopts MERO method to generate the initial test set. Besides, only one pair of test patterns is applied to the circuit under test at a time, which does not apply to existing SCA-based HT detection methods.

Unlike the above methods that are utilized in the post-silicon stage, a design for trust (DfTr) approach that aims to reduce HTs' activation time by implanting scan flip-flops in the design stage is proposed in Reference [32], but the consequent area overhead limits its applicability. Later, authors in Reference [34] proposed a method called PMTP, which can activate the HTs efficiently by constructing the maximum transition probability (MTP) propagation paths from primary inputs to rare nets. Whereafter, a new test pattern generation method is integrated with PMTP approach in Reference [33] to enhance the side channel sensitivity. Moreover, a method called PODEM is proposed in Reference [35] to lower the area overhead in devices with limited hardware resources. The final outputs of the methods in References [33–35] are modified circuit netlist and optimized test patterns. While the test generation methods used in the post-silicon stage focus on the optimization of the test vectors and no changes will be made to the circuit netlist, like our proposed method. Those methods and our method both aim to activate and highlight the HTs, only in different ways. As shown in Table 1, the functioning stage, target HT detection methods, and time

complexity between our proposed method and existing test generation schemes are compared. Our proposed method is utilized as an assistance for SCA-based HT detection. Compared with MERS method, our proposed method has lower time complexity.

In summary, existing test generation methods have limitations in HT trigger efficiency and highlighting HTs' impacts. Our proposed method improves the above situation and will be introduced in the next chapter.

2.3 Relation between Switching Activity and SCA Sensitivity

In this article, we focus on increasing the switching activities of inactive nodes to trigger HTs and highlight HTs' impacts on the whole circuit's side channel signatures. Taking power consumption as an example, the total power of the digital CMOS circuit is made of dynamic power and static power, which can be expressed as Equation (1):

$$P_{total} = P_{dynamic} + P_{static}. \tag{1}$$

The overall power of the circuit in operation is dominated by dynamic power, which consists of dynamic switching power and dynamic short-circuit power. Dynamic switching power is the power required to charge and discharge the output capacitance on a gate. Dynamic short-circuit power is the power introduced by short-circuit currents that occur when both the NMOS and PMOS transistors are on. The composition of dynamic power consumption can be expressed by Equation (2):

$$\begin{aligned} P_{dynamic} &= P_{switching} + P_{short} \\ &= \frac{1}{2} \cdot C \cdot {V_{DD}}^2 \cdot f \cdot N_{SW} + Q_{SC} \cdot V_{DD} \cdot f \cdot N_{SW} \\ &= N_{SW} \cdot \left(\frac{1}{2} \cdot C \cdot {V_{DD}}^2 + Q_{SC} \cdot V_{DD} \right) \cdot f, \end{aligned} \tag{2}$$

where C is the load capacitance, V_{DD} is the supply voltage, f is the operation frequency, N_{SW} is the switching activities in the circuit, Q_{SC} is the charge carried by short-circuit current per transition. For HT-inserted circuits, the difference in power consumption between the HT circuit and the whole circuit is caused by the difference in C and N_{SW}, since the other parameters are the same. The proportion of the HT circuit's power consumption to the whole circuit's power consumption is denoted as P_{ratio}, which can be expressed as Equation (3):

$$\begin{aligned} P_{ratio} &= P_{HT}/P_{circuit} \\ &= \frac{N_{SW_{HT}} \cdot \left(\frac{1}{2} \cdot C_{HT} \cdot {V_{DD}}^2 + Q_{SC} \cdot V_{DD} \right) \cdot f}{N_{SW_{circuit}} \cdot \left(\frac{1}{2} \cdot C_{circuit} \cdot {V_{DD}}^2 + Q_{SC} \cdot V_{DD} \right) \cdot f} \\ &= \frac{N_{SW_{HT}}}{N_{SW_{circuit}}} \cdot \frac{k1 \cdot C_{HT} + k2}{k1 \cdot C_{circuit} + k2}. \end{aligned} \tag{3}$$

where $k1 = \frac{1}{2} \cdot {V_{DD}}^2$ and $k2 = Q_{SC} \cdot V_{DD}$ are constants for HT circuit and the whole circuit. Since the output load of the HT circuit and the whole circuit are constants for specific HT-inserted circuits. The proportion of the HT circuit's power consumption to the whole circuit's power consumption is determined by the proportion of the HT circuit's switching activities to the whole circuit's switching activities. Therefore, the impacts of HTs on side channel signatures can be highlighted by increasing the proportion of the HT circuit's activities to the whole circuit's activities, and the sensitivity of SCA methods can be improved consequently.

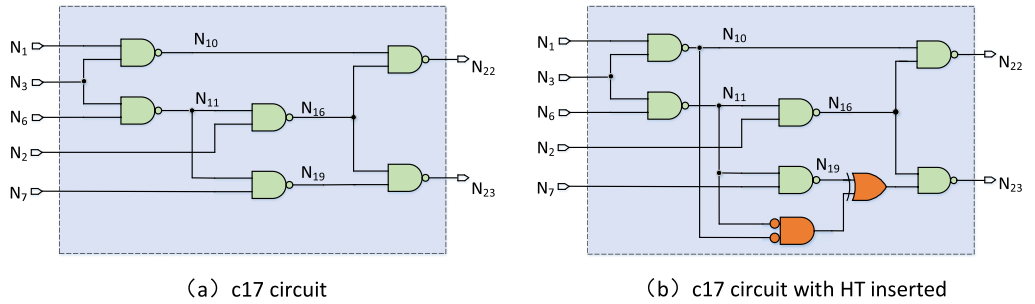


Fig. 1. The netlist of c17 circuit from ISCAS-85. The left picture shows the original netlist, the right picture shows the HT-inserted netlist where two internal inactive nodes are used to form up the trigger structure.

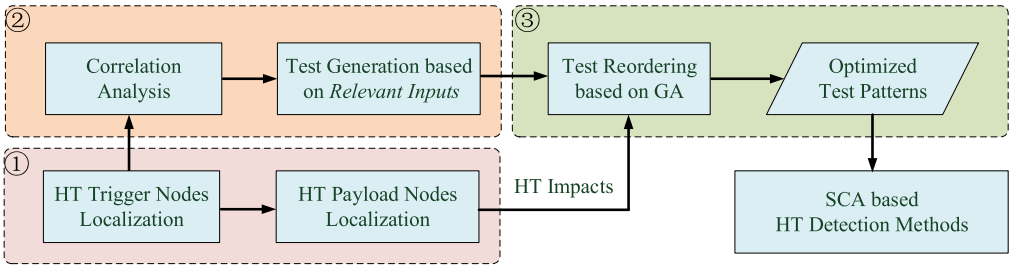


Fig. 2. The framework of our test generation and optimization method.

2.4 An Illustrative Example for Test Reordering

The switching activities on inactive nodes are the result of different logic values determined by two consecutive test vectors. The same set of test vectors will result in different switching activities of HTs under different orders. By applying test vectors in different orders, the proportion of the HT circuit's activities to the whole circuit's activities is different.

The c17 circuit from ISCAS-85 [3] is utilized as a proof-of-concept. Figure 1(a) depicts the original c17 circuit, while Figure 1(b) shows the c17 circuit with HT inserted where internal nodes N_{10} and N_{11} are used to form the HT trigger structure. N_{10} and N_{11} are identified as inactive nodes using random vector statistics method. When HT gets activated, the logic value of N_{19} will be modified. To trigger the HT, we can apply the test set T = (10100, 11100, 11101) to the circuit inputs. The number of switching activities of the original circuit and HT-inserted c17 circuit are 5 and 8, respectively. Therefore, the number of extra switching activities caused by HT is 3. The proportion of the HT circuit's activities to the whole circuit's activities is 0.375. Test set T_{op} = (11100, 10100, 11101) has the same test vectors with T, only in a different order. Applying T_{op} to the same circuits, the number of switching activities of the original circuit and HT-inserted c17 circuit are 9 and 15, respectively. Therefore, the number of extra switching activities caused by HT is 6, which is twice as much as the result of applying the test set T. The proportion of the HT circuit's activities to the whole circuit's activities is 0.4, which is also improved compared with 0.375. Therefore, it is convincing that the proportion of the HT circuit's activities to the whole circuit's activities can be improved by test reordering.

3 TEST GENERATION AND OPTIMIZATION METHODOLOGY

In this section, we discuss the overall framework, algorithms, and steps of the proposed test generation and optimization method. The overall framework is illustrated in Figure 2, where three steps

are implemented sequentially. ① represents the localization of the potential HT trigger nodes and payload nodes. ② is the test generation process based on the *relevant inputs* identified by correlation analysis method. The number of test vectors required to trigger HTs is reduced through this process. ③ is the test optimization process, where the test vectors get reordered to increase the proportion of the HT circuit's activities to the whole circuit's activities, thus SCA-based HT detection sensitivity can be improved. It should be noted that our test generation method is based on the gate-level netlist of circuits. Since the logic function of the gate-level netlist is equivalent to that of the final chip, the switching activity condition in the gate-level netlist can reflect the switching activity condition in the real chip.

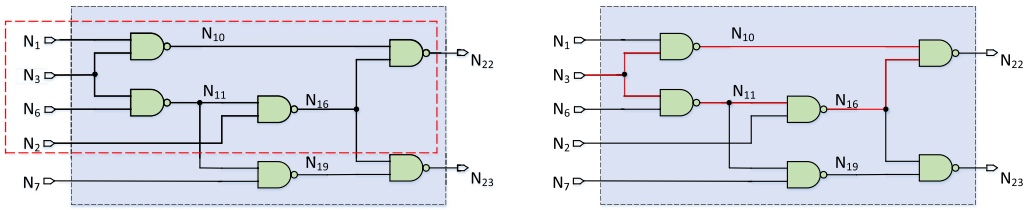
3.1 HT Nodes Localization

In this section, inactive nodes with low transition probability and circuit nodes with high observability value are located, since they are the most potential HT nodes. To locate inactive nodes, a random vector statistics method can be leveraged [16]. Random vector statistics method applies random test vectors to circuit inputs and extracts target nodes' switching activity conditions. After applying a sufficiently large number of test vectors to the circuit, the statistical measure of transition probability of nodes can be obtained, which is a fairly accurate result [16, 43]. Then the nodes whose transition probability is below the threshold θ are screened out to form the inactive nodes list, as Equation (4) shows. Where L represents the inactive nodes list, P_{trans} is the transition probability of circuit nodes. A reasonable selection of the threshold θ is very important in the process of locating inactive nodes. If the threshold θ is set too low, then the selected set of inactive nodes will be incomplete. However, if the threshold θ is set too high, then the selected set of inactive nodes will include several nodes that should be classified into active node sets, and the test generation efficiency will also be lowered. Thus, we set the inactive transition probability threshold θ to be 0.1, where the same setup is also used in other related papers [7, 26, 29].

$$L \leftarrow Nodes|(P_{trans} < \theta) \tag{4}$$

After the random simulation, inactive nodes that are impossible to switch and dependent nodes need to be removed from the inactive nodes list [38]. Inactive nodes that are impossible to switch are not ideal HT trigger nodes, because HTs that use those nodes to form trigger structures will never be triggered. For functionally and structurally dependent nodes, test vectors that can activate some of them will also activate the other nodes, thus, we only keep the key nodes in the inactive nodes list. At this step, for those nodes whose transition probability is still 0, they are removed from the inactive nodes list. Because the goal of HT designers is to make HTs stealthy, not dormant all the time.

To avoid being detected, well-designed HTs are more likely to select nodes with high observability value to deliver the payloads. Because logic changes on these nodes are usually masked from reaching the outputs. By using SCOAP method, the observability values of all circuit nodes are acquired. As an unsupervised clustering method, k-means algorithm can make the analysis less dependent on accurate observability values and circuit scales [16]. Therefore, the k-means clustering algorithm is leveraged to classify the nodes into two categories, which are high observability value nodes and low observability value nodes. The range of the observability value is $[0, \infty]$. The points with the maximum and minimum observability values are selected as the center points for two clusters, respectively. When there are multiple nodes whose observability value is equal to the maximum or minimum value, initial center points will be randomly selected from those nodes. Then every node's observability value is taken to compute its distance to two different center



（a）Logic cone in c17 circuit （b）Reconvergence in c17 circuit

Fig. 3. The netlist of c17 circuit from ISCAS-85. The left picture shows the logic cone of N_{22}, the right picture shows the reconverge path in c17 circuit.

points, which can be expressed as Equation (5):

$$D(x, \mu) = (|x - \mu|)^2, \tag{5}$$

where D is the distance value; x is the observability value of the selected node; μ represents the observability value of the current center point. Based on the distance value, every node is distributed to the cluster whose center point is closer to it. Next, for each cluster, the center point is recomputed and the distance between the new center point and the old one is evaluated. The calculation and classification process is repeated until the distance between the new center point and the old one is within the preset threshold, which means the position of the center point tends to be stable. At this time, the classification result will be accepted.

3.2 Correlation Analysis-based Test Generation

In this section, we propose a static method to perform the correlation analysis between different circuit inputs and inactive nodes. The logic states of the inactive nodes are determined by input test vectors and circuit structures jointly. Considering the MERO process, it can be sure that the logic changes on some of the inputs will not activate the inactive nodes, which means the influence capability of each input on inactive nodes differs from each other [24]. By identifying the *relevant inputs*, the transition probability of inactive nodes can be improved by flipping only these *relevant inputs*. The concept of the *relevant inputs* is similar to the stimulating inputs in Reference [33]: Both of the *relevant inputs* and the stimulating inputs are utilized to increase the switching activity of the inactive nodes. Compared to the PMTP approach used in Reference [33], our correlation analysis method has lower time complexity. The time complexity of PMTP method is $O(N * L)$, where N is the number of inactive nodes and L is the number of circuit levels. While the time complexity of our correlation analysis method is $O(n)$, where n is the number of the circuit inputs.

First, the topological analysis algorithm used to compute the signal probability and transition probability of circuit nodes is introduced, which is denoted as F. In F, the signal probability of circuit nodes is computed by constructing logic cones and using mathematical calculation method [12, 20, 32]. A logic cone of the circuit node consists of combinational logic fanning backward from the node to terminate at primary inputs. The c17 circuit is used to demonstrate the topological analysis algorithm F. As demonstrated in Figure 3(a), the gates and nodes marked within the red dash line is the corresponding logic cone for calculating the signal probability of N_{22}. Starting from circuit inputs, signal probability of target nodes is obtained by using different calculation rules for different gates in logic cones. The signal probability of circuit inputs is set to be 0.5 under uniform distribution condition, which means the probabilities of being 0 and being 1 are equal. For a node S, its transition probability can be computed as Equation (6), where $P_S(1)$ and $P_S(0)$ are the signal

probability of node S to be logic 1 and 0, respectively.

$$\hat{St} = P_S(1) \cdot P_S(0) = P_S(1) \cdot (1 - P_S(1)) \tag{6}$$

However, the main challenge in using the topological analysis algorithm is the presence of reconvergent fan-out nodes. Reconvergent fan-out nodes are those nodes whose fan out >1 and the corresponding branches reconverge within the circuit [12, 20]. Reconvergence of fan-out nodes will result in logical correlations between internal nodes, thus will interfere with the signal probability calculation process and lead to inaccurate signal probability results [16]. As shown in Figure 3(b), the nets in the red color show the reconverge path in the c17 circuit. Both N_{10} and N_{16} are fan-out nodes of input N_3, and they reconverge as a NAND gate's inputs to generate output signal N_{22}. Based on the calculation rules for combinational logic gates, the signal probability of N_{22} can be calculated using F as Equation (7):

$$\begin{aligned} P_{N_{22}}(1) &= 1 - P_{N_{10}}(1) \cdot P_{N_{16}}(1) \\ &= 1 - (1 - P_{N_1}(1) \cdot P_{N_3}(1)) \cdot (1 - (1 - P_{N_6}(1) \cdot P_{N_3}(1)) \cdot P_{N_2}(1)). \end{aligned} \tag{7}$$

It can be seen that the signal probability of N_{10} and N_{16} are both related to circuit input N_3, thus they have logical correlations. However, N_{10} and N_{16} are assumed to be independent of each other as a NAND gate's inputs in Equation (7). Therefore, the transition probability result of N_{22} computed using topological analysis algorithm F will be inaccurate due to the reconvergence of fan-out nodes of input N_3. It shows that the result of F is accurate under ideal conditions, which assumes there are no reconvergent fan-out nodes in the circuit.

Based on the analysis above, the influence of primary input on the target circuit node's logic value will cause the transition probability of the target node to deviate from the accurate value. For circuit input En_i and inactive node S, the signal probability of node S with respect to En_i is denoted as $\hat{S}_{En_i}$, which can be computed as Equation (8). To eliminate the influence of reconvergent fan-out nodes of En_i on S, the logic value of En_i is set to 0 and 1, respectively, to get the conditional signal probability $\hat{S}_{En_i}(0)$ and $\hat{S}_{En_i}(1)$.

$$\begin{aligned} \hat{S}_{En_i} &= \frac{\hat{S}_{En_i}(0) + \hat{S}_{En_i}(1)}{2} \\ &= \frac{F(En_i = 0) + F(En_i = 1)}{2}. \end{aligned} \tag{8}$$

Please note that Equation (8) assumes uniform distribution of the signal probability, which means $P_{En_i}(0) = P_{En_i}(1) = 1/2$. If the distribution is nonuniform, then the Equation (8) is developed to be as Equation (9):

$$\hat{S}_{En_i} = \hat{S}_{En_i}(0) \cdot P_{En_i}(0) + \hat{S}_{En_i}(1) \cdot P_{En_i}(1). \tag{9}$$

In Equation (8), the signal probability of inactive node S is calculated under two mutually exclusive and collectively exhaustive conditions for En_i. Thus, $\hat{S}_{En_i}$ is not affected by the reconvergence of the fan-out nodes of input En_i. The transition probability of node S with respect to En_i is denoted as $\hat{St}_{En_i}$, which can be calculated based on $\hat{S}_{En_i}$. For the inactive node S, its accurate transition probability $\hat{St}$ can be obtained by random vector statistics method. As Equation (10) shows, we denote $diff_i$ as the difference between $\hat{St}$ and $\hat{St}_{En_i}$, $diff_i$ reflects the influence capability that input En_i has on the logic value of inactive node S. In other words, the bigger $diff_i$ is, the more relevant En_i is to the logic state of S.

$$\begin{aligned} diff_i &= |\hat{St} - \hat{St}_{En_i}| \\ &= |\hat{St} - (\hat{S}_{En_i} \cdot (1 - \hat{S}_{En_i}))| \end{aligned} \tag{10}$$

Table 2. Conditional Signal Probability, Transition Probability, and *diff* Value Results of Node N_{22} with Respect to Different Inputs

$S = N_{22}$	$\hat{S}_{En_i}(1)$	$\hat{S}_{En_i}(0)$	$\hat{S}_{En_i}$	$\hat{St}_{En_i}$	*diff*
$En_i = N_1$	3/8	11/16	17/32	0.249	0.244
$En_i = N_3$	5/8	1/2	9/16	0.246	0.247
$En_i = N_6$	5/8	7/16	17/32	0.249	0.244
$En_i = N_2$	13/16	1/4	17/32	0.249	0.244
$En_i = N_7$	17/32	17/32	17/32	0.249	0.244

Using c17 circuit in Figure 3(b) for example, N_{22} is selected as the target node, and its transition probability result from the random vector statistics method is 0.493. The conditional signal probability, transition probability, and *diff* value results of N_{22} with respect to each circuit input are demonstrated in Table 2. It can be seen that the *diff* value of N_3 is higher than that of the other inputs, which means input N_3 is more relevant to the logic state of N_{22}.

Our goal is to identify the inputs that are more relevant to the activation of inactive nodes, rather than the exact values of their correlations. Therefore, the inputs are ranked following the descending order of *diff*, then the top k inputs with the largest *diff* value are screened out as *relevant inputs* list $Ci(S)$ for inactive node S, as Equation (11) shows. One thing to notice here, we did not select all the inputs whose *diff* value is nonzero as *relevant inputs* of node S, because we focus on the most related inputs for higher test generation efficiency.

$$Ci(S) = inputs \leftarrow diff[0:k] \tag{11}$$

Finally, the *relevant inputs* of each inactive node are added to the final *relevant inputs* collection Ci, as Equation (12) shows:

$$Ci = Ci \cup Ci(S). \tag{12}$$

ALGORITHM 1: Test Generation based on *Relevant Inputs*

Input: Random patterns V, inactive nodes list L, *relevant inputs* list Ci, switching requirement N, gate level netlist G.
Output: Test set T.

```
for v in V do
    for p in Ci do
        for S in L do
            S_w ← 0                          ▷ Initialize switching number of S
        Rs ← (G, V, L)                       ▷ Count the number of switches on L
        V' ← v' ← p = 1 − p                  ▷ Update test set
        Rs' ← (G, V', L)
        if Rs' > Rs then
            T ← V'
            else
                T ← V
        if S_w ≥ N for every S in L then
            Break
return T
```

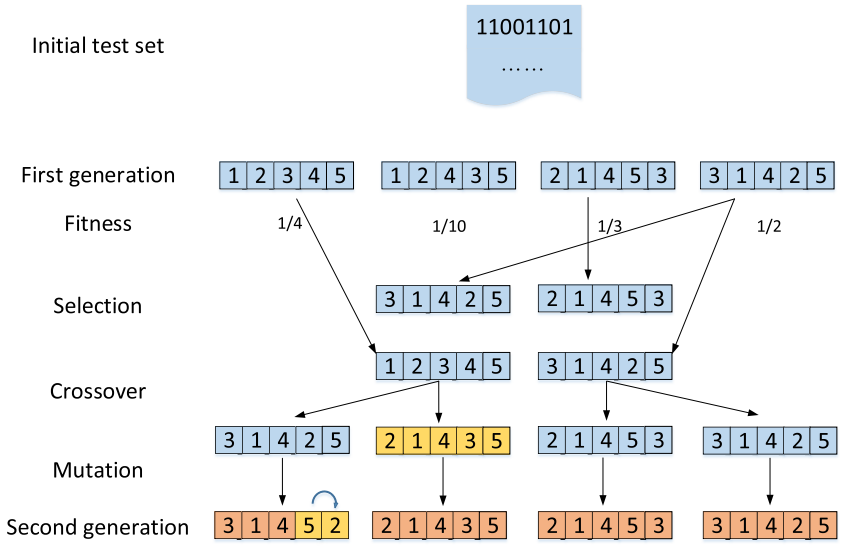


Fig. 4. The procedures of genetic algorithm.

Based on the *relevant inputs* set *Ci*, test patterns that can increase the transition probability of inactive nodes to threshold θ are generated. To maximize the trigger probability of inserted HTs, each of the inactive nodes is triggered to endure logic transition for at least N times. Algorithm 1 shows the steps for our proposed test generation method. For every random vector, the logic value on each *relevant input* is mutated successively (see lines 1~7). If the sum of switching number of all inactive nodes increases under the modified test set, then V' is updated as the new test set. Otherwise, this operation is undone and the test set is left unchanged (see lines 8~11). Lines 12~13 denote the termination condition of the algorithm.

3.3 Genetic Algorithm-based Test Reordering

In this section, we introduce our test reordering method based on genetic algorithm. GA is a well known iterative searching algorithm for seeking the optimal solution; it simulates the biological evolution process. Starting from the initial population, a group of new and more adaptable individuals are generated through selection, crossover, and mutation operations. Individuals in the population continue to breed and evolve, finally converging to a group of the most suitable individuals, so the optimal solution of object problem is obtained. Our goal is to find the best order for test patterns to improve the proportion of the HT circuit's activities to the whole circuit's activities, and GA is utilized to search for feasible orders extensively.

The reordering method based on GA can be expressed as Equation (13), where C is chromosome coding rules for individual, E is the fitness function, P_o is the initial generation, M is the size of the population, Φ is the selection operator, Γ is the crossover operator, Ψ is the mutation operator, and T_e is the termination condition of GA.

$$Reorder_{GA} = (C, E, P_o, M, \Phi, \Gamma, \Psi, T_e) \tag{13}$$

The flow of our reordering method based on GA is shown in Figure 4; the test set used for example contains five test vectors. The inputs of the reordering method include test set T generated based on *relevant inputs*, the genetic algorithm parameter such as population size M, mutation rate, crossover rate, and termination condition T_e.

First, the population is initialized. An individual in the population is a specific order for test set T, namely, S_i. While the genes on the only chromosome of the individual are the test patterns distributed to different positions. The first generation P_o that contains multiple distinct orders for test set T is generated randomly, as Equation (14) shows:

$$S_i = randperm(T). \tag{14}$$

Based on S_i, test patterns are reordered to form a new test set to compute the fitness value. Setting fitness function is the key to genetic algorithm. The fitness function directly determines the direction of evolution and optimization effect. In our method, the number of switching activities difference between HT-inserted circuit and golden reference is denoted as *extra_switching*. While the number of switching activities of HT-inserted circuit is denoted as *total_switching*. To improve the proportion of the HT circuit's activities to the whole circuit's activities, the fitness function is defined as Equation (15):

$$E(S_i) = \frac{extra\_switching}{total\_switching}. \tag{15}$$

Test patterns in T are reordered based on each S_i and applied to circuit inputs. After the simulation, the fitness value of each individual is calculated using Equation (15). Then, the selection process is carried out to pick out the parent individuals to reproduce the next generation. Ranking selection strategy is used in the selection process: All individuals in the population are ranked in descending order of fitness value. As Equation (16) shows, the probability of every individual being selected is denoted as $P_{selection}(S_i)$, which is assigned based on the ranking order $rank(S_i)$. One thing to notice here, the same individual is allowed to be selected more than once. As Figure 4 shows, the individual "31425" in the first generation has the largest fitness value and is selected twice.

$$\Phi = P_{selection}(S_i) \leftarrow rank(S_i) \tag{16}$$

The selected parent individuals reproduce children through the crossover process. Like the biological world, the chromosomes of filial individuals are the combination of the parents' chromosomes. In our method, the mean value of the two parent individuals $S_{parent1}$ and $S_{parent2}$ on the same gene locus is computed first, then the children individual S_{child} is generated by round up or down on each gene locus of the individual, as Equation (17) shows. To ensure a constant population size, the one with higher fitness value in the parents will be retained in the new generation. As Figure 4 shows, the individual "21435" in yellow color is the filial individual generated through crossover operation, and its parent individual "31425" is retained as the one with higher fitness value in parent individuals.

$$\Gamma = S_{child} \leftarrow round(mean(S_{parent1} + S_{parent2})) \tag{17}$$

After the crossover process is finished, the mutation process is performed to prevent the population from falling into a locally optimal solution. For every gene locus in an individual, when mutation condition is met, test vector on another randomly selected gene locus and test vector at current gene locus will be swapped, as Equation (18) shows. In Figure 4, the individual "31425" becomes "31452" after the mutation process.

$$\Psi = S_{child} \leftarrow randswap(S_{child}) \tag{18}$$

Finally, when the termination condition T_e is met, the best individual in the evolution process will be chosen as the final result. By adjusting test patterns with this order, the proportion of the HT circuit's activities to the whole circuit's activities can be improved to assist SCA-based HT detection.

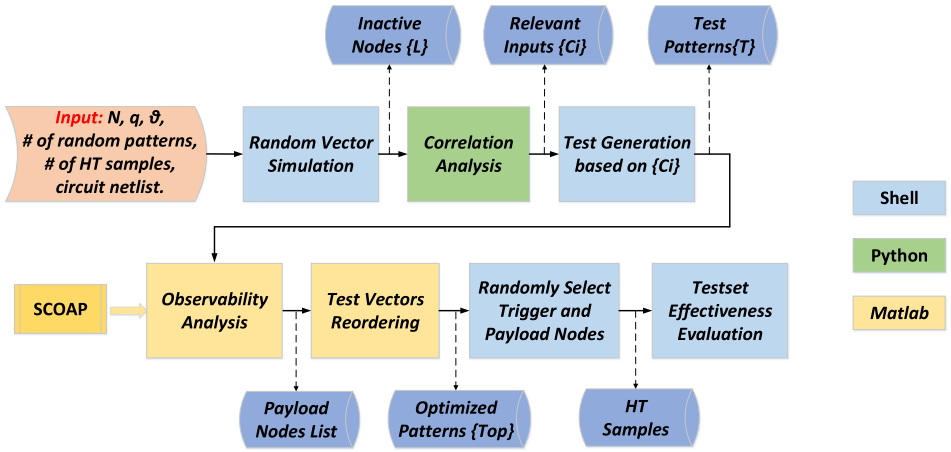


Fig. 5. The test generation and optimization procedures.

4 EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Simulation Setup

In this section, the test generation procedures, experimental settings, and corresponding tools are introduced. The inactive nodes' localization, test generation, and test effectiveness evaluation are implemented using the Shell script. Whereas correlation analysis is performed using Python, payload nodes localization and test optimization are implemented using Matlab.

Figure 5 shows the procedures of our proposed method. The inputs of our method are first demonstrated. N is the number of switching each inactive node needs to endure, q is the number of inactive nodes that will be combined to form HT trigger structure, and θ is the transition probability threshold. First, random patterns are generated and applied to circuits to filter out the inactive nodes whose transition probability is below the preset threshold θ. The switching activity extraction process is realized by analyzing the switching activity interchange format (SAIF) file generated by Verilog Compile Simulator (VCS). Based on the inactive nodes list L, correlation analysis is performed using Python and outputs the list of *relevant inputs Ci*. Then, test patterns that can activate the HTs efficiently are generated based on *relevant inputs*. Next, the k-means algorithm is used to sort out the nodes with higher observability value, which are more likely to be selected to deliver HT payloads. After this, the best order for test patterns is explored using GA. Finally, the effectiveness of test vectors is evaluated from the improvement of inactive nodes' transition probability, the number of test vectors required to trigger HTs, HT trigger coverage, test generation time, and the proportion of the HT circuit's activities to the whole circuit's activities. To validate the effectiveness and scalability of our proposed method, several representative test circuits are utilized, including a subset of ISCAS-85 and ISCAS-89 benchmark circuits, OpenCores benchmark circuits, and the ITC'99 benchmark circuits [31, 36]. All simulations and test generation are carried out on a Linux workstation with a 3.47 GHz Intel processor and 4 GB RAM.

4.2 Test Set Evaluation Results

Inactive nodes' transition probability: In this section, the improvement of inactive nodes' transition probability after applying optimized test patterns are first demonstrated in Table 3. The first column presents the benchmark circuits used in our experiments. Column 2 shows the number of *relevant inputs* and all the inputs in test circuits; it can be seen that only a small subsection

Table 3. Improvement of Inactive Nodes' Transition Probability

Benchmarks	Inputs (relevant/all)	Nodes (inactive/total)	Average Improve Factor
c880	28 / 60	17 / 443	19.4X
c2670	22 / 157	35 / 1,350	10.3X
c3540	27 / 50	55 / 1,719	5.3X
c5315	19 / 178	29 / 2,485	7.1X
c6288	31 / 32	43 / 2,448	7.0X
c7552	67 / 207	81 / 3,720	5.2X
s13207	41 / 62	304 / 8,651	6.5X
s15850	36 / 77	337 / 10,383	29.7X
s35932	30 / 35	270 / 17,825	5.6X
s38417	25 / 28	1148 / 23,843	19.3X
wb_conmax	304 / 1,128	1762 / 22,070	4.4X
b17	28 / 37	158 / 24,212	1.5X
Average	-	-	10.1X

of primary inputs are *relevant inputs* for most of the circuits. By focusing on these *relevant inputs*, the test generation process is more targeted and efficient compared with MERO and MERS method. Column 3 lists the number of total nodes and inactive modes identified by the random vector statistics method with a threshold $\theta = 0.1$. The transition probabilities are estimated under 1M random vectors. The average improve factor of inactive nodes' transition probability is demonstrated in column 4. Compared with random test vectors, the transition probability of inactive nodes is improved by 10.1 times on average, which is of great significance in increasing HT trigger probability.

Test length, HT trigger coverage, and test generation time comparison: Our test generation and optimization methods aim to activate the HTs that use inactive nodes' combination to form up their trigger structures by promoting inactive nodes' transition probabilities. Therefore, the number of test vectors required to trigger HTs and HT trigger coverage of MERO method, MERS method, and our method are compared. Since MERS uses MERO method to generate the initial test set and only reorders test vectors for optimization, the number of test vectors required to trigger HTs and trigger coverage is the same for MERO and MERS methods. To demonstrate the efficiency of our method, the test generation time of the MERO method, MERS method, and our method are also compared. The experiments are conducted under the same settings of $N = 1{,}000$, $q = 4$, and $\theta = 0.1$. Which means all the inactive nodes' transition probability is below 0.1. All the inactive nodes can endure at least 1,000 times of transitions under improved test vectors. When q gets larger, the trigger probability of HTs gets smaller. However, the size of HTs will be larger at the same time and HTs are more likely to be detected using SCA. Therefore, we set $q = 4$, the maximum trigger probability of HT samples can only reach 10^{-4}, which can provide enough concealment for HTs while maintaining a small size.

As shown in Table 4, the number of test vectors required to meet the switching requirement of MERO and MERS methods is listed in column 2. Column 3 lists the number of test vectors required by our method to meet the switching requirement. It is obvious that the switching requirement can be satisfied with fewer test vectors when using our test generation method. The number of test vectors needed by our method reduces 28.8% on average compared with MERO and MERS methods. This is due to the accurate identification of HT trigger nodes and significant improvement of inactive nodes' transition probability. In terms of the trigger coverage, our method also outperforms MERO and MERS methods. Compared with MERO and MERS methods,

Table 4. Comparison of Test Length, HT Trigger Coverage, and Test Generation Time among MERO Patterns, MERS Patterns, and Test Patterns Generated by Our Method for $N = 1{,}000, q = 4, \theta = 0.1$

Benchmarks	Number of test vectors			Trig.Cov.(%)			Test generation time(s)		
	MERO & MERS [29]	Our method	Reduction Rate(%)	MERO & MERS [29]	Our method	Improvement (%)	MERO [7]	MERS [17]	Our method
c880	6,284	5,000	20.4	75.92	97.94	29.00	-	-	5,371
c2670	9,340	7,375	21.0	62.66	90.26	44.05	30,051.53	13,370.86	16,800
c3540	15,900	10,060	36.7	55.02	75.63	37.46	9,403.11	6,097.51	10,127
c5315	15,850	7,060	55.5	43.50	84.33	93.86	80,241.52	45,595.97	7,409
c6288	5,014	4,032	19.6	92.50	91.29	-1.31	15,716.42	4,154.62	11,027
c7552	16,358	9,370	42.7	45.07	78.05	73.18	160,783.37	81,405.89	27,337
s13207	26,926	20,000	25.7	94.44	81.65	12.62	23,432.04	12,511.95	39,838
s15850	36,992	33,334	9.9	36.00	65.50	81.94	39,689.63	19,903.44	37,216
s35932	7,343	5,000	31.9	62.49	80.22	28.37	29,810.49	7,295.74	24,234
s38417	52,735	40,000	24.1	21.07	53.77	155.20	-	-	24,294
wb_conmax	-	60,300	-	-	67.54	-	-	-	43,261
b17	-	12,000	-	-	72.61	-	-	-	14,316
Average	-	-	28.8	-	-	55.44	48,641.01	23,792.00	21,769.17

trigger coverage for combinational HTs that use four inactive nodes to form trigger structures of our method improves by 55.44% on average. The premise to achieve this effect is that the number of test vectors needed in our method is less than any other method for the same switching requirement (N=1,000). The HT sample size used to estimate the trigger coverage is 100,000. As for the test generation time, the average test generation time of our method on different benchmarks is less than half of the result of the MERO method, because only a subsection of the circuit inputs will be modified during the iteration process. The average test generation time of our method is close to that of the MERS method. Since the MERS method uses the MERO method to generate the initial test set, the reason should be that the implementation manner and the experimental environment are different for our method and the MERS method. The MERS core algorithm is implemented using C [18], while our test generation process is controlled by a Shell script.

Test effectiveness evaluation for mixed-trigger type HTs and elaborated HTs: The HT samples used to estimate the trigger coverage in Table 4 are combinational HTs that use four inactive nodes as trigger signals. However, attackers may intentionally use active nodes and inactive nodes together to form HT trigger structures to avoid detection. To validate the applicability of our method to the HTs that have partial trigger signals coming from active nodes, we evaluate the trigger coverage on 100,000 randomly selected Trojan samples whose four trigger signals include both inactive nodes and active nodes. As shown in Table 5, the trigger coverage for mixed-trigger type HTs slightly decreases compared to the result for HTs whose trigger signals are all inactive nodes. The reason is that the test patterns are generated aiming at promoting the transition probabilities of the inactive nodes, thus the activities of the active nodes have not been significantly improved.

To further validate the effectiveness of our proposed method, we conduct the experiments on 10 HT-inserted benchmarks that contain elaborated combinational and sequential HTs. As shown in Table 6, all the combinational HTs are successfully triggered using our test generation method except for wb_conmax-T100. The wb_conmax-T100 is not triggered, because some of the HT trigger nodes are removed during the process of synthesizing, which is a common situation according to the study in Reference [28]. Even so, all the six remaining trigger nodes for wb_conmax-T100 are

Table 5. HT Trigger Coverage Evaluation under Test Patterns Generated by Our Method over 100,000 Random HT Samples That Use up to Four Inactive Nodes to Form Trigger Structures for $N = 1{,}000, \theta = 0.1$

Benchmarks	Trig.Cov.(%)			
	4-inactive/0-active	3-inactive/1-active	2-inactive/2-active	1-inactive/3-active
c880	97.94	87.31	85.89	89.62
c2670	90.26	88.61	84.78	89.47
c3540	75.63	72.71	67.76	74.52
c5315	84.33	80.29	75.64	77.12
c6288	91.29	88.25	84.32	85.44
c7552	78.05	73.34	69.18	70.01
s13207	81.65	75.20	70.16	71.78
s15850	65.50	60.54	56.61	58.86
s35932	80.22	75.66	70.18	74.15
s38417	53.77	48.65	44.81	45.53
wb_conmax	67.54	60.22	55.76	56.05
b17	72.61	69.76	60.15	67.45
Average	78.23	73.38	68.77	71.67

Table 6. Test Effectiveness Evaluation on HT-inserted Benchmarks That Contain Elaborated Combinational and Sequential HTs [4, 10]

HT-inserted circuits	Conbinational HTs				Sequential HTs					
	s35932-T300	s38417-T200	wb_conmax-T100	vga_lcd-T100	s13207-T400	s13207-T464	s15850-T419	s15850-T488	s35932-T405	s35932-T619
Triggered?	YES	YES	NO	YES	YES	YES	YES	NO	YES	YES

Table 7. Comparison of the Number of Test Vectors, HT Trigger Coverage, and Runtime with Different Values of θ for c880 Circuit($N = 1000, q = 4$)

θ	0.01	0.02	0.03	0.04	0.05	0.1	0.15	0.2	0.25	0.3
Number of test vectors	2,660	3,500	3,710	4,230	4,680	5,000	5,260	5,500	5,670	5,990
Trig.Cov.(%)	54.51	59.29	65.47	70.11	76.25	97.94	94.31	91.30	87.04	88.11
Run-time(s)	1,291	2,674	3,015	4,132	4,469	5,371	5,540	5,966	6,217	6,596

successfully triggered simultaneously. However, all the sequential HTs are successfully triggered except for s15850-T488. The reason why not all the sequential HTs are triggered is that the trigger conditions of sequential HTs are more complicated and diverse, such as the rare values of internal nodes and the state transition of FSM. In our test generation method, we focus on increasing the transition probability of inactive nodes, which could be not sufficient for some sequential HTs. But still, our method shows favorable effectiveness for triggering most of the sequential HTs.

Comparison of test effectiveness for varying θ: To demonstrate the influences that threshold θ has on test effectiveness and test generation efficiency, the number of test vectors required to trigger inactive nodes for N times, trigger coverage for HT samples that use four inactive nodes as trigger inputs, and the test generation time for varying θ values are compared for c880 benchmark circuit. As shown in Table 7, as the threshold θ increases, the number of test

Table 8. Comparison of the Proportion of the HT Circuit's Activities to the Whole Circuit's Activities among Random Patterns (10K), MERO Patterns, MERS Patterns, and Test Patterns Generated by Our Method over 1,000 Random HT Samples That Use Four Inactive Nodes to Form Trigger Structures for $N = 1{,}000, \theta = 0.1$

Benchmarks	HT circuit's activities / The whole circuit's activities						
	Random	MERO [18]	MERS-s [18]	Our Method	Improvement to Random(%)	Improvement to MERO(%)	Improvement to MERS-s(%)
c880	0.00519	-	-	0.04397	747.21	-	-
c2670	0.00368	0.02571	0.03308	0.02379	546.47	−7.47	−728.08
c3540	0.00458	0.04238	0.11067	0.01869	308.08	−755.90	−783.11
c5315	0.00156	0.01082	0.01586	0.00762	388.46	−729.57	−751.95
c6288	0.00339	0.00395	0.00896	0.01830	439.82	363.29	104.24
c7552	0.00300	0.00737	0.01168	0.01301	333.67	76.53	11.39
s13207	0.00635	0.00617	0.00826	0.03893	513.07	530.96	371.31
s15850	0.00427	0.00487	0.00634	0.03039	611.71	524.02	379.34
s35932	0.00174	0.00463	0.00512	0.00803	361.49	73.43	56.84
s38417	0.00108	-	-	0.00293	171.30	-	-
wb_conmax	0.08076	-	-	0.21751	169.33	-	-
b17	0.06660	-	-	0.08638	29.70	-	-
Avg.Improve	-	-	-	-	385.03	184.41	95.00

vectors required to meet the switching requirements also increases, because the number of selected inactive nodes gets larger. At the same time, the test generation time also increases for larger θ. It is also observed that our proposed test generation method achieves the maximum HT trigger coverage at $\theta = 0.1$. The HT trigger coverage will gradually decrease no matter whether the θ is set to higher or lower values. The reasons are as follows: For low θ values, the HTs that use those inactive nodes as trigger inputs are hard to trigger, so the trigger coverage is low. For higher θ values, more circuit nodes that have high activities will be introduced. Those nodes are easy to activate, thus the HT trigger coverage gets higher. With the increase of θ, the test generation process should consider more nodes simultaneously, thus the improvement of the activity regarding low transition probability nodes is unobvious. Consequently, the HT trigger coverage will decrease after reaching the maximum value.

Side channel sensitivity improvement: To improve the proportion of the HT circuit's activities to the whole circuit's activities, the test vectors get reordered. In our experiments, the population size is set to be 100, the crossover probability is set to be 0.6, the mutation probability is set to be 0.05, which are all typical values for corresponding parameters. The *extra_switching* and *total_switching* values are evaluated over 1,000 random HT samples that contain four trigger nodes and one payload node. When the HT gets activated, the logic value of the payload node will be modified. As shown in Table 8, column 1 lists the benchmark circuits used in our experiments, the ratio of *extra_switching* caused by HTs to the *total_switching* of the HT-inserted circuits under random patterns (10K), MERO patterns, MERS-s patterns, and test vectors generated by our method are presented in columns 2∼5. After the reordering process, the ratio of *extra_switching* to *total_switching* is improved by 385.03% on average compared with the result under random vectors. Compared to MERS-s vectors in Reference [18], the ratio of *extra_switching* to *total_switching* under reordered test vectors is improved by 95% on average. The experimental results show that our

reordering method can improve the proportion of the HT circuit's activities to the whole circuit's activities, thus can improve SCA-based HT detection sensitivity.

4.3 Application to Side Channel Analysis

As observed from the results presented in the former subsection, our proposed approach can increase inactive nodes' transition probability significantly. Thereby, HTs can be activated with fewer test vectors using our method under the same condition compared with MERO and MERS methods. To improve the proportion of the HT circuit's activities to the whole circuit's activities, the test patterns get further tuned by reordering. Our approach can be used to generate test vectors for SCA-based HT detection methods, which require highlighting HT circuit's impacts on side channel signatures like power consumption.

5 CONCLUSION AND FUTURE WORK

Existing test generation methods for SCA have disadvantages such as poor HT trigger efficiency and inadequate consideration of improving the proportion of the HT circuit's activities to the whole circuit's activities. We have presented a test generation and optimization method for SCA-based HT detection using correlation analysis and test reordering. Experiments on 10 selected ISCAS benchmarks, wb_conmax benchmark, and b17 benchmark show that the inactive nodes' transition probability is improved to 10.1 times on average compared with the result under random patterns. Our method outperforms MERO and MERS methods in the number of test vectors required to meet the same switching requirement. The reduction rate of the number of test vectors is 28.8% on average. At the same time, trigger coverage for HTs that use four inactive nodes to form trigger structures of our method improves by 55.44% on average, compared with MERO and MERS methods. After the test vector reordering process, the proportion of the HT circuit's activities to the whole circuit's activities is improved by 95% on average, compared with the result under MERS-s vectors. It shows that our proposed test generation method can improve HT detection ability and sensitivity using SCA.

Future work will involve improving the test vectors' effectiveness, which means higher HT trigger probability and more applicability to sequential HTs. Besides, more kinds of reordering methods will be combined with the correlation analysis method to meet the requirements of different application scenarios. For example, the simulation-based reordering method can be combined with the correlation analysis method to improve the side channel sensitivity for situations where the number of test vectors is not large.

REFERENCES

[1] M. E. Amyeen, S. Venkataraman, A. Ojha, and Sangbong Lee. 2004. Evaluation of the quality of N-detect scan ATPG patterns on a processor. In *the International Test Conference*. 669–678. DOI: https://doi.org/10.1109/TEST.2004.1387328

[2] C. Bao, D. Forte, and A. Srivastava. 2015. Temperature tracking: Toward robust run-time detection of hardware Trojans. *IEEE Trans. Comput.-aided Des. Integ. Circ. Syst.* 34, 10 (2015), 1577–1585.

[3] ISCAS benchmark circuits. 2007. Retrieved from http://www.pld.ttu.ee/ maksim/benchmarks/.

[4] Trust-Hub benchmark circuits. 2013. Retrieved from https://www.trust-hub.org/benchmarks/chip-level-trojan.

[5] R. S. Chakraborty, S. Narasimhan, and S. Bhunia. 2009. Hardware Trojan: Threats and emerging solutions. In *Proceedings of the IEEE International High Level Design Validation and Test Workshop*. 166–171. DOI: https://doi.org/10.1109/HLDVT.2009.5340158

[6] R. S. Chakraborty, S. Pagliarini, J. Mathew, S. R. Rajendran, and M. N. Devi. 2017. A flexible online checking technique to enhance hardware Trojan horse detectability by reliability analysis. *IEEE Trans. Emerg. Topics Comput.* 5, 2 (Apr. 2017), 260–270. DOI: https://doi.org/10.1109/TETC.2017.2654268

[7] Rajat Subhra Chakraborty, Francis Wolff, Somnath Paul, Christos Papachristou, and Swarup Bhunia. 2009. MERO: A statistical approach for hardware Trojan detection. In *Proceedings of the Cryptographic Hardware and Embedded Systems (CHES'09)*, Christophe Clavier and Kris Gaj (Eds.). Springer Berlin, 396–410.

[8] Mingsong Chen, Xiaoke Qin, Heon Mo Koo, and Prabhat Mishra. 2012. *System-level Validation: High-level Modeling and Directed Test Generation Techniques*. Springer Publishing Company, Incorporated.

[9] Wang Chin Chen and Augusli Kifli. 2011. High speed ATPG testing circuit and method. U.S. Patent 0050030 [P]. 2010-2-25.

[10] J. Cruz, Y. Huang, P. Mishra, and S. Bhunia. 2018. An automated configurable Trojan insertion framework for dynamic trust benchmarks. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'18)*. 1598–1603. DOI: https://doi.org/10.23919/DATE.2018.8342270

[11] S. Dupuis, M. Flottes, G. Di Natale, and B. Rouzeyre. 2018. Protection against hardware Trojans with logic testing: Proposed solutions and challenges ahead. *IEEE Des. Test* 35, 2 (2018), 73–90.

[12] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco. 1989. Estimate of signal probability in combinational logic networks. In *Proceedings of the 1st European Test Conference*. 132–138.

[13] L. H. Goldstein and E. L. Thigpen. 1980. SCOAP: Sandia controllability/observability analysis program. In *Proceedings of the 17th Design Automation Conference*. 190–196. DOI: https://doi.org/10.1109/DAC.1980.1585245

[14] Jiaji He, Yanjiang Liu, Yidong Yuan, Kai Hu, and Yiqiang Zhao. 2018. Golden chip free Trojan detection leveraging electromagnetic side channel fingerprinting. *ICE Electron. Exp.* 16, 2 (2018).

[15] J. He, Y. Zhao, X. Guo, and Y. Jin. 2017. Hardware Trojan detection through chip-free electromagnetic side-channel statistical analysis. *IEEE Trans. Very Large Scale Integ. (VLSI) Syst.* 25, 10 (2017), 2939–2948.

[16] Y. He and K. Huang. 2019. Trigger identification using difference-amplified controllability and dynamic transition probability for hardware Trojan detection. *IEEE Trans. Inf. Forens. Sec.* 15 (2019), 3387–3400.

[17] Yuanwen Huang, Swarup Bhunia, and Prabhat Mishra. 2016. MERS: Statistical test generation for side-channel analysis based Trojan detection. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*. ACM, New York, NY, 130–141. DOI: https://doi.org/10.1145/2976749.2978396

[18] Y. Huang, S. Bhunia, and P. Mishra. 2018. Scalable test generation for Trojan detection using side channel analysis. *IEEE Trans. Inf. Forens. Sec.* 13, 11 (Nov. 2018), 2746–2760. DOI: https://doi.org/10.1109/TIFS.2018.2833059

[19] C. H. Kok, C. Y. Ooi, M. Moghbel, N. Ismail, H. S. Choo, and M. Inoue. 2019. Classification of Trojan nets based on SCOAP values using supervised learning. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'19)*. 1–5. DOI: https://doi.org/10.1109/ISCAS.2019.8702462

[20] B. Krishnamurthy and I. G. Tollis. 1989. Improved techniques for estimating signal probabilities. *IEEE Trans. Comput.* 38, 7 (1989), 1041–1045.

[21] Yanjiang Liu, Jiaji He, Haocheng Ma, and Yiqiang Zhao. 2019. Hardware Trojan detection leveraging a novel golden layout model towards practical applications. *J. Electron. Test.* 35, 11 (2019).

[22] Y. Liu, Y. Zhao, J. He, and A. Liu. 2017. A novel test pattern optimization approach based on ring oscillator network. In *Proceedings of the 9th International Conference on Intelligent Human-machine Systems and Cybernetics (IHMSC'17)*, Vol. 1. 243–247.

[23] Y. Liu, Y. Zhao, J. He, A. Liu, and R. Xin. 2017. SCCA: Side-channel correlation analysis for detecting hardware Trojan. In *Proceedings of the 11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID'17)*. 196–200.

[24] Y. Lyu and P. Mishra. 2019. Efficient test generation for Trojan detection using side channel analysis. In *Proceedings of the Design, Automation Test in Europe Conference Exhibition (DATE'19)*. 408–413. DOI: https://doi.org/10.23919/DATE.2019.8715179

[25] S. Narasimhan, D. Du, R. S. Chakraborty, S. Paul, F. G. Wolff, C. A. Papachristou, K. Roy, and S. Bhunia. 2013. Hardware Trojan detection by multiple-parameter side-channel analysis. *IEEE Trans. Comput.* 62, 11 (2013), 2183–2195.

[26] M. A. Nourian, M. Fazeli, and D. Hely. 2018. Hardware Trojan detection using an advised genetic algorithm based logic testing. *J. Electron. Test.* 34, 4 (01 Aug. 2018), 461–470. DOI: https://doi.org/10.1007/s10836-018-5739-4

[27] I. Pomeranz and S. M. Reddy. 2004. A measure of quality for n-detection test sets. *IEEE Trans. Comput.* 53, 11 (Nov. 2004), 1497–1503. DOI: https://doi.org/10.1109/TC.2004.87

[28] T. Reece and W. H. Robinson. 2013. Analysis of data-leak hardware Trojans in AES cryptographic circuits. In *Proceedings of the IEEE International Conference on Technologies for Homeland Security (HST'13)*. 467–472. DOI: https://doi.org/10.1109/THS.2013.6699049

[29] Sayandeep Saha, Rajat Subhra Chakraborty, Srinivasa Shashank Nuthakki, Anshul, and Debdeep Mukhopadhyay. 2015. Improved test pattern generation for hardware Trojan detection using genetic algorithm and Boolean satisfiability. In *Proceedings of the Cryptographic Hardware and Embedded Systems (CHES'15)*, Tim Güneysu and Helena Handschuh (Eds.). Springer Berlin, 577–596.

[30] H. Salmani. 2017. COTD: Reference-free hardware Trojan detection and recovery based on controllability and observability in gate-level netlist. *IEEE Trans. Inf. Forens. Sec.* 12, 2 (Feb. 2017), 338–350. DOI: https://doi.org/10.1109/TIFS.2016.2613842

[31] H. Salmani, M. Tehranipoor, and R. Karri. 2013. On design vulnerability analysis and trust benchmarks development. In *Proceedings of the IEEE 31st International Conference on Computer Design (ICCD'13)*. 471–474. DOI: https://doi.org/10.1109/ICCD.2013.6657085

[32] H. Salmani, M. Tehranipoor, and J. Plusquellic. 2012. A novel technique for improving hardware Trojan detection and reducing Trojan activation time. *IEEE Trans. Very Large Scale Integ. (VLSI) Syst.* 20, 1 (Jan. 2012), 112–125. DOI: https://doi.org/10.1109/TVLSI.2010.2093547

[33] A. Shabani and B. Alizadeh. 2021. Enhancing hardware Trojan detection sensitivity using partition-based shuffling scheme. *IEEE Trans. Circ. Syst. II: Exp. Briefs* 68, 1 (Jan. 2021), 266–270.

[34] A. Shabani and B. Alizadeh. 2020. PMTP: A MAX-SAT-based approach to detect hardware Trojan using propagation of maximum transition probability. *IEEE Trans. Comput.-aided Des. Integ. Circ. Syst.* 39, 1 (2020), 25–33.

[35] Ahmad Shabani and Bijan Alizadeh. 2020. PODEM: A low-cost property-based design modification for detecting hardware Trojans in resource-constraint IoT devices. *J. Netw. Comput. Applic.* 167 (2020). Retrieved from http://dx.doi.org/10.1016/j.jnca.2020.102713.

[36] Bicky Shakya, Tony He, Hassan Salmani, Domenic Forte, Swarup Bhunia, and Mark Tehranipoor. 2017. Benchmarking of hardware Trojans and maliciously affected circuits. *J. Hardw. Syst. Sec.* 1 (04 2017). DOI: https://doi.org/10.1007/s41635-017-0001-6

[37] R. Shende and D. D. Ambawade. 2016. A side channel based power analysis technique for hardware Trojan detection using statistical learning approach. In *Proceedings of the 13th International Conference on Wireless and Optical Communications Networks (WOCN'16)*. 1–4.

[38] Sying-Jyan Wang, Jhih-Yu Wei, Shih-Heng Huang, and K. S. Li. 2016. Test generation for combinational hardware Trojans. In *Proceedings of the IEEE Conference on Asian Hardware-Oriented Security and Trust (AsianHOST'16)*. 1–6.

[39] M. Tehranipoor and F. Koushanfar. 2010. A survey of hardware Trojan taxonomy and detection. *IEEE Des. Test. Comput.* 27, 1 (Jan. 2010), 10–25. DOI: https://doi.org/10.1109/MDT.2010.7

[40] Xin Xie, Yangyang Sun, Hongda Chen, and Yong Ding. 2017. Hardware Trojans classification based on controllability and observability in gate-level netlist. *IEICE Electron. Exp.* 14, 18 (2017), 20170682–20170682. DOI: https://doi.org/10.1587/elex.14.20170682

[41] L. Zhang, Y. Dong, J. Wang, C. Xiao, and D. Ding. 2019. A hardware Trojan detection method based on the electromagnetic leakage. *China Commun.* 16, 12 (2019), 100–110.

[42] Z. Zhou, U. Guin, and V. D. Agrawal. 2018. Modeling and test generation for combinational hardware Trojans. In *Proceedings of the IEEE 36th VLSI Test Symposium (VTS'18)*. 1–6.

[43] M. Zou, X. Cui, L. Shi, and K. Wu. 2018. Potential trigger detection for hardware Trojans. *IEEE Trans. Comput.-aided Des. Integ. Circ. Syst.* 37, 7 (July 2018), 1384–1395. DOI: https://doi.org/10.1109/TCAD.2017.2753201

Received August 2020; revised December 2020; accepted January 2021