IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 12, 2019, accepted March 23, 2019, date of publication March 28, 2019, date of current version April 16, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2908088

A Hardware Trojan Detection Method Based on Structural Features of Trojan and Host Circuits

QIANG LIU, (Member, IEEE), PENGYONG ZHAO, AND FUQIANG CHEN
Tianjin Key Laboratory of Imaging and Sensing Microelectronic Technology, School of Microelectronics, Tianjin University, Tianjin 300072, China
Corresponding author: Qiang Liu (qiangliu@tju.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61574099 and in part by the Tianjin Municipal Transportation Science and Technology Development Plan Project under Grant 2017B-40.

ABSTRACT Modern IC designs often involve outsourced IP cores. It is convinced that there are opportunities in which the IP cores contain malicious logic, namely hardware Trojan (HT), which raises serious concerns about the trustworthiness of ICs used in mission-critical applications. This paper proposes an HT detection method by analyzing the combined structural features of HTs and host circuits. The structural features of combinational and sequential logic HTs are extracted and form an HT feature database. An efficient quantization approach on feature matching is proposed to search the features from circuit designs. The experiments conducted on TrustHub benchmarks show that the proposed method can successfully detect all stealth Trojans with runtime within 72 s on a platform with an AMD 2.00-GHz CPU with 4-GB RAM and a low false positive rate compared with the existing methods.

INDEX TERMS Structure features, hardware Trojan, gate-level circuits.

I. INTRODUCTION

Modern IC designs often involve third-party intellectual property (IP) cores because of the high design complexity, the restriction of time-to-market and the cost constraint of final products [1]. One of the challenges faced by the IP reuse-based design methodology is the untrustworthiness of the outsourced IPs. It is convinced that there are opportunities in which the IP cores contain malicious extra logic, namely Hardware Trojan (HT). HTs can cause the circuit failure or leak confidential information, which raise serious concerns about trustworthiness of ICs used in Internet of Things (IoT) [2] and consumer electronics (CE) [3], as well as mission critical applications [4].

In addition to the design stage, HTs can also be inserted during and after IC fabrication. Usually a HT is quietly hidden in its host circuit and can only be triggered in rare conditions [5]. To detect the silent HTs, currently there are three categories of method: logic testing [6]–[10], side-channel analysis [11]–[16] and feature analysis [17]–[22]. Logic testing approaches [6]–[10] attempt to generate a large number of test vectors to activate unknown HTs and propagate their effects to the output ports. The test vector generation and application usually require significant amount of time and do not guarantee a hundred percent coverage. The logic testing approaches can be applied to detect HTs at both pre- and post-fabrication stages. The side-channel analysis approaches detect HTs by finding circuit delay, power consumption and electromagnetic radiation differences caused by HTs, even not activated. The effectiveness of side-channel analysis approaches is greatly affected by process variations and usually depends on a golden reference design. The side-channel analysis approaches detect HTs at the post-fabrication stage. In contrast, feature analysis approaches [17]–[22], in which structural features in gate-level netlist of known HTs are extracted for identification, have been proven as an effective static approach and avoid test vector generation and application. Like software virus detection paradigm, HT feature database is constructed and maintained. When a new type of HT appears, the HT feature database is extended. The feature analysis approaches target at HT detection at the pre-fabrication stage. In fact, these three types of HT detection method can provide complementary capabilities in detecting Trojans inserted in different IC production chain stages.

Our previous work [21] based on the structural features in gate-level netlist successfully detected single-triggered HTs from TrustHub benchmarks [23] with low false positive rate. An HT is considered to be successfully detected if part of

The associate editor coordinating the review of this manuscript and approving it for publication was Cihun-Siyong Gong.

the HT gates is in the HT candidates [10]. In this paper, we extend the method [21] in the following ways: (a) the structural features of combinational logic HTs in addition to single-triggered HTs are extracted, (b) the structural features of sequential logic HTs are extracted, (c) structural features of the HT circuits and the host circuits are combined, and (d) time complexity of the feature matching algorithm is reduced by exploiting the topological sorting technique. Combining the four extensions, the HT detection coverage and efficiency are significantly improved.

The contributions of this paper are:

- To the best of our knowledge, the structural features in gate-level netlist of the HT circuits and the host circuits are examined and combined for the first time in HT detection. The idea is based on the fact that the elusiveness of HTs relies on not just HT structure design but also the insertion positions in the host circuits, in which HTs are usually inserted at either the low controllability position or the low observability position owning the stealth characteristic [19].
- The structural features of combinational logic HTs and sequential logic HTs in TrustHub benchmarks [23] are extracted. The new features make a good supplement to the existing HT feature database.
- An efficient quantization approach on feature matching is proposed based on the topological sorting technique in graph theory.
- The experiments conducted on TrustHub benchmarks show that our approach can successfully detect all stealth Trojans with short runtime and low false positive rate.

The rest of the paper is organized as follows. Section II briefly introduces the related work. The threat model and the overall flow of the proposed method are presented in Section III. In Sections IV–VII, the HT detection method proposed in the paper is described in detail. The experimental results are shown and analyzed in section VIII. Finally, we conclude this paper in Section IX.

II. RELATED WORK

HT is a piece of circuit that is added to the design or is modified from the original design for malicious purposes. In terms of activation characteristics, HTs could be *always on* or *condition based*. To be hidden in chips, the HTs usually are designed to be silent in most of time. A typical HT contains trigger and payload [4]. The payload circuit is responsible for implementing HT attacks, which may result in serious effects such as denial of service, confidential information leakage and chip reliability degradation. The trigger monitors a certain condition, which could be a specific logic state, a particular input pattern or a specific counter value.

Recently, a number of hardware Trojan detection methods based on circuit feature analysis have been proposed. In [17], the authors propose a HT detection method named FASTrust. They construct the flip-flop level control data flow graphs (CDFGs) in which flip-flops, primitive inputs, as well as primitive outputs are abstracted as nodes and combinational logic circuits are abstracted as directed edges. In a CDFG, all vertices are classified into two categories: loop vertices each containing a self-loop and normal vertices without self-loops. Loop vertices connected together can form a loop group. Four features are extracted from CDFGs to identify different types of HT: Feature 1 is that a time-triggered HT has a large loop group; Feature 2 is that a combinational logic-triggered HT contains a node with large in-degree; Feature 3 indicates that an sequential logic-triggered HT contains a loop group with large in-degree; Feature 4 extracts that all nodes in a DeTrust HT have small out-degrees. A threshold was set for each feature. If there are circuits whose feature values are larger than the thresholds, these circuits will be reported as suspicious circuits. Chen *et. al.* [22] further extended FASTrust to multilevel FASTrsut (ML-FASTrust). For each reported HT suspicious node, combinational logic circuit was constructed between the suspicious node and the nearest level of flip-flops in its fan-in cone. Then, the toggling rate at the combinational logic was obtained. If the toggle rate is smaller than a threshold, it is considered that the circuit contains HTs.

The controllability and observability analysis has been proven to be an effective technique for HT detection [19]. The controllability and observability well characterize the host circuit. HTs inserted at either the low controllability position or the low observability position own the stealth characteristic. The method proposed in [19] took a gate-level netlist as the input, and performed the controllability and observability analysis. Afterwards, the k-means clustering was performed to cluster gates based on their controllability and observability values, and gates having significant inter-cluster distance from the genuine gates are Trojan candidates. Xie *et. al.* [20] further developed the method [19]. In [20], the inter-cluster distance was calculated and combined with the number of primitives, AND gates, and OR gates in the circuit as a four-dimensional vector to input to a SVM (Support Vector Machine) classifier for HT detection. Here, the primitives include all the information of inputs/outputs, DFFs, BUFs, MUXs and other gates.

A variety of machine learning algorithms has been used in HT detection more recently, including the SVM [24], random forest classifier [25] and multilayer neural networks [26]. In [26], the number of logic gate fanins, DFFs, MUXs, loops, logic levels to primitive input/output ports were extracted as Trojan-net features. These features were used as inputs of a neural network. There were two units in the output layer of the neural network. One unit corresponds to the normal nets, and the other corresponds to the Trojan nets. When the output value of the first unit is larger than that of the second one, the net is considered as a normal net; otherwise, a Trojan net is found.

The above methods examined the structural features of HTs and host circuits separately. However, the stealth of HTs is determined by both features. Therefore, we combine them together so as to improve the efficiency of HT detection.

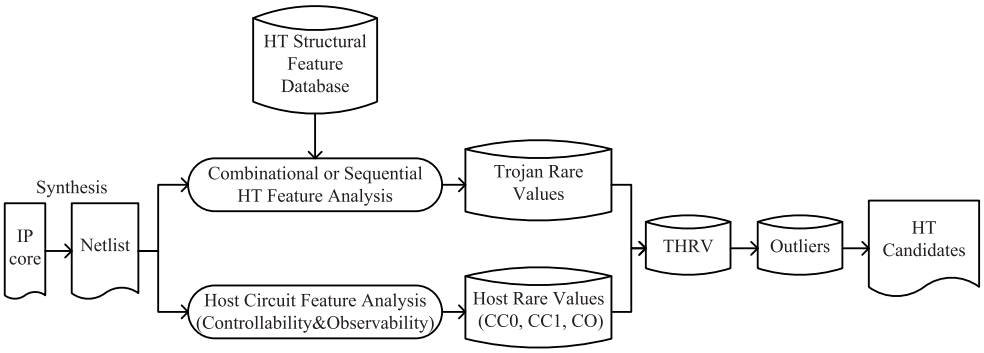

FIGURE 1. Flow of the proposed HT detection method.

III. FRAMEWORK OF THE PROPOSED METHOD

This section describes the threat model considered in this paper and gives an overview of the proposed HT detection method.

A. THREAT MODEL

The threat model considered in this paper is that the condition-based digital HTs, which are not easily detected by normal verification/testing approaches, are inserted into the IP cores by untrusted members or untrusted electronics design automation tools in outsourced vendors. Attackers by inserting HTs could steal privacy or make critical damages to the electronic systems under certain conditions. There are also always-on HTs which are not in the range of our study.

According to trigger mechanisms, HTs can be classified into combinational logic HTs and sequential logic HTs. Combinational logic HTs can be further classified into single pattern (SP) Trojans and case patterns (CP) Trojans. SP Trojans are activated when the monitored signals meet a unique specific pattern. CP Trojans are activated when the monitored signals meet any of the specific patterns.

The trigger circuits of the sequential logic HTs contain sequential elements such as flip-flops. Sequential logic HTs can be further classified into counter Trojans and state machine (SM) Trojans. For counter Trojans, the trigger condition is based on a single value (SV), or a range of values (RV) of an internal counter. SM Trojans are activated when a certain specific sequence of patterns is met at the monitored signals, which stimulate a sequence of state transitions. In order to improve the stealth, SM Trojans usually have a large state machine. The size of a state machine is reflected in both 'depth' and 'width'. The 'width' of a Trojan state machine is represented by the input width of the state machine; the 'depth' of a Trojan state machine is represented by the number of states. SM Trojans increase the stealth in the following two ways:

- Increasing the input width of the state machine. With wider bit-width inputs, the conditions for state transitions are more difficult to be satisfied. This kind of SM Trojans is defined as Wide State Machine (WSM) Trojans.
- Increasing the depth of the state machine. In the current state, if the input does not meet the condition of transition to the next state, it returns to the IDLE state. This is equivalent to indirectly increasing the number of states. This kind of SM Trojans is defined as Deep State Machine (DSM) Trojans.

The proposed method aims to detect the combinational logic and sequential logic HTs mentioned above. An overview of the method is shown next.

B. FRAMEWORK

The framework of our proposed method is shown in Fig.1. In the proposed method, a database that contains HT structural feature templates is established. The structural templates in the database are the basic elements. An HT can contain multiple and various basic elements cascaded deeply and widely. First, the third-party IP core under test is synthesized to a gate-level netlist. Then, the netlist goes through HT feature analysis and host circuit feature analysis. In the HT feature analysis process, the netlist is searched to find circuit elements which match to the structural feature templates in the database. Once a template is matched, an integer score is given to the circuit element. The score indicates how inactive the circuit element is and is called *Trojan rare value*. In the host circuit feature analysis process, the controllability and observability (called *host rare value*) of each circuit element including logical gates and flip-flops are examined. Afterwards, the Trojan rare value and the host rare value are combined to be a comprehensive rare value defined as *THRV* (*Trojan-Host Rare Value*) for each circuit element. Finally, the circuit elements are sorted in a descending order of THRV values and HT candidates are found by looking for THRV outliers.

Different parts of the framework will be presented in the following sections, including feature extraction, HT feature analysis, host circuit feature analysis, THRV rare value combination and outlier determination.

IV. STRUCTURE FEATURE EXTRACTION OF HTS

Feature-based HT detection methods mainly extract Trojan features from known Trojan benchmarks [22]. Based on the

above HT classification and HT circuits from [23], we extract two sets of structural features, one set for SP, CP, SV, RV and WSM Trojans and another set for DSM Trojans, respectively.

1) COMBINATIONAL STRUCTURE FEATURES

We analyze the gate-level netlists of several typical combinational HT circuits. The trigger circuits indeed bare some common features. We abstract the features as a number of structure templates, which serve as basic elements.

TABLE 1. AONN Structure Templates ($m, n = 2, 3, \ldots$).

Number	1st level-2nd level
1	NORm - NANDn
2	ANDm - NANDn
3	NORm - ANDn
4	ANDm - ANDn
5	NANDm - NORn
6	ORm - NORn
7	NANDm - ORn
8	ORm - ORn

The established templates are shown in Table 1. Each template contains two levels of logic gates with m and n inputs respectively, each from the four types of AONN (AND, OR, NAND, NOR) gates. Note that, m and n are integers which are 2 or more (m, n = 2, 3, 4, 5…). There may be cases, although not found in existing HTs, in which inverters or buffers exist between the two levels of AONN gates. The feature extraction algorithm can extract equivalent logic. For example, buffers and the even number of inverters are ignored, and the odd number of inverters is equivalent to logically inverting the gate in the first level.

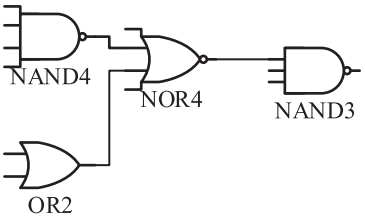

FIGURE 2. A circuit with three levels of AONN gates.

According to the principle of permutation, there should be 16 combinations of the AONN gates. However, the 8 templates we summarized are particular gate combinations that result in '1' or '0' infrequently, while the other 8 combinations do not and hence are not included in our templates. Given a concrete number of m and n to a template from Table 1, a case is instantiated. Note that, the output of the first level could be connected to any input of the second level. Fig.2 illustrates a circuit consisting of cases of Templates 5, 6 and 1. The number of input combinations of this circuit is 1024 (2^{10}). Given the probability of each input being '0' and '1' is 1/2, the probability of the output being '0' is 1/1024. If being '0', which is a low probability event for the circuit, is the trigger of a HT, then the HT is only activated when the input pattern is '1111000011'. The HT is an SP Trojan. The trigger circuits of HTs could contain multiple similar structures cascaded. This is consistent with the stealth property of HTs. Therefore, the HTs could be detected by looking for the structures. In Section V we will show that the structures are identified by calculating rare values of the logic gates.

For SV, RV and WSM sequential Trojans, they all use comparators to determine if the trigger condition is satisfied. If k-bit outputs of the counter or state machine are regarded as k signals of the host circuit, the trigger structure features of SV, RV and WSM Trojans also match the structure templates shown in Table 1. An example of comparators used in SV, RV and WSM trojans and in normal circuits is shown in Fig. 3. We can see that in Fig. 3 (a) there are NAND4-NOR2 and NOR2-NAND2 cases, while in Fig. 3 (b) there is not structure matched to the eight templates. Therefore, the SP, CP, SV, RV and WSM Trojans share the same combinational logic structure features.

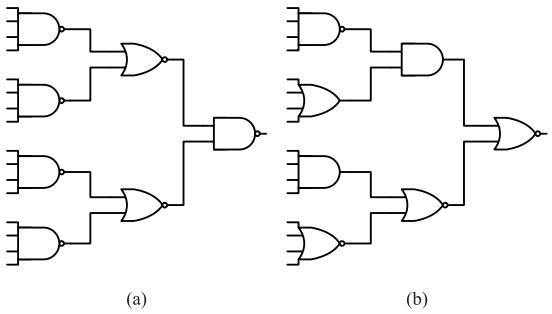

FIGURE 3. Gate-level structure of comparators used in (a) SV, RV, and WSM Trojans and (b) in normal circuits.

2) SEQUENTIAL STRUCTURE FEATURES

The DSM Trojans indirectly increase the depth of the state machine without increasing the number of states. This leads to the increase of the number of interconnections between state registers and between state registers and state machine input signals.

Fig. 4 shows the circuit structure of an FSM (Finite State Machine) with 1-bit input and 5 states, and the structure of a DSM derived from the FSM by adding transitions from each state to the IDLE state. The figure illustrates the input signal paths of each state register $R0$, $R1$ and $R2$ from state registers $R0 - R2$ (represented by circles) and state machine input (represented by input ports), respectively. The figure points out the structural feature of the state registers that each state register has a circle path from itself and the cross feedback paths from the other state registers. The obvious structure difference of FSM and DSM is that the number of paths from $R0 - R2$ and the state machine input to each state register increases in DSM. The number increases from 14 to 20 in total for three state registers. If the number of state machine inputs increases, the difference becomes larger as shown in Table 2. Therefore, the total number of paths from state registers and state machine inputs to each state register

TABLE 2. The number of paths from state registers and state machine inputs to each state register in FSM and DSM as the number of state machine inputs increases.

#state machine inputs	FSM			DSM		
	#paths from I[a] (in-degree)	#paths among R[b]	Total	#paths from I[a] (in-degree)	#paths among R[b]	Total
1	4	10	14	5	15	20
2	20	24	44	20	29	49
3	20	24	44	30	38	68
4	32	30	62	37	34	71
5	34	18	52	49	29	78
6	41	16	57	59	28	87
7	50	16	66	66	20	86
8	55	15	70	80	28	108

[a] paths from state machine inputs i.e. in-degree
[b] paths among registers

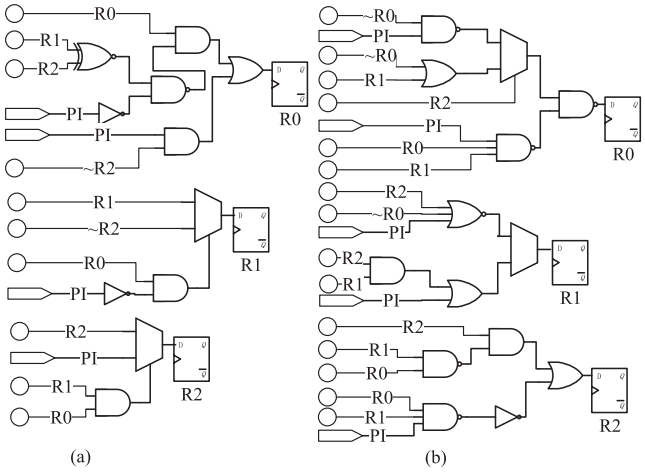

FIGURE 4. Circuit structures of (a) an FSM and (b) a DSM. The input signals of each state register *R*0, *R*1 and *R*2 are state registers *R*0 − *R*2 outputs (represented by circles) and state machine inputs (represented by input ports), respectively.

is extracted as the structural feature of DSM Trojans. In [22] the feature is the total in-degree of the group of state registers, while the feature in our work considers both in-degree of group of state registers and interconnections among the state registers in the group. By considering the interconnections within the group, the difference between FSM and DSM is amplified. For example, in Table 2, when only in-degree is considered, the FSM and DSM cannot be identified for 2 state machine inputs. Though the number of paths depends on the gate-level implementation, the number of paths of a DSM is always larger than that of the FSM, from which the DSM is derived, in a circuit under a given gate-level implementation.

The structural features of combinational and sequential logic Trojans described above form a database for HT detection.

V. STRUCTURAL FEATURE MATCHING ALGORITHMS

The proposed method identifies suspicious circuits from a circuit design by looking for small piece of circuits which has the structural features shown in the previous section. We propose feature matching algorithms for the combinational and sequential logic structural feature analysis, respectively. The matching degree is quantified as integer values, Trojan rare values. Higher the matching degree is, larger the rare value is.

A. COMBINATIONAL LOGIC STRUCTURAL FEATURE MATCHING

The procedure for calculating the combinational rare values of the AONN gates is the following [21]. In the first step, all AONN gates are extracted from a given gate-level netlist to form a set N_1. The AONN gates extraction significantly reduces the complexity of the HT detection, because the number of circuit elements which need to be analyzed is significantly reduced.

In the second step, for each gate G_j in N_1, its rare value crv_j is initialized to its input number. In the third step, each input pin I_i of G_j is evaluated according to the following three criteria:

1) There is a gate G_i connected to I_i in N_1.
2) Two adjacent gates $G_i - G_j$ match one of the cases of the AONN templates.
3) G_i's fan-out is one.

If all three criteria are satisfied, the rare value crv_i of G_i is accumulated into the rare value crv_j of G_j. Afterwards, G_j is added to set N_2. If all input pins of G_j do not satisfy the criteria, G_j is put into set N_3.

In the last step, if set N_2 is empty, then all gates obtain their final values and the algorithm terminates; otherwise go back to the third step and the process iterates. Let us continue with the example in Fig.2. Following the above procedure, the initial rare values of NAND4, OR2, NOR4 and NAND3 are 4, 2, 4 and 3, respectively. Because NAND4-NOR4, OR2-NOR4 and NOR4-NAND3 match the Templates 5, 6 and 1, the accumulated rare values of the four gates in the end are 4, 2, 8 and 10, respectively. As we can see, the logic gates? transition probability decreases and the rare value increases with the cascaded structure.

The time complexity of the above procedure is approximately $\boldsymbol{O(V{\cdot}E)}$, where $\boldsymbol{V}$ and $\boldsymbol{E}$ are the number of gates and the number of interconnections in a netlist, respectively. To reduce the time complexity, we consider the topologi-

cal sorting algorithm which has linear complexity ***O(V+E)***. Topological sorting [28] is a technique in graph theory. It sorts all the nodes in a directed acyclic graph in a certain order. The procedure for calculating the combinational rare values can be realized by the modified sorting method for reduced complexity. The gate-level netlist is essentially a directed graph in which nodes represent basic circuit elements and edges represent the signal propagation paths. The netlist may contain cyclic paths, especially in state registers of FSMs. It is not a problem for netlist with only AONN gates because sequential paths with feedback are removed.

Algorithm 1 Combinational Rare Value Calculation for AONN Structure Feature Matching

Input: Gate-level netlist
Output: Gate list L with rare values
1: extract all AONN gates and abstract them as a directed graph G
2: initialize rare value crv_i of each vertex v_i in G
3: find all nodes in G with 0 in-degree and put them into a queue *zeroInDeg*
4: **repeat**
5: pop one vertex v_i from *zeroInDeg*
6: **for** each v_i' successor v_j
7: **if** out-degree(v_i) == 1
8: **if** v_i-v_j matches to any structure template
9: $crv_j = crv_j + crv_i$-1
10: **end if**
11: **if** in-degree(v_j) == 1
12: add v_j to *zeroInDeg*
13: **end if**
14: **end if**
15: **end for**
16: remove v_i from G and add v_i to L
17: **until** G is empty
18: return L

The modified matching algorithm is shown in Algorithm 1. Lines 3–14 are designed based on the topological sorting.

Line 1: All AONN gates are extracted from a given gate-level netlist and are abstracted as a directed graph G.

Line 2: Combinational rare value of each node in G is initialized to the number of inputs of the corresponding gate.

Line 3: All nodes with 0 in-degree in G are found and popped into the queue *zeroInDeg*.

Lines 4–14: Node v_i is popped out from *zeroInDeg*. The condition that out-degree of v_i is 1 meets Criterion 1. v_i only has one successor node v_j. If v_i-v_j matches one of templates in Table 1, the rare value of v_i is added to the rare value of v_j. At the same time, if the in-degree of v_j is 1, v_j is one of the next batch of nodes having zero in-degree after removing v_i and is popped in *zeroInDeg*.

Line 15: Node v_i is removed from G and added to list L.

Line 16–18: The process is continued until G is empty and a list L of gates with rare values is obtained.

It will be shown in the result section that the modified algorithm has superior performance compared to the original method in [21]. Following the similar procedure, we also develop the feature matching algorithm for the sequential structure features in the next subsection.

B. SEQUENTIAL LOGIC STRUCTURAL FEATURE MATCHING

To match the feature of DSM HTs, we need to calculate the total number of paths (defined as *sequential rare value*) from state registers and state machine inputs to each state register. To amplify the feature, we calculate the total value of a group of state registers which belong to an FSM. The procedure for calculating sequential rare value is shown in algorithm 2.

Algorithm 2 Sequential Rare Value Calculation for Sequential Logic Structure Feature Matching

Input: Gate-level netlist
Output: State register group list L with the sequential rare value
1: convert the gate-level netlist to a directed graph G
2: extract all registers with self-feedback and partition them into
groups S_i, $1 \leq i \leq m$
3: **for each** group S_i **do**
4: initialize the sequential rare value srv_i of S_i to 0
5: **for each** register in S_i **do**
6: add the register vertex to a queue Q
7: **repeat**
8: pop a vertex v_k from Q
9: **for each** v_k' predecessor v_j in G **do**
10: **if** v_j is a state machine input or a register belonging to S_i
11: srv_i += 1
12: **else if** v_j is a combinational logic element
13: add v_j to Q
14: **end if**
15: **end for**
16: **until** Q is empty
17: **end for**
18: add group S_i to L
19: **end for**

Line 1: The gate-level netlist is converted to a directed graph G. Vertex in G represent primary input ports, registers and combinational logic elements such as gates and MUX. Edges in G represent the logic interconnections which exist in the netlist.

Line 2: All registers with self-feedback are extracted from G and the registers having cross feedback paths form a group S_i.

Lines 5–18: Find all paths from the state registers and state machine inputs to a stage register in S_i. Once a path is found, the sequential rare value srv_i of the group is added one (lines 9 and 10).

Lines 19–20: Once the total value srv_i is calculated for group S_i, group S_i is put into list L. The algorithm iterates for the other groups.

So far, we have presented the logic structural features of stealth HTs and the feature matching methods. The elusiveness of HTs is determined not only by the HT structures, but also by the HT insertion positions in the host circuits. Therefore, the structural features of the HTs and the host circuits are combined in a way shown in the next section.

VI. COMBINATION OF STRUCTURAL FEATURES OF HTS AND HOST CIRCUITS

The insertion positions of HTs are usually chosen to be the circuit nodes which are difficult to control and observe from the primary ports. The controllability and observability analysis of a circuit has been used for HT detection in [19]. We adopt the approach for analysis of the HT insertion positions, and combine it with the HT structural features.

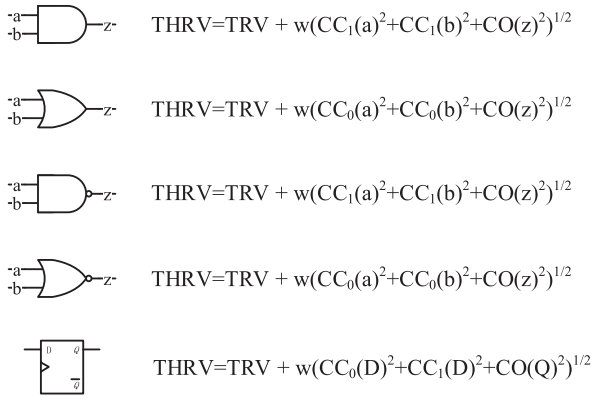

FIGURE 5. Definition of THRV. CC_0 means 0-controllability. CC_1 means 1-controllability. *CO* means observability.

The SCOAP method [30] is used to measure the controllability and observability of nodes at the circuit level. We examine AONN gates and registers, and define a combined rare value *THRV* for each element as shown in Fig.5. *THRV* consists of two parts. The first part *TRV* is the Trojan rare value that reflects the rarity of the Trojan structures. The second part SCOAP value is the host rare value that reflects the rarity of insertion positions in the host circuit. The definition of the second part is determined by the type of circuit elements [19]. For example, for an AND gate, 1 is a rare output compared to 0. Therefore, CC_1 (1-controllability) of inputs represents its controllability. Its observability is reflected by the *CO* (observability) of the output. The weight coefficient w is dynamically adjusted to keep the SCOAP value in the same order of magnitude as Trojan rare value. As will be seen in the result section, using *THRV* in HT detection leads to higher detection accuracy than using *TRV*.

VII. OUTLIER DETERMINATION

The feature matching algorithms have been applied to a set of HT benchmarks. From the rare value *THRV* distributions we made three observations. First, there are always a few outliers in the rare value distribution of each benchmark. Second, the outliers tend to have the highest rare values. Third, the gates or register groups corresponding to the rare value outliers are part of HTs. Therefore, the problem of HT detection is transformed to rare value outlier determination. Based on the three points, we propose an algorithm to identify outliers as shown in Algorithm 3. For a given *THRV* list, the elements are first sorted by value in a descending order. During the sorting, the number of elements which have the same rare value $THRV_i$ is also counted as m_i.

Algorithm 3 Outlier Determination

Input: Rare value list
Output: Outliers set O
1: sort the list by value in descending order $S = [s_1, s_2, \ldots, s_n]$, each s_i having rare value $THRV_i$
2: **for each** s_i in S **do**
3: calculate $\text{Avg}(s_i)=\frac{1}{n-i}\sum_{k=i+1}^{n} THRV_k$, the average of the rare values
of the elements $s_{i+1} \sim s_n$
4: **if** ($THRV_i$ - $THRV_{i+1} > T_1$ * $\text{Avg}(s_i)$) **and** ($m_i < T_2 \times n$) **and**
($i < n \times T_3$)
5: add s_i to O
6: **end if**
7: **end for**

There are three criteria to determine an outlier s_i as shown in line 4 of Algorithm 3.

1) The difference between s_i and s_{i+1} is greater than the average of the rare values of elements $s_{i+1}, \ldots, s_n$. This criterion means that there should be an obvious gap between s_i and the subsequent elements in terms of the rare values.
2) The number of elements which have the same large value $THRV_i$ should be small. This criterion corresponds to the first observation that the number of outliers is small, and to the fact that the number of elements constituting HTs is small.
3) An outlier s_i should be at the top of a descending list and thus its index i is small. This criterion corresponds to the second observation that the outliers tend to have highest rare value.

The three parameters T_1, T_2 and T_3 in Algorithm 3 are decided by the rare value list and data fitting in practice. Through the three parameters T_1, T_2 and T_3 we could achieve some guarantees on the detectability of the targeted HTs. Following the theoretical analysis approach [22] on the detectability, a HT with q-bit trigger inputs should select $q \gg \log_2 l$, where l is the number of clock cycles for which the functional verification process tests a circuit, in order to escape from the verification process. If the threshold of HT features is larger than $\log_2 l$, then all HTs having the features can be detected. The same conclusion is achievable

TABLE 3. Detection results of combinational logic HTs. Each circuit is labeled with HT type. C: the number of candidates of HT gates. H: the number of real HT gates.

Circuit	Circuit_scale(K)	HT_scale	C	H	Circuit	Circuit_scale(K)	HT_scale	C	H
rs232_t100(SP)	0.2	7	1	1	aes_t1000(SP)	176.7	44	3	3
rs232_t200(SP)	0.2	5	3	3	aes_t1300(SP)	176.6	44	3	3
rs232_t400(SP)	0.2	5	1	1	rsa_t100(SP)	1.9	11	4	1
rs232_t800(SP)	0.2	5	1	1	rsa_t200(SP)	2.0	13	4	1
rs232_t1000(SP)	0.3	13	2	2	vga_t100(SP)	69.2	5	4	3
rs232_t1100(SP)	0.3	13	2	2	s15850_t100(SP)	2.2	26	7	6
rs232_t1200(SP)	0.3	10	1	1	s35932_t100(CP)	5.4	22	8	8
rs232_t1300(SP)	0.3	9	1	1	s35932_t200(SP)	5.4	11	2	2
rs232_t1400(SP)	0.3	12	2	2	s35932_t300(SP)	5.5	35	4	4
rs232_t1500(SP)	0.3	13	2	2	s38417_t100(SP)	5.3	12	9	1
rs232_t1600(SP)	0.3	10	1	1	s38417_t200(CP)	5.3	15	9	1
wb_t100(SP)	20.5	15	20	3	s38417_t300(CP)	5.4	16	21	13
wb_t300(SP)	33.0	168	40	40	s38584_t100(SP)	6.5	8	12	0
aes_t400(SP)	177.1	46	3	3	ethernet_t700(CP)	102.1	12	21	12
aes_t1800(SP)	176.6	44	3	3	ethernet_t710(CP)	102.1	12	21	12
aes_t600(SP)	176.6	46	3	3	ethernet_t720(CP)	102.1	12	21	12
aes_t700(SP)	176.7	44	3	3	ethernet_t730(CP)	102.1	12	21	12

* The circuit scale means the number of elements including AND, NAND, OR, NOR, INV, BUF, MUX and DFF.
* The HT scale means the number of HT gates or register groups.

in our method. In Algorithm 3, a conservative selection of T_1, T_2 and T_3 can ensure a threshold larger than $\log_2 l$, leading to negligible false negative rate. Section VIII will show the determination method of the parameters on the tested HT benchmarks and their impacts on HT detection efficiency.

VIII. EXPERIMENTAL RESULTS

To evaluate the proposed approach, HT benchmarks from TrustHub are used. The benchmarks include two sets of circuits, one in the register transfer level description and another in the gate-level description. Synopsys Design Complier under saed90nm technology library is used for synthesizing the benchmarks described in the register transfer level. Synopsys TetraMAX is used to obtain CC_0, CC_1, and CO. Algorithm 1, Algorithm 2 and Algorithm 3 are implemented in Python. Experimental platform is an AMD A8-6410 2.00GHz CPU with 4GB RAM.

The three parameters in the outlier determination algorithm are determined as below. T_2 is determined according to the size n of the rare value list S. For HTs we have seen, the maximum number of elements with the same rare value is 32. Therefore, we set T_2 to 0.1^x $(x = 0, 1, 2\ldots)$ so that $T_2 \times n$ is less than 32. T_1 and T_3 are decided by data fitting. Because the rare values of sequential logic structure features are generally larger than the rare values of AONN structure features, T_1 and T_3 are determined separately. We pick 10 rare value distributions from SP, CP, SV, RV and WSM HTs, and 8 rare value distributions from DSM HTs. Rare value outliers are known for these distributions. T_1 is tuned within the range of [0.1, 1] with step 0.1, and T_3 is tuned within the range of [0.02, 0.2] with step 0.02. In total, there are 100 combinations of T_1 and T_3. For every pair of T_1 and T_3, Algorithm 3 is performed on the training set to obtain the outliers. The values of T_1 and T_3 are picked, respectively, so that the best match between the obtained outliers and the expected outliers is achieved. In our experiment, finer tuning of T_1 is also carried out from 0.4 to 0.6 with the step of 0.01 after the 100 combinations. The final values of T_1 and T_3 used for SP, CP, SV, RV, WSM HTs detection are 0.56 and 0.02, respectively. The values of T_1 and T_3 used for DSM HTs detection are 0.47 and 0.2, respectively.

A. DETECTION RESULTS

The detection results of combinational logic Trojans are shown in Table 3. Columns 'C' and 'H' show the number of suspicious HT gates (Candidates) and the number of real HT gates (Hits, indicating true positives) from the candidate list. The circuit scale means the number of elements including AND, NAND, OR, NOR, INV, BUF, MUX, DFF. The HT scale means the number of HT gates or DFF groups. As shown in Table 3, 33 of 34 HTs are detected. The HT in s38584_t100 that is not detected has been proven to be easily activated [19], so the Trojan rare value and SCOAP rare value are both small. It could be detected by normal functional verification and testing techniques.

The detection results of SV, RV and WSM Trojans are shown in Table 4. HTs are detected from all circuits except from the pic series. In pic series circuits, the HT trigger, a 13-bit counter, observes the number of executions of a specific instruction. When the number is above 100 the Trojan is triggered. The counter is incremented by one if a 4-bit signal satisfies some specific values. When this 4-bit signal is equal to 1101, the counter is reset to zero. Regarding the trigger condition, the HT activation probability is $1-100/2^{13}$. As a result, its combinational rare value is small. However, the reset operation actually makes the HT not easy to be activated, which is equivalent to indirectly increasing the counter value. This feature is the same as DSM HTs. Fig. 6 illustrates the structure of the counter like in the pic series circuits, where the registers have cross feedback paths.

The detection results of pic series and the DSM Trojans are shown in Table 5. As can be seen, all pic series and the DSM Trojans are detected.

Except for one easily activated HT, all rarely activated HTs from TrustHub benchmarks are detected efficiently by our approach. Accuracy rate (R_{acc}) is defined as the percentage of true positives of candidates [9], [17]. As shown

TABLE 4. Detection results of SV, RV and WSM HTs. Each circuit is labeled with HT type. C: the number of candidates of HT gates. H: the number of real HT gates.

Circuit	Circuit_scale(K)	HT_scale	C	H	Circuit	Circuit_scale(K)	HT_scale	C	H
rs232_t300(SV)	0.3	11	3	3	s38584_t200(SV)	6.6	53	2	2
rs232_t500(SV)	0.3	11	3	3	s38584_t300(RV)	7.2	154	10	10
aes_t900(SV)	177.1	171	3	3	pic_t100(RV)	1.5	1 (register group)	10	0
aes_t1200(SV)	177.1	171	3	3	pic_t200(RV)	1.4	1 (register group)	10	0
aes_t1500(SV)	177.0	171	3	3	pic_t300(RV)	1.4	1 (register group)	10	0
aes_t1700(SV)	177.5	171	3	3	pic_t400(RV)	1.4	1 (register group)	10	0
aes_t1900(SV)	176.9	172	3	3	wb_t200(WSM)	32.8	27	2	2
aes_t2100(SV)	176.9	172	3	3	aes_t500(WSM)	176.7	86	8	8
rsa_t300(SV)	2.0	11	4	1	aes_t800(WSM)	176.8	86	8	8
rsa_t400(SV)	2.1	11	4	1	aes_t1100(WSM)	176.8	86	8	8
b19_t200(RV)	62.9	53	8	2	aes_t1400(WSM)	176.7	86	8	8
b19_t300(SV)	54.1	288	114	60	aes_t1600(WSM)	177.2	91	11	11
b19_t400(SV)	53.4	288	114	60	aes_t2000(WSM)	176.7	86	8	8
b19_t500(SV)	52.1	96	74	20					

* The circuit scale means the number of elements including AND, NAND, OR, NOR, INV, BUF, MUX and DFF.
* The HT scale means the number of HT gates or register groups.

TABLE 5. Detection results of pic series and DSM HTs. Each circuit is labeled with HT type. C: the number of candidates of HT register groups. H: the number of real HT register groups.

Circuit	Circuit_scale(K)	HT_scale	C	H	Circuit	Circuit_scale(K)	HT_scale	C	H
pic_t100(RV)	1.5	1 (register group)	2	1	rs232_t900(DSM)	0.2	1 (register group)	2	1
pic_t200(RV)	1.4	1 (register group)	2	1	rs232_t600(DSM)	0.2	1 (register group)	2	1
pic_t300(RV)	1.4	1 (register group)	2	1	rs232_t700(DSM)	0.2	1 (register group)	2	1
pic_t400(RV)	1.4	1 (register group)	2	1	rs232_t901(DSM)	0.2	1 (register group)	2	1

* The circuit scale means the number of elements including AND, NAND, OR, NOR, INV, BUF, MUX and DFF.
* The HT scale means the number of HT gates or register groups.

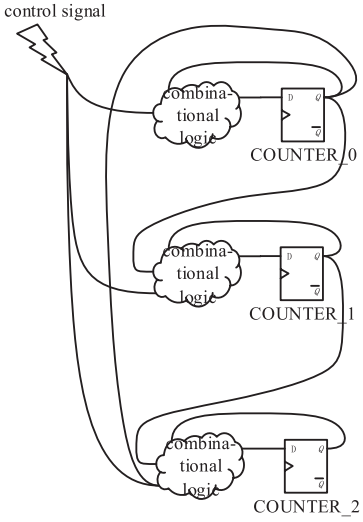

FIGURE 6. Structure of a 3-bit counter like in the pic series circuits.

in Table 3, 4 and 5, among 64 detected HT circuits, the average of accuracy rate is 78%. That is, 78 out of 100 suspicious circuit elements, determined by our approach, belong to HTs. The accuracy rate is 100% for 42 detected HT circuits.

In addition, we also explore the impacts of the parameters T_1 and T_3 in Algorithm 3 on the HT detection results. In the exploration, benchmark circuits with HTs and without HTs are used, and T_1 varies within the range of [0.1, 2] with step 0.1 and T_3 varies within the range of [0.02, 0.2] with step 0.02. The detection results are shown in Table 6. In Table 6, the check mark indicates that HT is detected from a circuit. The results show that when $T_1 = 0.2$ and $T_3 = 0.1$, HTs are successfully detected from all HT-infected circuits, as well as from 8 HT-free circuits. As T_1 gets bigger and T_3 smaller, the number of detected HT-infected circuits slightly decreases and more HT-free circuits are detected as HT free. When $T_1 = 1.6$ and $T_3 = 0.02$, HTs are detected from 45 of 56 HT-infected circuits and 2 of 10 HT-free circuits. Therefore, the selection of the parameters T_1 and T_3 allows tradeoff between false negative rate and false positive rate. In practice, one may want to keep low false negative rate at the loss of false positive rate. The false positive cases may be identified by further processes.

B. RUN TIME ANALYSIS OF THE PROPOSED METHOD

As shown in Table 7, the run time of the method proposed in this paper is divided into three parts: the time consumed by Trojan rare value calculation (t_1), *i.e.* the time consumed by Algorithm 1 or Algorithm 2, the time consumed by host rare value calculation (t_2), and the time consumed by outlier determination (t_3), *i.e.* the time consumed by Algorithm 3. As we can see, for the largest AES series circuits which have 176.9K elements, the average runtime is 71.353 seconds, and for the smallest rs232 series which have 0.252K elements, the average runtime is 0.252 seconds. The time for host rare value calculation dominates the runtime as the circuit size increases. The runtime comparison with exiting methods is shown in Fig. 7. The average detection time of our method is 2.56 seconds longer than that in [22], because the method [22] is based on the information flow graph which has less circuit information than the netlist. However, our approach has lower false positive rate than [22] which will be seen next.

TABLE 6. Detection results in HT-infected and HT-free Circuits under various parameters T_1 and T_3. A check mark indicates that HT is detected from a circuit.

Circuit	$T_1 = 0.5$ $T_3 = 0.08$	$T_1 = 0.7$ $T_3 = 0.06$	$T_1 = 1.2$ $T_3 = 0.04$	$T_1 = 1.6$ $T_3 = 0.02$	Circuit	$T_1 = 0.5$ $T_3 = 0.08$	$T_1 = 0.7$ $T_3 = 0.06$	$T_1 = 1.2$ $T_3 = 0.04$	$T_1 = 1.6$ $T_3 = 0.02$
rs232_t100	√	√	√	√	aes_t2100	√	√	√	√
rs232_t200	√	√	√	√	rsa_t100	√	√	√	√
rs232_t300	√	√	√	√	rsa_t200	√	√	√	√
rs232_t400	√	√	√	√	rsa_t300	√	√	√	√
rs232_t500	√	√	√	√	rsa_t400	√	√	√	√
rs232_t800	√	√	√	√	b19_t200	√	√	√	√
rs232_t1000	√	√	√	√	b19_t300	√	√	√	√
rs232_t1100	√	√	√	√	b19_t400	√	√	√	√
rs232_t1200	√	√	√	√	b19_t500	√	√	√	√
rs232_t1300	√	√	√	√	vga_t100	√	√		
rs232_t1400	√	√	√	√	s15850_t100	√	√	√	
rs232_t1500	√	√	√	√	s35932_t100	√	√	√	√
rs232_t1600	√	√	√	√	s35932_t200	√	√	√	√
wb_t100	√	√	√		s35932_t300	√	√	√	√
wb_t200	√	√	√		s38417_t100	√	√		
wb_t300	√	√	√	√	s38417_t200	√	√		
aes_t400	√	√	√	√	s38417_t300	√	√		
aes_t500	√	√	√	√	s38584_t200	√	√	√	√
aes_t600	√	√	√	√	s38584_t300	√	√	√	√
aes_t700	√	√	√	√	ethernet_t700	√			
aes_t800	√	√	√	√	ethernet_t710	√			
aes_t900	√	√	√	√	ethernet_t720	√			
aes_t1000	√	√	√	√	ethernet_t730	√			
aes_t1100	√	√	√	√	aes(HT-free)				
aes_t1200	√	√	√	√	b19(HT-free)	√			
aes_t1300	√	√	√	√	ethernet(HT-free)	√	√	√	√
aes_t1400	√	√	√	√	rs232(HT-free)				
aes_t1500	√	√	√	√	s15850(HT-free)	√	√	√	√
aes_t1600	√	√	√	√	s35932(HT-free)				
aes_t1700	√	√	√	√	s38417(HT-free)	√	√		
aes_t1800	√	√	√	√	s38584(HT-free)	√	√	√	
aes_t1900	√	√	√	√	vga(HT-free)	√			
aes_t2000	√	√	√	√	wb_conmax(HT-free)	√			

TABLE 7. Run Time of the Proposed Method.

Circuit	t_1(s)	t_2(s)	t_3(s)	Total(s)
rs232 series	0.022	0.225	0.005	0.252
pic series	4.305	0.343	0.011	4.659
rsa series	0.081	0.575	0.014	0.67
s15850 series	0.138	0.780	0.185	1.103
s38417 series	0.277	1.723	0.613	2.613
s35932 series	0.190	1.620	0.023	1.833
s38584 series	0.387	1.795	1.184	3.366
wb_conmax series	1.496	3.650	1.050	6.196
b19 series	4.061	12.763	13.531	30.355
vga series	2.709	21.870	0.149	24.728
ethernet series	4.227	28.715	0.480	33.422
aes series	23.365	41.696	6.292	71.353

Compared with [21] whose runtime only contains t_1 and t_3, our method is 75.4 times faster due to the use of topological sorting.

C. COMPARISON WITH EXISTING METHODS

Finally, We compare the false positive rate (R_{fp}) and accuracy rate (R_{acc}) of our method with three existing methods proposed in [21], [22], [26]. R_{fp} is defined as the percentage of false positives of total logic elements [9]. The values R_{fp} and R_{acc} of the three methods are computed using the reported data from the published papers. The comparison is shown in Fig. 8 and Fig. 9. It can be seen from the figures that our method successfully detect HTs with lower false positive rate and higher accuracy rate. Several values of R_{fp} and R_{acc} are

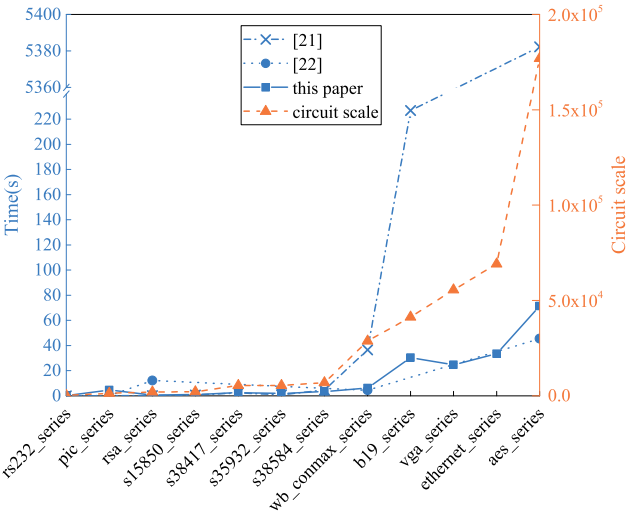

FIGURE 7. Runtime comparison with existing methods.

blanked for the three methods, because the detection results were not reported in their papers.

Table 8 summaries HT detection results of the methods [21], [22], [26] and our proposed method. In [21], [22], [26], only a subset of benchmarks is tested, while our method is applied to all benchmarks and can successfully detect all HTs. Columns 5, 6, 12 and 13 show the detection

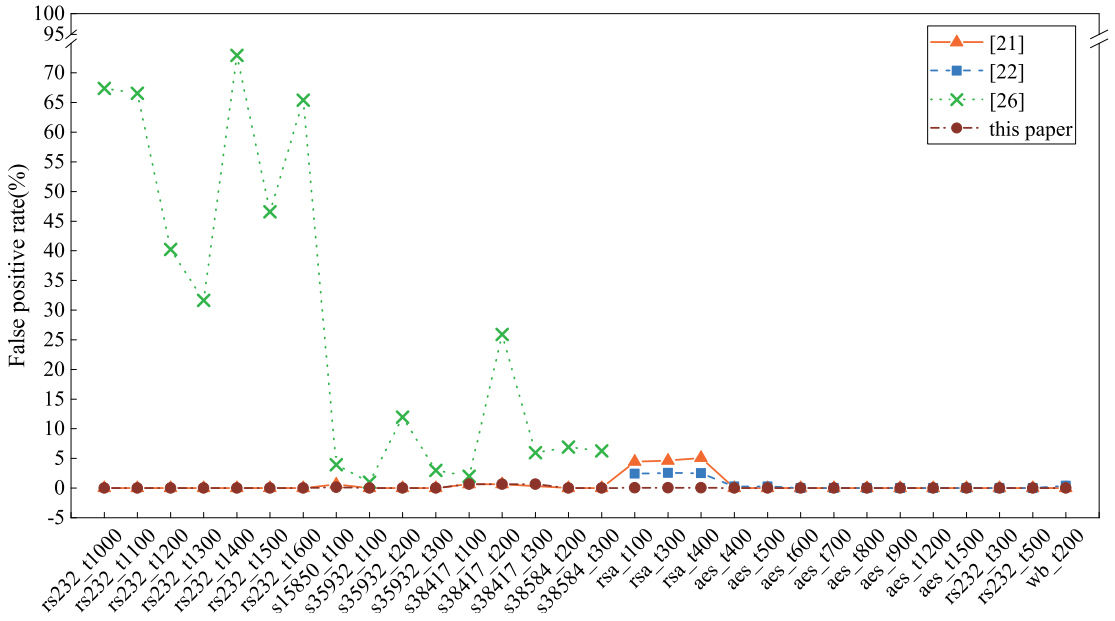

FIGURE 8. Comparison of R_{fp} of our method and existing methods. R_{fp}: false positive rate as lower as better.

TABLE 8. Comparison of HT Detection Results. The check marks indicate that HT is successfully detected. N/A means that the circuits were not tested by [21], [22], [26].

Circuit	[21]	[22]	[26]	Only TRV	Only HRV	THRV	Circuit	[21]	[22]	[26]	Only TRV	Only HRV	THRV
rs232_t100	√	N/A	N/A	√	√	√	aes_t1000	√	N/A	N/A	√		√
rs232_t200	√	N/A	N/A	√	√	√	aes_t1300	√	N/A	N/A	√		√
rs232_t400	√	N/A	N/A	√	√	√	rsa_t100	√	√	N/A	√		√
rs232_t800	√	N/A	N/A	√	√	√	rsa_t200	√	N/A	N/A	√		√
rs232_t1000	√	N/A	√	√	√	√	vga_t100	N/A	N/A	N/A		√	√
rs232_t1100	√	N/A	√	√	√	√	s15850_t100	√	N/A	√	√	√	√
rs232_t1200	√	N/A	√	√	√	√	s35932_t200	√	N/A	√	√		√
rs232_t1300	√	N/A	√	√	√	√	s35932_t300	√	N/A	√	√	√	√
rs232_t1400	√	N/A	√	√	√	√	s38417_t100	√	N/A	√	√	√	√
rs232_t1500	√	N/A	√	√	√	√	s35932_t100	N/A	N/A	√	√	√	√
rs232_t1600	√	N/A	√	√	√	√	s38417_t200	N/A	N/A	√		√	√
wb_t100	√	N/A	N/A	√	√	√	s38417_t300	N/A	N/A	√		√	√
wb_t300	√	N/A	N/A	√		√	ethernet_t700	N/A	N/A	N/A		√	√
aes_t400	√	√	N/A	√		√	ethernet_t710	N/A	N/A	N/A		√	√
aes_t1800	√	N/A	N/A	√		√	ethernet_t720	N/A	N/A	N/A		√	√
aes_t600	√	√	N/A	√		√	ethernet_t730	N/A	N/A	N/A		√	√
aes_t700	√	√	N/A	√		√	s38584_t200	√	N/A	√	√	√	√
rs232_t300	√	√	N/A	√	√	√	s38584_t300	N/A	N/A	√	√	√	√
rs232_t500	√	√	N/A	√	√	√	pic_t100	N/A	√	N/A	√		√
aes_t900	√	√	N/A	√		√	pic_t200	N/A	√	N/A	√		√
aes_t1200	√	√	N/A	√		√	pic_t300	N/A	√	N/A	√		√
aes_t1500	√	√	N/A	√		√	pic_t400	N/A	√	N/A	√		√
aes_t1700	√	N/A	N/A	√		√	wb_t200	N/A	√	N/A	√		√
aes_t1900	√	N/A	N/A	√		√	aes_t1600	N/A	N/A	N/A	√		√
aes_t2100	√	N/A	N/A	√		√	aes_t500	N/A	√	N/A	√		√
rsa_t300	N/A	√	N/A	√		√	aes_t2000	N/A	N/A	N/A	√		√
rsa_t400	N/A	√	N/A	√		√	aes_t800	N/A	√	N/A	√		√
b19_t300	√	N/A	N/A	√	√	√	aes_t1100	N/A	N/A	N/A	√		√
b19_t400	√	N/A	N/A	√	√	√	aes_t1400	N/A	N/A	N/A	√		√
b19_t500	√	N/A	N/A	√		√	rs232_t700	N/A	N/A	N/A	√		√
rs232_t600	N/A	N/A	N/A	√		√	rs232_t901	N/A	√	N/A	√		√
rs232_t900	N/A	√	N/A	√		√	b19_t200	√	N/A	N/A	√	√	√

* TRV: Trojan rare values, HRV: host rare values, THRV: Trojan-host rare values

results while only using Trojan rare values and only using host rare values. The trigger inputs of some HTs are the primitive inputs and the payloads are the primitive outputs, so the SCOAP rare values of the host circuit are small. These HTs cannot be detected by only host rare values. On the other hand, there are HTs whose trigger circuit is simple resulting in small Trojan rare values. As a result, these HTs cannot be detected by only Trojan rare values. All above HTs

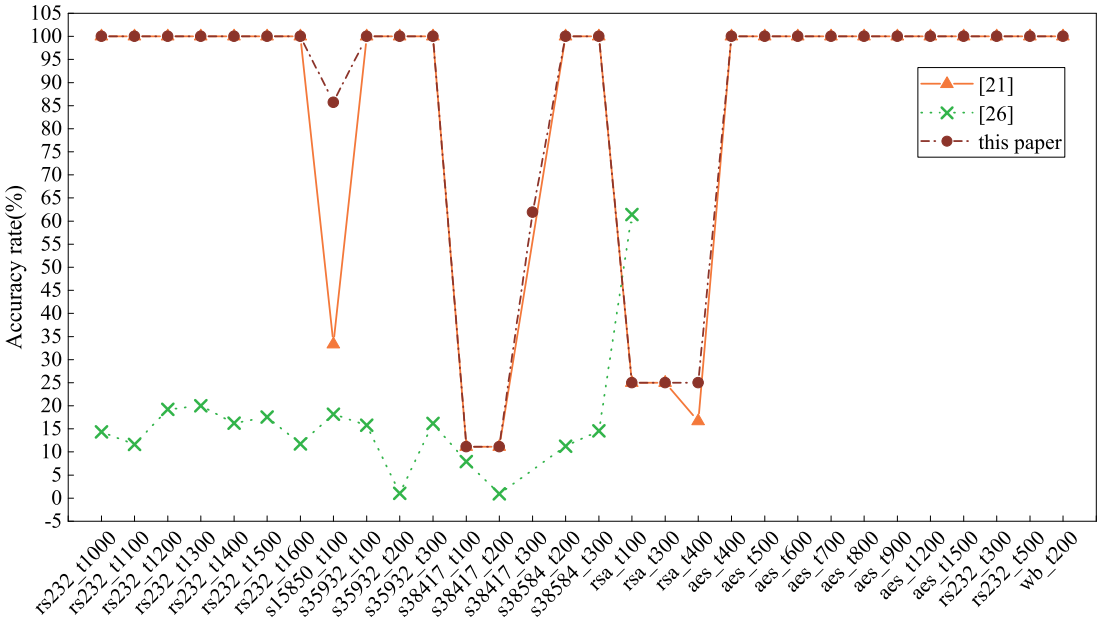

FIGURE 9. Comparison of R_{acc} of our method and existing methods. R_{acc}: accuracy rate as higher as better.

are detected by using Trojan host rare values because they contain the information of both Trojan rare values and host rare values.

IX. CONCLUSION

In this paper, we propose a HT detection method based on structural features of Trojans and host circuits. We extract a number of structural features of HTs from gate-level netlists and construct a HT database covering the combinational logic HT features and sequential logic HT features. Efficient feature analysis algorithms are developed to search small piece of circuits which match the features in the database and are assigned with a score. A score outlier determination algorithm is developed to identify suspicious Trojan elements. The experimental results show that the proposed method is capable of detecting all stealth Trojans from the benchmarks with short runtime and low false positive rate, compared to the existing HT detection methods.

REFERENCES

[1] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," in *Proc. IEEE Int. High Level Des. Test Workshop*, Nov. 2009, pp. 166–171.

[2] A. Syed and R. M. Lourde, "Hardware security threats to DSP applications in an IoT network," in *Proc. IEEE Int. Symp. Nanoelectr. Inf. Syst. (iNIS)*, Dec. 2016, pp. 62–66.

[3] A. Sengupta, "Hardware vulnerabilities and their effects on ce devices: Design for security against Trojans [Hardware Matters]," *IEEE Consum. Electron. Mag.*, vol. 6, no. 3, pp. 126–133, Jul. 2017.

[4] S. Mitra, H.-S. P. Wong, and S. Wong, "The Trojan-proof chip," *IEEE Spectrum*, vol. 52, no. 2, pp. 46–51, Feb. 2015.

[5] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware Trojan detection and reducing Trojan activation time," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 112–125, Jan. 2012.

[6] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia, "MERO: A statistical approach for hardware Trojan detection," in *Proc. Cryptograph. Hardw. Embedded Syst.*, 2009, pp. 396–410.

[7] H. Li and Q. Liu, "Hardware Trojan detection acceleration based on word-level statistical properties management," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2014, pp. 153–160.

[8] S. Saha, R. S. Chakraborty, S. S. Nuthakki, and D. Mukhopadhyay, "Improved test pattern generation for hardware Trojan detection using genetic algorithm and Boolean satisfiability," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.* Berlin, Germany: Springer, Sep. 2015, pp. 577–596.

[9] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of stealthy malicious logic using Boolean functional analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, Nov. 2013, pp. 697–708.

[10] J. Zhang, F. Yuan, L. Wei, Y. Liu, and Q. Xu, "Veritrust: Verification for hardware trust," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 34, no. 7, pp. 1148–1161, Jul. 2015.

[11] J. He, Y. Zhao, X. Guo, and Y. Jin, "Hardware Trojan detection through chip-free electromagnetic side-channel statistical analysis," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 10, pp. 2939–2948, Oct. 2017.

[12] A. Amelian and S. E. Borujeni, "A side-channel analysis for hardware Trojan detection based on path delay measurement," *J. Circuits, Syst. Comput.*, vol. 27, no. 9, 2018, Art. no. 1850138.

[13] Y. Cao, C.-H. Chang, and S. Chen, "Cluster-based distributed active current timer for hardware Trojan detection," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2013, pp. 1010–1013.

[14] Y. Huang, S. Bhunia, and P. Mishra, "Scalable test generation for Trojan detection using side channel analysis," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 11, pp. 2746–2760, Nov. 2018.

[15] X. Cui, K. Wu, and R. Karri, "Hardware Trojan detection using path delay order encoding with process variation tolerance," in *Proc. IEEE 23rd Eur. Test Symp. (ETS)*, May/Jun. 2018, pp. 1–2.

[16] M. Cozzi, J.-M. Galliere, and P. Maurine, "Thermal scans for detecting hardware Trojans," in *Proc. Int. Workshop Constructive Side-Channel Anal. Secure Design*, 2018, pp. 117–132.

[17] S. Yao *et al.*, "Fastrust: Feature analysis for third-party ip trust verification," in *Proc. IEEE Int. Test Conf. (ITC)*, Oct. 2015, pp. 1–10.

[18] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, "A score-based classification method for identifying hardware-Trojans at gate-level netlists," in *Proc. Design, Autom. Test Eur. Conf. Exhibit.*, Mar. 2015, pp. 465–470.

[19] H. Salmani, "COTD: Reference-free hardware Trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 2, pp. 338–350, Feb. 2017.
[20] X. Xie, Y. Sun, H. Chen, and Y. Ding, "Hardware Trojans classification based on controllability and observability in gate-level netlist," *IEICE Electron. Exp.*, vol. 14, no. 18, 2017, Art. no. 20170682.
[21] F. Chen and Q. Liu, "Single-triggered hardware Trojan identification based on gate-level circuit structural characteristics," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.
[22] X. Chen *et al.*, "Hardware Trojan detection in third-party digital intellectual property cores by multilevel feature analysis," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1370–1383, Jul. 2018.
[23] M. Tehranipoor, R. Karri, F. Koushanfar, and M. Potkonjak. (2016). *Trusthub*. [Online]. Available: https://www.trust-hub.org
[24] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists based on machine learning," in *Proc. IEEE 22nd Int. Symp. On-Line Test. Robust Syst. Des. (IOLTS)*, Jul. 2016, pp. 203–206.
[25] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.
[26] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," in *Proc. IEEE 23rd Int. Symp. On-Line Test. Robust Syst. Des. (IOLTS)*, Jul. 2017, pp. 227–232.
[27] A. B. Kahn, "Topological sorting of large networks," *Commun. ACM*, vol. 5, no. 11, pp. 558–562, 1962.
[28] Z.-Q. Ma, H.-W. Yang, W.-D. Hu, and W.-X. Yu, "New topological sort algorithm based on adjacency matrix," *J. Comput. Appl.*, vol. 27, no. 9, pp. 2307–2309, 2007.
[29] L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia controllability/observability analysis program," in *Proc. 17th Design Automat. Conf.*, Jun. 1980, pp. 190–196.
[30] L. Goldstein, "Controllability/observability analysis of digital circuits," *IEEE Trans. Circuits Syst.*, vol. 26, no. 9, pp. 685–693, Sep. 1979.

QIANG LIU (M'14) received the Ph.D. degree from the Department of Electrical and Electronic Engineering, Imperial College London, London, U.K., in 2008. From 2009 to 2011, he was a Research Associate with the Department of Computing, Imperial College London. He is currently an Associate Professor with the School of Microelectronics, Tianjin University. His research interests include hardware security, VLSI design optimization, and reconfigurable computing.

PENGYONG ZHAO received the B.Eng. degree from the Shaanxi University of Science and Technology, Shaanxi, China, in 2017. He is currently pursuing the master's degree with the School of Microelectronics, Tianjin University. His current research interest includes hardware Trojan detection.

FUQIANG CHEN received the bachelor's degree from the School of Microelectronics, Tianjin University, Tianjin, China, in 2015, where he is currently pursuing the master's degree. His current research interest includes hardware Trojan design and detection.