Journal of Electronic Testing (2024) 40:741–759
https://doi.org/10.1007/s10836-024-06150-6

RESEARCH

A Feature-Adaptive and Scalable Hardware Trojan Detection Framework For Third-party IPs Utilizing Multilevel Feature Analysis and Random Forest

Yanjiang Liu[1] · Junwei Li[1] · Pengfei Guo[1] · Chunsheng Zhu[1] · Junjie Wang[1] · Jingxin Zhong[1] · Lichao Zhang[1]

Received: 1 April 2024 / Accepted: 24 October 2024 / Published online: 6 December 2024

Abstract
The trustworthiness of integrated circuits is susceptible to the hardware Trojans in the third-party intellectual property (IP) cores. Various hardware Trojan detection approaches have been proposed to identify the Trojans from a given netlist at the design stage, however, such methods suffer from low detection accuracy, poor scalability, small-scale circuit, and inability to cover more types of Trojans. To address these issues, a hardware Trojan detection framework for third-party IPs is proposed, which is capable of detecting the various types of Trojans with high accuracy and scalable to find the unknown Trojans. Some typical gate-level Trojans in the benchmark circuits are analyzed and 13 common features are introduced as the Trojan features. Furthermore, a feature extraction method is presented to extract 13 features of each node from 23 gate-level circuits and form the Trojan's feature set. Besides, a feature rebalancing method is proposed to balance the distribution of genuine and Trojan class, and a feature combination selection method is proposed to choose the optimal Trojan feature combination of each Trojan circuit, and thus the Trojan's feature database is established. Finally, the Trojan detection is formulated as the anomalous feature identification problem, and a feature-adaptive classifier based on the random forest is proposed to identify various types and features of Trojans. Simulation results of 23 Trojan circuits demonstrate that the proposed approach achieves a comparable precision (*TPR*=90.48%), recall (*TNR*=97.92%), and accuracy (*ACC*=97.96%) compared to the existing methods. Besides, the proposed approach is not only able to detect the existing Trojans in the circuits that scale from 0.2K to 120K nodes, but more importantly, it also has the unique advantage of identifying more types of Trojans by adding new Trojan features to the Trojan feature database.

Keywords Integrated circuit · Hardware Trojan · Multilevel feature analysis · Imbalanced data · Random forest

Responsible Editor: S. Bhunia

✉ Yanjiang Liu
liuyj_1013@126.com

Junwei Li
lijunwei3@163.com

Pengfei Guo
pfguo_hit@126.com

Chunsheng Zhu
cszhu01@126.com

Junjie Wang
njueewang@163.com

Jingxin Zhong
zhjx_08@163.com

Lichao Zhang
lichaozhang79@126.com

[1] School of Cryptographic Engineering, Information Engineering University, Shangcheng Road No.12, Zhengzhou 450000, Henan, China

1 Introduction

With the continuous development of integrated circuit design technology, modern designs equip themselves with several memory modules, various interfaces, and numerous processors or co-processors to seek high performance, deploying them in critical applications and sensitive fields. The complexity of ultra-large design, the urgent time-to-market, and the limited developing cost motivate the designers to reuse numerous third-party IP cores during the design stage. For the third-party IP cores, adversaries can easily modify the original design and insert hardware Trojans into it. Figure 1 shows the insertion stage of hardware Trojans during the design stage. Hardware Trojan is a well-designed malicious alteration of the original design, which can change the function specified by the design specification, leak the sensitive information of the circuit, or even make the entire system disabled. Consequently, the trustworthiness of third-party IP

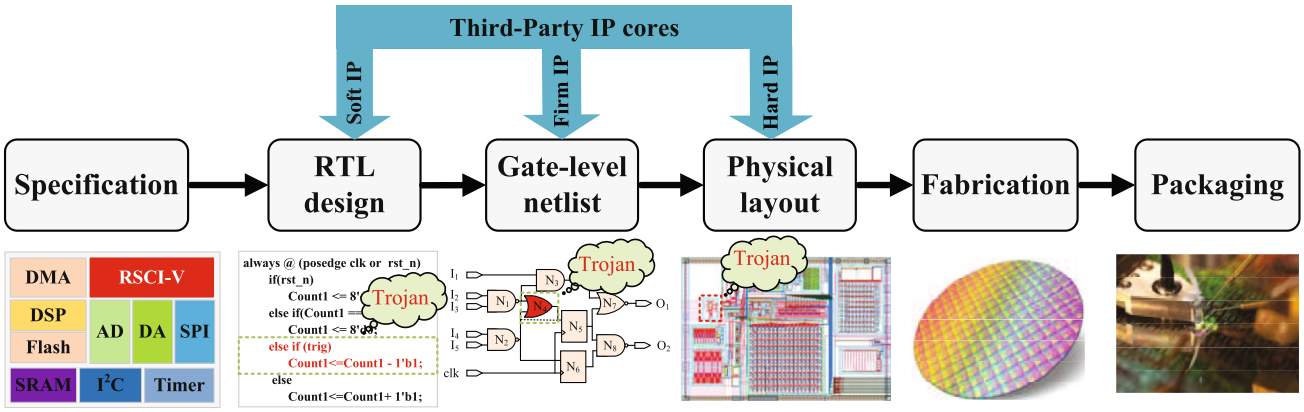

Fig. 1 Overview of the IC design flow and its security risk caused by hardware Trojan

cores is challenged by adversaries using the hardware Trojan, which has caused a tremendous threat to the security of integrated circuits and even information systems.

Recently, various hardware Trojan detection techniques have been explored in the literature over the past decades, including the design-time, run-time, and test-time countermeasures. Run-time countermeasures employ built-in-test logics (e.g. thermal sensors [1], dynamic function verification [2], and time-to-digital converter [3]) to monitor the anomalous behaviors in a given design. These techniques, however, need to modify the original design, which can degrade the main performance (e.g. power consumption, delay overhead, and area cost) that may not be allowed for the resource-constrained fields. Test-time countermeasures, including reverse engineering [4, 5], logic testing [6], and side-channel analysis [7–9], validate the trustworthiness of fabricated chips. However, once the hardware Trojan is detected at the test-time stage, it will cause a tremendous economic loss for integrated circuit suppliers. Considering the test cost and time, it is significant to have extra detection steps to evaluate the trustworthiness of the original design before fabrication. Recently, various design-time countermeasures have been explored over the past decades, including formal verification [10, 11], proof-carrying hardware [12], feature analysis [13], and so on. Formal verification and proof-carrying hardware exploit the specified security properties to validate the violations of design, which can only detect the functional-change hardware Trojans. Feature analysis has proven as a scalable and practicable software Trojan detection solution, and the Trojan feature database is extended when a new Trojan appears. Inspired by this, several feature analysis approaches for hardware Trojan, including the UCI [14], VeriTrust [15, 16], FANCI [17], FIGHT-metric[18], FASTrust [19], DeTrust [20], COTD [21], ML-FASTrust [13], and so on, have introduced. Specifically, various structural Trojan features [22–27], functional Trojan features [28–30], and multilevel Trojan features [31–33] are extracted from the Trust-HUB website benchmarks and used to identify the existence of hardware Trojan during the design stage. But those features are limited to several types of hardware Trojans. In the actual case, an imbalanced data problem exists wherein the samples belonging to genuine class heavily outnumber the Trojan class, and the conventional supervised classifiers are biased towards the genuine class. However, feature analysis approaches pay little attention to the imbalanced dataset problem, and the trained classifiers predict a poor performance of the other type of Trojans.

In this paper, a scalable hardware Trojan detection framework is proposed to identify the Trojan nodes from the gate-level netlist. The structural and stealthy characteristics of several gate-level Trojans are analyzed, and 13 Trojan features are introduced to cover all the existing Trojans. Further, a multi-level feature extraction method is proposed to extract the features of the gate-level netlist. Specifically, we establish the node-level directed graph for a gate-level netlist and calculate seven structural features upon the directed graph using the breadth-first search approach, and also analyze the stealthy behavior of Trojan and extract six testability features. Moreover, the synthetic minority over-sampling technique (SMOTE) and K-nearest neighborhood (KNN) are combined to generate synthetic data of the Trojan class and balance the Trojan feature database. Then, a feature-adaptive Trojan identification method based on the random forest is proposed to identify the Trojan features. Multiple decision trees are trained with a balanced dataset of all the possible feature combinations for learning all the Trojan features sufficiently. Experimental results show that the proposed framework scales well with large circuits and various types of Trojans. The main contributions are listed as follows.

- A data rebalancing method is proposed to expand the Trojan feature database. The classifier trained with the balanced dataset is sufficient to learn the Trojan features,

and the proposed approach achieves a comparable detection accuracy compared to the existing methods.

- A feature-adaptive Trojan detection method based on the random forest is proposed to establish multiple decision trees with the feature combinations and determine the Trojan nodes using the maximum voting principle of multiple decision trees. The proposed approach covers the existing Trojans and has detected 23 gate-level Trojan circuits.
- The proposed hardware Trojan detection framework scales well with different types of Trojans and various sizes of benchmarks. Besides, the features of the emerging Trojan can be also added to the Trojan feature database, which makes the proposed approach scalable to emerging Trojan.

The rest of this paper is organized as follows. Section 2 investigates the related work of feature analysis for hardware Trojan detection. Section 3 discusses the attack model and motivation of this paper. Section 4 introduces the multi-level Trojan feature analysis. Section 5 presents the overall framework of hardware Trojan detection. Section 6 gives the simulation results and discusses the proposed approach. Section 7 concludes this paper.

2 Previous Work

Since the hardware Trojan was proposed by Agrawal [34], numerous hardware Trojan detection approaches have been proposed over the past decades. Among them, feature analysis for hardware Trojan has become the most promising approach during the design stage. Feature analysis approaches, including structural feature analysis, functional feature analysis, and multi-level feature analysis, are aimed to extract and identify the specific features of Trojan. Next, we present the state-of-the-art feature analysis approaches for hardware Trojans.

Structural feature analysis: Hicks et al. propose a MUX-based Trojan that can evade the code coverage test and introduce the UCI algorithm to identify this type of Trojan[14]. Further, four stealthier features of Trojan are extracted from the control-data flow graph [19]. Authors propose a third-party IP trust verification framework named FANTrust, which identifies the time-triggered, data-triggered, sequential-triggered, and implicitly triggered Trojan. After that, a new feature quantitative analysis approach is proposed to calculate the Trojan rank by summing up three Trojan features. If the Trojan rank value exceeds the predefined threshold, it is regarded as a Trojan net [24]. The VeriTrust technique presented in [15] regards the redundant inputs as Trojan and identifies the potential trigger inputs at the design stage. In [23], the authors propose nine structural Trojan features and detect the Trojan net when the sum of these features exceeds the threshold. However, a fixed decision boundary will cause high false negatives and false positives on various benchmarks [23, 24]. Therefore, several machine learning algorithms, such as support vector machine [25, 27] and random forest [22], propose to improve the Trojan detection accuracy using an adaptive threshold. Further, Chen et al. introduce 13 structural features as Trojan features and exploit the local outlier factor algorithm to identify the anomalous features [34].

Functional feature analysis: In [17], functional analysis, namely FANCI, is proposed to identify the nodes with low control values as hardware Trojans. The low transition probability is considered as a Trojan feature [35]. However, Ref. [17] and [35] only aim at the combinational logic. After that, Sullivan et al. calculate the control value of sequential logic and identify the sequential Trojans according to the controllability and control value of internal signals [18]. Moreover, Zou et al. regard the extreme state and low switching probability as Trojan features. The author proposes a fast heuristic method to identify the nodes with low state and switching probability [36]. Further, Both the Ref. [37], [21], [38], and [39] regard the low testability nodes as the best candidate for the insertion sites of Trojan [21, 37–39]. Specifically, Sayandeep et al. detect the small pairwise correlation value of low controllability nodes [37], Salmani and Xie et al. assume the combinational controllability and observability features as the Trojan feature [21, 30], Priyadharshini et al. use several tools to extract the testability features, and Tebyanian et al. regard the combinational and sequential testability features as the Trojan feature [39]. However, a hardware Trojan design method presented in [40, 41] can disable the Trojan detection based on testability metrics. Further, the testability metric and dynamic transition analysis are combined to identify the inactive nodes of the netlist and locate the Trojan's trigger circuit [33].

Multilevel feature analysis: Structural feature analysis usually remarks numerous genuine nodes as Trojan because the structure of Trojan circuits is much similar to the original circuit, and functional feature analysis cannot cover all the types of Trojans. Therefore, several structural and testability features are combined to extend the scalability to different types of Trojans. In [32], two sets of structural features and three combinational testability features form the Trojan feature set. Further, the authors propose a feature-matching algorithm to match and identify the Trojan features from the netlist. Chen et al. exploit four structural Trojan features and a functional Trojan trigger feature to build a hardware Trojan detection framework [13]. Although the multi-level feature

analysis is an efficient and scalable hardware Trojan detection approach, researchers have given little attention to the imbalanced data problem and the bias of the classification model toward the majority class. Therefore, several genuine nodes are classified as Trojan nodes, and a functional testing or formal verification is needed to validate the trustworthiness of suspicious nodes.

3 Attack Model and Motivation

3.1 Attack model

In this paper, our proposed approach aims at distinguishing the Trojan features in a gate-level netlist using the Trojan feature database. We assume that the Trojan attack scenario is at the design stage, and the other stage is trustworthy. Specifically, the adversaries hidden in the design team may insert additional circuits into the original netlist, or the malicious module already exists in the third-party intellectual property cores. We focus on the structural and functional feature extraction of digital Trojans. However, analog Trojans, mixed-type Trojans, and parametric Trojans are not taken into consideration in this paper.

3.2 Motivation

Since the concept of hardware Trojan was proposed by Agrawal in 2007 [42], scholars have designed various Trojans with different types, sizes, and functions over the past decades. As mentioned above, numerous hardware Trojan detection approaches, including formal verification, logic testing, side-channel analysis, and so on, have been proposed to ensure the trustworthiness of circuits. However, there is no universal method to detect all known and unknown Trojans. Thus, feature analysis for software Trojan is applied to the hardware Trojan detection because the feature analysis can identify all the existing and future software Trojans. Similarly to the software Trojan, hardware Trojans with different types have common features that make it possible to propose a scalable hardware Trojan detection.

Based on this idea, several structural and testability features have been proposed over the past decades. More specifically, several structural features, including the unused circuit [14], redundant inputs [15], trigger structure, and typical structures, are extracted from benchmarks [22, 23, 25] and are used as the structural features of Trojans. Meanwhile, the control value [17], transition probability [35, 36], testability metrics [21, 30, 33, 37–39] are adopted as testability features of Trojan. Besides, various classification algorithms (e.g. K-means clustering [21], SVM [25, 27], and RF [22]) are used to match the features with the Trojan feature database and identify the existence of hardware Trojan.

The existing structural feature analysis approaches achieve a high hardware Trojan detection accuracy on various benchmarks with some false positives and negatives. Because the Trojan is a well-designed malicious modification for a specific design, and the structure of the Trojan is nothing different from the other design. Moreover, the size of Trojan nodes is much smaller than genuine nodes, and the genuine feature set outnumbers the Trojan feature set. Based on these imbalanced feature sets, the training model using traditional machine learning algorithms biases toward the genuine class, and several Trojan nodes are misclassified as genuine nodes using this classifier. There is no doubt that the increasing number of false alarms increases the workload of the manual check. To the best of our knowledge, the majority of the supervised learning algorithms ignore this imbalanced data problem during the classification model training stage. Besides, most of these techniques focus on the combinational features of Trojan's trigger part and ignore the sequential trigger and payload features. Moreover, all the existing functional feature analysis approaches can only detect triggered hardware Trojans and do not mention the other types of hardware Trojans. Consequently, multi-level feature analysis methods have been proposed to strengthen the current feature analysis approaches, which combine structural features and testability features to cover more types of Trojans. However, Ref. [13] exploits four structural features and a testability feature to detect the Trojans, and Ref. [32] uses six structural and three combinational testability features to identify the existence of Trojans. Besides, multi-level feature analysis methods mainly focus on the Trojan's trigger features and rarely mention the payload features. The combination of trigger and payload features may cover more types of Trojans. Although the multi-level feature analysis is an efficient hardware Trojan detection approach, the Trojan feature database should expand continuously with the increasing number of emerging Trojans.

Inspired by this, 13 features are proposed to make the hardware Trojan framework more universal. More specifically, we introduce six testability features (including the four trigger features and two payload features) and seven typical structural features extracted from the benchmarks to form the Trojan feature database that can cover all the types of existing hardware Trojans. Furthermore, a data rebalancing method based on the SMOTEKNN is proposed to balance the Trojan feature database. The supervised classification model using the balanced Trojan feature database learns well with the Trojan features and genuine features. Then, a Trojan feature identification method based on the random forest is proposed to identify the Trojan features from a netlist. Wherein multiple random forests with numerous decision trees predict the class of nodes with a maximum voting rule, which improves the Trojan detection accuracy and reduces the Trojan misclassification ratio simultaneously. Finally, a hardware Trojan

detection framework is introduced, which can be scalable to detect the future Trojan and locate the vulnerability of a gate-level netlist.

4 Multilevel Hardware Trojan Feature Analysis

In general, hardware Trojan consists of two parts: trigger and payload. The trigger condition can be a single/multiple logic value, a fixed time, a specific input pattern sequence, and a given state change. The payload implements information leakage, functional change, performance degradation, and so on. In this section, we analyze the characteristics of several hardware Trojans and introduce a set of Trojan features.

1) Feature of single-logic-triggered Trojans

The single-logic-triggered Trojan is activated when the predefined logic value reaches. From the testability perspective, high controllability signals are hard to reach a specified logic value. Therefore, signals with high controllability value are the ideal candidate for the insertion site of a single-logic-triggered Trojan. Several works have regarded the high observability signals as the Trojan nodes[21, 30, 36, 39]. The controllability metrics include the combinational controllability of logic 0 (CC0), combinational controllability of logic 1 (CC1), sequential controllability of logic 0 (SC0), and sequential controllability of logic 1 (SC1). The controllability value ranges from 1 and ∞, and the detailed calculation formulas are presented in [21, 39]. The feature of a single-logic-triggered Trojan can be summarized as follows.

Lemma 1 *The trigger signal of a single-logic-triggered Trojan has a high controllability (CC0, CC1, SC0, and SC1) value.*

2) Feature of multiple-logic-triggered Trojans

The multiple-logic-triggered Trojan is triggered when all the trigger inputs meet the predefined values. Due to its stealthy nature, adversaries typically select high observability signals as the trigger inputs. Purely using **Lemma** 1 can detect this Trojan, but some rare nodes may be misjudged as Trojan. Besides, the number of trigger inputs should be large enough to obtain a low activation probability. We use the value of FAN_IN to show the number of trigger inputs. The larger the FAN_IN is, the greater the number of trigger inputs is. Therefore, we use the **Lemma** 1 and the value of FAN_IN to detect the multiple-logic triggered Trojans. This observation can be summarized as **Lemma** 2.

Lemma 2 *All the trigger inputs of multiple-logic-triggered Trojans have a high controllability value and the trigger part has a large FAN_IN value.*

3) Feature of time-triggered Trojans

The time-triggered Trojans, like the AES-T2100, B19-T200, and RS232-T500, use a counter to count the number of clock cycles or transitions of internal signals. This Trojan is activated when the current counter is equal to a predefined value. In general, the size of a counter should be large enough to evade the functional verification methods. In [31], authors assume the counter size of time-triggered Trojans is at least larger than 20. Notice that a counter includes several flip-flips that form a loop group. The predecessor and successor of each counter node have several flip-flops. To detect the time-triggered Trojans, we use the number of flip-flops from the input side (FF_IN) and output side (FF_OUT) of the signal in a loop group as the Trojan feature. For the time-triggered Trojans, the FF_IN or FF_OUT of counter nodes tends to be large. The feature of time-triggered Trojans is described as follows.

Lemma 3 *All the trigger signals of time-triggered Trojans form a large loop group, and the trigger signal has a large FF_IN and FF_OUT value.*

4) Feature of pattern-triggered Trojans

The pattern-triggered Trojans, such as AES-T1000, BASICRSA-T100, and MC8051-T800, monitor whether the input pattern matches the specified trigger pattern. In this case, adversaries select an extremely low activation probability as the trigger pattern. Specifically, the width or the length of the trigger pattern's sequences should be large enough to avoid the spurious triggering of Trojan. For the former case, we use the minimum level from the primary inputs to the internal signal (LPI) to show the relationship between the input and internal node. The smaller the LPI value is, the higher the correlation between the input and internal node is. For the latter, such a trigger part has a large number of multiplexers. We use the number of multiplexers (MUX) for the path where the internal signal is located as the Trojan feature. The feature of pattern-triggered Trojan is presented as **Lemma** 4.

Lemma 4 *The pattern-triggered Trojans have a small LPI value or large MUX value.*

5) Feature of functional-change Trojans

The functional-change Trojans (e.g. B19-T100, RS232-T100, and WB_CONMAX-T300) modify the original specification after the Trojan is activated. It is well known that signals with a high observability value are difficult to propagate to the outputs. To better hide the function-change Trojan, adversaries generally choose the high observability signals as the implantation site of the Trojan's payload part. In the testability metrics, the observability metrics include combinational observability (CO) and sequential observability (SO). The observability value ranges from 0 to ∞, and the detailed calculation formulas are presented in [21]. Thus, we adopt the observability metrics (CO and SO) as the

functional-change Trojan's feature. This feature can be summarized as **Lemma** 5.

Lemma 5 *The payload signal of functional-change Trojans has a large observability (CO and SO) value.*

6) Feature of information-leakage Trojans

The information-leakage Trojans leak the secret information through the primary outputs (e.g. BASICRSA-T100, AES-T400, S38584-T200, and S38584-T300), covert channel (e.g. AES-T700, AES-T800, and AES-T900), or side-channel (e.g. AES-T600, AES-T1500, and S38417-T300). For the first case, Trojans use the primary outputs as the insertion site of the payload. Therefore, the level from the Trojan signal to the primary outputs is much smaller than other genuine signals. We exploit the minimum level from the signal to the primary output (LPO) as the Trojan feature. For the second and third cases, the payload part does connect with the original design, which will be removed during the synthesis process. Besides, Trojans use a large bit-width shift register and eight flip-flops to create a covert code-division multiple access channels in the second case. Such a structure has numerous flip-flops, thus, the payload signal has a large FF_IN and FF_OUT value. We can detect this type of Trojan using **Lemma** 3. For the third case, Trojans exploit a large bit-width shift register or ring oscillator with an odd number as the leakage circuit. **Lemma** 3 can detect the first type of Trojans in the third case. The ring oscillator is a loop. Further, we use the number of inverters in a loop group (denoted as INV) as the Trojan feature, which can detect the second type of Trojans in the third case. The feature of information-leakage Trojans can be summarized as follows.

Lemma 6 *The information-leakage Trojans have a small LPO value, large FF_IN and FF_OUT value, or large INV value.*

7) Feature of performance-degradation Trojans

The performance-degradation Trojans decrease the expected lifetime of the battery by increasing the power consumption (e.g. AES-T500, AES-T1800 and AES-T1900) or reduce the critical frequency by extending the critical path (e.g. S35932-T300). In the former case, the Trojan payload continuously rotates after the Trojan is activated. We can detect this hardware Trojan using **Lemma** 3. For the latter, the Trojan payload is a ring oscillator along the critical path, which slows down this path when the ring oscillator oscillates. It can also detect this case by employing the INV value of **Lemma** 6. Therefore, combining the **Lemma** 3 and **Lemma** 6 can identify the performance-degradation Trojans.

Lemma 7 *The performance-degradation Trojans have a large FF_IN and FF_OUT value, or large INV value.*

In summary, we propose seven structural and six testability features to detect the above types of Trojans in the Trust-Hub benchmark. The description of Trojan features is shown in Table 1. Regarding Table 1, those features can be scalable to more types of other Trojans. For the always-on Trojans (e.g. AES-T100, AES-T200, and AES-T300), the payload part includes a large bit-width shift register, which can be detected using **Lemma** 3. Moreover, denial-of-service Trojans (e.g. AES-T1800, RS232-T1500, S38417-T200, and VGA_LCD-T100) are identified with several features. We can lower the number of suspicious signals using multiple features. To be specific, RS232-T1500 is triggered when the 32-bit counter reaches the maximum value and keeps an output signal stuck at 0. We can use the **Lemma** 3 to identify the 32-bit counter of the trigger part and exploit the **Lemma** 6 to locate the payload part. Therefore, we can conclude that the proposed Trojan features not only identify existing Trojans in the Trust-hub

Table 1 A summary of Trojan features

Feature type	Trojan feature	Description
Testability features	CC0	The combinational controllability of logic 0 of the node *n*.
	CC1	The combinational controllability of logic 1 of the node *n*.
	CO	The combinational observability of the node *n*.
	SC0	The sequential controllability of logic 0 of the node *n*.
	SC1	The sequential controllability of logic 1 of the node *n*.
	SO	The sequential observability of the node *n*.
Structural features	LPI	The minimum level from the primary inputs to the node *n*.
	LPO	The minimum level from the node *n* to the primary output.
	FAN_IN	The number of inputs of the node *n*.
	INV	The number of inverters for the loop path where the node *n* is located.
	MUX	The number of multiplexers for the path where the node *n* is located.
	FF_IN	The number of flip-flops from the input side of the node *n* in a loop group.
	FF_OUT	The number of flip-flops from the output side of the node *n* in a loop group.

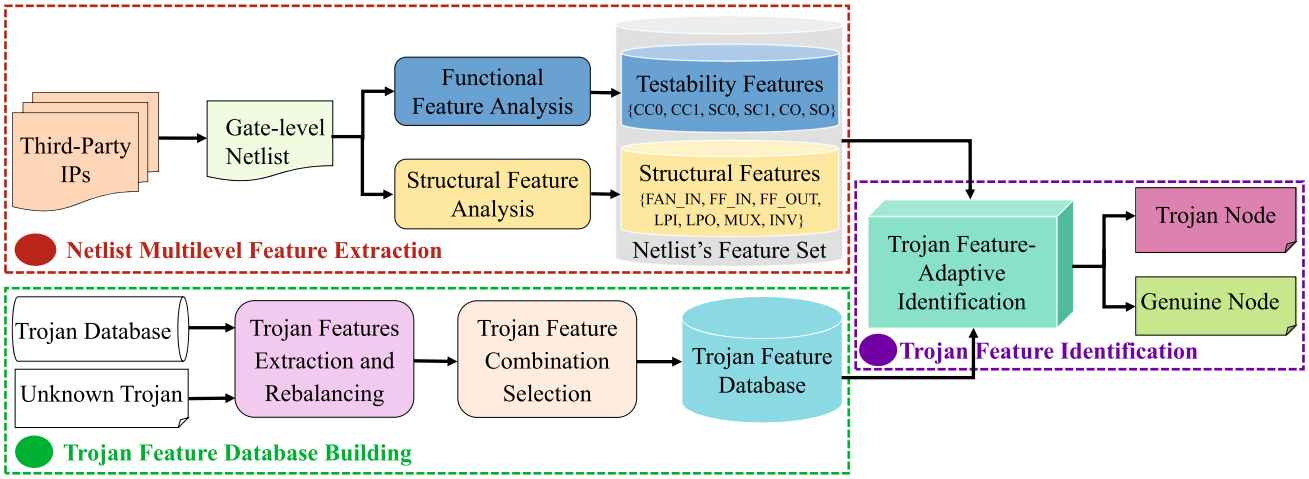

Fig. 2 The overall framework of hardware Trojan detection

benchmarks, but also be scalable to defend against unknown Trojans.

5 Hardware Trojan Detection Framework Utilizing Multilevel Feature Analysis and Random Forest

5.1 Overall framework

The overall framework is illustrated in Fig. 2, including the gate-level netlist's feature extraction, Trojan's feature database building, and Trojan feature identification. In the gate-level netlist's feature extraction, testability and structural features of each node are extracted from the gate-level netlist. More specifically, functional feature analysis is performed to calculate the controllability value (CC0, CC1, SC0, and SC1) and observability value (CO and SO) of each node, and structural feature analysis is conducted to calculate the structural feature value (FAN_IN, FF_IN, FF_OUT, LPI, LPO, MUX, and INV). In the Trojan's feature database building stage, the features of Trojans in the Trust-Hub benchmark are extracted, re-sampled and selected to build a balanced Trojan feature database. Finally, a feature-adaptive classifier based on the random forest is built and trained with the Trojan feature database. The feature set of the netlist under test is fed into the trained classifier and the Trojan nodes are found by identifying the feature outliers.

5.2 Gate-level netlist's structural and testability features extraction method

As described in Section 4, signals with high controllability and observability value are the ideal implantation site for hardware Trojans. In this paper, we use testability metrics as the Trojan testability features. The testability calculation method is provided in Ref. [21] and [39]. For the sake of simplicity, we implement a testability measurement tool to calculate the testability features of each node in a circuit.

Further, we build the directed graph of a gate-level netlist and extract the proposed structural features of each node. Figure 3 (a) shows an example circuit. Where $I_1, I_2, \cdots, I_5$

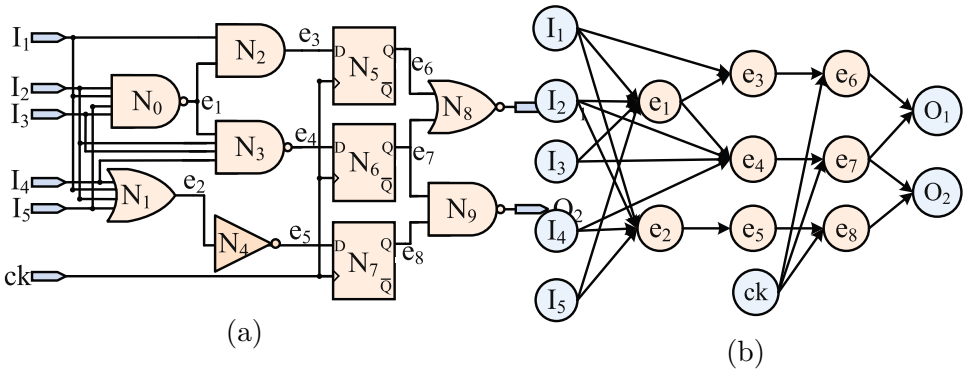

Fig. 3 The gate-level schematic of a circuit (a) and corresponding directed graph (b)

and ck are the inputs, O_1 and O_2 are the outputs, N_0, N_1, $\cdots$, N_9 are the instances, and $e_1, e_2, \cdots, e_8$ are the internal nodes. In the directed graph, the inputs, internal nodes, and outputs are modeled as vertexes $\mathbf{V}=\{I_1, I_2, \cdots, I_5, ck, O_1, O_2, e_1, e_2, \cdots, e_8\}$, and the instances are expressed as edges $\mathbf{E}=\{N_0, N_1, \cdots, N_9\}$. Therefore, the corresponding directed graph is shown in Fig. 3 (b). We take vertex e_4 as an example to calculate the structural features of a circuit. The vertexes pointing to the e_4 are e_1, I_2, I_3, and I_4, thus, the FAN_IN of e_4 is equal to 4. Regarding Fig. 3 (a), the path from the inputs to the internal node e_4 has no flip-flops and only a flip-flop (N_6) exists from vertex e_4 to outputs (e.g. O_1 or O_2). Therefore, the FF_IN and FF_OUT of e_4 are 0 and 1, respectively. The value of inputs (I_1, I_2, I_3, I_4, and I_5) directly propagates to the e_4 through the instance N_3, thus, the LPI of e_4 is 0. The e_4 propagates to the e_7 through the instance N_6, and the e_7 flows to the outputs after passing through the instance N_8 or N_9 ($e_4 \rightarrow e_7 \rightarrow O_1$ or $e_4 \rightarrow e_7 \rightarrow O_2$). So, the LPO of e_4 is 1. The path where the e_4 is located has no inverters and multiplexers, thus, the INV and MUX are equal to 0.

A large-scale design includes millions or even billions of instances and nodes, and the corresponding directed graph has a similar amount of vertexes and edges. Based on **Lemma** 2, the FAN_IN of the current vertex is the number of adjacent input edges. However, the other structural features (FF_IN, FF_OUT, LPI, LPO, INV, and MUX) are correlated with the predecessor and successor vertexes of the directed graph. The computational complexity of feature extraction is the exponential order of the number of vertexes. To improve the efficacy of feature extraction, a vertexes-searching method based on the breadth-first search is introduced to search the vertexes of the graph.

We use the i-th vertex v_i of $\mathbf{G}$ to explain the vertexes searching process. The v_i is the starting point, and the adjacent first-level vertexes $\mathbf{W}=\{w_1, w_2, \cdots, w_n\}$ of v_i are extracted from the $\mathbf{G}$ using the function $\mathscr{A}$, which is expressed in Eq. 1.

$$\mathbf{W} = \mathscr{A}(\mathbf{G}, v_i) \tag{1}$$

The vertex in $\mathbf{W}$ is marked as accessed in order, and the first accessed vertex is added into the marked vertex set $\mathbf{X}$. If the vertex w_i has no next-level adjacent vertex, the searching process is jumped to the w_{i+1}, otherwise, the next-level adjacent vertexes of vertex w_i are searched and the searching results are added into the second-level adjacent vertexes set $\mathbf{H}=\{H_{w_1}, H_{w_2}, \cdots, H_{w_n}\}$. The searching process of $\mathbf{W}$ is expressed in Eq. 2. Where the H_{w_i} is the next-level adjacent vertex set of w_i.

$$H_{w_i} = \begin{cases} \mathscr{A}(\mathbf{G}, w_i), & w_i \notin \mathbf{X}. \\ \varnothing, & w_i \in \mathbf{X}. \end{cases} \tag{2}$$

Algorithm 1 : structural features extraction algorithm.

Input: $\mathbf{G}$, v_i; ▷ $\mathbf{G}$ is the directed graph of gate-level netlist and v_i is a vertex of $\mathbf{G}$.
Output: FAN_IN(v_i), FF_IN(v_i), FF_OUT(v_i), LPI(v_i), LPO(v_i), INV(v_i) and MUX(v_i); ▷ 7 features of v_i.
1: $\mathbf{P}=\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n\} \leftarrow path\_search(\mathbf{G}, v_i)$; ▷ search all the paths where the v_i locates.
2: $In\text{-}degree(v_i) \leftarrow Indegree(\mathbf{G}, v_i)$; ▷ calculate the value of feature FAN_IN of v_i.
3: **for** $\mathbf{p}_i \in \mathbf{P}$ **do**
4: **lpio_min**$(i,1{:}2) \leftarrow Inout\_Num(\mathbf{p}_i, v_i)$; ▷ determine the minimum number of inputs and outputs to the v_i.
5: **mux_max**$(i) \leftarrow Mux\_Num(\mathbf{p}_i, v_i)$; ▷ determine the maximum number of multiplexers along the v_i.
6: $\mathbf{K}=\{\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_m\} \leftarrow Loopgroup(\mathbf{p}_i)$; ▷ determine the loop set where the v_i locates.
7: **if** $\mathbf{K} \neq \varnothing$ **then**
8: **for** $\mathbf{k}_j \in \mathbf{K}$ **do**
9: **ffinout_loop**$(i, j{:}j{+}1) \leftarrow FF\_Num(\mathbf{k}_j)$; ▷ count the maximum number of flip-flops from the input and output side.
10: **inv_loop**$(i, j) \leftarrow INV\_Num(\mathbf{k}_j)$; ▷ count the maximum number of inverters.
11: **end for**
12: **else**
13: **ffinout_loop**$(i, j{:}j{+}1)$, **inv_loop**$(i, j) \leftarrow 0$;
14: **end if**
15: **end for**
16: FF_IN(v_i), FF_OUT(v_i) $\leftarrow$ max(**ffinout_loop**); INV(v_i) $\leftarrow$ max(**inv_loop**); MUX(v_i) $\leftarrow$ max(**mux_max**);
17: LPI(v_i), LPO(v_i) $\leftarrow$ min(**lpio_num**); ▷ output the feature value of v_i.

Once a vertex is marked as accessed, the next-level adjacent vertexes searching process of this vertex is ignored and the searching process is jumped to the next vertex. If the marked vertexes set $\mathbf{X}$ equal to $\mathbf{V}$, the searching process is completed, otherwise, an unmarked vertex is chosen from the graph as the starting point for the next search. The structural feature extraction process is presented in algorithm 1. All the paths ($\mathbf{P}=\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n$) of v_i are searched using the function $path\_search$, and the number $In\text{-}degree(v_i)$ of input edges for v_i is calculated using the function $Indegree$. For each path $\mathbf{p}_i$, the minimum number **lpio_min** of vertexes from the inputs and outputs to the v_i is calculated using the function $Inout\_Num$, and the maximum number **mux_max** of multiplexers along the path $\mathbf{p}_i$ is determined using the function Mux_Num, and all the loops ($\mathbf{K}=\{\mathbf{k}_1, \mathbf{k}_2, \cdots, \mathbf{k}_m\}$) where the v_i locates are obtained using the function $Loopgroup$. If $\mathbf{K}$ is a nonempty set, it indicates the path $\mathbf{p}_i$ exists several loops, otherwise, the **ffin_loop**, **ffout_loop**, and **inv_loop** are equal to 0. For each loop $\mathbf{k}_j$, the maximum number of flip-flops from the input side **ffin_loop** and output side **ffout_loop** to v_i is counted with the function FF_Num, and the maximum number **inv_loop** of inverters is calculated using the function INV_Num. Finally, the maximum value of **ffin_loop**, **ffout_loop**, **inv_loop**, and **mux_max** is the value of feature FF_IN, FF_OUT, INV, and MUX of v_i, and the minimum

value of **lpio_min** is the value of feature LPI and LPO of v_i, respectively.

5.3 Trojan features database building method

Since 2013, various types, sizes, and functions of Trojans are proposed in the Trust-Hub Trojan benchmark [43]. All the features of each benchmark circuit are extracted to form the Trojan feature database. The Trojan and genuine vertexes are labeled as Trojan and genuine classes, respectively. As shown in the Trojan benchmark, the size of the Trojan circuit is much smaller than the original design and the number of genuine nodes is significantly greater than Trojan nodes. In this case, the distribution of genuine nodes and Trojan nodes is unbalanced. The imbalanced ratio, the ratio between the number of the Trojan nodes and genuine nodes, is as high as 1:236 in the s35932-T200 and even 1:3318 in the EthernetMAC10GE_T700. Therefore, an imbalanced data problem exists in the Trojan feature database building process. The classifier learned from the unbalanced data sets biases towards the genuine class and obtains a poor prediction performance of the Trojan class. Consequently, a data rebalancing method is needed to solve the imbalanced data problem and establish a fair classifier with a balanced data set.

Various techniques, including over-sampling, under-samp ling, and hybrid sampling, have been explored to solve the imbalanced data problem. The over-sampling methods, including the synthetic minority over-sampling technique (SMOTE) [44], borderline-SMOTE [45], and adaptive synthetic sampling technique (ADASYN) [46], add new synthetic samples into the minority class, while the under-sampling approaches, such as Tomek Links [45], K-nearest neighborhood (KNN) [47] and ensemble [48], remove the samples of the majority class. Thus, the hybrid sampling technique combines the advantages of over-sampling and under-sampling methods to eliminate the effect of overfitting and loss of essential information, which has been shown as a promising approach in recent years. Among all the hybrid sampling techniques, SMOTEKNN increases the number of minority class samples efficiently and removes the samples concentrated at the boundary, which is very suitable for Trojan detection. This is mainly because the new Trojan feature samples should reflect the total characteristics of Trojan features and avoid the unclear boundaries caused by excessive data expansion. Thus, we use the SMOTEKNN to rebalance the Trojan features, which generate the synthetic samples of the minority class using the SMOTE over-sampling algorithm and remove the over-sampled synthetic samples using the KNN under-sampling algorithm.

Let the genuine samples $\Psi=\{\psi_1, \psi_2, \cdots, \psi_n\}$ and Trojan samples $\Lambda=\{\lambda_1, \lambda_2, \cdots, \lambda_m\}$ explain the feature rebalancing process. Where ψ_i and λ_j are the i-th genuine sample and j-th Trojan sample respectively, and $n \gg m$. The k nearest neighbor samples $\Gamma=\{\gamma_1, \gamma_1, \cdots, \gamma_k\}$ of λ_j are selected from the Λ, which is expressed as Eq. 3. Where f is the Euclidean distance calculation function and Γ is a subset of Λ.

$$\Gamma \leftarrow \underset{k}{argmin} f(\lambda_j, \Lambda) \tag{3}$$

The synthetic sample ξ_{ji} is the linear interpolation between the λ_j and γ_i, which is expressed in Eq. 4. Where α scales from 0 to 1 and γ_i is the i-th nearest neighbor sample of λ_j.

$$\xi_{ji} = \lambda_j + \alpha(\lambda_j - \gamma_i), i = 1, 2, \cdots, k \tag{4}$$

Combining with Eqs. 3 and 4, all the synthetic samples of each λ_j in Λ are obtained and the synthetic data set is Ξ. The Ξ, Ψ, and Λ form the Trojan feature set. However, the synthetic samples in Ξ may be closer to the genuine class instead of the Trojan class and those synthetic samples are labeled as the Trojan class. The classifier trained with those mislabeled samples builds a distortion decision boundary that can not reflect the actual distribution of genuine class and the Trojan class. To address this issue, we use the KNN to remove the synthetic samples that are close to the genuine class. The KNN determines the class of each synthetic data by measuring the similarity between Ψ and Λ. If the ξ_{ji} is classified as genuine class, ξ_{ji} is removed from the Ξ and a new synthetic data μ_{ji} is generated using the SMOTE again. Otherwise, the ξ_{ji} is retained as a synthetic data ξ'_{ji}. The data-cleaning process of KNN is shown in Eq. 5.

$$\xi'_{ji} = \begin{cases} \xi_{ji}, & C_{\xi_{ji}} = 1; \\ \mu_{ji}, & C_{\xi_{ji}} = 0; \end{cases} \tag{5}$$

In the Trust-Hub benchmark site, Trojans have more than one feature and the features differ from each other. For example, the trigger node of a single-logic-triggered Trojan has a large CC0, CC1, SC0, and SC1 value, while a Trojan payload node only has a large CO and SO value. As mentioned in Section 4, Trojans have several features that reflect the characteristics of Trojan from the different perspectives. Considering the various types of Trojans, there might be considerable differences in several features between the genuine nodes and Trojan nodes, while the other features of genuine nodes are similar to the Trojan nodes. Actually, not all the features in Table 1 are suitable for the hardware Trojan detection. To ensure a high detection accuracy, we calculate all the Trojan feature combination and choose the optimal Trojan feature combination as the Trojan feature. In this paper, a cost function is used to assess the quality of feature combination. The cost function Υ is expressed in Eq. 6. Where n is the number of features for this feature combination, m is the number of Trojan nodes, T_{ij} is the value of i-th feature

of j-th Trojan node and G_i is the centroid of the i-th feature of all the genuine node.

$$\Upsilon = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(T_{ij} - G_i)^2 \tag{6}$$

Regarding Eq. 6, the cost function measures the distance between the features of genuine and Trojan nodes. The larger the value of Υ_i, the greater the feature difference between the genuine and Trojan nodes, and the better the Trojan feature combination is. In other word, the larger the value of Υ_i, the more efficacy the Trojan feature is, the easier it is for analyzer to detect the Trojans. Therefore, we sort the cost functions of all the feature combinations and choose the Trojan feature combination Γ corresponding to maximum cost function value as the feature of this Trojan. The sorting criterion is expressed in Eq. 5. Where T is the current Trojan circuit, Γ is all the feature combinations and $\mathcal{F}$ is the optimal feature combination for T. The selected Trojan feature combination is used to build an effective Trojan feature database that can cover more types of Trojans.

$$\mathcal{F} = \underset{T,\Gamma}{argmax}(\Upsilon) \tag{7}$$

5.4 Trojan identification method based on the feature-adpative random forest classifier

Random forest is a popular ensemble learning approach, which is widely applied to data mining, pattern recognition, fault prediction, and density estimation fields. The random forest selects a given set of samples from the training data stochastically, constructs numerous decision trees, and makes the prediction from the results of decision trees. Random forest uses voting as the decision-making rule, which achieves good accuracy even if a large proportion of the critical features of the dataset are missing.

As mentioned above, the Trojan includes one or more features, and each feature should be considered during the Trojan detection. Moreover, the features of the Trojans differ from each other. If all the features are considered using the random forest, the minor features caused by hardware Trojan may be masked. Inspired by this, a feature-adaptive random forest is adopted to detect the features of Trojan nodes in a design. Multiple decision trees are trained with a set of feature combinations from the balanced Trojan feature database, which can learn well with the Trojan and genuine features. The trained decision trees are not only able to cover the current Trojans available in the Trojan benchmarks but also scalable to the emerging Trojans that contain one or more of those features.

The Trojan feature identification process is shown in Fig. 4. The balanced Trojan feature database has k feature combinations ($f_1, f_2, \cdots, f_k$), and all the samples are marked as two classes: 1 (Trojan class) and 0 (genuine class). 13 features of the netlist under test are obtained using the feature extraction method, and the k feature combinations are selected from the Trojan feature database according to the cost function. We use each feature combination to train a decision tree. More specifically, the training data of i-F classifier is the i-th feature combination $\mathcal{F}_i$, $\mathcal{F}_i$=$\{f_1^i, f_2^i, \cdots, f_j^i, \cdots, f_n^i\}$ and n is the number of nodes of the netlist. f_j^i is a $e\times1$ vector, e is the number of features in this feature combination and each column of f_j^i represents the corresponding feature value of j-th nodes. k decision trees are trained with the training data $\mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_k$, respectively, and each decision tree is used to predict the class of test data $\mathcal{V}$. The prediction process of i-th decision tree $\mathcal{D}t_i$ can be expressed in Eq. 8. Where $\mathbf{C}^i$ is the prediction result of test data, and $\mathbf{C}^i$=$\{C_1^i, C_2^i, \cdots, C_n^i\}$.

$$\mathbf{C}^i = \mathcal{D}t_i(\mathcal{V}) \tag{8}$$

Finally, all the decision trees are used to evaluate the trustworthiness of each node of netlist under test and make the prediction with a majority vote of the decision trees. If the sum of the decision trees' outputs C_j^i is greater than the specified threshold F_{th}, the j-th node is classified as a Trojan node, otherwise, the j-th node is identified as a genuine node. The class C_j of the j-th node is expressed in Eq. 9.

$$C_j = \begin{cases} 1. & \sum_{i=1}^{k} C_j^i \geq F_{\text{th}}; \\ 0. & \sum_{i=1}^{k} C_j^i < F_{\text{th}}; \end{cases} \tag{9}$$

6 Simulation Results and Analyses

6.1 Simulation Results

In this paper, 23 gate-level Trojan circuits available in the Trust-Hub benchmark site are used to demonstrate the efficacy of the proposed approach. The details of Trojan circuits are given in Table 2. Where the genuine circuit VGA_LCD, WB_CONMAX, EthernetMAC10GE respectively are denoted as V, W and Ethernet, the number of genuine nodes and Trojan nodes respectively are denoted as T_r and T_j, and the corresponding results are presented in the second and third columns of Table 2. From the results of T_r and T_j in Table 2, the size of Trojan circuits ranges from 0.2K to 102K. Moreover, the ratio R_a=$\frac{T_r}{T_j}$ between the genuine nodes and Trojan nodes is calculated and the results are given in the fourth column of Table 2. The R_a ranges from 4.67 to 5372, which shows that the distribution of genuine

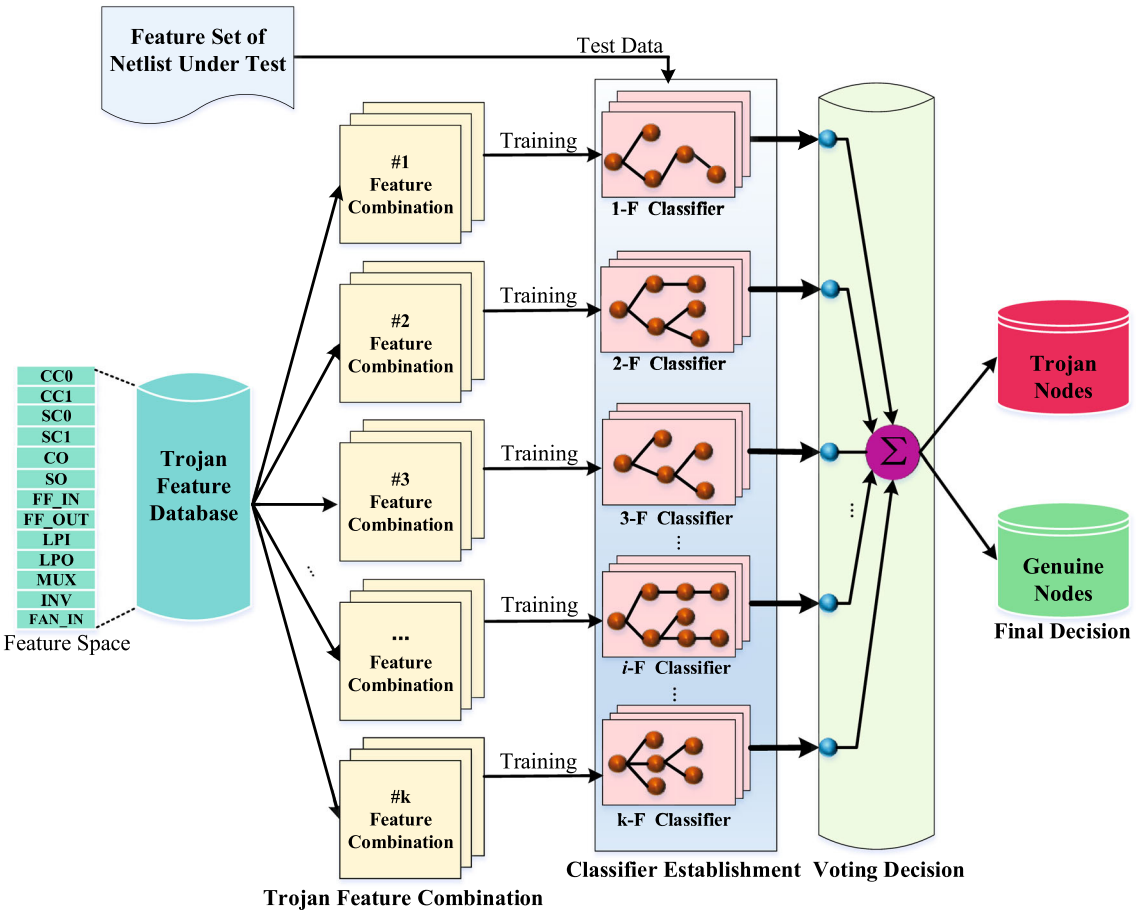

Fig. 4 The procedure of Trojan feature identification based on the feature-adaptive random forest classifier

nodes and Trojan nodes is unbalanced. Figure 5 (a) shows the distribution of genuine nodes and Trojan nodes in the RS232-T1000. As illustrated in Fig. 5 (a), the number of genuine nodes is greater than the Trojan nodes. Besides, we choose 3 features (CO, SO and FAN_IN) to show the feature distribution of genuine and Trojan nodes. The clustering of genuine nodes is evident, while the clustering of Trojan nodes are dispersed. Considering the dispersed distribution of Trojan nodes, it is extremely difficult to determine a reasonable hardware Trojan feature boundary. It's necessary to consider the imbalanced data problem and introduce a data rebalancing method to solve this problem. The ratio of R_a of samples from minority classes to samples from majority classes is used in the fitness function of data balancing algorithm. The over-sampling process is stopped when the R_a reaches 50%. Several over-sampled synthetic samples under the boundary are cleaned, and the value of R_a is approximately equal to 50%. Figure 5 (b) shows the balanced distribution of genuine and Trojan nodes. The number of Trojan nodes is equal to genuine nodes.

Furthermore, we analyze the characteristics of Trojan circuits and list the features of each Trojan circuit. Regarding the sixth column of Table 2, all the Trojans have more than one feature. Regarding the features of RS232-T1000 and RS232-T1100 in the Table 2, RS232-T1000 and RS232-T1100 have a large testability value compare to the genuine nodes. Figure 6 shows the value of two features of RS232-T1000 and RS232-T1100. We classify the genuine nodes as the normal class and the Trojan nodes as the Trojan class. As shown in Fig. 6 (a), the CC and CO of Trojan nodes are greater than genuine nodes. Similarly, the SC and SO of Trojan nodes outweigh the genuine nodes in Fig. 6 (b).

Actually, the features of hardware Trojans reflect the stealthy characteristics of Trojan from the different perspectives. To improve the detection efficacy of Trojan, we should consider all the features during the Trojan feature identification process. However, different types of Trojans have varied features. From the feature distribution of Trojan feature database, we find that not all the Trojan features have significant differences from genuine nodes. Besides, some Trojan features are similar with the genuine nodes. Figure 7 shows the similarity features between the genuine nodes and Trojans. Although the S35932 and S38584 have a good stealthy characteristics, both the feature CC and CO of Tro-

Table 2 23 Trojan circuits available in the Trust-Hub benchmark site for the experiments

Circuits	T_r	T_j	R_a	Trojan Function	Trojan Features
RS232–T1000	215	44	4.88	Change-functionality	CC0, CC1, CO, SC0, SC1, SO, LPO
RS232–T1100	216	43	5.02	Change-functionality	CC0, CC1, CO, SC0, SC1, SO
RS232–T1200	222	37	6	Change-functionality	CC0, CC1, SC0, SC1, LPO
RS232–T1300	223	31	7.19	Change-functionality	CC0, CC1, SC0, SC1, LPO
RS232–T1400	213	45	4.73	Change-functionality	CC0, CC1, SC0, SC1, LPO
RS232–T1500	215	46	4.67	Change-functionality	CC0, CC1, CO, SC0, SC1, SO, LPO
RS232–T1600	228	30	7.6	Change-functionality	CC0, CC1, SC0, SC1, LPO
S15850–T100	2386	60	39.76	Denial-of-service / change-functionality	CC0, CC1, SC0, SC1, FAN_IN, LPO
S35932–T100	6389	32	199.65	Leak-information / change-functionality	CC0, CC1, SC0, SC1, FAN_IN, LPI, LPO
S35932–T200	6390	27	236.66	Denial-of-service / change-functionality	CC0, CC1, CO, SC0, SC1, SO, FAN_IN, LPI
S35932–T300	6386	55	116.1	Denial-of-service / degrade-performance	CC0, CC1, SC0, SC1, FAN_IN, LPI, INV
S38417–T100	5782	29	199.37	Denial-of-service / change-functionality	CC0, CC1, CO, SC0, SC1, SO, FAN_IN
S38417–T200	5779	35	165.11	Denial-of-service / change-functionality	CC0, CC1, CO, SC0, SC1, SO, FAN_IN
S38417–T300	5785	61	94.83	Denial-of-service / change-functionality	CC0, CC1, CO, SC0, SC1, SO, FAN_IN, INV
S38584–T100	7344	19	386.52	Denial-of-service / change-functionality	CC0, CC1, CO, SC0, SC1, SO, FAN_IN, LPI
S38584–T200	7335	136	53.93	Leak-information / change-functionality	CC0, CC1, CO, SC0, SC1, SO, LPO, MUX, FF_IN, FF_OUT
S38584–T300	7334	1154	6.35	Leak-information / change-functionality	CC0, CC1, CO, SC0, SC1, SO, LPO, FF_IN, FF_OUT, MUX
V_T100	69836	13	5372	Leak-information / change-functionality	CC0, CC1, SC0, SC1, LPO
W_T100	22162	35	633.2	Leak-information / change-functionality	CC0, CC1, CO, SC0, SC1, SO, FAN_IN
Ethernet_T700	102888	31	3318.96	Change-functionality	CC0, CC1, CO, SC0, SC1, SO, FAN_IN
Ethernet_T710	102888	31	3318.96	Change-functionality	CC0, CC1, CO, SC0, SC1, SO, FAN_IN
Ethernet_T720	102888	31	3318.96	Change-functionality	CC0, CC1, CO, SC0, SC1, SO, FAN_IN
Ethernet_T730	102889	30	3429.63	Change-functionality	CC0, CC1, CO, SC0, SC1, SO, FAN_IN

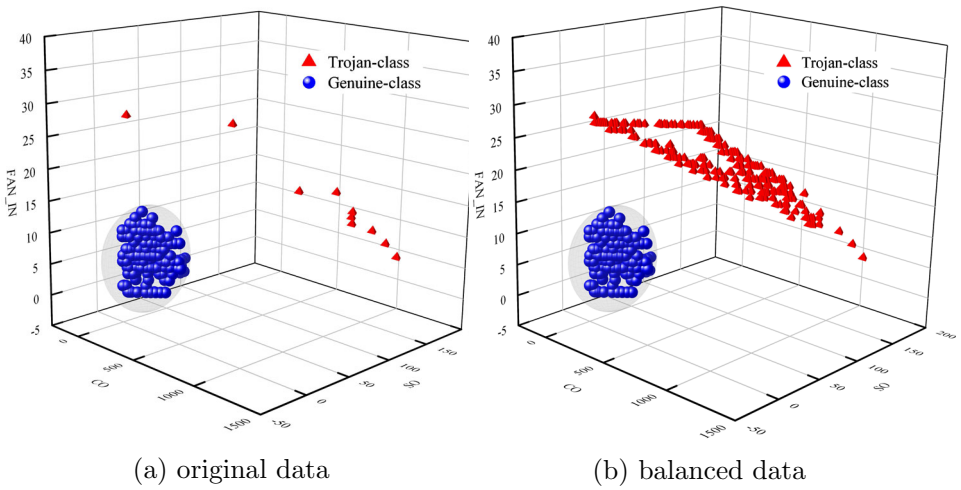

Fig. 5 The distribution of genuine nodes and Trojan nodes in the RS232-T1000

(a) original data (b) balanced data

jan nodes are similar to the genuine nodes. Thus, the value of CC and CO falls within the genuine nodes.

Thus, we can conclude that not all the Trojan features are suitable for detecting the hardware Trojans. It is necessary to select the optimal combinations of Trojan features for improving the Trojan detection efficacy. Figure 8 shows several feature combination scheme of RS232-T1200 and S15850. Regarding the Fig. 8 (a) and (c), the genuine nodes gather near the origin of coordinate system composed of feature CC and CO, while the Trojan nodes distribute far from the origin. More specifically, the genuine nodes are densely distributed with high local density, while Trojan nodes are sparse with low local density. Similarly, the distribution of feature SC and SO in Fig. 8 (b) and (d) is accord with the Fig. 8 (b) and (d). Therefore, we sort all the feature combinations and choose the optimal feature combination.

After sorting the Trojan's feature combinations, the corresponding feature combinations of all the Trojan circuits form the Trojan feature database. Further, we use the random forest to train the classification model with the Trojan feature database. In the training process, six parameters, including the split criterion, number of decision trees (ω), maximal depth of decision trees (ϱ), the minimum number of observations required to split internal nodes (ρ), and the minimal number of observations per tree leaf (φ), are adjusted to achieve an efficient classification model. We choose the gini as the split criterion of decision trees and the validation accuracy of decision trees can reach 99.82%. Similarly, the other parameters are determined according to the validation results. More specifically, ω ranges from 10 to 210, ϱ scales from 10 to 31, ρ varies from 2 to 10, and φ fluctuates from 2 to 15. To improve the searching efficiency, an iteration script is developed to change the value of parameters and search for the optimized parameter with maximal validation accuracy. The proposed framework is implemented in Python and a personal computer with Intel Core i5-8265U CPU@1.60GHz and 8GB of RAM are used for experiments. In this paper, 23 circuits were tested and the total time complexity of the algorithms is calculated. The total time required for the feature extraction, data balancing, and prediction phase is 116.68 minutes, 3.43 seconds, and 18.47 seconds, respectively. WWe only need an average of 5.086 minutes to analyze the trustworthiness of a circuit. Finally, ω, ϱ, ρ, and φ are equal to 201, 22, 33, 1, respectively, and the validation accuracy reaches 99.83%.

After extracting the multi-level feature extraction of the netlist under test, the trained classifiers are used to classify the samples and distinguish the Trojan nodes from the

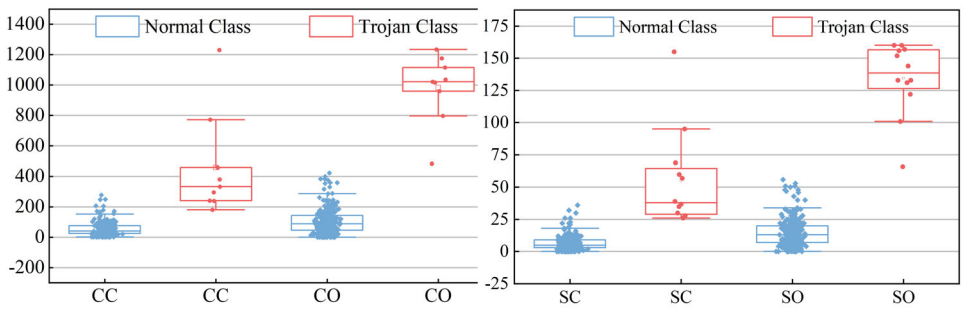

Fig. 6 The feature comparison of two features for RS232-T1000 (a) and RS232-T1100 (b)

(a) (b)

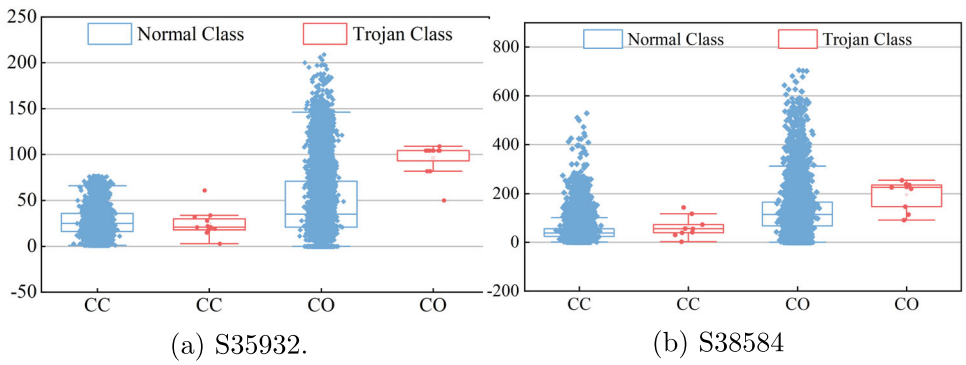

Fig. 7 The distribution of feature CC and CO of S35935 and S38584

netlist. All the Trojan nodes should be considered to avoid the catastrophic consequences caused by hardware Trojans. In this case, the shared nodes of Trojans and genuine circuits are labeled as Trojan nodes during the training phase. The predoction model may cuase a high number of false negatives, and some genuine nodes may be misclassified as Trojan nodes. There is no doubt that the increasing number of false alarms increases the workload of the manual check. In this paper, we sacrifice the verification costs to seek a high-level security of the proposed approach. The Trojan detection results of 23 Trojan circuits are shown in Fig. 9. Where the true positive (TP) and false negative (FN) show the number of Trojan samples identified to be Trojan samples and genuine samples respectively, whereas the true negative (TN) and the false positive (FP) imply the number of genuine samples identified to be genuine samples and Trojan samples respectively. In this case, we use precision $TPR=\frac{TP}{TP+FN}$, recall $TNR=\frac{TN}{TN+FP}$, and accuracy $ACC=\frac{TP+TN}{TP+FP+FN+TN}$ to evaluate the classification results. The circuit S15850, VGA-LCD, WB_CONMAX and EthernetMAC-10GE are denoted as S1, V, W and Ethernet respectively. As shown in Fig. 9, the majority of Trojan and genuine samples are correctly identified, and only a small number of samples, which is less than 10% of total samples, are wrongly classified. Among 23 Trojan circuits, the *TPR*, *TNR*, and *ACC* of the proposed approach reach 90.48%, 97.92%, and 97.92%. Therefore, we can confirm that the proposed approach can

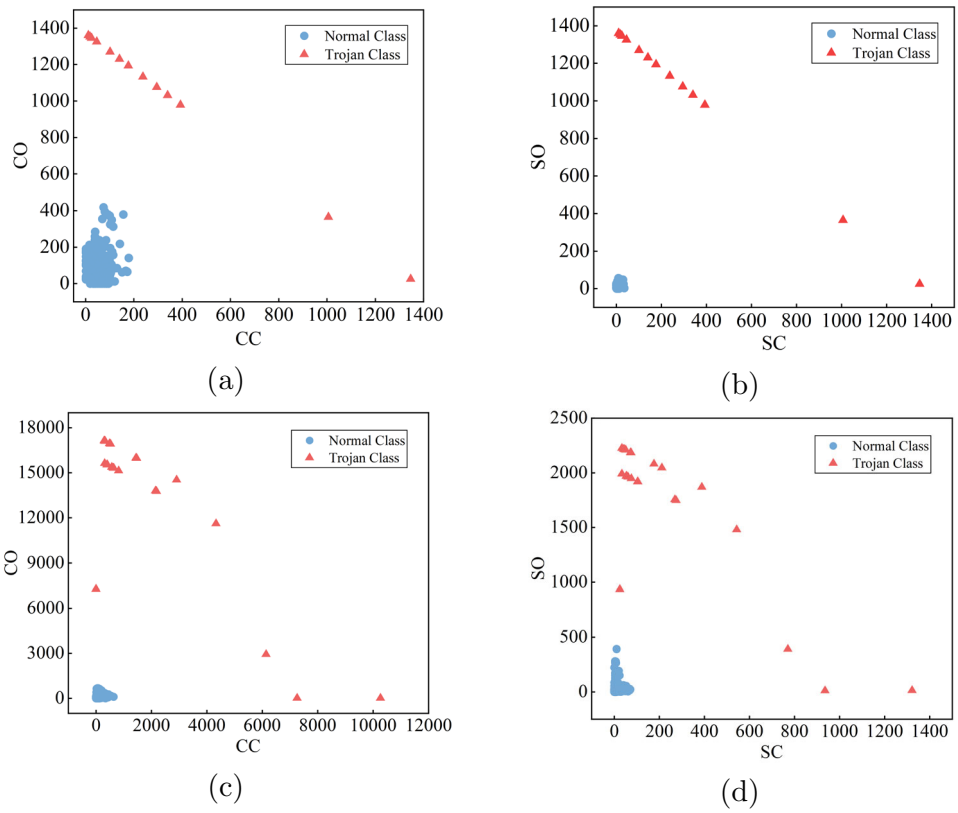

Fig. 8 Several feature combinations of RS232-T1200 and S15850. (a) The combination of CC and CO for RS232-T1200. (b) The combination of SC and SO for RS232-T1200. (c) The combination of CC and CO for S15850. (d) The combination of CC and CO for S15850

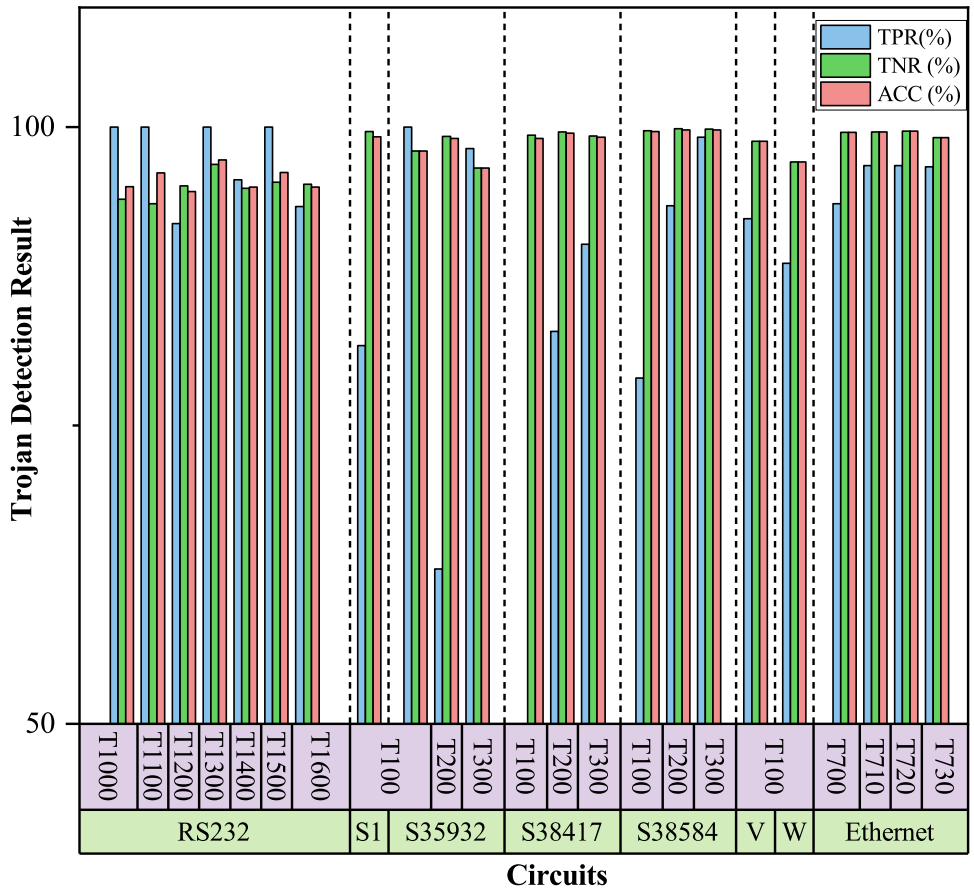

Fig. 9 Overall framework of hardware Trojan detection

identify all the Trojans only with a small number of false alarms.

Further, we compare the detection results of the proposed approach with the other popular methods (Ref. [39], [49], [22], [25], and [34]), and the comparison results are presented in Table 3. As shown in Table 3, the averaged *TPR* of the proposed approach is 90.48%, which is greater than other methods (90.48% vs. 68.32%, 82.59%, 88.56%, 82.03% and 42.42%). This implies that the proposed approach can identify more Trojan nodes than other methods. Compared with Ref. [39], [22], [49], and [34], the proposed approach achieves a similar level of performance with TNR=97.92%. However, Ref. [39] uses four testability metrics as the Trojan features, while Ref. [49], Ref. [34] and [49] use several structural features to detect several specified types of small-scale Trojans. For the [22, 39, 49] and [34], the scale of original circuits below 10K gates and the number of Trojan nodes is smaller than 57. The detection results of those approaches are not cover all the types of existing Trojans and large-scale original circuits. Moreover, the proposed approach outperforms the other three methods (Ref. [22], [25] and [34]) with *ACC*, which shows that the proposed approach distinguishes the Trojan nodes with a significantly smaller number of false alarms compared to the Ref. [22, 25] and [34].

Besides, several large circuits scale from 22K to 102K nodes (e.g. VGA_LCD-T100, WB_CONMAX-T100, EthernetMAC10GE-T700, and so on) are further used to evaluate the efficacy of the proposed approach. Regarding Fig. 9 and Table 3, the proposed approach can identify most of the Trojan nodes and only leave no more than four misclassified Trojan nodes. In addition, there are 2.08% of genuine nodes are misclassified as Trojan nodes, which further reduces the cost of manual review to check whether or not the suspicious nodes are Trojan. Therefore, we can conclude that the proposed approach scales well with large circuits. In summary, the proposed approach identifies the Trojan nodes from the gate-level netlist with a few false alarms that are much smaller than the existing methods.

6.2 Scalablity Analysis of Proposed Approach

The scalability of the proposed approach, including the netlist-level, feature-level, and algorithm-level, is analyzed. The scalability analysis is listed as follows.

Table 3 Comparison of Trojan detection results with the proposed approach and existing methods

Circuits	Ref. [22]			Ref. [25]			Ref. [39]			Ref. [49]			Ref. [34]			Ours		
	TPR	*TNR*	*ACC*	*TPR*	*TNR*	*ACC*	*TPR*	*TNR*	*ACC*	*TPR*	*TNR*	*ACC*	*TPR*	*TNR*	*ACC*	*TPR*	*TNR*	*ACC*
RS232-T1000	100%	98.9%	92.3%	53%	31%	37.75%	83%	100%	99.24%	84.09%	99.25%	97.11%	50%	96.43%	94.83%	100%	93.95%	94.98%
RS232-T1100	50%	98.2%	78.3%	58%	27%	28.2%	83%	100%	99.24%	80.95%	100%	97.44%	45.45%	96.43%	94.83%	100%	95.37%	96.14%
RS232-T1200	88.2%	100%	100%	80%	26%	27.62%	85%	100%	99.22%	78.79%	95.39%	93.65%	46.15%	97.14%	94.56%	91.89%	95.05%	94.59%
RS232-T1300	100%	100%	100%	89%	26%	27.69%	100%	100%	100%	87.1%	98.19%	97.07%	57.14%	96.04%	95.09%	100%	96.89%	97.24%
RS232-T1400	97.8%	100%	100%	83%	22%	24.12%	85%	100%	99.22%	86.96%	99.62%	97.75%	41.67%	96.40%	94.14%	95.56%	94.84%	94.96%
RS232-T1500	94.9%	99.6%	97.4%	83%	24%	26.11%	77%	100%	98.84%	93.33%	96.48%	96.18%	45.45%	96.45%	94.54%	100%	95.35%	96.17%
RS232-T1600	93.1%	99%	90%	89%	26%	27.69%	100%	100%	100%	80%	95.74%	94.46%	44.44%	96.11%	94.52%	93.33%	95.18%	94.96%
S15850-T100	77.8%	100%	95.5%	93%	66%	66.29%	65%	100%	99.43%	–	–	–	73.08%	95.81%	95.54%	81.67%	99.62%	99.18%
S35932-T100	73.3%	100%	100%	93%	60%	60.08%	–	–	–	80%	100%	99.95%	7.14%	99.11%	99.13%	100%	97.98%	97.99%
S35932-T200	8.3%	100%	100%	100%	59%	59.1%	92%	100%	99.98%	83.33%	100%	99.97%	33.33%	99.90%	99.76%	62.96%	99.19%	99.03%
S35932-T300	81.1%	100%	96.8%	27%	58%	57.82%	–	–	–	30.56%	100%	99.61%	–	–	–	98.18%	96.55%	96.57%
S38417-T100	33.3%	100%	100%	100%	76%	76.04%	92%	100%	99.98%	91.67%	99.95%	99.93%	–	–	–	41.38%	99.31%	99.02%
S38417-T200	46.7%	100%	100%	73%	76%	75.99%	55%	100%	99.85%	80%	99.95%	99.89%	–	–	–	82.86%	99.57%	99.47%
S38417-T300	75%	100%	100%	100%	72%	75.63%	–	–	–	100%	99.95%	99.95%	22.73%	99.96%	99.37%	90.16%	99.22%	99.13%
S38584-T100	5.3%	100%	33.3%	100%	62%	62.05%	100%	100%	100%	73.68%	99.99%	99.92%	–	–	–	78.95%	99.67%	99.62%
S38584-T200	–	–	–	94%	64%	64.79%	–	–	–	–	–	–	–	–	–	93.38%	99.85%	99.73%
S38584-T300	–	–	–	89%	66%	70.43%	–	–	–	–	–	–	–	–	–	99.13%	99.82%	99.73%
V-T100	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	99.31%	99.82%	99.73%
W-T100	–	–	–	–	–	–	–	–	–	100%	100%	100%	–	–	–	88.57%	97.07%	97.06%
Ethernet-T700	–	–	–	–	–	–	100%	100%	100%	–	–	–	–	–	–	93.55%	99.54%	99.54%
Ethernet-T710	–	–	–	–	–	–	100%	100%	100%	–	–	–	–	–	–	96.77%	99.56%	99.56%
Ethernet-T720	–	–	–	–	–	–	100%	100%	100%	–	–	–	–	–	–	96.77%	99.66%	99.66%
Ethernet-T730	–	–	–	–	–	–	100%	100%	100%	–	–	–	–	–	–	96.67%	99.10%	99.10%
Average	68.32%	99.71%	92.24%	82.59%	49.47%	50.84%	88.56%	100%	99.69%	82.03%	98.97%	98.19%	42.42%	97.25%	96.00%	90.48%	97.92%	97.92%

- For the design-level scalability, the proposed approach can also be scalable to evaluate the trustworthiness of register transfer level design. Before the validation, the register transfer level design should be synthesized into the gate-level netlist.
- For the feature-level scalability, the proposed approach is scalable to emerging Trojans. The Trojan features shown in Table 1 can cover all the types of existing Trojans, and the emerging Trojans may include one or more of those features. In this paper, a feature-adaptive classifier with multiple classifiers can detect any feature combinations of Trojans. Thus, we can conclude that the proposed approach may be scalable for detecting emerging Trojans.
- For the algorithm-level scalability, the other machine learning algorithms can be expanded to identify the features of hardware Trojans. The Trojan detection problem is formulated as a feature outliers identification problem. In this paper, we use the random forest to identify the abnormal features caused by the Trojan implantation. Similarly, the existing machine learning algorithms can be used to identify the minor feature differences between the original nodes and Trojan nodes. The proposed hardware Trojan framework can also be updated with the latest machine learning algorithms.

7 Conclusion

In this paper, we present a hardware Trojan detection framework to identify the Trojan nodes from the gate-level netlist. A multi-level features extraction approach is applied to extract 13 features from the gate-level netlist, and a feature rebalancing method is introduced to build the Trojan feature database from the Trojans in the benchmark. Further, a feature-adaptive Trojan identification method based on the random forest is introduced to identify the Trojan nodes from the gate-level netlist. 23 gate-level Trojan circuits are used to evaluate the efficacy of the proposed approach. Experimental results prove that the proposed approach outperforms the other methods with comparable detection accuracy. In the future, more Trojan features will be explored to further cover other types of emerging Trojans, including analog Trojans, mixed-type Trojans, parametric Trojans, and so on. Moreover, larger designs are also required to validate the limitation of the proposed approach, and new types of Trojans are designed to evaluate the scalability of the proposed approach.

Funding This work was supported by the National Natural Science Foundation of China under Grant 62302519. This work has no financial and personal relationships with other people or organizations that can inappropriately influence our work. There is no professional or other personal interest of any nature or kind in any product, service and/or company.

References

1. Bao C, Forte D, Srivastava A (2015) Temperature tracking: Toward robust run-time detection of hardware trojans. IEEE Trans Comput Aided Des Integr Circuits Syst 34(10):1577–1585
2. Lok-Won K, Villasenor JD (2015) Dynamic function verification for system on chip security against hardware-based attacks. Reliability, IEEE Transactions on 64(4):1229–1242
3. Ma H, He J, Liu Y, Kuai J, Zhao Y (2020) On-chip trust evaluation utilizing tdc-based parameter-adjustable security primitive. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems PP(99):1–1
4. Rajendran S, Regeena ML (2022) A novel algorithm for hardware trojan detection through reverse engineering. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 41
5. Bao C, Forte D, Srivastava A (2015) On reverse engineering-based hardware trojan detection. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 1–1
6. Reece T, Robinson WH (2016) Detection of hardware trojans in third-party intellectual property using untrusted modules. IEEE Trans Comput Aided Des Integr Circuits Syst 35(3):357–366
7. Liu Y, He J, Ma H, Zhao Y (2020) Golden chip free trojan detection leveraging probabilistic neural network with genetic algorithm applied in the training phase. Science China(Information Sciences) 63(02):242–244
8. Liu Y, He J, Ma H, Zhao Y (2019) Hardware trojan detection leveraging a novel golden layout model towards practical applications. J Electron Test 35(11)
9. He J, Ma H, Liu Y, Zhao Y (2020) Golden chip-free trojan detection leveraging trojan trigger's side-channel fingerprinting. ACM Transactions on Embedded Computing Systems (TECS)
10. Hafeez AI, Faiq K, Osman H, Mehmood KA, Muhammad S (2018) Mcsevic: a model checking based framework for security vulnerability analysis of integrated circuits. IEEE Access 6:1–1
11. Veeranna N, Schafer BC (2017) Hardware trojan detection in behavioral intellectual properties (ip's) using property checking techniques. IEEE Trans Emerg Top Comput 5(4):576–585
12. Guo X, Dutta RG, Mishra P, Jin Y (2017) Automatic code converter enhanced pch framework for soc trust verification. IEEE Trans Very Large Scale Integr (VLSI) Syst 25(12):3390–3400
13. Chen X, Liu Q, Yao S, Wang J, Xu Q, Wang Y, Liu Y, Yang H (2017) Hardware trojan detection in third-party digital intellectual property cores by multilevel feature analysis. IEEE Trans Comput Aided Des Integr Circ Syst
14. Hicks M, Finnicum M, King ST, Martin M, Smith JM (2010) Overcoming and untrusted computing base: Detecting and removing malicious hardware automatically. Proc. of IEEE Symposium on Security and Privacy
15. Zhang J, Yuan F, Wei L, Sun Z, Xu Q (2013) Veritrust: verification for hardware trust. In: Proc. of Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE
16. Feng Y, Jie Z, Qiang X, Yannan L, Linxiao W (2015) Veritrust: verification for hardware trust. IEEE Transactions on computer aided design of integrated circuits & system
17. Adam W, Matthew S, Simha S (2013) Fanci: Identification of stealthy malicious logic using boolean functional analysis. In: Proc. of Acm sigsac conference on computer & communications security
18. Dean S, Jeff B, Guidong Z, Shaojie Z, Yier J (2014) Fight-metric: functional identification of gate-level hardware trustworthiness. In: Proc. of design automation Conference
19. Song Y, Xiaoming C, Jie Z, Qiaoyi L, Huazhong Y (2015) Fastrust: Feature analysis for third-party ip trust verification. In: Proc. of IEEE International test Conference
20. Jie Z, Feng Y, Qiang X (2014) Detrust: defeating hardware trust verification with stealthy implicitly-triggered hardware trojans. In:

Proc. of ACM conference on computer and communications Security
21. Hassan S (2017) Cotd: reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist. IEEE Trans Inf Forensics Secur 12(2):338–350
22. Kento H, Masao Y, Nozomu T (2017) Trojan-feature extraction at gate-level netlists and its application to hardware-trojan detection using random forest classifier. In: Proc. of 2017 IEEE International Symposium on Circuits and Systems (ISCAS)
23. Masaru O, Youhua S, Masao Y, Nozomu T (2015) A score-based classification method for identifying hardware-trojans at gate-level netlists. In: Proc. of design, automation & test in Europe conference & exhibition
24. Masaru O, Noritaka Y, Toshihiko O, Yukiyasu T, Masao Y, Nozomu T (2016) Hardware-trojans rank: Quantitative evaluation of security threats at gate-level netlists by pattern matching. IEICE Trans Fundam Electron Commun Comput Sci 99(12):2335–2347
25. Kento H, Masaru O, Masao Y, Nozomu T (2016) Hardware trojans classification for gate-level netlists based on machine learning. In: Proc. of 2016 IEEE 22nd International symposium on on-line testing and robust system design (IOLTS)
26. Fuqiang C, Qiang L (2017) Single-triggered hardware trojan identification based on gate-level circuit structural characteristics. In: IEEE International symposium on circuits & systems, pp 1–4
27. Shichao Y, Chongyan G, Weiqiang L, Maire O (2020) A novel feature extraction strategy for hardware trojan detection. In: Proc. of international symposium on circuits and systems
28. Lang L, Markus K, Tim G, Christof P, Wayne B (2009) Trojan side-channels: Lightweight hardware trojans through side-channel engineering. In: Proc. of International workshop on cryptographic hardware and embedded systems
29. Jayesh P, Usha M (2017) Transition probabilistic approach for detection and diagnosis of hardware trojan in combinational circuits. In: Proc. of India Conference
30. Xin X, Yangyang S, Hongda C, Yong D (2017) Hardware trojans classification based on controllability and observability in gate-level netlist. IEICE Electronics Express 14(18):20170682–20170682
31. Xiaoming C, Qiaoyi L, Song Y, Jia W, Qiang X, Yu W, Yongpan L, Huazhong Y (2017) Hardware trojan detection in third-party digital intellectual property cores by multilevel feature analysis. IEEE transactions on computer-aided design of integrated circuits and systems
32. Qiang L, Pengyong Z, Fuqiang C (2019) A hardware trojan detection method based on structural features of trojan and host circuits. IEEE Access
33. Kai H, Yun H (2020) Trigger identification using difference-amplified controllability and dynamic transition probability for hardware trojan detection. IEEE Trans Inf Forensics Secur 15:3387–3400
34. Chen D, Yulin L, Jinghui C, Ximeng L, Yuzhong C (2020) An unsupervised detection approach for hardware trojans. IEEE Access PP(99):1–1
35. Jayesh P, Usha M (2017) Transition probabilistic approach for detection and diagnosis of hardware trojan in combinational circuits. In: Proc. of India Conference
36. Minhui Z, Xiaotong C, Liang S, Kaijie W (2017) Potential trigger detection for hardware trojans. IEEE Trans Comput Aided Des Integr Circ Syst 1–1
37. Sayandeep S, Subhra CR, Debdeep M (2016) Testability based metric for hardware trojan vulnerability assessment. In: Digital system Design, pp 503–510
38. Priyadharshini M, Saravanan P (2020) An efficient hardware trojan detection approach adopting testability based features. Proc. of IEEE International test conference India
39. Mahshid T, Azadeh M, Alireza S (2021) Sc-cotd: hardware trojan detection based on sequential/combinational testability features using ensemble classifier. J Electron Test 37(4):473–487
40. Yu S, Haihua S, Renjie L, Yunying Y (2021) A stealthy hardware trojan design and corresponding detection method. In: Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)
41. Ning Z, Zhiqiang L, Yanlin Z, Haiyang L, Weiqing H (2020) Novel design of hardware trojan: A generic approach for defeating testability based detection. In: Proc. of IEEE 19th International conference on trust, security and privacy in computing and communications (TrustCom)
42. Dakshi A, Selcuk B, Deniz K, Pankaj R, Berk S (2008) Trojan detection using ic fingerprinting. In: Proc. of IEEE Symposium on Security & Privacy
43. benchmarks T-h (2024) https://www.trust-hub.org
44. Huaikuan Y, Qingchao J, Xuefeng Y, Bei W (2021) Imbalanced classification based on minority clustering synthetic minority oversampling technique with wind turbine fault detection application. IEEE transactions on industrial informatics (17-9)
45. Ning Q, Zhao X, Ma Z (2021) A novel method for identification of glutarylation sites combining borderline-smote with tomek links technique in imbalanced data. IEEE/ACM Trans Comput Biol Bioinf 19:2632–2641
46. Ery KY, Erna PA, Silmi F (2018) Adaptive synthetic-nominal (adasyn-n) and adaptive synthetic-knn (adasyn-knn) for multiclass imbalance learning on laboratory test data. In: Proc. of 2018 4th International Conference on Science and Technology (ICST), pp 1–6. https://doi.org/10.1109/ICSTC.2018.8528679
47. Luqyana WA, Ahmadie BL, Supianto AA (2019) K-nearest neighbors undersampling as balancing data for cyber troll detection. Proc. of 2019 International Conference on Sustainable Information Engineering and Technology (SIET) 322–325
48. Zhang M, Daoxiong G (2022) Combining clustering undersample and ensemble learning for wearable fall detection. In: Proc. of 2022 IEEE International Conference on Real-time Computing and Robotics (RCAR), pp 545–550. https://doi.org/10.1109/RCAR54675.2022.9872285
49. Hoo KC, Yee OC, Mehrdad M, Nordinah I, Sim CH, Michiko I (2019) Classification of trojan nets based on scoap values using supervised learning. In: Proc. of 2019 IEEE International Symposium on Circuits and Systems (ISCAS), pp 1–5. https://doi.org/10.1109/ISCAS.2019.8702462

Yanjiang Liu Liu received the B.S. degree from the Zhoukou Normal University, Zhoukou in 2013, M.S. degree from the Guangdong University of Technology in 2016, and Ph.D. degree from the Tianjin University in 2020, and engaged in postdoctoral research with the Information Engineering University from 2020 to 2023. He is now a lecturer at the Information Engineering University. His current research interests include hardware Trojan detection, side-channel attack and protection technique, physical unclonable function design, secure digital circuit design, and EDA for security.

Junwei Li He is currently pursuing the Ph.D. degree at the Information Engineering University. His research interests include Embedded system security and Trusted SoC Design.

Pengfei Guo is a lecturer at the Information Engineering University. His current research interests include EDA for security, cache attack and secure digital circuit design.

Chunsheng Zhu is a lecturer at the Information Engineering University. His current research interests include advanced packaging and IC design.

Junjie Wang is a lecturer at the Information Engineering University. His current research interests include EDA for security, random number generator circuit design and secure digital circuit design.

Jingxin Zhong received the Ph.D degree from Air Force Engineering University in 2018. He is now a lecturer at the Information Engineering University. His current research interests include trusted circuit design and IC security.

Lichao Zhang is an associate professor at the Information Engineering University. His current research interests include trusted circuit design and IC security.