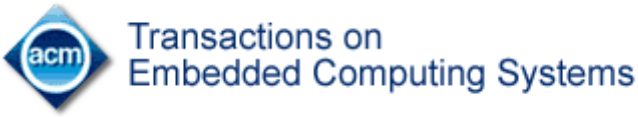

A Scalable Hardware Trojan Detection Framework Utilizing Multi-level Features and Random Forest

Journal:	*Transactions on Embedded Computing Systems*
Manuscript ID	TECS-2022-0212
Manuscript Type:	Paper for Regular Issues
Date Submitted by the Author:	30-Sep-2022
Complete List of Authors:	Liu, Yanjiang; Information Engineering University, School of Cryptographic Engineering Dai, Zibin; Information Engineering University, School of Cryptographic Engineering Guo, Pengfei; Information Engineering University, School of Cryptographic Engineering Li, Junwei; Information Engineering University, School of Cryptographic Engineering Yan, Yingjian; Information Engineering University, School of Cryptographic Engineering
Topic - A primary topic is required. You may also select a secondary topic.:	19. Secure, Reliable, and Trustworthy Embedded Systems, 6. Computer Aided Design Tools for Embedded Systems
Computing Classification Systems:	

A Scalable Hardware Trojan Detection Framework Utilizing Multi-level Features and Random Forest

YANJIANG LIU, ZIBIN DAI, PENGFEI GUO, JUNWEI LI, and YINGJIAN YAN, Information Engineering University, China

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 high-efficient hardware Trojan detection framework for gate-level netlist 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 are analyzed and 13 common features are introduced as the Trojan features. Furthermore, a feature extraction method is presented to extract the features of the gate-level netlist and form the netlist's feature set, and a feature rebalancing method is proposed to balance the distribution of Trojan-free and Trojan class and establish the Trojan's feature database. The Trojan detection is formulated as the anomalous feature identification problem, and a feature-adaptive Trojan detection method based on the random forest is proposed to detect the Trojan features and identify more types 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 being scalable to detect emerging Trojans by adding new features to the Trojan feature database.

CCS Concepts: • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

Additional Key Words and Phrases: Integrated circuit, hardware Trojan, multi-level feature analysis, imbalanced data, random forest.

ACM Reference Format:
Yanjiang Liu, Zibin Dai, Pengfei Guo, Junwei Li, and Yingjian Yan. 2018. A Scalable Hardware Trojan Detection Framework Utilizing Multi-level Features and Random Forest. 1, 1 (September 2018), 19 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With an ever-developing of integrated circuit design technology, modern designs are equipped with several memory modules, various interfaces, and numerous processors or co-processors to seek high performance, which deploys in the critical applications and sensitive fields. The high complexity of design, the urgent time-to-market, and the limited developing cost motivate the designers to reuse numerous IP cores during the design stage that are provided by third-party outsourced vendors. For the third-party IP cores, adversaries can easily modify the original design and insert the hardware Trojans into it. Fig. 1 shows the design stage where the hardware Trojan can be inserted by adversaries.

Authors' address: Yanjiang Liu, liuyj_1013@126.com; Zibin Dai; Pengfei Guo; Junwei Li; Yingjian Yan, Information Engineering University, Shangcheng Road No. 12, Zhengzhou, Henan Province, China, 450004.

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 total system fails to work. Consequently, the trustworthiness of third-party IP cores is challenged by hardware Trojan which has become a serious problem for the integrated circuits.

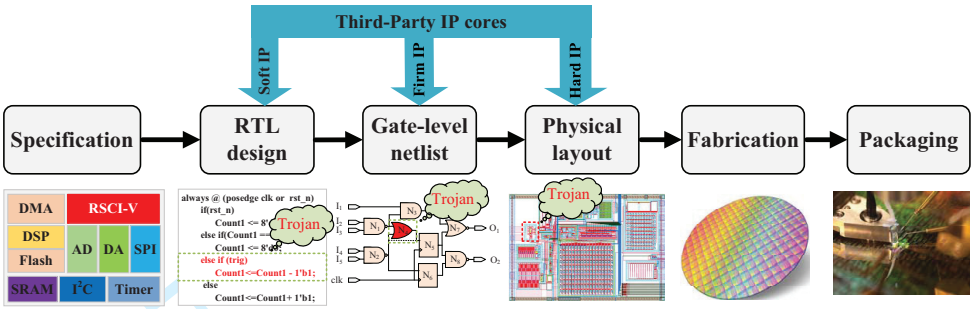

FIGURE 1. Overview of the IC design flow and its security risk caused by hardware Trojan.

Recently, various hardware Trojan detection techniques have 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], functional test [6], and side-channel analysis [7–9], validate the trustworthiness of fabricated chips. However, once the hardware Trojan is identified 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 been introduced. Specifically, various structural Trojan features [22–27], functional Trojan features [28–30] and multi-level 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 these features are limited to several types of hardware Trojans In the real case, an imbalanced data problem exists wherein the samples belonging to Trojan-free class is heavily outnumber the Trojan class, and the conventional supervised classifiers are biased towards the Trojan-free 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 nets from gate-level netlist. We analyze the structural and stealthy characteristics of several gate-level Trojans and introduce 13 Trojan

features that can cover all the types of 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 oversampling technique (SMOTE) and K-nearest neighborhood (KNN) are combined to generate synthetic data of the Trojan class efficiently 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, which is sufficient to learn all the Trojan features. Experimental results show that the proposed framework scales well with the large circuits and various types of Trojans. The main contributions are listed as follows.

- 13 features, including the structural and functional features, are proposed as the Trojan features set, which can cover all the types of existing Trojans in benchmarks. The features of the emerging Trojan are also added to the feature set, which makes the proposed approach scalable to emerging Trojan.
- A multi-level feature extraction method is introduced to extract the Trojan features from the Trojan benchmark site automatically, and a data rebalancing method is proposed to generate a new synthetic dataset for the Trojan class and emend the distribution of the Trojan feature database. The classification model 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 identification method based on the random forest classifier is proposed, which establishes multiple decision trees with the feature combinations and determines the label of samples with the maximum voting results of decision trees. The proposed approach covers all the types of existing Trojans and has successfully 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 proposed framework can be easily integrated into the integrated circuit design process and used to assess the trustworthiness of the gate-level netlist before fabrication.

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, aimed at extracting and identifying the specific features of Trojan. Next, we present the state-of-the-art feature analysis approaches for hardware Trojan.

1) structural feature analysis: Hicks et al. propose a MUX-based Trojan that can evade the code coverage test, and then 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 FASTrust, 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 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], authors propose nine structural Trojan features and detect the Trojan net when the sum of these features exceed 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 the Trojan features and exploit the local outlier factor algorithm to identify the anomalous features [35].

2) functional feature analysis: In [17], functional analysis, namely FANCI, is proposed to identify the nets with low control values as hardware Trojan. The low transition probability is considered a Trojan feature [36]. However, Ref. [17] and [36] only aim at the combinational logic. After that, Sullivan et al. calculate the control value of sequential logics 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 nets with low state and switching probability [37]. Further, Both the Ref. [38], [21], [39] and [40] regard the nets with low controllability and observability value as the best candidate for the insertion sites of Trojan [21, 38–40]. Specifically, Sayandeep et al. detect the small pairwise correlation value of low controllability nets [38], 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 [40]. However, a hardware Trojan design method presented in [41, 42] can disable the Trojan detection based on testability metrics. Further, the testability metric and dynamic transition analysis are combined to identify the inactive nets of the netlist and locate the Trojan's trigger circuit [33].
3) multi-level feature analysis: Structural feature analysis usually remarks numerous Trojan-free nets 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 functional features are combined to extend the scalability to different types of Trojan. 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, little attention had paid to the imbalanced data problem and the classification model bias toward the majority class. Therefore, several Trojan-free nets are classified as Trojan nets, and functional test or formal verification is needed to validate the trustworthiness of suspicious nets.

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 the 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 features 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 [34], scholars have designed various Trojans with different types, sizes, and functions over the past decade. As mentioned above, numerous hardware Trojan detection approaches, including formal verification, logic test, 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 functional features have proposed over the past decades. More specifically, several structural features, including the unused circuit [14], redundant inputs [15], trigger structure, and typical structures that extracted from benchmarks [22, 23, 25], are used as the structural features of Trojan, while the control value [17], transition probability [36, 37], testability metrics [21, 30, 33, 38–40] are adopted as functional 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 good 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 nets is much smaller than Trojan-free nets and the Trojan-free feature set outnumbers the Trojan feature set. Based on these imbalanced feature sets, the training model using traditional machine learning algorithms biases toward the Trojan-free class, and several Trojan nets are misclassified as Trojan-free nets 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, 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 are not mention the other types of hardware Trojans. Consequently, multi-level feature analysis methods have proposed to strengthen the current feature analysis approaches, which combine structural features and functional features to cover more types of Trojans. However, Ref. [13] exploits four structural and a functional 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. Actually, 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. To be specific, 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 Trojan-free 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 nets 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 MULTI-LEVEL 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 single-logic triggered Trojan. Several works have regarded the high observability signals as the Trojan nets[21, 30, 37, 40]. 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, 40]. The feature of single-logic triggered Trojan can be summarized as follows.

Lemma 4.1. *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

For the multiple-logic triggered Trojan, it is triggered when all the trigger inputs meet the predefined values. Due to its stealthy nature, adversaries typically select the high observability signals as the trigger inputs. Purely using feature 4.1 can detect this Trojan, but some rare nets may be misjudged as Trojan. Besides, the number of trigger inputs should be large enough to obtain a low activation probability. We use the In-degree to show the number of trigger inputs. The larger the In-degree is, the greater the number of trigger inputs is. Therefore, we use feature 4.1 and In-degree value to detect the multiple-logic triggered Trojans. This observation can be summarized as the feature 4.2.

Lemma 4.2. *All the trigger inputs of multiple-logic triggered Trojans have a high controllability value and the trigger part has a large In-degree 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 clcok 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 4.3. *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 net. The smaller the LPI value is, the higher the correlation between the input and net is. For the latter, such trigger part has a large number of multiplexers. We use the number of multiplexers (MUX) for the path where the internal signal located as the Trojan feature. The feature of pattern-triggered Trojan is presented as feature 4.4.

Lemma 4.4. *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 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 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 feature 4.5.

Lemma 4.5. *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 Trojan-free 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 is not connected 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 structure has a large number of flip-flops, thus, the payload signal has a large FF_IN and FF_OUT value. We can detect this type of Trojan using the feature 4.3. For the third case, Trojans exploit a large bit-width shift register or ring oscillator with an odd number as the leakage circuit. Feature 4.3 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 4.6. *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 main frequency by extending the critical path (e.g. S35932-T300). For the former case, the Trojan payload continuously rotates after the Trojan is activated. We can detect this hardware Trojan using the feature 4.3. For the latter, the Trojan payload is a ring oscillator along the critical path, which slows down this path when the ring oscillator oscillates. This case can also be detected by employing the INV value of feature 4.6. Therefore, combining the feature 4.3 and 4.6 can detect the performance-degradation Trojans.

Lemma 4.7. *The performance-degradation Trojans have a large FF_IN and FF_OUT value, or large INV value.*

In this paper, we propose seven structural features to detect the above types of Trojans in the Trust-Hub benchmark. However, 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 feature 4.3. Moreover, denial-of-service Trojans (e.g. AES-T1800, RS232-T500, S38417-T200, and VGA_LCD-T100) are identified with several features. Thus, we can lower the number of suspicious signals using multiple features. To be specific, AES-T1800 is triggered when observing a predefined input plaintext and rotates the shift register, which can be detected using the features 4.3 and 4.4. RS232-T500 is triggered when the 32-bit counter reaches the maximum value and keeps an output signal stuck at 0. We use the feature 4.3 to identify the 32-bit counter of the trigger part and exploit the feature 4.6 to locate the payload part. Therefore, we can conclude that the proposed Trojan features not only identify existing Trojans in the Trust-hub benchmarks, but also be scalable to defend against unknown Trojans.

5 HARDWARE TROJAN DETECTION FRAMEWORK UTILIZING MULTI-LEVEL FEATURE ANALYSIS AND RANDOM FOREST

5.1 Overall framework

The overall framework of Trojan detection 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, functional and structural features of each net are extracted from the gate-level netlist and the results form the feature set. 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 net, and structural feature analysis is conducted to calculate the structural feature value (In-degree, 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 and resampled 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 nets are found by identifying the feature outliers.

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

As described in Section 4, signals with high controllability and observability value are the ideal implantation site of hardware Trojan. In this paper, we use testability metrics as the Trojan functional features. The testability calculation

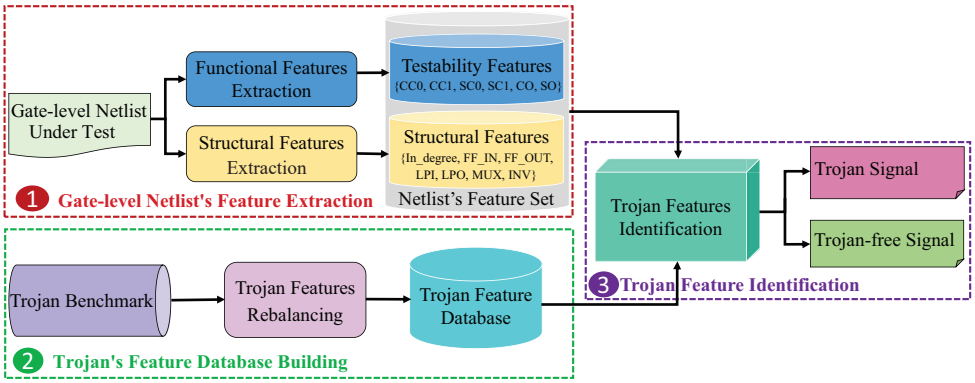

FIGURE 2. Overall framework of hardware Trojan detection.

method is provided in [21, 40]. For the sake of simplicity, we implement a testability measurement tool to calculate the testability features of each net in a circuit.

Further, we build the directed graph of a gate-level netlist and extract the proposed structural features of each net. Fig. 3 (a) shows an example circuit. Where I_1, I_2, $\cdots$, I_5 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 nets. In the directed graph, the inputs, internal nets 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 In-degree value of e_4 is 4. Regarding Fig. 3 (a), the path from the inputs to the internal net 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 propopagates 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 located has no inverters and multiplexers, thus, the INV and MUX are equal to 0.

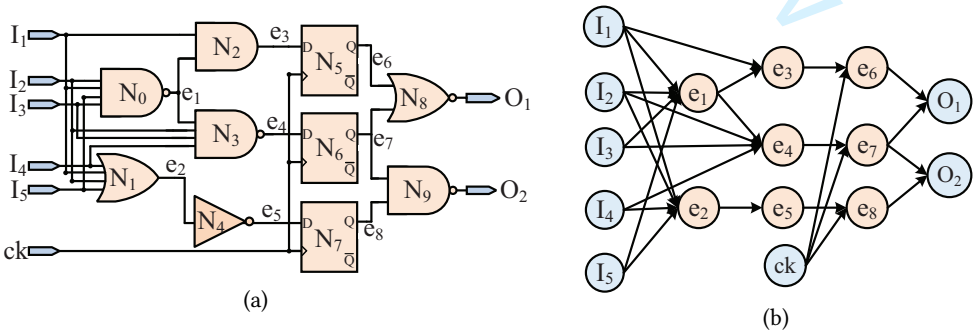

FIGURE 3. The gate-level schematic of a circuit (a) and corresponding directed graph (b).

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 the feature 4.2, the In-degree 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 Equation 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 rusults 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 Equation 2. Where the H_{w_i} is the next-level adjacent vertex set of w_i.

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

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 the $\mathbf{V}$, the searching process is completed, otherwise, an unmarked vertex is chosen from the graph as the starting point for the next searching. 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-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 unempty set, it indicates the path $\mathbf{p}_i$ exist 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 using data rebalancing algorithm

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 Trojan-free vertexes are labeled as Trojan and Trojan-free class, respectively. As shown in the Trojan benchmark, the size of the Trojan circuit is much smaller than the original design and the number of Trojan-free nodes is significantly greater than Trojan nodes. In this case, the distribution of Trojan-free nodes and Trojan nodes is unbalanced. The imbalanced ratio, the ratio between the number of the Trojan nodes and Trojan-free 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 Trojan-free class and

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: In-degree(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: $\mathbf{P}$={$\mathbf{p}_1$, $\mathbf{p}_2$, $\cdots$, $\mathbf{p}_n$} ← *path_search*($\mathbf{G}$, v_i); ▷ search all the paths where the v_i locates.
2: *In-degree*(v_i) ← *Indegree*($\mathbf{G}$, v_i); ▷ calculate the value of feature In-degree of v_i.
3: **for** $\mathbf{p}_i \in \mathbf{P}$ **do**
4: **lpio_min**(i,1:2) ← *Inout_Num*($\mathbf{p}_i$, v_i); ▷ determine the minimum number of inputs and outputs to the v_i.
5: **mux_max**(i) ← *Mux_Num*($\mathbf{p}_i$, v_i); ▷ determine the maximum number of multiplexers along the v_i.
6: $\mathbf{K}$={$\mathbf{k}_1$, $\mathbf{k}_2$, $\cdots$, $\mathbf{k}_m$} ← *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) ← *FF_Num*($\mathbf{k}_j$); ▷ count the maximum number of flip-flops from the input and output side.
10: **inv_loop**(i, j) ← *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) ← 0;
14: **end if**
15: **end for**
16: FF_IN(v_i), FF_OUT(v_i) ← *max*(**ffinout_loop**); INV(v_i) ← *max*(**inv_loop**); MUX(v_i) ← *max*(**mux_max**);
17: LPI(v_i), LPO(v_i) ← *min*(**lpio_num**); ▷ output the feature value of v_i.

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 the oversampling, undersampling and hybrid sampling, have explored to slove the imbalanced data problem. The oversampling methods, including the synthetic minority oversampling technique (SMOTE) [44], borderline-SMOTE [45], and adaptive synthetic sampling technique (ADASYN) [46], add new synthetic samples into the minority class, while the undersampling 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 oversampling and undersampling methods to eliminate the effect of overfitting and loss of essential information, which have shown as a promising approach in recent years. In this paper, we combine SMOTE and KNN, namely SMOTEKNN, to generate the synthetic samples of the minority class using the SMOTE oversampling algorithm and remove the oversampled synthetic samples using the KNN undersampling algorithm.

Let the Trojan-free samples Ψ={ψ_1, ψ_2, $\cdots$, ψ_n} and Trojan samples Λ={λ_1, λ_2, $\cdots$, λ_m} explain the feature rebalancing process. Where ψ_i and λ_j are the i-th Trojan-free sample and j-th Trojan sample respectively, and $n \gg m$. The k nearest neighbor samples Γ={γ_1, γ_1, $\cdots$, γ_k} of λ_j are selected from the Λ, which is expressed as Equation 3. Where f is the Euclidean distance calculation function and Γ is the subset of Λ.

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

The synthetic sample ξ_{ji} is the linear interpolation between the λ_j and γ_i, which is expressed in Equation 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 the Equation 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 Trojan-free 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 Trojan-free class and the Trojan class. To address this issue, we use the KNN to remove the synthetic samples that is close to the Trojan-free 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 Trojan-free 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 Equation 5.

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

5.4 Feature-adpative Trojan identification method based on random forest

In the Trust-Hub benchmark site, Trojans have more than one feature and the features differ from each other. For example, the trigger net of a single-logic triggered Trojan has a large CC0, CC1, SC0, and SC1 value, while a Trojan payload net only has a large CO and SO value. Considering the various types of Trojans, there might be considerable differences in several features between the Trojan-free nets and Trojan nets, while the other features of Trojan-free nets are similar to the Trojan nets. To ensure a high detection accuracy, all the features should be considered to build a feature-adaptive classifier that can cover more types of Trojan.

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 the voting as the decision-making rule, which achieves good accuracy even if a large proportion of the critical features of the dataset are missing. Inspired by this, random forest is adopted to detect different features of Trojan nets in a netlist. 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 Trojan-free 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 those features.

The Trojan feature identification process is shown in Fig. 4. The balanced Trojan feature database $\mathcal{F}$=$\{f_1, f_2, \cdots, f_n\}$ has n samples, and all the samples are marked as two classes: 1 (Trojan class) and 0 (Trojan-free class). f_i is a 13×1 vector and each row of f_i represents the corresponding feature value. 70% samples of $\mathcal{F}$ form the training data $\mathcal{K}$ and the remaining 30% samples set as the testing data $\mathcal{V}$. 13 features of netlist under test are obtained using the feature extraction method, and the results are the test data $\mathcal{Z}$=$\{\mathfrak{z}_1, \mathfrak{z}_2, \cdots, \mathfrak{z}_k\}$ and k is the number of vertexes of the directed graph of netlist. We select m samples from the $\mathcal{K}$ randomly as the randomized data $\mathcal{D}$=$\{\mathfrak{d}_1, \mathfrak{d}_2, \cdots, \mathfrak{d}_m\}$, and choose these samples with 1, 2, 3, $\cdots$, and 13 features respectively to train the decision trees. According to the permutation and combination theory, there are C_{13}^i feature combinations when we select i features from 13 features. C_{13}^i decision trees are built and the decision tree Υ_{ij} can be expressed in Equation 6. Where i is the i-th feature combination that includes i features, $\mathcal{D}t$ is the decision tree function, j is the number of the i-th feature combination and j scales from 1 to C_{13}^i.

$$\Upsilon_{ij} \leftarrow \mathcal{D}t(\mathcal{K}), i = 1, 2, \cdots, 13; j = 1, 2, \cdots, C_{13}^i \tag{6}$$

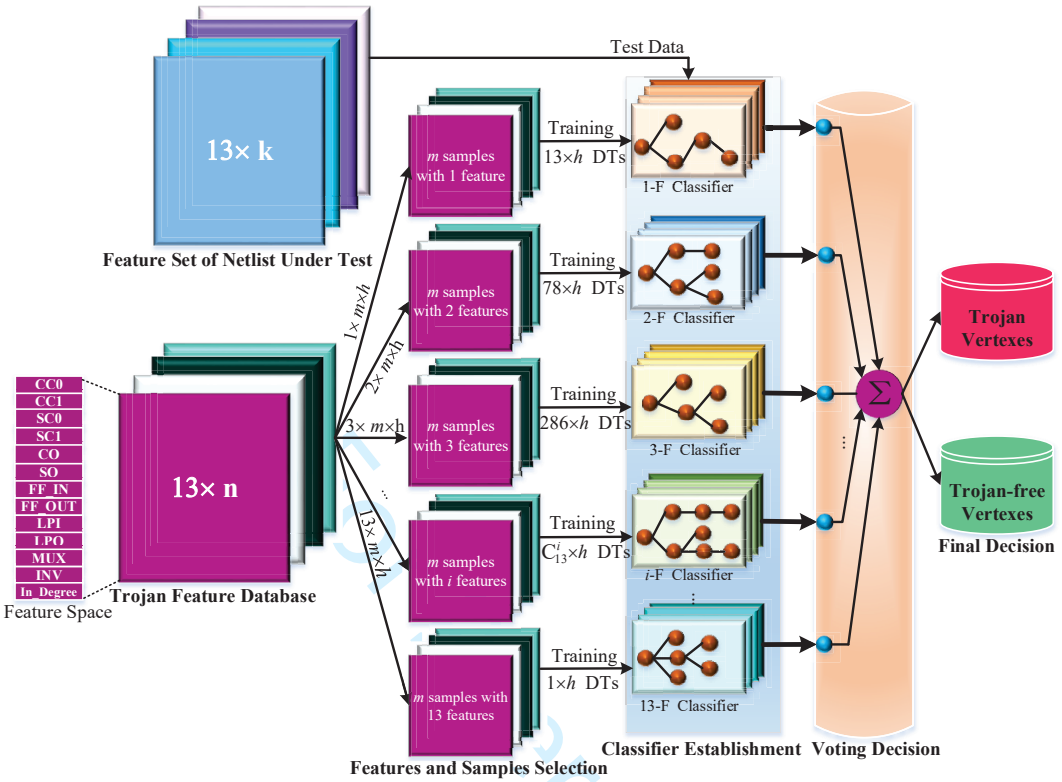

FIGURE 4. The procedure of Trojan feature identification based on the feature-adaptive random forest classifier.

To ensure the effectiveness of Υ_{ij}, h training data subsets ($\mathcal{K}_1, \mathcal{K}_2, \cdots, \mathcal{K}_h$) are selected from the $\mathcal{K}$ to build h decision trees ($\Upsilon_{ij}^1, \Upsilon_{ij}^2, \cdots, \Upsilon_{ij}^h$). Further, h decision trees predict the class of each sample in $\mathcal{V}$, and all the classification results of decision trees vote for the final class of each sample in $\mathcal{V}$. If all the samples in $\mathcal{V}$ are correctly classified, it shows that the corresponding decision tree can identify the features of the Trojan. Otherwise, we select the decision tree with the maximum accuracy (Acc) and recall (Rec) to identify the Trojan features and detect their existence of Trojan. The selection of decision tree is shown in Equation 7.

$$\Upsilon_{ij} \leftarrow \arg\max_{\Upsilon_{ij}^k} \{Acc, Rec\}, k = 1, 2, \cdots, h; \tag{7}$$

Finally, all the decision trees are used to evaluate the trustworthiness of each net of netlist under test and make the prediction with a majority vote of the decision trees. If the sum of the decision trees' outputs $C_{\eth e}^{ij}$ is greater than the specified threshold F_{th}, $\eth e$ is classified as Trojan net, otherwise, $\eth e$ is identified as Trojan-free net. The category $C_{\eth e}$ of $\eth e$ can be expressed in Equation 8. Where e ranges from 1 to k.

$$C_{\eth e} = \begin{cases} 1. & \sum_{i=1}^{13} \sum_{j=1}^{C_{13}^i} C_{\eth e}^{ij} \geq F_{\text{th}}; \\ 0. & \sum_{i=1}^{13} \sum_{j=1}^{C_{13}^i} C_{\eth e}^{ij} < F_{\text{th}}; \end{cases} \tag{8}$$

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 1. The number of Trojan-free nodes and Trojan nodes respectively are denoted as T_r and T_j, which is shown in the second and third column of Table 1. From the results of T_r and T_j in Table 1, the size of Trojan circuits ranges from 0.2K to 102K. Moreover, the ratio R_a=$\frac{T_r}{T_j}$ between the Trojan-free nodes and Trojan nodes is calculated and the results are given in the fourth column of Table 1. The R_a ranges from 4.67 to 5372, which shows that the distribution of Trojan-free nodes and Trojan nodes is unbalanced. It's necessary to take the imbalanced data problem into consideration and introduce a data rebalancing method to solve this problem. Furthermore, we analyze the characteristics of Trojan circuits and list the features of each Trojan circuit. Regarding the sixth column of Table 1, all the Trojans have more than one feature. Therefore, all the possible features should be considered during the Trojan feature identification process. The proposed hardware Trojan detection framework is implemented in Python, and a personal computer with Intel Core i5-8265U CPU@1.60GHz and 8GB of RAM is used for experiments.

Table 1. 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, In-Degree, LPO
S35932-T100	6389	32	199.65	leak-information / change-functionality	CC0, CC1, SC0, SC1, In-Degree, LPI, LPO
S35932-T200	6390	27	236.66	denial-of-service / change-functionality	CC0, CC1, CO, SC0, SC1, SO, In-Degree, LPI
S35932-T300	6386	55	116.1	denial-of-service / degrade-performance	CC0, CC1, SC0, SC1, In-Degree, LPI, INV
S38417-T100	5782	29	199.37	denial-of-service / change-functionality	CC0, CC1, CO, SC0, SC1, SO, In-Degree
S38417-T200	5779	35	165.11	denial-of-service / change-functionality	CC0, CC1, CO, SC0, SC1, SO, In-Degree
S38417-T300	5785	61	94.83	denial-of-service / change-functionality	CC0, CC1, CO, SC0, SC1, SO, In-Degree, INV
S38584-T100	7344	19	386.52	denial-of-service / change-functionality	CC0, CC1, CO, SC0, SC1, SO, In-Degree, 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
VGA-LCD_T100	69836	13	5372	leak-information / change-functionality	CC0, CC1, SC0, SC1, LPO
WB_CONMAX_T100	22162	35	633.2	leak-information / change-functionality	CC0, CC1, CO, SC0, SC1, SO, In-Degree
EthernetMAC10GE_T700	102888	31	3318.96	change-functionality	CC0, CC1, CO, SC0, SC1, SO, In-degree
EthernetMAC10GE_T710	102888	31	3318.96	change-functionality	CC0, CC1, CO, SC0, SC1, SO, In-degree
EthernetMAC10GE_T720	102888	31	3318.96	change-functionality	CC0, CC1, CO, SC0, SC1, SO, In-degree
EthernetMAC10GE_T730	102889	30	3429.63	change-functionality	CC0, CC1, CO, SC0, SC1, SO, In-degree

We build the Trojan feature database using the feature extraction method and 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 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 the optimized parameter with maximal validation accuracy. Finally, ω, ϱ, ρ and φ respectively are 201, 22, 33, 1, and the validation accuracy reaches 99.83%.

Table 2. Trojan detection results of the proposed approach.

Circuits	*TP*	*TN*	*FP*	*FN*	*TPR*	*TNR*	*ACC*
RS232-T1000	44	202	13	0	100%	93.95%	94.98%
RS232-T1100	43	206	10	0	100%	93.57%	96.14%
RS232-T1200	34	211	11	3	91.89%	95.05%	94.59%
RS232-T1300	31	216	7	0	100%	96.86%	97.24%
RS232-T1400	43	202	11	2	95.56%	94.84%	94.96%
RS232-T1500	46	205	10	0	100%	95.35%	96.17%
RS232-T1600	28	217	11	2	93.33%	95.18%	94.96%
S15850-T100	49	2377	9	11	81.67%	99.62%	99.18%
S35932-T100	32	6260	129	0	100%	97.98%	97.99%
S35932-T200	17	6338	52	10	62.96%	99.19%	99.03%
S35932-T300	54	6166	220	1	98.18%	96.55%	96.57%
S38417-T100	12	5742	40	17	41.38%	99.31%	99.02%
S38417-T200	29	5754	25	6	82.86%	99.57%	99.47%
S38417-T300	55	5740	45	6	90.16%	99.22%	99.13%
S38584-T100	15	7320	24	4	78.95%	99.67%	99.62%
S38584-T200	127	7324	11	9	93.38%	99.85%	99.73%
S38584-T300	1144	7321	13	10	99.13%	99.82%	99.73%
VGA-LCD_T100	12	69004	832	1	92.31%	98.81%	98.81%
WB_CONMAX_T100	31	21513	649	4	88.57%	97.07%	97.06%
EthernetMAC10GE_T700	29	102412	476	2	93.55%	99.54%	99.54%
EthernetMAC10GE_T710	30	102435	453	1	96.77%	99.56%	99.56%
EthernetMAC10GE_T720	30	102537	351	1	96.77%	99.66%	99.66%
EthernetMAC10GE_T730	29	101965	924	1	96.67%	99.10%	99.10%
Average					90.48%	97.92%	97.92%

After extracting the multi-level feature extraction of netlist under test, the trained classifiers are used to classify the samples and distinguish the Trojan nets from the netlist. The Trojan detection results are shown in Table 2. Where the true positive (*TP*) and false negative (*FN*) show the number of Trojan samples identified to be Trojan samples and Trojan-free samples respectively, whereas the true negative (*TN*) and the false positive (*FP*) imply the number of Trojan-free samples identified to be Trojan-free samples and Trojan samples respectively. In this case of classification, 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. As shown in Table 2, the majority of Trojan and Trojan-free 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 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. [40], [49], [22], [25] and [35]) and the comparison results are shown 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 nets than other methods. Compared with the Ref. [40], [22], [49] and [35], the proposed approach achieves a similar level of performance with *TNR*=97.92%. However, Ref. [40] uses four testability metrics as the Trojan features, while Ref. [49], Ref. [35] and [49] use several structural features to detect several specified types of small-scale Trojans. For the [40], [22], [49] and [35], 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 [35]) with *ACC*, which shows that the proposed approach distinguishes the Trojan nets with a significantly smaller number of false alarms compared to the Ref. [22], [25] and [35].

Besides, several large circuits scale from 22K to 102K nodes (e.g. WB_CONMAX-T100, VGA_LCD-T100, EthernetMAC10GE-T700, and so on) are further used to evaluate the efficacy of the proposed approach. Regarding Tables 2 and 3, the proposed approach can identify most of the Trojan nets and only leave no more than four misclassified Trojan nets. In addition, there are 2.08% Trojan-free nets are misclassified as Trojan nets, which further reduces the cost of manual review to check whether the suspicious nets are Trojan or not. Therefore, we can conclude that the proposed approach scales well with large circuits. In summary, the proposed approach identifies the Trojan nets from the gate-level netlist with a small number of false alarms that is much smaller than the existing methods.

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

Circuits	Ref. [22]			Ref. [25]			Ref. [40]			Ref. [49]			Ref. [35]			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%
VGA_LCD-T100	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	99.31%	99.82%	99.73%
WB_CONMAX-T100	-	-	-	-	-	-	-	-	-	100%	100%	100%	-	-	-	88.57%	97.07%	97.06%
EthernetMAC10GE-T700	-	-	-	-	-	-	100%	100%	100%	-	-	-	-	-	-	93.55%	99.54%	99.54%
EthernetMAC10GE-T710	-	-	-	-	-	-	100%	100%	100%	-	-	-	-	-	-	96.77%	99.56%	99.56%
EthernetMAC10GE-T720	-	-	-	-	-	-	100%	100%	100%	-	-	-	-	-	-	96.77%	99.66%	99.66%
EthernetMAC10GE-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%

6.2 Discussion

In this paper, we introduce a hardware Trojan detection framework to evaluate the trustworthiness of third-party intellectual property cores at the design stage. The proposed approach complements the other methods (e.g. run-time and test-time approaches) to ensure the trustworthiness of circuits at all stages of the design. It is also scalable to the register transfer level design that synthesizes it into the gate-level netlist before validation. Multi-level feature analysis is the most innovative hardware Trojan detection approach at the design stage, which this paper investigates. In this paper, six functional features and seven structural features are extracted from the Trojan benchmark to form a Trojan feature database that can cover all the types of existing Trojans. Besides, new features of emerging Trojans can be included into the Trojan feature database, and thus the proposed approach can be scalable to the emerging Trojans. Unlike the existing feature analysis approaches focusing on the specific types of Trojans, the proposed approach can detect the change-functionality, denial-of-service, leak-information, and degrade-performance Trojans. The proposed approach identifies the Trojan by checking whether the features of the netlist match with the Trojan feature database or not. The Trojan feature identification process does not require any information about Trojans during the detection

stage and does not need a set of specified security attributes. Thus, the proposed framework has fewer assumptions than the other methods, which is much closer to the real case.

As mentioned above, the existing hardware Trojans have more than one feature, and all the possible feature combinations should be considered during the Trojan feature identification process. Furthermore, a feature-adaptive classifier based on the random forest is proposed to detect the Trojan nodes of a netlist under test, wherein multiple decision trees train with feature combinations of the Trojan feature database and the voting results of decision trees to detect the Trojan. The detection results show that the proposed approach achieves a comparable *TPR*, *TNR*, and *ACC* compared to the existing methods. Therefore, we can claim that the proposed approach can cover more types of hardware Trojans and is scalable to emerging Trojans.

In the real case, the size of the Trojan is much smaller than the original design, and a small number of Trojan circuits appear in the literature. Therefore, an imbalanced data problem exists where the samples belonging to the Trojan-free class heavily outnumber the Trojan class. Several machine learning algorithms have been proposed to identify the features or behaviors of Trojan, and little attention pays to the imbalanced data problem. The classifier trained with this imbalanced data is insufficient learning for the minority class and biased toward the majority class, which causes a low detection accuracy of the existing detection approaches. There is doubt that those biased classifiers make a poor prediction performance for the unknown Trojans. In this paper, we propose the SMOTEKNN to generate new synthetic data of the Trojan class and build a balanced Trojan feature database to train the classifier. The detection results show that the proposed classifier has successfully detected 23 gate-level Trojan circuits of different sizes.

7 CONCLUSION

In this paper, a hardware Trojan detection framework is presented to identify the Trojan nets from the gate-level netlist. A multi-level features extraction approach is used to extract 13 features from the gate-level netlist, and a feature rebalancing method is presented 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 nets 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.

REFERENCES

[1] Chongxi Bao, Domenic Forte, and Ankur Srivastava. Temperature tracking: Toward robust run-time detection of hardware trojans. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(10):1577–1585, 2015.

[2] Lok-Won Kim and John D. Villasenor. Dynamic function verification for system on chip security against hardware-based attacks. *IEEE Transactions on Reliability*, 64(4):1229–1242, 2015.

[3] Haocheng Ma, Jiaji He, Yanjiang Liu, Jun Kuai, He Li, Leibo Liu, and Yiqiang Zhao. On-chip trust evaluation utilizing tdc-based parameter-adjustable security primitive. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(10):1985–1994, 2021.

[4] Sreeja Rajendran and Mary Lourde R. A novel algorithm for hardware trojan detection through reverse engineering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2021.

[5] Chongxi Bao, Domenic Forte, and Ankur Srivastava. On reverse engineering-based hardware trojan detection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(1):49–57, 2016.

[6] Trey Reece and William H. Robinson. Detection of hardware trojans in third-party intellectual property using untrusted modules. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(3):357–366, 2016.

[7] Yanjiang, LIU, Jiaji, HE, Haocheng, MA, Yiqiang, and ZHAO. Golden chip free trojan detection leveraging probabilistic neural network with genetic algorithm applied in the training phase. *Science China(Information Sciences)*, v.63(02):242–244, 2020.

[8] Y. Liu, J. He, H. Ma, and Y. Zhao. Hardware trojan detection leveraging a novel golden layout model towards practical applications. *Journal of Electronic Testing*, 35(11), 2019.

[9] Golden chip-free trojan detection leveraging trojan trigger's side-channel fingerprinting. *ACM Transactions on Embedded Computing Systems (TECS)*, 2020.

[10] Imran Hafeez Abbassi, Faiq Khalid, Osman Hasan, Awais Mehmood Kamboh, and Muhammad Shafique. Mcsevic: A model checking based framework for security vulnerability analysis of integrated circuits. *IEEE Access*, 6:32240–32257, 2018.

[11] Nandeesha Veeranna and Benjamin Carrion Schafer. Hardware trojan detection in behavioral intellectual properties (ip's) using property checking techniques. *IEEE Transactions on Emerging Topics in Computing*, 5(4):576–585, 2017.

[12] Xiaolong Guo, Raj Gautam Dutta, Prabhat Mishra, and Yier Jin. Automatic code converter enhanced pch framework for soc trust verification. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(12):3390–3400, 2017.

[13] Xiaoming Chen, Qiaoyi Liu, Song Yao, Jia Wang, Qiang Xu, Yu Wang, Yongpan Liu, and Huazhong Yang. 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*, 37(7):1370–1383, 2018.

[14] Matthew Hicks, Murph Finnicum, Samuel T. King, Milo M. K. Martin, and Jonathan M. Smith. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *2010 IEEE Symposium on Security and Privacy*, pages 159–172, 2010.

[15] Jie Zhang, Feng Yuan, Lingxiao Wei, Zelong Sun, and Qiang Xu. Veritrust: Verification for hardware trust. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–8, 2013.

[16] Jie Zhang, Feng Yuan, Linxiao Wei, Yannan Liu, and Qiang Xu. Veritrust: Verification for hardware trust. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(7):1148–1161, 2015.

[17] Adam Waksman, Matthew Suozzo, and Simha Sethumadhavan. Fanci: Identification of stealthy malicious logic using boolean functional analysis. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, CCS '13, page 697?708, New York, NY, USA, 2013. Association for Computing Machinery.

[18] Dean Sullivan, Jeff Biggers, Guidong Zhu, Shaojie Zhang, and Yier Jin. Fight-metric: Functional identification of gate-level hardware trustworthiness. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–4, 2014.

[19] Song Yao, Xiaoming Chen, Jie Zhang, Qiaoyi Liu, Jia Wang, Qiang Xu, Yu Wang, and Huazhong Yang. Fastrust: Feature analysis for third-party ip trust verification. In *2015 IEEE International Test Conference (ITC)*, pages 1–10, 2015.

[20] Jie Zhang, Feng Yuan, and Qiang Xu. Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans. In *In Proc. ACM Conference on Computer and Communications Security (CCS*, 2014.

[21] Hassan Salmani. Cotd: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist. *IEEE Transactions on Information Forensics and Security*, 12(2):338–350, 2017.

[22] Kento Hasegawa, Masao Yanagisawa, and Nozomu Togawa. Trojan-feature extraction at gate-level netlists and its application to hardware-trojan detection using random forest classifier. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4, 2017.

[23] Masaru Oya, Youhua Shi, Masao Yanagisawa, and Nozomu Togawa. A score-based classification method for identifying hardware-trojans at gate-level netlists. In *Design, Automation and Test in Europe Conference and Exhibition*, 2015.

[24] M. Oya, N. Yamashita, T. Okamura, Y. Tsunoo, M. Yanagisawa, and N. Togawa. Hardware-trojans rank: Quantitative evaluation of security threats at gate-level netlists by pattern matching. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, 99(12):2335–2347, 2016.

[25] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa. Hardware trojans classification for gate-level netlists based on machine learning. In *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2016.

[26] F. Chen and Q. Liu. Single-triggered hardware trojan identification based on gate-level circuit structural characteristics. In *IEEE International Symposium on Circuits and Systems*, pages 1–4, 2017.

[27] Shichao Yu, Chongyan Gu, Weiqiang Liu, and Maire O'Neill. A novel feature extraction strategy for hardware trojan detection. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2020.

[28] M. Zou, X. Cui, L. Shi, and K. Wu. Potential trigger detection for hardware trojans. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2017.

[29] J. Popat and U. Mehta. Transition probabilistic approach for detection and diagnosis of hardware trojan in combinational circuits. In *India Conference*, 2017.

[30] X. Xie, Y. Sun, H. Chen, and Y. Ding. Hardware trojans classification based on controllability and observability in gate-level netlist. *IEICE Electronics Express*, 14(18):20170682–20170682, 2017.

[31] X. Chen, Q. Liu, S. Yao, J. Wang, Q. Xu, Y. Wang, Y. Liu, and H. Yang. 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*, 2017.

[32] Q. Liu, P. Zhao, and F. Chen. A hardware trojan detection method based on structural features of trojan and host circuits. *IEEE Access*, 7:44632–44644, 2019.

[33] K. Huang and Y. He. Trigger identification using difference-amplified controllability and dynamic transition probability for hardware trojan detection. *IEEE Transactions on Information Forensics and Security*, 15:3387–3400, 2020.

[34] Dakshi Agrawal, Selcuk Baktir, Deniz Karakoyunlu, Pankaj Rohatgi, and Berk Sunar. Trojan detection using ic fingerprinting. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pages 296–310, 2007.

[35] Chen Dong, Yulin Liu, Jinghui Chen, Ximeng Liu, Wenzhong Guo, and Yuzhong Chen. An unsupervised detection approach for hardware trojans. *IEEE Access*, 8:158169–158183, 2020.

[36] Jayesh Popat and Usha Mehta. Transition probabilistic approach for detection and diagnosis of hardware trojan in combinational circuits. In *2016 IEEE Annual India Conference (INDICON)*, pages 1–6, 2016.

[37] Minhui Zou, Xiaotong Cui, Liang Shi, and Kaijie Wu. Potential trigger detection for hardware trojans. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(7):1384–1395, 2018.

[38] Sayandeep Saha, Rajat Subhra Chakraborty, and Debdeep Mukhopadhyay. Testability based metric for hardware trojan vulnerability assessment. In *2016 Euromicro Conference on Digital System Design (DSD)*, pages 503–510, 2016.

[39] M. Priyadharshini and P. Saravanan. An efficient hardware trojan detection approach adopting testability based features. In *2020 IEEE International Test Conference India (ITC India)*, 2020.

[40] M. Tebyanian, A. Mokhtarpour, and A. Shafieinejad. Sc-cotd: Hardware trojan detection based on sequential/combinational testability features using ensemble classifier. *Journal of Electronic Testing*, pages 1–15, 2021.

[41] Yu Su, Haihua Shen, Renjie Lu, and Yunying Ye. A stealthy hardware trojan design and corresponding detection method. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–6, 2021.

[42] Ning Zhang, Zhiqiang Lv, Yanlin Zhang, Haiyang Li, Yixin Zhang, and Weiqing Huang. Novel design of hardware trojan: A generic approach for defeating testability based detection. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 162–173, 2020.

[43] Trust-hub benchmarks. https://www.trust-hub.org, 2021.

[44] Huaikuan Yi, Qingchao Jiang, Xuefeng Yan, and Bei Wang. Imbalanced classification based on minority clustering synthetic minority oversampling technique with wind turbine fault detection application. *IEEE Transactions on Industrial Informatics*, 17(9):5867–5875, 2021.

[45] Qiao Ning, Xiaowei Zhao, and Zhiqiang Ma. A novel method for identification of glutarylation sites combining borderline-smote with tomek links technique in imbalanced data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1, 2021.

[46] Yulia Ery Kurniawati, Adhistya Erna Permanasari, and Silmi Fauziati. Adaptive synthetic-nominal (adasyn-n) and adaptive synthetic-knn (adasyn-knn) for multiclass imbalance learning on laboratory test data. In *2018 4th International Conference on Science and Technology (ICST)*, pages 1–6, 2018.

[47] Luqyana Wanda Athira, Ahmadie Beryl Labique, Supianto Ahmad Afif. K-Nearest Neighbors Undersampling as Balancing Data for Cyber Troll Detection. In *2019 International Conference on Sustainable Information Engineering and Technology (SIET)*, pages 322–325, 2019.

[48] Meng Zhang, Gong Daoxiong. Combining Clustering Undersample and Ensemble Learning for Wearable Fall Detection. In *2022 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 545–5506, 2022.

[49] Chee Hoo Kok, Chia Yee Ooi, Mehrdad Moghbel, Nordinah Ismail, Hau Sim Choo, and Michiko Inoue. Classification of trojan nets based on scoap values using supervised learning. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, 2019.