GNN4HT: A Two-Stage GNN-Based Approach for Hardware Trojan Multifunctional Classification

Lihan Chen, Chen Dong *Member, IEEE*, Qiaowen Wu, Ximeng Liu, *Senior Member, IEEE*, Xiaodong Guo, Zhenyi Chen, Hao Zhang, and Yang Yang, *Senior Member, IEEE*

***Abstract*—Due to the complexity of integrated circuit design and manufacturing process, an increasing number of third parties are outsourcing their untrusted intellectual property (IP) cores to pursue greater economic benefits, which may embed numerous security issues. The covert nature of hardware Trojans (HTs) poses a significant threat to cyberspace, and they may lead to catastrophic consequences for the national economy and personal privacy. To deal with HTs well, it is not enough to just detect whether they are included, like the existing studies. Same as malware, identifying the attack intentions of HTs, that is, analyzing the functions they implement, is of great scientific significance for the prevention and control of HTs. Based on the fined detection, for the first time, this article proposes a two-stage Graph Neural Network model for HTs' multifunctional classification, GNN4HT. In the first stage, GNN4HT localizes HTs, achieving a notable true positive rate (TPR) of *94.28%* on the Trust-Hub dataset and maintaining high performance on the TRTC-IC dataset. GNN4HT further transforms the localization results into HT information graphs (HTIGs), representing the functional interaction graphs of HTs. In the second stage, the dataset is augmented through logical equivalence for training and HT functionalities are classified based on the extracted HTIG from the first stage. For the multifunctional classification of HTs, the correct classification rate reached as high as *80.95%* at gate-level and *62.96%* at register transfer level. This article marks a breakthrough in HT detection, and it is the first to address the multifunctional classification issue, holding significant practical importance and application prospects.**

***Index Terms*—Gate level, golden free, hardware Trojan (HT), HT information graph (HTIG), HT location, HT multifunctional classification, register transfer level (RTL).**

Manuscript received 30 November 2023; revised 29 May 2024; accepted 7 July 2024. Date of publication 15 July 2024; date of current version 26 December 2024. This work was supported in part by the Fund of Fujian Province Digital Economy Alliance, National Natural Science Foundation of China under Grant 62372110, Grant 62072109, Grant 2021J06013, and Grant U1804263; and in part by the Natural Science Foundation of Fujian Province under Grant 2020J01500, Grant 2021J01616, Grant 2020J01891, and Grant 2023J02008. This article was recommended by Associate Editor Y. Lyu. *(Corresponding author: Chen Dong.)*

Lihan Chen, Chen Dong, Qiaowen Wu, Ximeng Liu, Xiaodong Guo, and Hao Zhang are with the College of Computer and Data Science, Fuzhou University, Fuzhou 350108, Fujian, China (e-mail: lh08190527@gmail.com; dongchen@fzu.edu.cn; qiaowenwu1@gmail.com; snbnix@gmail.com; xiaodong.guo0328@gmail.com; zhanghao@fzu.edu.cn).

Zhenyi Chen is with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 32306 USA (e-mail: zhenyichen@usf.edu).

Yang Yang is with the School of Computing and Information Systems, Singapore Management University, Singapore (e-mail: yang.yang.research@gmail.com).

Digital Object Identifier 10.1109/TCAD.2024.3428469

I. Introduction

THE EMERGENCE of cyber-physical systems (CPSs) and the consequent growth of the Internet of Things have heightened the reliance on electronic devices, placing integrated circuits at a security crossroads. Security risks in these circuits threaten national structures, economic stability, and personal financial and privacy security [1], [2]. The increasing focus on these hardware security issues is evident across various sectors of society [3], [4]. System-on-Chip (SoC), a critical cyberinfrastructure component, presents distinct security challenges that cannot be ignored [5], [6].

Driven by market forces, chip producers increasingly outsource design, depending on third-party automation tools and intellectual property (IP) cores, shortening development times but introducing vulnerabilities [7], [8]. These vulnerabilities may be exploited by adversaries embedding hardware Trojans (HTs) that escape late-stage detection and activate in operation, compromising device integrity [1]. Moreover, chip assembly from globally sourced components complicates security, amplifying the costs and revealing weaknesses in chip design methodologies [9].

HTs are recognized as a profound threat to IP core security within integrated circuits and are composed of triggers and payloads that are deftly concealed [10], [11]. Their objectives include leaking sensitive information, changing functionality, performance degradation, etc. As the linchpin of cyberspace, integrated circuits are foundational in multiple sectors; hence, undetected Trojans that become active could wreak havoc [12], [13], [14]. Unknown trojans, hidden in the depths, trigger panic with their undisclosed functions, constantly threatening to compromise privacy and property, and possessing the potential to ignite warfare. In recent years, scientists worldwide have been committed to solving the problem of HT detection and have made significant progress [15].

In the semiconductor manufacturing process, two critical stages are delineated: 1) the presilicon and 2) post-silicon phases. The presilicon phase, preceding actual production, involves only the conceptual and design-level documentation. Early detection of HTs during this phase can significantly reduce potential losses and promote the timely implementation of defensive measures. Benefiting from the augmented computational power of modern chips, machine learning-based approaches to HT detection have gained prominence. Distinct from traditional Trojan detection methodologies, such

© 2024 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

as side-channel analysis, this novel approach obviates the need for a golden reference, which is often elusive [16]. Furthermore, machine learning methods have demonstrated enhanced efficacy in Trojan detection, especially at gate-level [17], [18], [19].

During the presilicon stage, it is imperative to focus on analyzing register transfer level (RTL) descriptions and gate-level netlists. RTL descriptions encapsulate the hardware's logic operations and data flow processes, whereas gate-level netlists articulate the connections between logic gates. Traditional machine learning methods typically involve extensive feature extraction from each component detailed at RTL descriptions or gate-level netlists. This conventional approach, while thorough, often depends significantly on the distribution of the dataset and may inadvertently neglect essential netlist-specific information. The recent approach of representing circuits as graph structures has demonstrated promising results. These graph models encapsulate the intricate relationships within circuitry, offering a richer and more natural representation for HT detection. Envisioning logic gates as nodes and their interconnects as edges within gate-level netlists and leveraging these constructs in graph models has yielded encouraging outcomes. At RTL, converting hardware design into data flow graphs (DFGs) for comprehensive graph classification marks a novel direction [20].

The goals of HT detection split into two primary strands [21]: 1) partial detection, which interrupts the design workflow upon detecting a Trojan and 2) complete detection, which aims to deactivate only the Trojan-compromised sections. However, existing methods based on machine learning or graph models have not yet truly achieved comprehensive complete detection. To deal with HTs effectively, it is not enough to just detect their presence or absence. Like malware, understanding the attack intentions of HTs, that is, analyzing the functions they implement, is of great scientific significance for studying the prevention and control of HTs. The most refined approaches among the latest work have managed to localize Trojans only at gate-level; for analyzing functionality, i.e., the different purposes of Trojans, a novel and robust solution is still needed.

The HT multifunctional classification has three significant implications.

1) *Improved Detection Accuracy:* The HT multifunctional classification aids in more accurately identifying different types of HTs, thereby enhancing the precision and efficiency of detection methods. This categorization allows security systems to more effectively discern true threats rather than misidentifying normal behavior as malicious.
2) *Tailored Defensive Strategies:* Through HT multifunctional classification, security levels can be assigned to HTs, enabling more targeted defensive measures.
3) *Deeper Insight Into Trojan Threats:* Understanding the diverse functions in HT multifunctional classification assists in better-predicting attack mechanisms and targets, guiding more precise defensive strategies.

As with the functionality detection of malicious code, identifying the functionality of malicious code first requires determining its location or the module they inhabit. Similarly, locating the Trojan is the initial step for complete HT detection, followed by HT multifunctional classification. Therefore, precision in accurately locating Trojan hardware is crucial. Unlike the solid logical relationships found in malicious code, HTs lack substantial logical interconnections. Characterizing their combined effects becomes a significant challenge once they are located. Most importantly, the scarcity of current datasets limits the possibility of providing sufficient data for model training.

To address the critical challenges of HT multifunctional classification, we introduce *GNN4HT*, a pioneering two-stage, golden-free approach. Our strategy innovatively classifies HT functionalities, marking a first in the field. The precision of the system in HT multifunctional classification reaches *80.95%* at gate-level and *62.96%* at RTL.

Our contributions are as follows.

1) *GNN4HT* is developed, a groundbreaking two-stage HT complete detection framework that localizes and classifies HT functionalities accurately without the need for a golden reference, significantly advancing the field.
2) *GNN4HT* transforms gate-level netlists into undirected graphs, incorporating graph isomorphism network (GIN) and global graph features that substantially improve the accuracy of HT localization, which achieves the true positive rate (TPR) of *94.28%* and demonstrate the model's generalizability to TRIT-TC benchmark with *95.81%* TPR and *99.02%* true negative rate (TNR).
3) *GNN4HT* introduces the hardware Trojan information graph (HTIG), effectively connecting the interrelations between Trojan gates and unifying the representation of Trojan functionalities.
4) *GNN4HT* employs a GNN whole-graph classifier. It augments the training set using a logic equivalence method and further classifies the Trojan functionalities based on the localization results from the first stage, achieving an unprecedented *80.95%* accuracy rate. To further validate the effectiveness of our second stage, we synthesized RTL designs into gate-level netlists to serve as our test set, ultimately achieving an accuracy of *62.96%*

The remainder of this article is organized as follows: Section II reviews related work. Section III describes the Problem Model. Section IV details the Methodology, encompassing graph modeling and threshold-based data augmentation, among other techniques. Section V discusses the Experimental Results, validating the efficacy of our proposed methods. Finally, Section VI offers concluding remarks.

II. Related Works and Motivation

A. Related Works

Machine learning excels at unraveling intricate patterns within complex datasets that typically elude human scrutiny. It enables models to learn from data and autonomously adjust to emerging patterns. In the pioneering work of Hasegawa et al. [22], the extraction of netlist features and the conceptualization of networks were introduced. Their

approach utilized support vector machines (SVMs) for binary classification, creatively mitigating the issue of data imbalance by duplicating Trojan instances. Further advancements were made by incorporating random forests for refined classification as explored in [23]. In a subsequent study, a multilayer perceptron was employed [21]; however, the features used were overly dependent on dataset distribution and could not detect unknown Trojans.

While conventional machine learning techniques for HT detection are promising, pathway-level analysis represents a specialized niche. Pioneering works by Lu et al. [12] treated each pathway as an individual entity, utilizing the netlist's entirety as a training corpus for a natural language processing (NLP) model, yielding encouraging outcomes. Moreover, LMDet [24] harnessed NLP methodologies to discern the "unnaturalness" in HTs vis-à-vis legitimate circuits. This was accomplished through statistical language models that sequence circuit gates, scrutinizing their probabilistic fit within the model to signal circuit sequences as suspicious when exhibiting low-probability thresholds, indicative of potential Trojan activity.

Graph-based models have significantly advanced the analytical prowess at RTL and gate-level, illuminating pathways to detect and classify HTs with enhanced precision. Techniques like GNN4TJ [25] and HW2VEC [26] at RTL have pioneered the use of DFGs, leveraging the structural intricacies of netlists through GNNs. The DFG structure, which matches the RTL designs, brings high-precision reporting, but it cannot accurately locate Trojans and can even only achieve partial detection.

At gate-level, innovations, such as NHTD-GL [27] and TrojanSAINT [28], have achieved marked success in precisely pinpointing Trojan locations. NHTD-GL, employing a pioneering graph structure, provided a novel method for Trojan localization, achieving high precision. Nonetheless, it stopped short of realizing complete detection, such as multifunctional classification. TrojanSAINT also achieved precise localization of HTs based on gate-level netlists, demonstrating high accuracy. This study further pointed out that the experimental approach commonly adopted in existing literature, which involved reserving one netlist for validation and testing, lacked generalizability and did not correspond to real-world scenarios.

While transformative in HT detection, graph models are not impervious to exploitation. The BadGNN attack [29] underscores the imperative to scrutinize GNNs for vulnerabilities. This method injected backdoor triggers into otherwise benign models by introducing subtle perturbations into the circuit's graph representation; when these compromised models undergo training for graph classification, they adeptly bypass detection mechanisms, posing a hidden threat.

Against this backdrop, the principle of logical equivalence emerges as a bulwark to bolster the robustness of detection models. A pioneering approach [30] harnessing machine learning has been introduced, utilizing data augmentation to strengthen Trojan detection. This technique generated a diversified dataset by substituting gates within Trojan circuits with their logical equivalents, enhancing the model's exposure to varied Trojan patterns. The R-HTDetector [31] pushed the envelope further by employing adversarial training rooted in logical equivalence, achieving strides toward machine learning models that are inherently more resilient. Despite these advancements, integrating logical equivalence into existing graph-based models remains untapped.

B. Research Challenges

At present, the solutions to the HT multifunctional classification problem face the following challenges.

1) *Golden-Free:* The lack of readily available golden chips necessitates a detection methodology that does not rely on such references, making it critically important for practical applications.
2) *Feature Representation:* Selecting and engineering features that effectively capture the essence of the problem is vital in machine learning. For HT detection, this involves converting circuit data into a format understandable to the model, which is inherently challenging.
3) *Scalability:* A model's ability to effectively adapt to different netlists is crucial for widespread adoption. Strong generalization capability allows for more efficient design processes, reducing time and resources spent on adaptation.
4) *Unknown Trojan Functionality:* Detecting and neutralizing Trojan components does not inherently disclose their intended function. Understanding the functionality of a Trojan goes beyond mere detection and is key to developing a proactive defense strategy.

To address the abovementioned challenges, this article introduces a two-stage, golden-free complete detection approach to achieve high-precision localization and multifunctional classification of HTs.

III. Problem Model

In the context of this research, we proceed with a foundational hypothesis that an attacker may implant HTs during the design or manufacturing phases, especially within third-party IP cores or outsourced production scenarios. The attacker's objectives may include leaking sensitive information, compromising chip functionality or performance, or activating malevolent actions under certain conditions. The detection based on gate-level netlists can be provided by suppliers or obtained through reverse engineering principles involving the reading and restoration from schematics.

In the given netlists, each logic gate i can be encoded as a vector $\mathbf{X}_i \in \mathbb{R}^d$, where d represents the dimension of the encoding space. Consequently, a netlist can be represented as a matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$, where n denotes the number of logic gates. We formulate the HT location task as a Trojan-gate classification problem, which can be described as follows:

$$y = f_{\text{Location}}(X), \quad y^{(i)} \in y \tag{1}$$

$$y^{(i)} = \begin{cases} (1,0), \text{ if the node is a Trojan gate} \\ (0,1), \text{ otherwise.} \end{cases} \tag{2}$$

Definition (HTIG): Let Φ be a mapping function. We extract the set of gates characterizing the HT functionality information

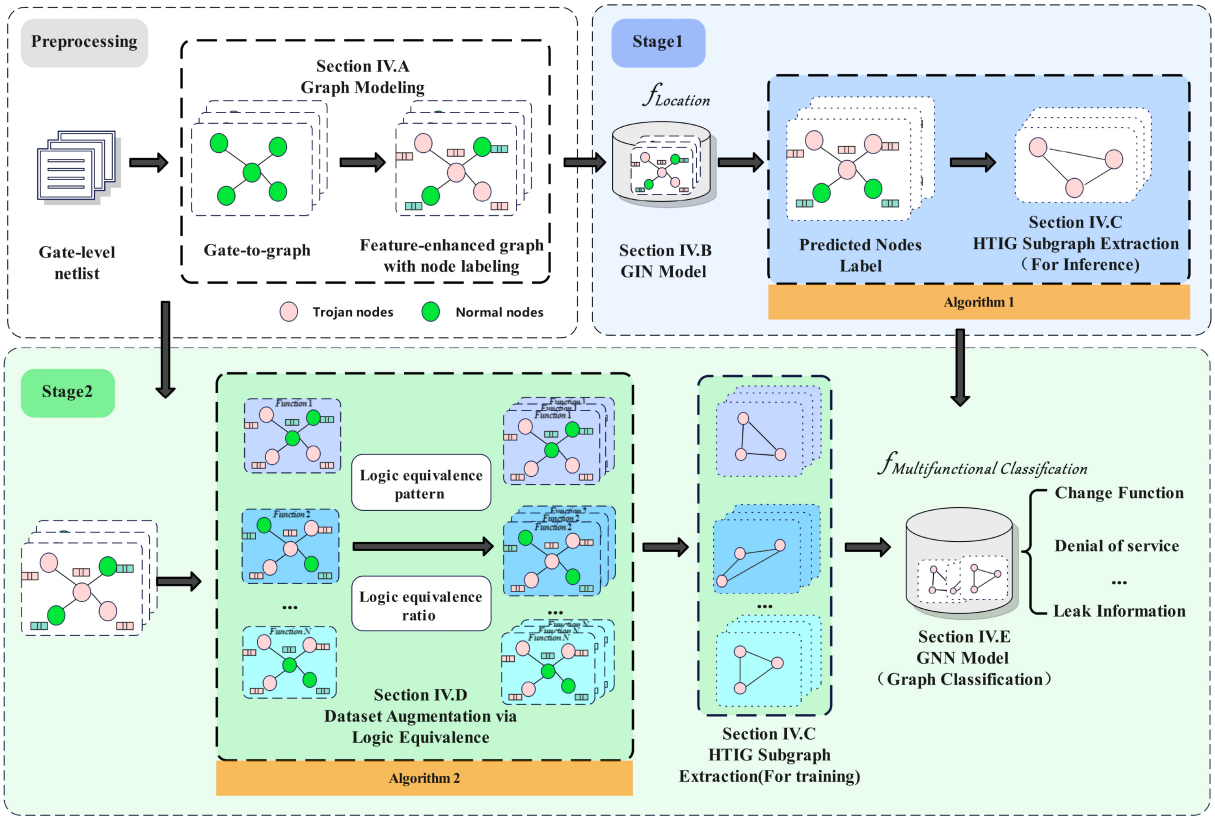


Fig. 1. Overall process of GNN4HT. The entire process includes two stages, location and multifunctional classification.

into a new structure, which is defined as

$$\mathbf{HTIG} = \Phi(\mathbf{X}, \mathbf{y}) \tag{3}$$

where Φ is a transformation that encapsulates the localized Trojan behavior within the netlist.

To further categorize the HTIG, that is, to ultimately classify the function of the Trojan within each netlist, we proceed as follows:

$$\mathbf{y}_{\mathrm{HT}} = f_{\text{Multifunction Classification}}(\mathrm{HTIG}), \quad \mathbf{y}_{\mathrm{HT}} \in \mathbb{R}^n \tag{4}$$

where the vector $\mathbf{y}_{\mathrm{HT}} = [c_1, c_2, \ldots, c_n]$ is composed of elements c_i that indicate the presence of specific functionalities, where

$$c_i = \begin{cases} 1, \text{ if HTIG corresponds to functionality class } i \\ 0, \text{ otherwise} \end{cases} \tag{5}$$

each c_i is a binary indicator within the one-hot encoded vector $\mathbf{y}_{\mathrm{HT}}$, representing whether a particular Trojan functionality.

In this study, we introduce the GNN4HT framework, designed to develop two models that approximate the functions f_{Location} and $f_{\text{Multifunction Classification}}$, respectively.

IV. Methodology

In this section, we outline the comprehensive framework of GNN4HT and then delve into the specific methodologies that support each stage of this framework.

A. GNN4HT Framework

As depicted in Fig. 1, the entire architecture of GNN4HT consists of two main stages, each relying on a distinct graph neural network model to solve specific target tasks. The first stage aims to localize HTs in the target gate-level netlist and extract related information. The second stage focuses on addressing the issue of data scarcity by augmenting the dataset and then performing further multifunctional classification based on the information acquired in the first stage. The GNN4HT process is as follows:

Preprocessing: Each gate-level netlist is represented as a graph, and nodes and edges are endowed with specialized features. It contains two processes as follows.

1) *Graph Construction:* Extract all logical gates from the gate-level netlist as the nodes in the graph, and the connections between these logical gates form the edges of the undirected graph.
2) *Node and Edge Feature Allocation:* Features are assigned to each node based on its original information and topological location in the graph. To more accurately locate HTs, global features are added to the nodes, and port information is incorporated into the edge features.

Stage One: The objective of this stage is to achieve the localization of the Trojan as well as to extract relevant information about the Trojan; it primarily includes the following two key steps.

1) *Model Training and Inference:* f_{Location} is GIN model which is aimed at node classification. Each node is assigned a unique computation graph and classified

using the GIN for more accurate localization of the Trojan; the unique computation graph is articulated in the following equation:

$$f_v : G_v \to F \tag{6}$$

where f_v is an injective function, ensuring that the computation graph G_v for each node v is unique and F represents the feature space to which each node's computed graph is mapped.

2) *HTIG Extraction(for Inference):* Nodes classified as HTs are extracted to form HTIG, which serves as the input for the model in the second stage.

Stage Two: This stage is dedicated to enlarging the dataset and implementing the HT multifunctional classification. It fundamentally comprises three pivotal steps.

1) *Logical Equivalence Handling:* To alleviate the problem of data scarcity, logical equivalence is used to transform the *Trojan nodes* in the undirected graph constructed in the preprocessing stage. Let the new graph after logical equivalence transformation be denoted as $G' = (V', E')$, where V' and E' can be defined as follows:

$$V' = (V \setminus V_{\mathrm{HT}}) \cup \phi(V_{\mathrm{HT}}) \tag{7}$$

$$E' = E \tag{8}$$

where $\phi : V_{\mathrm{HT}} \to V'$ is the logical equivalence transformation function.

2) *Dataset Threshold Filtering:* To filter the new dataset derived from logical equivalence, we established a threshold filtering mechanism. The threshold is calculated based on the intraclass similarity of the HTIG, transformed from the undirected graphs associated with each category of HT functionalities. The specific calculation is as follows:

$$\mu = \frac{1}{\binom{|D_{\mathrm{function}}|}{2}} \sum_{\substack{\mathrm{HTIG}_i, \mathrm{HTIG}_j \in D_{\mathrm{function}} \\ i \neq j}} \mathrm{Similarity}\left(\mathrm{HTIG}_i, \mathrm{HTIG}_j\right). \tag{9}$$

If the similarity $\theta_{\mathrm{HTIG'}}$ derived from G' and its corresponding class in the dataset is less than the established threshold μ for that class, HTIG' is permitted to be added to the dataset D_{function}. Otherwise, it is not included.

3) *Model Training and Inference:* $f_{\mathrm{Multifunctional\ Classification}}$ is a GNN model specifically devised for whole graph classification. The HTIGs, augmented and categorized based on distinct functionalities, will be employed as the training dataset to cultivate a robust model. The HTs located in the initial stage and subsequently transformed into HTIGs will function as the inputs for $f_{\mathrm{Multifunctional\ Classification}}$, culminating in the achievement of precise functional classification outcomes.

B. Graph Modeling

A common challenge in using graph models for HT issues is deciding whether to represent circuit schematics as directed or undirected graphs. Utilizing directed graphs preserves the fundamental direction of signal propagation in the circuitry, which seems apt for this problem. However, for graph models, the aggregation function only collects information in one direction of each edge, consequently neglecting the nodes connected on the opposite side [27]. Thus, we opt to represent circuit diagrams as undirected graphs, facilitating better convergence of the graph neural networks.

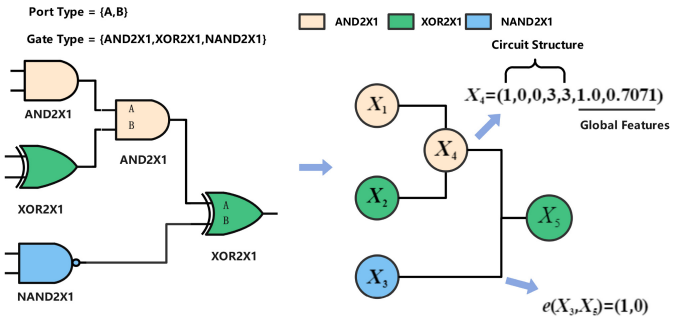


Fig. 2. Circuit graph modeling process of the gate-level netlist.

Each gate-level netlist is constructed as an undirected graph $G = \langle V, E \rangle$. The node set $V = \{v_0, v_1, \ldots, v_{n-1}\}$ comprises n logic gates derived from the netlist. The edge set $E = \{e_{st} | s, t = 0, \ldots, n-1\}$ signifies the connections among the gates. A bidirectional edge e_{st} is in E if the output port of the logic gate v_s is wired to the input port of the gate v_t.

To represent the information and graph structure of the circuit diagram and to enable better convergence of the graph model, we have encoded the following features for each node and edge in the graph as depicted in Fig. 2. Gate-level netlists are converted into undirected graphs by a script we developed, and features are automatically extracted from these graphs.

1) Circuit Structure: To represent the information in the circuit, we extracted three features, with two serving as node features and one as edge features.

Gate Type: Each gate type is uniquely identified through one-hot encoding, with the bit corresponding to the specific gate type encoded as one and all others as 0.

Port Information: The logic gate port data is embedded into the edge feature. Specifically, for an edge where the output of one gate connects to the input of another gate, the port number type of the input is used as the edge feature and is represented using one-hot encoding. As Fig. 2 shows, The graph includes two types of connections where the outputs of certain gates are linked to the inputs at ports A and B. In the original gate-level netlist, the output of X_3 is connected to the input at port B of X_5. Therefore, the edge $e(x_3, x_5)$ is $(1, 0)$.

Out-Degree and In-Degree: The degrees of outgoing and incoming edges will be incorporated as features to characterize the nodes, representing the logic gates.

2) Global Features: To enhance the localization of trojans, we introduce global features to enable the model to more effectively capture the distinct positions of nodes within the graph. Global features are represented as node features to indicate the positions of gates within the gate-level netlist.

Betweenness Centrality: This measure of centrality gauges a node's importance across all shortest paths and can be a vital feature [32]. In the context of this study, nodes with low-betweenness values may be integral components of HTs,

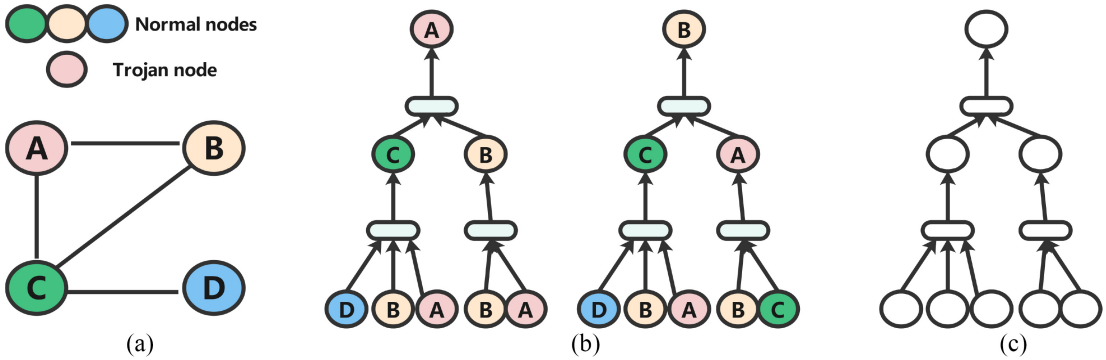


Fig. 3. (a) Demonstrates a fundamental graph structure. (b) Indicates that in this graph structure, when nodes A and B aggregate the same points, due to GIN's independent mapping function, the final embeddings generated in their computation graphs are different. (c) Represents that in the traditional GNN perspective, the computation graphs for nodes A and B are identical.

influencing numerous signal paths

$$C_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)} \tag{10}$$

where s and t are nodes in the graph, $\sigma(s,t)$ is the total number of shortest paths from node s to node t, and $\sigma(s,t|v)$ is the number of those paths that pass through v.

Eigenvector Centrality: Employing eigenvector centrality gives a multifaceted assessment of a node's influence[33]. Nodes with significant eigenvector centrality may constitute key elements of HTs, given their considerable effect on the entire graph

$$\mathbf{x}^{(k+1)} = \frac{A\mathbf{x}^{(k)}}{|A\mathbf{x}^{(k)}|} \tag{11}$$

where A is the adjacency matrix of the graph, $\mathbf{x}^{(k)}$ is the vector of centrality scores at iteration k, and $\mathbf{x}^{(k+1)}$ is the vector of centrality scores at the next iteration.

C. Graph Isomorphism Network for HT Location

The effectiveness of GNN in various fields has been overshadowed by specific critical limitations in capturing underlying graph structures. We address this challenge by implementing our model using the GIN [34], renowned for its superior ability to discern complex structures that traditional GNN or GraphSaint models might miss.

Classic GNNs or GraphSaint models may misclassify identical structures within a graph, leading to possible misclassification of HT nodes as normal nodes if they exist within the same subgraph structure as depicted in Fig. 3. This misclassification can result in a significant drop in accuracy.

Conversely, GIN mitigates this challenge by associating each node's computational graph with unique functions. This implies that GIN can distinguish between nodes that exhibit structural resemblance or are identical by correlating them with disparate functions. The integration of a specialized global structure, as delineated in Section IV-B, further augments the GIN's proficiency in identifying Trojan nodes. The node update function is articulated in the following equation:

$$h_v^{(k+1)} = \mathrm{MLP}\left((1+\epsilon)\cdot h_v^{(k)} + \sum_{u \in N(v)} h_u^{(k)}\right) \tag{12}$$

where $h_v^{(k+1)}$ signifies the updated feature of node v at the $(k+1)$th iteration, $N(v)$ is the set of neighboring nodes of v, and ϵ is a trainable parameter.

The refined node features are then fed into a Softmax layer to produce the final classification

$$Y = \mathrm{Softmax}\left(\mathrm{FC}(h_v^{(k+1)})\right) \tag{13}$$

where FC denotes the Fully Connected layer, and $h_v^{(k+1)}$ are the node features processed by the GIN model.

In our methodology, we optimize the model using a specialized weighted cross-entropy loss function, emphasizing the prediction of Trojan nodes

$$\mathrm{Loss} = -\sum_{i=1}^{N} w_i \left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right] \tag{14}$$

here, N indicates the total number of data points, w_i refers to the weights for each category (based on the ratio of Trojan and non-Trojan nodes), y_i represents the proper categories, and p_i are the predicted probabilities. This loss function makes the model more biased toward predicting Trojan nodes, enhancing accuracy in locating HTs.

D. Hardware Trojan Information Graph

HTs Typically Comprise Two Elements: a trigger and a payload part. The trigger and payload often reside within low-activity, inconspicuous networks, modifying the propagation of signals to achieve their objective [35], [36]. To express their combined effect and characterize their joint representation, we propose the concept of HTIG. The following outlines the derivation method for HTIG.

1) *Node Selection:* We extract the nodes associated with HTs from the undirected graph G. This is defined as

$$\mathcal{V}_{\mathrm{HTIG}} = \{v_i \mid v_i \in \mathcal{V}_G, \text{and } v_i \text{ is a trojan node}\} \tag{15}$$

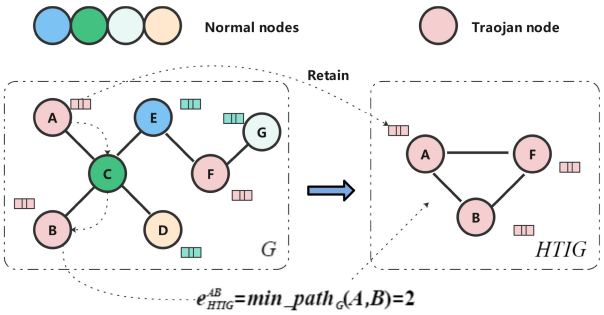


Fig. 4. Construction of HTIG.

Algorithm 1 Extract HTIG

Input: Graph G, HT nodes set H
Output: *HTIG*
Initialize an empty graph *HTIG*
for each node n in G **do**
  if n is in H **then**
    Add node n to *HTIG*
  end if
end for
for each pair of nodes n_1, n_2 in *HTIG* **do**
  if $n_1 \neq n_2$ **then**
    The shortest distance between node n_1 and n_2 in the original graph G is used as the edge weight in *HTIG*
  end if
end for
return HTIG

the vectors of nodes in the original graph G will be retained, which can characterize the information of this Trojan gate in the original graph.

2) *Edge Definition:* For the edges within HTIG, they are defined as follows:

$$\mathcal{E}_{\text{HTIG}} = \left\{ (v_i, v_j, w_{ij}) \mid v_i, v_j \in \mathcal{V}_{\text{HTIG}} \right\} \tag{16}$$

here, w_{ij} signifies the number of *hops* between identified Trojan gates, depicted as the shortest distance between nodes in the original graph, encapsulating the interconnections among Trojan components.

With the initial stage of the GNN4HT framework concluded, wherein Trojans are located, and their nodes are harvested into HTIG for assembling the test set for multifaceted classification tasks, attention shifts toward the onset of the second stage. This stage, illustrated in Fig. 1, entails preparing the training set, which HTIG is similarly extracted from the graph refined during the preprocessing stage. To alleviate the constraints of a limited dataset, logical equivalence transformations are strategically deployed on the nodes within the graph. This critical augmentation is executed on the preprocessed undirected graph rather than the HTIG to avoid diluting the nodes' inherent global features. This measure is pivotal to maintaining the fidelity of Trojan functionality classification. The trojan nodes' unique attributes are preserved through careful dataset augmentation via the undirected graph, thereby empowering a more robust and precise classification.

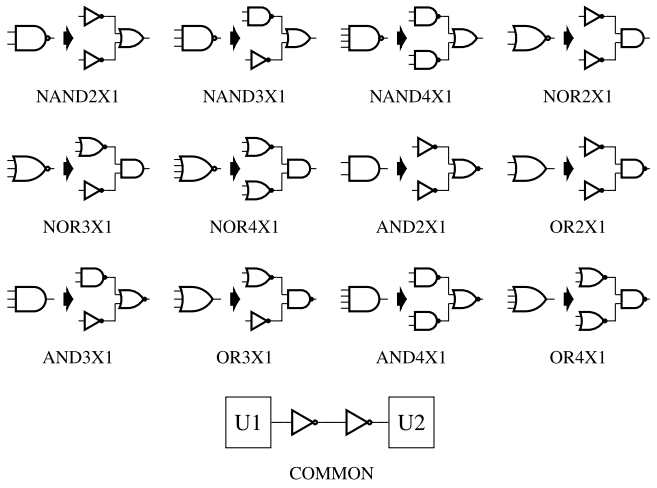


Fig. 5. Logic gate replacement patterns.

E. Dataset Augmentation via Logic Equivalence

HT multifunctional classification has two predominant challenges: 1) the effective representation of Trojan information and 2) the prevalent scarcity of datasets. The preceding section addresses the former by detailing a robust extraction method for salient Trojan information. This section pivots to introducing methodologies devised to bolster the HT dataset, laying solid groundwork for our forthcoming multiclassification experiments. Hasegawa et al. [30] originally introduced the notion of logical equivalence, a concept markedly distinct from traditional image data augmentation, owing to the unique limitations of integrated circuit augmentation. The augmentation must preserve the circuit's logical functionality to maintain its operational integrity; concurrently, extensive modifications are eschewed to circumvent any undue increase in power consumption or significant latency.

In a departure from Hasegawa et al.'s logical equivalence schema, our approach harnesses De Morgan's theorem to facilitate logical equivalence substitutions within gate-level netlists. We conceptualize De Morgan's theorem as a transformational function that prescribes an equivalent gate combination for any given gate type; this prescribed combination conservatively maintains both the count of input and output wires and the gate's logical functionality intact. Recognizing the constraints of De Morgan's theorem and the rich diversity of gate types present in netlists, we incorporate a comprehensive suite of gates, such as NAND, NOR, AND, OR, XOR, and XNOR. The granularity of our approach extends to distinguishing between gates with two, three, and four inputs, reflecting their structural and functional variances. The transformation process consequently generates an array of corresponding replacement patterns. Moreover, the four transformation strategy employs NOT gates to execute logical equivalence or the more intricate and sequential components like flip-flops and multiplexers; these replacement patterns and their implications are graphically elucidated in Fig. 5.

In our dataset augmentation process, each undirected graph set $\mathcal{G}_i$ is initially transformed into its corresponding HTIG collection, denoted as $\mathcal{H}_{\text{orig}}$. A threshold τ_i is established based

Algorithm 2 Threshold-Based Logical Equivalence Data Augmentation for a Specific Function Set $\mathcal{G}_i$.

Input: Specific function set $\mathcal{G}_i$, logical equivalence ratio α, substitution pattern p
Output: Augmented HTIG set $\mathcal{H}_{\text{aug}}$
Initialize the HTIG collection $\mathcal{H}_{\text{orig}}$ from $\mathcal{G}_i$
Compute intraclass similarity threshold τ_i for $\mathcal{H}_{\text{orig}}$
Randomly select a graph $\mathcal{G}$ from $\mathcal{G}_i$
Apply logical equivalence to $\mathcal{G}$ using p to generate $\mathcal{G}'$
Convert $\mathcal{G}'$ to HTIG $\mathcal{H}'$
Calculate the average similarity $S(\mathcal{H}')$ with all HTIGs in $\mathcal{H}_{\text{orig}}$
if $S(\mathcal{H}') \leq \tau_i$ **then**
    Add $\mathcal{H}'$ to $\mathcal{H}_{\text{orig}}$
end if
return the augmented HTIG collection $\mathcal{H}_{\text{aug}}$

on the internal cosine similarity within each $\mathcal{H}_{\text{orig}}$ collection, serving as a crucial filter to preserve the diversity of the dataset.

The graph embedding $E(\mathcal{G})$ for each graph $\mathcal{G}$ is computed utilizing the aggregate function

$$E(\mathcal{G}) = \text{Agg}\left(\left\{h_v^{(L)} : v \in V(\mathcal{G})\right\}\right). \tag{17}$$

The cosine similarity between two HTIG embeddings, $E(\mathcal{H}_a)$ and $E(\mathcal{H}_b)$, is given by

$$\text{Cosine Similarity}(E(\mathcal{H}_a), E(\mathcal{H}_b)) = \frac{E(\mathcal{H}_a) \cdot E(\mathcal{H}_b)}{\|E(\mathcal{H}_a)\| \|E(\mathcal{H}_b).\|} \tag{18}$$

The threshold τ_i for each functional class in $\mathcal{H}_{\text{orig}}$ is the mean cosine similarity across all HTIG pairs

$$\tau_i = \frac{1}{|\mathcal{G}_i|(|\mathcal{G}_i| - 1)} \sum_{\substack{\mathcal{H}_a, \mathcal{H}_b \in \mathcal{H}_{\text{orig}}, \\ a \neq b}} \text{Cosine Similarity}(E(\mathcal{H}_a), E(\mathcal{H}_b)). \tag{19}$$

For each new graph $\mathcal{G}'$ generated through logical equivalence transformations, which is transformed into HTIG $\mathcal{H}'$, its average cosine similarity with HTIGs in $\mathcal{H}_{\text{orig}}$ is computed. HTIG $\mathcal{H}'$ is included in $\mathcal{H}_{\text{orig}}$ only if its similarity $S(\mathcal{H}')$ is less than or equal to τ_i

$$S(\mathcal{H}') = \frac{1}{|\mathcal{G}_i|} \sum_{\mathcal{H} \in \mathcal{H}_{\text{orig}}} \text{Cosine Similarity}(E(\mathcal{H}'), E(\mathcal{H})). \tag{20}$$

Algorithm 2 details the threshold-based selection method for logical equivalence augmentation applied to specific functional categories of HT sets under a given logical equivalence ratio. This approach effectively mitigates the issue of dataset insufficiency, enhancing the robustness of our multifunctional classification model.

F. HT Multifunctional Classification

This section will integrate the content above to finalize the HT multifunctional classification.

A GNN model tailored for whole-graph classification at this stage as depicted in Fig. 6. This model augments the analytical scope from the targeted node classification of the initial stage to encompass the entire graph, focusing on HTIG that captures the essence of HT functionalities.

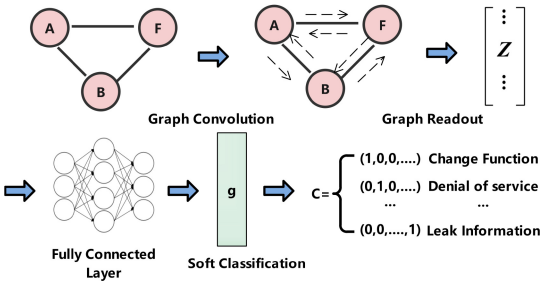


Fig. 6. Process of multifunctional classification of HTs.

The node update mechanism for our GNN model, which serves as the backbone for learning graph representations, is defined by the equation

$$h_v^{(l+1)} = \sigma\left(W^{(l)} h_v^{(l)} + \sum_{u \in N(v)} W^{(l)} h_u^{(l)}\right) \tag{21}$$

where $h_v^{(l)}$ signifies the hidden state of node v at layer l, σ represents the nonlinear activation function, and $N(v)$ denotes the set of neighboring nodes of v.

For whole-graph classification, the GNN model leverages the composite graph embedding obtained by aggregating the node embeddings from the last layer L

$$Z = \frac{1}{|V|} \sum_{v \in V} h_v^{(L)}. \tag{22}$$

This embedding is then projected through a fully connected layer and processed by a Softmax function to yield the final classification vector

$$C = \text{Softmax}(\text{FC}(Z)) \tag{23}$$

where FC is the fully connected layer and C denotes the predicted classifications.

Utilizing the enriched HTIG as the training set, the model is optimized using the Adam optimizer. The objective is to minimize the cross-entropy loss function L

$$L = -\sum_{g \in G} \sum_{k=0}^{K} Y_{gk} \ln\left(Z_{gk}\right) \tag{24}$$

here, G represents the set of graphs, K is the number of HT functionalities, Y is the ground truth, and Z is the predicted label distribution for each graph.

After training, we leverage the outcomes from stage one as inputs to refine the model further, enabling us to obtain accurate classification results for HT functionalities.

V. Experiment Results and Analysis

A. Experiments of Preprocessing

In our experiments, we selected netlists provided by Trust-Hub [37] and TRIT-TC [38] as our experimental benchmarks. The gate-level netlists involved in the experiments are

TABLE I
COMPARE THE LOCALIZATION EFFICACY OF GNN4HT ACROSS VARIOUS NETLISTS WITH ADVANCED METHODOLOGIES

Benchmark	Normal Gate	Trojan Gate	Effect Type	[39](2020)		[28](2023)		[27](2023)		GNN4HT			
				TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR	LT(ms)	ET(s)
RS232-T1000	202	13	CF	100.00	97.50	100.00	60.00	100.00	100.00	100.00	99.01	11.56	0.08
RS232-T1100	204	12	CF	80.00	97.60	92.00	68.00	100.00	99.66	100.00	99.02	9.51	0.11
RS232-T1200	202	14	CF	81.80	98.00	41.00	80.00	100.00	99.66	100.00	99.01	7.51	0.14
RS232-T1300	204	9	CF	100.00	97.60	100.00	74.00	100.00	100.00	100.00	99.01	11.51	0.04
RS232-T1400	202	13	CF	97.70	100.00	92.00	50.00	100.00	99.66	100.00	99.50	15.02	0.08
RS232-T1500	202	14	CF	100.00	99.20	71.00	82.00	100.00	100.00	100.00	99.01	7.50	0.14
RS232-T1600	202	12	CF	100.00	97.20	73.00	57.00	69.20	100.00	100.00	100.00	9.00	0.07
s35932-T100	5426	15	LI	76.40	100.00	100.00	100.00	80.00	100.00	93.33	96.19	18.02	0.60
s35932-T200	5422	16	DoS	58.30	100.00	100.00	100.00	73.30	100.00	93.75	91.83	19.02	14.81
s35932-T300	5426	36	DoS	94.40	100.00	97.00	100.00	80.80	99.95	100.00	97.97	19.02	8.69
s38417-T100	5329	12	DoS	100.00	100.00	92.00	92.00	66.70	100.00	100.00	95.62	18.02	9.34
s38417-T200	5329	15	DoS	100.00	99.80	40.00	99.90	100.00	100.00	100.00	97.24	19.02	13.18
s38417-T300	5358	15	DoS	93.60	100.00	13.00	98.00	93.30	99.99	53.33	95.13	19.87	9.12
s15850-T100	2155	27	DoS	-	-	35.00	97.00	100.00	100.00	59.26	90.35	9.51	15.69
s38584-T100	6473	9	DoS	-	-	100.00	95.00	33.30	100.00	88.89	87.72	24.02	9.12
s38584-T200	6473	83	LI	-	-	90.00	98.00	98.80	100.00	97.06	97.03	13.51	143.91
s38584-T300	6473	731	LI	-	-	13.00	98.00	75.80	100.00	94.28	99.04	12.54	2197.23
EthernetMAC10GE-T700	102047	13	CF	-	-	-	-	100.00	100.00	100.00	99.86	105.63	0.35
EthernetMAC10GE-T710	102047	13	CF	-	-	-	-	100.00	100.00	100.00	99.90	65.08	0.36
EthernetMAC10GE-T720	102047	13	CF	-	-	-	-	92.30	100.00	100.00	99.63	78.59	0.31
EthernetMAC10GE-T730	102047	13	CF	-	-	-	-	100.00	100.00	100.00	99.60	67.43	0.33
Total Average	-	-	-	90.94	98.99	78.00	85.00	88.72	**99.98**	**94.28**	97.22	-	-

* *CF: Change Function; DoS: Denial of Service; LI: Leak Information; LT: Locating Time; ET: Extracting Time*

TABLE II
BEST PARAMETERS FOR THE FIRST STAGE OF GNN4HT

Method	Model	#epochs	#layers	#lr	#dropout	#units
Proposed	GIN	100	2	0.001	0.1	95

TABLE III
COMPARING THE PERFORMANCE OF GNN4HT ON THE TRIT-TC DATASET WITH EXISTING METHODS, WITHOUT ALTERING THE MODEL PARAMETERS

Benchmark	Normal Gate	Trojan Gate	Effect Type	GNN4HT		[31](2023)	
				TPR	TNR	TPR	TNR
c2670-T000	686	5	CF	80.00	99.27	100.00	85.90
c2670-T001	686	7	CF	85.71	99.27	100.00	84.00
c2670-T002	686	5	CF	100.00	100.00	75.00	90.90
c3540-T000	1134	6	CF	100.00	99.12	100.00	93.50
c3540-T001	1134	7	CF	85.71	96.90	100.00	64.60
c3540-T002	1134	7	CF	85.71	98.77	100.00	68.00
c5315-T000	2307	9	CF	100.00	100.00	87.50	78.40
c5315-T001	2307	10	CF	100.00	100.00	77.80	86.30
c5315-T002	2307	6	CF	100.00	97.40	100.00	71.00
s1423-T000	479	5	CF	100.00	100.00	100.00	90.80
s1423-T001	479	7	CF	100.00	100.00	83.30	91.90
s1423-T002	479	9	CF	100.00	100.00	100.00	86.90
s13207-T000	2246	6	CF	100.00	97.95	100.00	96.20
s13207-T001	2246	7	CF	100.00	98.70	100.00	96.10
s13207-T002	2246	5	CF	100.00	97.95	100.00	95.50
Total Average	-	-	-	**95.81**	**99.02**	94.90	85.30

presented in Tables I and III. Twenty-one netlists are chosen from Trust-Hub and fifteen from TRIT-TC, which encompasses three types of Trojan functionalities, with the number of gates in the gate-level netlists ranging from hundreds to hundreds of thousands. The netlists from Trust-Hub were used for the experiments in both the first and second stages, while the netlists from TRIT-TC were employed to validate the effectiveness of the GIN model in the first stage.

As Section IV-B describes, each netlist from the aforementioned benchmarks is constructed into an undirected graph. In the Trust-Hub netlist collection, each node of the transformed undirected graph is built into a 95-D feature vector, where 93 dimensions represent circuit features, including a 91-D one-hot encoding of gate types and two dimensions for the in-degree and out-degree. The remaining two dimensions represent global features. Edge features are expressed in a 50-D vector, reflecting the complexity of 50 different port numbers. In the undirected graphs constructed from the TRIT-TC dataset, node features are encoded into 48-D feature vectors, with 46 dimensions for circuit features and 2 for global features. Port information is encapsulated within the 45-D edge feature vectors.

B. Experiments of HT Location

Experiment Setup: Consistent with [27], We adopted the leave-one-out cross-validation (LOOCV) method. For instance, in evaluating the RS232-T1000 netlist, the remaining netlists served as the training set, and RS232-T1000 was exclusively used as the test set. The experimental parameters of the models used in the experiment are shown in Table II. In the GIN model configuration, `learn_eps` was set to `True` to better adapt to the data. The aggregation method chosen was `sum`, which preserves the uniqueness and discriminative power of the neighboring node features. The experiments were performed on a computer with an Intel Core i7-12700H (2.30 GHz) and an NVIDIA GeForce RTX 3070 Laptop GPU (8 GB of GDDR6 memory).

Throughout the experiments, classification outcomes for each netlist are meticulously documented, including counts of true positives (TPs), true negatives (TNs), false positives (FPs), and false negatives (FNs). The associated models are evaluated using two specific machine learning metrics: 1) true positive rate (TPR, or Recall) and 2) TNR. These metrics are calculated as follows:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{TNR} = \frac{\text{TN}}{\text{FP} + \text{TN}}.$$

Tables I and III compare the stage one detection model of GNN4HT to other state-of-the-art gate-level netlist location methods. By examining Table I, it is evident that GNN4HT

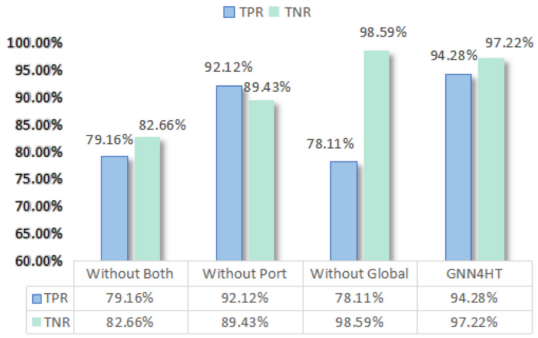


Fig. 7. Results of ablation experiments in four different cases.

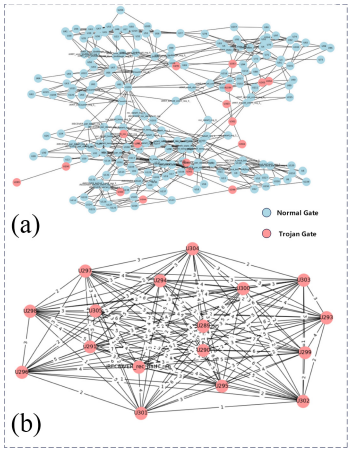


Fig. 8. Schematic of the HTIG extracted from the undirected graph converted from the RS232-T1000 with the predicted Trojan nodes. (a) Displays the node classification results in the undirected graph transformed from the netlist predicted by the GIN model. (b) Shows the result of extracting the HTIG, with edge weights representing the shortest distance between nodes in the original undirected graph.

exhibits an exceptional performance in the experiments, with TPR and TNR reaching *94.28%* and *97.22%*, respectively. The astonishing *94.28%* TPR surpasses other gate-level netlist location models, affording the best TPR, indicating our superior capability in locating Trojans. Furthermore, the exceptional results are not derived through leave-one-out split, where one sample is reserved as both the test and validation set. This method is often considered in literature due to its ability to showcase a model's optimal performance. In [28], they discuss an optimal scenario where leave-one-out validation yields precision with a TPR of up to *96%* and a TNR of *98%*. In this experiment, we do not endorse this unrealistic scenario because our localization performance is already exceedingly close to what they deem optimal in practical applications. This success can be attributed to several factors: using global features to help the graph model more accurately identify Trojans, utilizing port information to aid the model's convergence, and employing the GIN model to provide individual mapping functions for each node. These advantages are validated in the ablation study. Due to the parallel computing acceleration of the GPU, the model can locate the trojans of the netlist containing 10^5 gates within 105.63ms.

Ablation Study: This study was designed to elucidate the contributions of global features and port information encoding in HT location. To this end, we executed ablation studies under three distinct configurations, each juxtaposed against our standard research model. The ablation configurations are as follows.

1) *"Without Global Features":* This setting solely relies on port information, excluding global features.
2) *"Without Port Information":* We focused on global features, omitting the port encoding.
3) *"Without Global Features & Port Information":* Both global features and port information were omitted, providing a baseline for comparison.
4) *"GNN4HT":* An optimal localization approach integrating global features and port information.

The outcomes of these studies, depicted in Fig. 7, it can be observed that without the aid of port information and global features, GIN fails to effectively learn the distinctions between Trojan and normal gates in the graph, resulting in TPR and TNR. With the addition of port information, GIN's sensitivity to normal gates improves, yet it still struggles to accurately identify Trojan gates. However, incorporating global features significantly enhances GIN's ability to effectively locate Trojan gates.

To verify the generalizability of our method, we conducted Trojan localization on netlists using the TRIT-TC dataset under the same experimental settings, with results shown in Table III, GNN4HT achieved the best performance to date, confirming the robustness and effectiveness of our model. A TPR of *95.81%* and a TNR of *99.02%* surpass existing methods.

Upon completion of the localization process, the results are encapsulated within HTIGs, which serve as a collective representation of the identified Trojan gates and their functionalities. As depicted in Fig. 8, in the case of RS232-1000, the locations of Trojans identified by the GIN model were transformed into HTIG. They served as the test set for the final HT multifunctional classification model. We reported the time taken to extract the HTIG after locating the Trojans. Most netlists could be processed within 3 min. However, the netlist s38584-T300 contains more Trojans, which requires more time to traverse the graph and calculate the hop counts between Trojan gates.

C. Experiments of HT Multifunctional Classification

In this section, the experimental results of dataset augmentation based on logical equivalence are presented, as well as the training and final functional classification results of the second-stage model. To verify the feasibility of GNN4HT, we synthesized the RTL benchmarks into gate netlists to test the effectiveness of the second stage model.

Tables IV and V summarize the classifications of the netlists involved in three types of functionalities as well as the interclass similarity of the functionalities associated with the extracted HTIGs. Clearly, dataset augmentation is necessary for each type of functionality due to the common issues of insufficient datasets and overly high-interclass similarity, especially in the category of leak information, where the

TABLE IV
SUMMARY OF TROJAN FUNCTIONALITIES AND CORRESPONDING DATASETS

Trojan Functionality	Datasets
Change Function	RS232-T1000,RS232-T1100,RS232-T1200,RS232-T1300,RS232-T1400,RS232-T1500,RS232-T1600, EthernetMAC10GE-T700, EthernetMAC10GE-T710,EthernetMAC10GE-T720, EthernetMAC10GE-T730
Denial of Service	s35932-T200,s35932-T300,s38417-T100,s38417-T200,s38417-T300,s15850-T100,s38584-T100
Leak Information	s35932-T100, s38584-T200, s38584-T300

TABLE V
SUMMARY OF TROJAN FUNCTIONALITIES AND SIMILARITY MEANS

Trojan Functionality	Similarity Mean
Change Function	0.6333
Denial of Service	0.4915
Leak Information	0.5167

TABLE VI
BEST PARAMETERS FOR THE SECOND STAGE OF GNN4HT

Method	Model	#epochs	#layers	#lr	#dropout	#units
Proposed	GNN	50	2	0.01	0.1	64

TABLE VII
PERFORMANCE OF MULTIFUNCTIONAL CLASSIFICATION MODEL ON 21 BENCHMARKS WITH AND WITHOUT LOGIC EQUIVALENCE EXPANSION

Benchmark	Effect Type	Post Aug. Count	Ori. Dataset	Aug. Dataset	Time (ms)
RS232-T1000	CF	65	✔	✔	0.245
RS232-T1100	CF	62	✔	✔	0.239
RS232-T1200	CF	65	✔	✔	0.222
RS232-T1300	CF	61	✔	✔	0.233
RS232-T1400	CF	65	✔	✔	0.222
RS232-T1500	CF	65	✔	✘	0.199
RS232-T1600	CF	65	✔	✔	0.219
S15850-T100	DoS	69	✘	✔	1.518
S35932-T100	LI	71	✘	✘	1.211
S35932-T200	DoS	66	✔	✔	1.677
S35932-T300	DoS	68	✘	✔	2.292
S35417-T100	DoS	68	✘	✔	1.212
S35417-T200	DoS	68	✔	✔	1.666
S35417-T300	DoS	68	✘	✔	1.512
S38584-T100	DoS	69	✘	✘	1.211
S38584-T200	LI	65	✘	✔	3.287
S38584-T300	LI	66	✘	✔	5.245
EthernetMAC10GE-T700	CF	65	✘	✔	0.311
EthernetMAC10GE-T710	CF	65	✔	✔	0.302
EthernetMAC10GE-T720	CF	65	✔	✔	0.311
EthernetMAC10GE-T730	CF	65	✘	✘	0.211
Average		**66**	52.38%	**80.95%**	-

availability of only three datasets severely hampers the model's learning capabilities.

Experiment Setup: Similar to the experimental setup in the first stage, leave-one-out validation is used for each tested netlist, ensuring that each tested netlist remains unknown. For instance, if the netlist RS232-T1000 is considered as the test netlist, the first-stage model will be trained on the remaining 20 netlists to locate Trojans and extract them as HTIGs. The second-stage functional classification model will also not see this netlist. The remaining 20 netlists will be used as the training set, with logical augmentation performed only on these 20 netlists. Subsequently, they will be extracted as HTIG and used for model training, and finally, the functional classification will be performed on the test netlist. Regarding dataset augmentation, we selected a range of logical equivalence ratios from 5% to 95%, with increments of 5%. For each functionality, at each ratio, we randomly selected a graph from the collection to perform augmentation based on logical equivalence. The parameters related to the functional classification model are shown in Table VI.

Table VII shows the number of netlists used as the training set, which are the remaining 20 netlists processed through logical equivalence when each netlist was used as the test set and the classification results for the original dataset of 20 netlists and the augmented dataset.

With only 20 netlists used as the training set, the accuracy rate for Trojan functionality classification was merely 52.38%. In the category of changing function, most netlists were correctly classified, however, in the denial of service category, due to low-intraclass similarity and an insufficient dataset, the model could only correctly identify the functionality of two netlists. In the leak information category, not a single netlist was correctly identified due to only having two netlists in this category when one was used as the test set, which prevented the model from learning sufficient information about the class. Additionally, the results of Trojan localization in the first stage also impact the function classification outcomes. Even though we achieved a high TPR and effectively located Trojans in the first stage, the misclassification of normal gates as Trojan gates, followed by their extraction into HTIGs, also affected the representation of Trojan functions.

After logical equivalence processing, the average number of training netlists per test netlist increased to *66*, significantly boosting the training data volume to *3.14* times the original amount. Moreover, only netlists that passed a threshold selection could be added to the dataset, effectively reducing interclass similarity. By applying logical equivalence to Trojan gates, we enhanced the different representations of HTIGs under the same function, which increased the model's robustness and sensitivity to Trojan-related information, the model could better learn the Trojan functionality represented by HTIGs. In both the denial of service and leak information categories, only one netlist's functionality was not correctly identified, and the accuracy of function classification reached *80.95%*. Additionally, we have reported the detection times of our multifunctional classification model. Our model is capable of detecting the functionality of any netlist that has been located in the first stage and extracted as a HTIG within *5.245 ms*.

To verify the effectiveness of our second stage model, we used an open-source RTL synthesis tool called Yosys [40]. We synthesized RTL designs into gate-level netlists and tested them. We expanded the dataset through logical equivalence on 21 netlists from Trust-Hub and trained the entire set. To prevent Yosys from optimizing away the relevant HT code, we specified in Yosys not to delete any user-defined cells for netlists that lacked Trojan information after synthesizing to gate-level, ensuring that the Trojan gates are preserved. In these RTL codes, we marked all modules related to Trojans, and upon conversion to gate-level, all gates from these modules were treated as Trojan gates. These gates were then extracted as HTIGs to form the test set.

TABLE VIII
COMPARE THE PERFORMANCE OF GNN4HT WITH THE STATE-OF-THE-ART HT DETECTION

		Stage 1					Stage 2			
Paper (Year)	**Method**	**TPR**	**TNR**	**Location**	**Golden free**	**Unknown Trojan Detection**	**Precision (gate-level)**	**Precision (RTL)**	**Complete Detection**	**Unknown Trojan Detection**
[25] (2021)	Graph Neural Network	**84%**	*NA*	✘	✔	✔	*NA*	*NA*	✘	*NA*
[26] (2021)	Graph Neural Network	*NA*	*NA*	✘	✔	✔	*NA*	*NA*	✘	*NA*
[2] (2021)	Graph Neural Network (Reverse Engineering)	*NA*	*NA*	✔	✘	✔	*NA*	*NA*	✔	✔
[28] (2023)	Graph Neural Network	**78%**	**85%**	✔	✔	✘	*NA*	*NA*	✘	*NA*
[39] (2020)	XGBoost (Hybrid detection)	**90.94%**	**98.99%**	✔	✔	✔	*NA*	*NA*	✘	*NA*
[31] (2023)	Graph Neural Network	**83.5%**	**94.6%**	✘	✔	✔	*NA*	*NA*	✘	*NA*
[27] (2023)	Graph Neural Network	**88.72%**	**99.98%**	✔	✔	✔	*NA*	*NA*	✘	*NA*
Ours	Graph Neural Network	**94.28%**	**97.22%**	✔	✔	✔	**80.95%**	**62.96%**	✔	✔

TABLE IX
BEST PARAMETERS FOR THE SECOND STAGE OF GNN4HT FOR RTL DESIGNS

Method	Model	#epochs	#layers	#lr	#dropout	#units
Proposed	GNN	10	2	0.1	0.01	128

TABLE X
PERFORMANCE OF MULTIFUNCTIONAL CLASSIFICATION MODEL ON RTL BENCHMARKS WITH AND WITHOUT LOGIC EQUIVALENCE EXPANSION

Benchmark	Trojan Function	Original Dataset	Augmented Dataset
AES-T100	Leak Information	✔	✔
AES-T200	Leak Information	✘	✔
AES-T300	Leak Information	✘	✔
AES-T400	Leak Information	✘	✔
AES-T500	Denial of Service	✔	✔
AES-T600	Leak Information	✘	✔
AES-T700	Leak Information	✘	✔
AES-T800	Leak Information	✘	✔
AES-T900	Leak Information	✘	✘
AES-T1000	Leak Information	✘	✘
AES-T1100	Leak Information	✘	✔
AES-T1200	Leak Information	✘	✔
AES-T1300	Leak Information	✘	✔
AES-T1400	Leak Information	✘	✔
AES-T1500	Leak Information	✘	✔
AES-T1600	Leak Information	✘	✔
AES-T1700	Leak Information	✘	✔
AES-T1800	Denial of Service	✔	✘
AES-T1900	Denial of Service	✘	✘
AES-T2000	Leak Information	✘	✔
AES-T2100	Leak Information	✘	✔
AES-T2300	Change Function	✘	✘
AES-T2400	Change Function	✘	✘
AES-T2500	Change Function	✘	✘
AES-T2600	Change Function	✘	✘
AES-T2700	Change Function	✘	✘
AES-T2800	Change Function	✔	✘
Average		14.81%	**62.96%**

The parameters and results of the RTL designs multifunctional classification model are shown in Tables IX and X. It can be observed that before applying logical equivalence, the model's accuracy was merely *14.81%*. This low accuracy is due to the insufficiency of the dataset and because our test dataset, aimed at including more Trojan information, contained a lot of redundant information. In contrast, the Trojan gates in the gate-level netlists on Trust-Hub are inserted directly, designed to be covert and hard to detect without redundant information. Logical equivalence resolved this issue by expanding the dataset, which improved the model's robustness and increased its accuracy by *48.15%*. The training set did not contain RTL designs and was trained only with gate-level netlists. Testing conducted by converting RTL to gate-level netlists also achieved an accuracy of *62.96%*, demonstrating the viability of this method.

Table VIII presents the comparison results of our model with existing advanced HT detection technologies. It can be observed that existing models were mainly focused on detecting Trojan components in netlists (i.e., stage one). Yasaei et al. [25], [26] both developed models for detecting Trojans at RTL and gate-level, but these models can only determine whether the entire netlist contains Trojans without the ability to detect specific Trojan components. Among the models that can locate Trojans, GNN4HT achieved the highest TPR of *94.28%*, and our model has also demonstrated excellent performance in other datasets. GNN-RE [2] achieved good results in locating gates and classifying their functions in gate-level netlists, but it cannot classify the functions of Trojans. Building on the results of the first stage, GNN4HT introduced the innovative HTIG. It expanded the dataset using logical equivalence, achieving accuracies of *80.95%* and *62.96%* at gate-level and RTL, respectively. This introduces a new direction for HT detection and marks a significant advancement in hardware security.

VI. CONCLUSION

In this study, we introduced a two-stage model based on GNNs that effectively addresses the multifunctional classification of HTs. GNN4HT addresses the three main challenges in the multifunctional classification of HTs: 1) high-precision localization; 2) structural representation of HT functionalities; and 3) dataset insufficiency. In the first stage, GNN4HT employs the unique node computational graph mapping method and global structure of GIN to achieve a high TPR of *94.28%*, ensuring precise localization. Furthermore, its

performance on the TRIT-TC dataset demonstrates its generalizability, surpassing existing HT localization methods. To better represent the interactions between Trojan gates and the expression of HT functionalities, we introduced the HTIG and encapsulated the localization results into HTIG. In this stage, GNN4HT adopts a threshold-based logic equivalence data augmentation method to enrich the dataset diversity and alleviate the issue of dataset insufficiency. GNN4HT trains a robust GNN whole-graph classification model and further classifies the functionalities of the located HTs, achieving a *80.95%* accuracy rate at gate-level and *62.96%* at RTL. This is the first time in the world that Trojan functionalities have been classified, representing a significant breakthrough with important practical implications and broad application prospects.

REFERENCES

[1] X. Chen et al., "Hardware trojan detection in third-party digital intellectual property cores by multilevel feature analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 7, pp. 1370–1383, Jul. 2018.

[2] L. Alrahis et al., "GNN-RE: Graph neural networks for reverse engineering of gate-level netlists," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 8, pp. 2435–2448, Aug. 2022.

[3] C. Liu, P. Cronin, and C. Yang, "Securing cyber-physical systems from hardware trojan collusion," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 3, pp. 655–667, Sep. 2020.

[4] A. K. M. J. Alam Majumder, C. B. Veilleux, and J. D. Miller, "A cyber-physical system to detect IoT security threats of a smart home heterogeneous wireless sensor node," *IEEE Access*, vol. 8, pp. 205989–206002, 2020.

[5] A. Jain, Z. Zhou, and U. Guin, "Survey of recent developments for hardware trojan detection," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2021, pp. 1–5.

[6] C. Dong, G. He, X. Liu, Y. Yang, and W. Guo, "A multi-layer hardware trojan protection framework for IoT chips," *IEEE Access*, vol. 7, pp. 23628–23639, 2019.

[7] A. Basak, S. Bhunia, T. Tkacik, and S. Ray, "Security assurance for system-on-chip designs with untrusted IPs," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 1515–1528, 2017.

[8] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of Hardware Trojans and maliciously affected circuits," *J. Hardw. Syst. Secur.*, vol. 1, pp. 85–102, Mar. 2017.

[9] S. Charles, V. Bindschaedler, and P. Mishra, "Digital watermarking for detecting malicious intellectual property cores in NoC architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 7, pp. 952–965, Jul. 2022.

[10] N. Vashistha, H. Lu, Q. Shi, D. L. Woodard, N. Asadizanjani, and M. M. Tehranipoor, "Detecting Hardware Trojans using combined self-testing and imaging," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 6, pp. 1730–1743, Jun. 2022.

[11] G. Piliposyan and S. Khursheed, "PCB hardware trojan run-time detection through machine learning," *IEEE Trans. Comput.*, vol. 72, no. 7, pp. 1958–1970, Jul. 2023.

[12] R. Lu, H. Shen, Y. Su, H. Li, and X. Li, "GramsDet: Hardware trojan detection based on recurrent neural network," in *Proc. IEEE 28th Asian Test Symp. (ATS)*, 2019, pp. 111–115.

[13] H. Salmani, "Gradual-N-Justification (GNJ) to reduce false-positive hardware trojan detection in gate-level netlist," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 4, pp. 515–525, Apr. 2022.

[14] T. Kurihara, K. Hasegawa, and N. Togawa, "Evaluation on hardware-trojan detection at gate-level IP cores utilizing machine learning methods," in *Proc. IEEE 26th Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, 2020, pp. 1–4.

[15] Y. Wang, P. Liu, X. Han, and Y. Jiang, "Hardware trojan detection method for inspecting integrated circuits based on machine learning," in *Proc. 22nd Int. Symp. Qual. Electron. Design (ISQED)*, 2021, pp. 432–436.

[16] S. Faezi, R. Yasaei, and M. A. Al Faruque, "HTnet: Transfer learning for golden chip-free hardware trojan detection," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2021, pp. 1484–1489.

[17] Y. Wang, T. Han, X. Han, and P. Liu, "Ensemble-learning-based Hardware Trojans detection method by detecting the trigger nets," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2019, pp. 1–5.

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

[19] Z. Huang, Q. Wang, Y. Chen, and X. Jiang, "A survey on machine learning against hardware Trojan attacks: Recent advances and challenges," *IEEE Access*, vol. 8, pp. 10796–10826, 2020.

[20] R. Yasaei, L. Chen, S.-Y. Yu, and M. A. Al Faruque, "Hardware Trojan detection using graph neural networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, May 26, 2022, doi: 10.1109/TCAD.2022.3178355.

[21] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," in *Proc. IEEE 23rd Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, 2017, pp. 227–232.

[22] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists based on machine learning," in *Proc. IEEE 22nd Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, 2016, pp. 203–206.

[23] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to Hardware-Trojan detection using random forest classifier," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2017, pp. 1–4.

[24] H. Shen, H. Tan, H. Li, F. Zhang, and X. Li, "LMDet: A 'naturalness' statistical method for Hardware Trojan detection," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 4, pp. 720–732, Apr. 2018.

[25] R. Yasaei, S.-Y. Yu, and M. A. Al Faruque, "GNN4TJ: Graph neural networks for Hardware Trojan detection at register transfer level," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2021, pp. 1504–1509.

[26] S.-Y. Yu, R. Yasaei, Q. Zhou, T. Nguyen, and M. A. Al Faruque, "HW2VEC: A graph learning tool for automating hardware security," in *Proc. IEEE Int. Symp. Hardw. Orient. Secur. Trust (HOST)*, 2021, pp. 13–23.

[27] K. Hasegawa, K. Yamashita, S. Hidano, K. Fukushima, K. Hashimoto, and N. Togawa, "Node-wise hardware trojan detection based on graph learning," *IEEE Trans. Comput.*, early access, May 25, 2023, doi: 10.1109/TC.2023.3280134.

[28] H. Lashen, L. Alrahis, J. Knechtel, and O. Sinanoglu, "TrojanSAINT: Gate-level netlist sampling-based inductive learning for Hardware Trojan detection," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2023, pp. 1–5.

[29] L. Alrahis and O. Sinanoglu, "Graph neural networks for hardware vulnerability analysis—Can you trust your GNN?" in *Proc. IEEE 41st VLSI Test Symp. (VTS)*, 2023, pp. 1–4.

[30] K. Hasegawa, S. Hidano, K. Nozawa, S. Kiyomoto, and N. Togawa, "Data augmentation for machine learning-based Hardware Trojan detection at gate-level netlists," in *Proc. IEEE 27th Int. Symp. Line Test. Robust Syst. Design (IOLTS)*, 2021, pp. 1–4.

[31] K. Hasegawa, S. Hidano, K. Nozawa, S. Kiyomoto, and N. Togawa, "R-HTDetector: Robust Hardware-Trojan detection based on adversarial training," *IEEE Trans. Comput.*, vol. 72, no. 2, pp. 333–345, Feb. 2023.

[32] K. Madduri, D. Ediger, K. Jiang, D. A. Bader, and D. Chavarria-Miranda, "A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, 2009, pp. 1–8.

[33] G. L. D. l. Sarracén and P. Rosso, "Offensive keyword extraction based on the attention mechanism of BERT and the eigenvector centrality using a graph representation," *Pers. Ubiquitous Comput.*, vol. 27, pp. 45–57, Feb. 2023.

[34] X. Wang and M. Zhang, "How powerful are spectral graph neural networks," in *Proc. 39th Int. Conf. Mach. Learn.*, 2022, pp. 23341–23362.

[35] H. Salmani, "COTD: Reference-free Hardware Trojan detection and recovery based on controllability and observability in gate-level netlist," *IEEE Trans. Inf. Forensics Security*, vol. 12, pp. 338–350, 2017.

[36] A. Mondal, S. Karmakar, M. H. Mahalat, S. Roy, B. Sen, and A. Chattopadhyay, "Hardware Trojan detection using transition probability with minimal test vectors," *ACM Trans. Embed. Comput. Syst.*, vol. 22, no. 1, pp. 1–21, Oct. 2022.

[37] "Trust-HUB."Accessed: May 15, 2024. [Online]. Available: https://trust-hub.org

[38] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, 2013, pp. 471–474.
[39] S. Li, Y. Zhang, X. Chen, M. Ge, Z. Mao, and J. Yao, "A XGBoost based hybrid detection scheme for gate-level Hardware Trojan," in *Proc. IEEE 9th Joint Int. Inf. Technol. Artif. Intell. Conf. (ITAIC)*, 2020, pp. 41–47.
[40] C. Wolf. "Yosys open synthesis suite." Accessed: May 15, 2024. [Online]. Available: http://www.clifford.at/yosys/

Lihan Chen received the B.S. degree from the College of Computer and Data Science, Fuzhou University, Fuzhou, China, in 2022, where he is currently pursuing the master's degree in artificial intelligence.

His research interests include hardware security and graph neural networks.

Chen Dong (Member, IEEE) received the B.S. and M.S. degrees from the College of Computer and Data Science, Fuzhou University, Fuzhou, China, in 2002 and 2005, respectively, and the Ph.D. degree in computer science from the Computer School, Wuhan University, Wuhan, China, in 2011.

She was a Visiting Researcher with the University of California at Los Angeles, Los Angeles, CA, USA, from 2015 to 2016. She is currently an Associate Professor with the College of Computer and Data Science, Fuzhou University. Her research interests include artificial intelligence, hardware security, intelligent computing, and integrated circuit physical design.

Qiaowen Wu received the B.S. degree in electronic and information engineering from the Wuhan University of Science and Technology, Wuhan, China, in 2019. She is currently pursuing the master's degree in computer software and theory with the College of Computer and Data Science, Fuzhou University, Fuzhou, China.

Her research interests include hardware security and artificial intelligence.

Ximeng Liu (Senior Member, IEEE) received the B.S. degree in electronic engineering and the Ph.D. degree from Xidian University, Xi'an, China, in 2010 and 2015, respectively.

He is currently a Full Professor with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. He was a Research Fellow with the School of Information System, Singapore Management University, Singapore. He has published more than 250 papers on the topics of cloud security and big data security, including papers in the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON SERVICE COMPUTING, and the IEEE INTERNET OF THINGS JOURNAL. His research interests include cloud security, applied cryptography, and big data security.

Dr. Liu was awarded the ACM SIGSAC China Rising Star Award.

Xiaodong Guo received the B.S. degree in software engineering from the College of Computer and Science, Huaqiao University, Quanzhou, China, in 2020. He is currently pursuing the M.S. degree with the College of Computer and Big Data Science, Fuzhou University, Fuzhou, China.

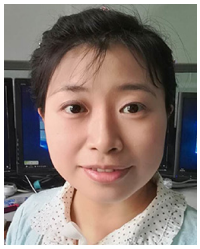
His research interests include IC security, biochip security, and design.

Zhenyi Chen received the B.S. and M.S. degrees in computer science from the College of Computer and Data Science, Fuzhou University, Fuzhou, China, in 2000 and 2005, respectively, and the first Ph.D. degree in applied computer technology from the Computer School, Wuhan University, Wuhan, China, in 2012. He is currently pursuing the second Ph.D. degree with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL, USA.

His research interests include swarm intelligence, optimization, and parallel computing.

Hao Zhang received the B.S. and M.S. degrees in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2002 and 2006, respectively, and the Ph.D. degree in applied mathematics from Fuzhou University, Fuzhou, China, in 2015.

He is currently an Associate Professor with the College of Computer and Data Science, Fuzhou University. He is also a Visiting Scholar with the Department of Electrical and Computer Engineering, Dalhousie University, Halifax, NS, Canada. His research interests include VLSI physical design and machine learning.



Yang Yang (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees from Xidian University, Xi'an, China, in 2006 and 2011, respectively.

She is a Full Professor with the College of Computer Science and Big Data, Fuzhou University, Fuzhou, China. She is also a Research Fellow with the School of Computing and Information System, Singapore Management University, Singapore. She has published more than 60 papers in IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON SERVICES COMPUTING, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. Her research interests are in the area of information security and privacy protection.