

Article

Directed and Resolution-Adaptive Louvain Community Method for Hardware Trojan Detection and Localization in Gate-Level Netlists

Hongxu Gao ¹, Dong Ding ^{1,*}, Cai Zhen ², Xin Liu ³, Yu Li ³, Jinping Li ⁴, Yuning Zhao ¹ and Quan Wang ¹

¹ School of Computer Science and Technology, Xidian University, Xi'an 710071, China

² School of Computer Science and Technology, North University of China, Taiyuan 030051, China

³ School of Electrical Engineering, Xi'an Jiaotong University, Xi'an 710049, China

⁴ School of Computer Science and Technology, Xi'an University of Posts & Telecommunications, Xi'an 710121, China

* Correspondence: dingdong@xidian.edu.cn; Tel.: +86-157-6362-2701

Abstract

The increasing complexity of modern gate-level circuits significantly degrades the efficiency of existing Hardware Trojan detection methods. Community partitioning is an efficient structural decomposition technique to address efficiency and scalability issues, yet current community-based detection schemes rely primarily on undirected graph modeling. To address these issues, we propose an improved structure-aware community detection method for gate-level netlists, aiming to enhance the detection and localization capabilities of small-scale Hardware Trojans. First, an expanded dataset with structural diversity of clean and Trojan-inserted circuits is constructed by extending Trust-Hub benchmark circuits. Then, a directed and resolution-adaptive Louvain community detection algorithm is proposed—by introducing directed modularity, resolution parameters, and logic-gate semantic weighting, fine-grained community partitioning is achieved. On this basis, topological, functional, and anomaly features are extracted from community subgraphs, and a detection framework is built by combining graph neural networks and traditional detection models. All experiments are conducted on a unified platform equipped with an Intel (R) Core (TM) i7-10750H processor and an NVIDIA GeForce RTX 2060 GPU. Experimental results show that compared with configurations using the original Louvain partitioning and traditional features, the proposed method achieves significant improvements in both detection accuracy and localization capability. After introducing the improved community partitioning and feature design, the optimal model (CommunityGAT) yields a 3.3% increase in TPR and a 10.8% increase in ALC, verifying the method's effectiveness in detecting small-scale concealed Trojans.



Academic Editor: Paris Kitsos

Received: 7 January 2026

Revised: 12 February 2026

Accepted: 17 February 2026

Published: 28 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

Keywords: Hardware Trojan; gate-level netlist; DAG; improved Louvain community detection algorithm; feature extraction

1. Introduction

The globalized division of labor models in the semiconductor industry has significantly increased the security risks of the integrated circuit (IC) supply chain, which has caused a serious impact on the security of the Internet of Things (IoT). As a type of malicious circuit with strong concealment and harsh trigger conditions in the underlying hardware of IoT

devices, Hardware Trojans (HTs) may be implanted into chips through untrusted third-party links. After long-term latent periods, HTs can pose severe threats to various critical information systems by manipulating logic, stealing data, or destroying functions [1]. Since Hardware Trojans usually remain dormant under normal operating conditions and are difficult to detect through traditional functional testing methods, research on effective Trojan detection and localization at the gate-level netlist stage has gradually attracted widespread attention.

Hardware Trojan detection methods are roughly classified into register-transfer-level (RTL) detection methods, gate-level netlist detection methods, physical layer detection methods, and dynamic behavior and side-channel detection methods, depending on the design abstraction level and detection stage. Gate-level netlist detection methods have received considerable attention because they directly act on the synthesized logical structure and can characterize actual gate-level connection relationships and control logic. Gate-level netlist detection methods can be further divided into pre-silicon and post-silicon stages. The pre-silicon stage only performs an analysis based on gate-level netlists and their derived structures, without relying on physical measurements. In recent years, with the development of graph representation learning technology, modeling gate-level netlists as graph structures for learning has gradually become an important direction for pre-silicon detection [2]. Despite notable progress in Hardware Trojan detection for gate-level netlists in recent years, most existing methods (e.g., structural statistics and graph neural network (GNN)-based whole-graph reasoning) directly perform modeling and inference on the entire netlist graph [3]. To address this issue, community-level modeling and learning methods have started to gain attention, but current community partitioning-based Hardware Trojan detection still faces several challenges. At the dataset level, current research highly relies on public benchmark datasets such as Trust-Hub [4], but the number of available Trojan samples for each type of circuit is limited, which is difficult to support the stable training of deep learning models. Meanwhile, these Trojans are mostly manually designed with relatively fixed structural patterns and trigger logic, making it hard to cover complex and variable real-world attack scenarios. In terms of structural modeling, existing community partitioning methods are mostly based on the traditional undirected graph assumption, ignoring the inherent signal flow characteristics in gate-level netlists. Moreover, classic modularity optimization algorithms are generally limited by the resolution problem and tend to generate large communities, thereby concealing stealthy Trojans hidden in small-scale substructures. In terms of feature extraction and representation, existing methods mostly rely on simple structural statistical features [5], which cannot fully characterize the complex behaviors of Trojans.

To address the issues of limited data scale, coarse-grained structural modeling, and insufficient feature expression capability in existing gate-level Hardware Trojan detection, this paper systematically improves the detection and localization capabilities for small-scale, stealthy Hardware Trojans in gate-level netlists from three aspects: data construction, structural partitioning, and feature modeling. The contributions are as follows:

- An expanded dataset with structural diversity is constructed based on classical small-to-medium-scale Trust-Hub benchmark circuits (c880, s27, s1488). Structural diversity is enhanced by generating functionally equivalent clean circuits through multiple logic optimization sequences, while Trojan-infected circuits are created by integrating parameterized Trojan templates. This strategy effectively mitigates the issues of limited sample size and homogeneous Trojan patterns in public datasets.
- Gate-level netlists are uniformly modeled as directed acyclic graphs (DAGs), and a direction-aware Louvain community partitioning algorithm with adjustable resolution is proposed. By incorporating signal flow information and controllable community

granularity, the proposed algorithm enables fine-grained decomposition of gate-level circuits, facilitating accurate isolation of small-scale and stealthy Trojan subcircuits.

- A community-level feature extraction framework is developed, combining topological characteristics, functional attributes, and anomaly deviation metrics. Furthermore, graph neural networks are employed to learn embedded representations of community subgraphs, yielding detection features with improved discriminability and generalization capability for Hardware Trojan detection and localization.

The remainder of this paper is organized as follows: Section 2 reviews the related research work on Hardware Trojan detection and graph-based methods. Section 3 introduces the construction process of the experimental dataset, the modeling method of converting gate-level netlists into directed acyclic graphs, elaborates on the proposed directed and resolution-adaptive Louvain community detection algorithm and the community-based feature extraction method, and then presents the design of the detection model. Section 4 provides the experimental results and analysis, and the final section summarizes the full paper and prospects future work.

2. Related Works

In recent years, Hardware Trojan detection methods can be roughly categorized into four types based on the design abstraction level and the source of detection information: register-transfer-level (RTL) detection methods [6–8], gate-level netlist detection methods [9], layout-level or physical layer detection methods [10], and detection methods based on dynamic behavior and side-channel analysis [11]. RTL-level methods primarily target high-level hardware descriptions and achieve early detection through rule checking, information flow analysis [12], or formal verification [13], but they have limited ability to characterize low-level implementation details. Gate-level netlist detection methods [14–16] act directly on the synthesized circuit structure, can fully reflect the actual logical implementation of the circuit, and achieve a good balance between detection accuracy and data accessibility, thus attracting widespread attention in recent years. Layout-level and physical layer detection usually rely on placement and routing information or post-silicon measurement data, offering high detection accuracy but suffering from high costs and significant application limitations. Methods based on side-channel and dynamic behavior utilize power consumption, timing, or electromagnetic characteristics for analysis [17], which are mainly applicable to the post-silicon stage and have relatively limited applicability in the design phase. Therefore, this study focuses on the application of pre-silicon Hardware Trojan detection technology at the gate-level netlist level, which can not only effectively achieve structural fidelity but also meet scalability requirements.

In the field of pre-silicon gate-level netlist Hardware Trojan detection, existing methods can be roughly divided into several categories: those based on global structural statistics, whole-graph graph neural networks (GNNs), subgraph-level GNNs, and community-level modeling. Early methods mostly take the complete netlist as input, relying on manually designed structural statistical features combined with traditional machine learning models for detection [18], which are simple to implement but have limited expressive ability for small-scale, stealthy Trojans and low localization accuracy. Subsequently, whole-graph GNN methods [19,20] improved detection performance through automatic structural representation learning, but their input remains the entire circuit graph, where Trojan structures are easily diluted by a large amount of normal logic, and precise localization is difficult to achieve. To enhance local perception capability, subgraph-level GNNs [21] introduce local structural modeling, but subgraph partitioning relies on heuristic rules or sliding window strategies, suffering from unstable partitioning and high computational complexity. In recent years, community-level GNNs have shown potential in balancing detection and

localization by first performing structural decomposition on the netlist and then conducting detection at the community level. However, existing community-based methods are mostly based on undirected graph modeling [22,23], with uncontrollable community granularity, and ignore logic gate semantics and signal directionality, making it still difficult to accurately separate compact small-scale Trojan logic.

In summary, current research on gate-level Hardware Trojan detection still faces the following challenges: first, the inherent signal directionality and functional hierarchy in gate-level netlists are not fully modeled; second, there is a lack of mechanisms for automated and adaptive substructure partitioning of circuits, leading to small-scale Trojan features being easily submerged [24]; third, the support for structured localization of Trojans [25] is insufficient. To address these issues, this paper proposes a Louvain community detection method integrating directional information and adjustable resolution, which realizes automatic community partitioning directly on gate-level directed acyclic graphs (DAGs), and then performs feature learning and anomaly detection at the community granularity, aiming to improve the accuracy and localizability of Trojan detection.

3. HT Detection Method

This paper first preprocesses gate-level netlists by uniformly modeling them as standardized directed acyclic graphs (DAGs) to explicitly characterize signal flow relationships within circuits. Building upon this foundation, an improved Louvain community detection algorithm is employed to partition the circuit graph. This algorithm enhances modeling capabilities for gate-level circuit structure and functional characteristics by introducing directed modularity functions, resolution parameters, and logic gate semantic weighting strategies. Following community partitioning, the original circuit is decomposed into several structurally compact, functionally coherent community subgraphs. For these subgraphs, this paper enriches the feature system beyond traditional extraction methods, characterizing potential Hardware Trojan behavior patterns across three dimensions: topological features, functional features, and anomaly metrics. Specifically, topological features focus on structural attributes within communities, such as the proportion of XOR/NAND gates, equivalent feedback paths, and maximum path depth. Functional features describe the dynamic behavior of subcircuits through output entropy and power variance descriptors under random or specific stimuli. Anomaly metrics utilize KL divergence and Jensen–Shannon distance to evaluate the deviation in feature distributions between communities and their corresponding clean circuits. Finally, these community-level features are fed into models such as CommunityGNN, CommunityGAT, Graph-Based Detector, and Local Outlier Factor to achieve effective detection and precise localization of Hardware Trojans. The overall detection methodology is illustrated in the figure below Figure 1.

3.1. Expanded Circuit Dataset

In practical model training, the Trust-Hub database contains a limited number of original benchmark circuits for Hardware Trojan detection. Each benchmark typically includes only dozens of instances, making it difficult to support the demand for large-scale training samples in deep learning models. To address this, this paper generates many functionally equivalent clean circuits and corresponding trojan-injected circuits through an automated process based on the original benchmarks. This approach builds a comprehensive and structurally diverse experimental dataset.

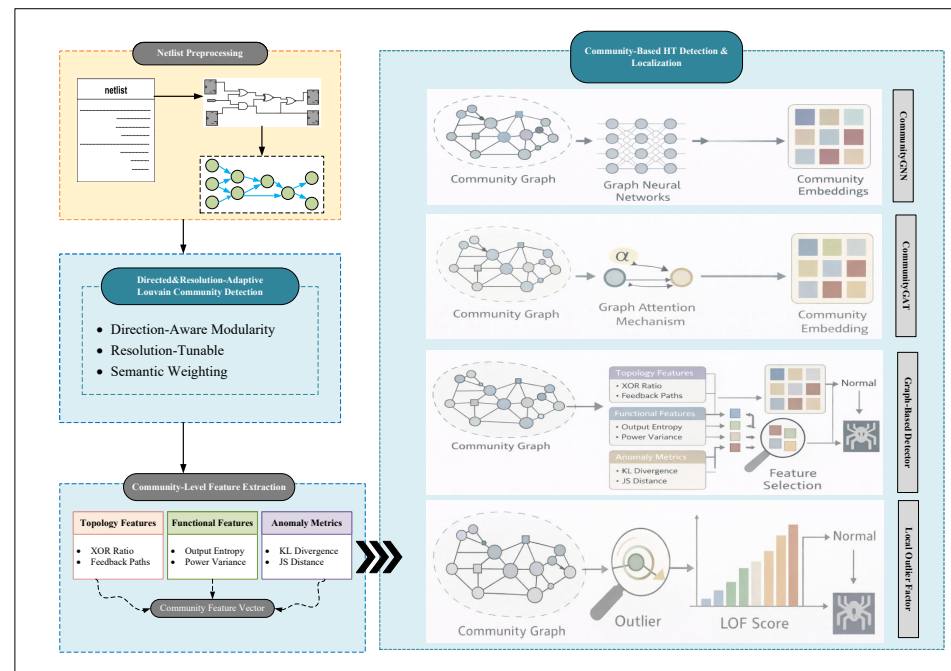


Figure 1. Gate-level HT detection workflow.

This paper evaluates the proposed method against post-synthesis Trojan insertion by an untrusted third-party foundry, using c880, s27, and s1488 from Trust-Hub as foundational circuits. The attacker model assumes complete netlist access with functional equivalence constraints, targeting rare-trigger activation (combinational or sequential) to evade detection. c880 is a classic combinational logic benchmark, featuring moderate structural complexity and logic functionality entirely determined by current inputs. Its verification process is straightforward, and simulation efficiency is high, making it suitable for generating diverse structural variants. Additionally, its logic cone and critical paths are well-defined, facilitating structural feature extraction. s27 and s1488 are classic small-scale and medium-scale sequential logic benchmarks proposed at ISCAS'89, respectively. They feature finite state spaces and strong controllability, supporting exhaustive or near-exhaustive verification. These characteristics facilitate sequential trigger condition design and rapid algorithm iteration. These three circuit types complement each other at both combinational and sequential logic levels, providing a unified and representative experimental platform for generating diverse combinational-triggered and sequential-triggered Hardware Trojans.

Clean circuit generation prioritizes functional equivalence while introducing structural variations through multiple logic optimization sequences. Specifically, operations such as redundant logic removal, node restructuring, rewriting and substitution, and multi-level logic optimization from synthesis tools (e.g., ABC) are employed. Different execution sequences and parameter configurations generate functionally equivalent yet structurally distinct logic networks. Subsequently, the universal Boolean network is mapped to a specific process cell library. Unlike area-optimized or delay-optimized mapping, this paper employs a random decision-based mapping strategy. This allows the synthesizer to randomly select from multiple equivalent decomposition schemes, yielding clean circuits with significantly different topologies and enhanced unpredictability. Such circuits are better suited for constructing diverse training samples.

To prevent generating invalid circuits or highly similar samples, diversity constraints are introduced during generation (1). Newly generated circuits are filtered using structural distance constraints, ensuring sufficient differentiation from existing samples in node

distribution, edge connectivity characteristics, and topological depth. This effectively enhances the structural diversity of the dataset while reducing the risk of model overfitting:

$$D(G_i, G_j) = \alpha \cdot d_{\text{node}}(G_i, G_j) + \beta \cdot d_{\text{edge}}(G_i, G_j) + \gamma \cdot d_{\text{topology}}(G_i, G_j) \quad (1)$$

where d_{node} , d_{edge} and d_{topology} denotes node type distribution difference (L1 distance, KL divergence), edge count and fanout distribution difference, topological depth or hierarchical distribution difference. A minimum distance threshold τ is set. If (2):

$$D(G_i, G_j) \geq \tau, \quad \forall G_j \in G_{\text{existing}} \quad (2)$$

If the similarity is too high, the circuit is discarded, and a new random mapping is performed. This iterative process ensures the generated clean circuits exhibit sufficient structural diversity.

Before generating trojan circuits, this paper constructs a series of malicious logic variants with enhanced stealthiness, and diverse trigger conditions based on the original baseline circuit through an automated process. These variants are then embedded into the original circuit to form many trojan training samples. To ensure the systematic and scalable generation of trojan circuits, a parameterized design approach based on classic trojan templates is adopted. Specifically, drawing from the open-source Hardware Trojan dataset Trust-Hub and related research, five representative Hardware Trojan trigger templates were summarized and selected: counter-triggered [26], rare-event signal-triggered [27], state machine-triggered [28], comparator-triggered [29], and multi-point distributed trojans [30]. The trigger mechanisms and key parameter configurations for each trojan type are shown in Table 1.

Table 1. Parameterized Hardware Trojan Trigger and Payload Templates.

Trojan Type	Trigger Parameters
Counter-based Trigger	Bit width selection (4, 6, 8, 12 bits); Threshold value; Increment condition (edge/pattern)
Rare-Signal Trigger	Matching mask width (8, 12, 16 bits); Required consecutive matching cycles
FSM Trigger	State count (3, 4, 6 states); Activation path depth; Entry signal selection
Comparator Trigger	Comparison bit width; Predefined reference value; Comparison type
Multi-point Payload	1–4 target locations; Selection of output/internal/high-fanout nodes

Based on the structural characteristics of the selected original circuit and the trigger conditions of different templates, trojan templates are randomly selected and their trigger parameters are instantiated during the generation process. This approach constructs Trojan circuit variants with diverse structures and behaviors. The trigger mechanism is a core factor in Hardware Trojan design, directly determining the difficulty of activation and the concealment of the Trojan. This paper employs a probability-based trigger design strategy. First, extensive logic simulation is performed on the target circuit. By analyzing tens of thousands to millions of test vectors, the signal probabilities at each node under normal operating conditions are statistically determined. This process identifies rare nodes with low switching probabilities as potential trigger sources. To further reduce the

risk of accidental activation, multiple rare nodes are combined to form composite trigger conditions, significantly enhancing trigger rarity and concealment.

After completing the Trojan horse logic embedding, the generated circuit samples undergo rigorous validity verification. First, structural consistency checks are performed to ensure the absence of multiple drivers, undriven nodes, and topological loops within the circuit. Subsequently, for combinational logic circuits (e.g., c880), formal equivalence verification is employed to guarantee functional consistency. For sequential circuits containing flip-flops (e.g., s27 and s1488), functional verification is achieved through large-scale random simulation. Simultaneously, the Trojan horse must be triggerable under random inputs while maintaining a low activation probability. Only when all structural and functional constraints are satisfied is the corresponding circuit sample included in the final experimental dataset. In total, 2400 circuit instances are generated (1200 clean and 1200 Trojan-inserted), with specific per-type composition and train/validation/test split detailed in Section 4.2.

3.2. Gate-Level Netlist to Directed Acyclic Graph Conversion

When constructing the directed acyclic graph (DAG) corresponding to the gate-level netlist, the original netlist is first parsed using standardized methods to remove redundant formatting characters, yielding a unified, parsable netlist representation. Subsequently, the netlist structure is parsed to extract logic gates, flip-flops, and port instances, assigning a unique identifier to each node to establish one-to-one mapping between netlist objects and graph nodes.

During graph construction, directed edges are established based on each component's fan-in and fan-out relationships to explicitly represent signal drive directions. Concurrently, necessary timing dependencies are introduced by incorporating sequential element boundaries (i.e., flip-flops/registers) as natural breakpoints. To satisfy DAG modeling requirements, pure combinational loops (cycles within combinational logic without intervening flip-flops) are detected and resolved through topological sorting and depth-first search. These sequential elements serve as breakpoints to sever combinational feedback paths: edges crossing flip-flop boundaries are preserved as directed connections (FF input to FF output), while cyclical paths within pure combinational logic are disrupted. This approach ensures that sequential structures—including sequential Trojan triggers such as state machines and counters—are retained as directed paths through flip-flop nodes rather than being eliminated.

Following loop elimination, lightweight optimizations are applied to the graph structure, including redundant node merging and path rearrangement, yielding a compact, semantically clear DAG. The resulting DAG serves as a unified intermediate representation that fully preserves node logic attributes and signal dependencies, providing a foundation for subsequent community partitioning and Hardware Trojan feature extraction.

3.3. Improvements to Community Partitioning Algorithms

This paper addresses the structural characteristics of directed acyclic graphs generated from gate-level netlists and the typical small scale and covert structure of Hardware Trojans. The proposed improved Louvain algorithm integrates three complementary enhancement mechanisms, as shown in Algorithm 1. First, the algorithm introduces a directed modularity function to explicitly model signal propagation directions in gate-level netlists, forming the theoretical foundation of this method. Second, by incorporating a resolution parameter to control the granularity of community partitioning, the algorithm can detect small, covert Hardware Trojans often overlooked by traditional modularity-based methods. Finally, the algorithm employs a logic-gate-semantics weighting mechanism to enhance focus on trojan-sensitive components like XOR gates, multiplexers, and flip-flops. These three mechanisms

operate at distinct levels yet are equally crucial, collectively achieving precise, fine-grained, and semantically aware community discovery tailored for Hardware Trojan analysis.

Algorithm 1 Directed and Resolution-Adaptive Louvain Community Method

Input: $\mathcal{V}$ (node set), $w[i][j]$ (weighted adjacency matrix), γ (resolution parameter), max_iter (maximum iterations)

Output: community_assignment (final node-community labels)

```

1 Initialization:
2 Assign community[i] = i for all  $i \in \mathcal{V}$ ; set improved = true
3 Greedy Optimization:
4 While improved and iter < max_iter
5   improved = false; shuffle  $\mathcal{V}$  to avoid order bias
6   For each  $i \in \mathcal{V}$ 
7     Compute  $\Delta Q$  for moving  $i$  to neighbor communities (via ComputeDeltaQ( $i, c, w, \gamma$ ))
8     Move  $i$  to the community with maximum  $\Delta Q > 0$  (set improved = true if moved)
9 Graph Aggregation:
10 Build aggregated graph where each node represents a community; compute aggregated edge
    weights  $w_{\text{agg}}[\mathcal{C}_u][\mathcal{C}_v] = \sum_{i \in \mathcal{C}_u, j \in \mathcal{C}_v} w[i][j]$ 
11 Iteration & Mapping:
12 Repeat Steps 2–3 on the aggregated graph until no improvement; map aggregated communities
    back to original nodes
13 Return community_assignment
14 Function ComputeDeltaQ( $i, c, w, \gamma$ )
15 Calculate modularity change for moving node  $i$  to community  $c$ :
16    $\Delta Q = \left( \frac{k_{i,\text{out} \rightarrow c} + k_{i,\text{in} \leftarrow c}}{m} - \gamma \cdot \frac{k_{i,\text{out}}k_{\text{in},c} + k_{i,\text{in}}k_{\text{out},c}}{m^2} \right) - \left( \frac{k_{i,\text{out} \rightarrow \text{curr}} + k_{i,\text{in} \leftarrow \text{curr}}}{m} - \gamma \cdot \frac{k_{i,\text{out}}k_{\text{in},\text{curr}} + k_{i,\text{in}}k_{\text{out},\text{curr}}}{m^2} \right)$ 
17 Return  $\Delta Q$ 
18 Notation:
19    $k_{i,\text{out}/\text{in}}$ : Total out/in-weight of node  $i$ ;  $k_{i,\text{out} \rightarrow X}/k_{i,\text{in} \leftarrow X}$ : Weight from  $i$  to  $X$  to  $i$ ;
     $m = \sum_{i,j} w[i][j]$ .
```

3.3.1. Directed Modularity Function

Given the inherent structural property of gate-level netlists possessing signal-driven directionality, this paper introduces directed modularity to theoretically extend the traditional Louvain algorithm. Unlike the default symmetric connection assumption in undirected networks, logic gate connections in gate-level netlists exhibit explicit input–output directionality, which directly reflects the functional dependencies within the circuit. Based on the directed modularity definitions proposed by Newman and Leicht & Newman, this paper represents gate-level netlists as directed weighted graphs and employs a directed modularity function of the following form (3):

$$Q_{\text{dir}} = \frac{1}{m} \sum_{i,j} \left[A_{ij} - \gamma \cdot \frac{k_i^{\text{out}} k_j^{\text{in}}}{m} \right] \delta(c_i, c_j) \quad (3)$$

where, A_{ij} denotes the weight of the directed edge from node i to node j , and the asymmetric definitions of in-degree and out-degree explicitly capture the unidirectional signal propagation in gate-level circuits, thereby enabling community partitions that preserve logical dependencies and signal flow characteristics.

3.3.2. Introduction of the Resolution Parameter

To address the tendency of conventional modularity-based methods to favor large communities and obscure small-scale anomalous structures, a resolution parameter γ is introduced into the directed modularity formulation to control the granularity of community partitioning. A larger γ leads to finer-grained communities, which facilitates the isolation of small-scale Hardware Trojans, whereas a smaller γ results in coarser partitions that better capture high-level functional modules. By incorporating γ , the proposed approach

effectively mitigates the resolution limit inherent in the traditional Louvain algorithm, enabling controllable community scales and significantly improving the visibility of small Trojan structures while preserving global structural consistency.

The resolution parameter γ in Algorithm 1 is not tuned using Trojan labels. It is fixed based on unsupervised structural statistics on the training set and then applied to all test circuits. Preliminary analysis shows that the performance is stable within a reasonable range of γ . Here, “resolution-adaptive” denotes the capability of adjusting community granularity via γ rather than label-dependent adaptation.

3.3.3. Logic Gate Semantic Weighting

Given the significant variations in semantic importance among different logic gates in Hardware Trojan design, this paper further introduces a logic gate semantic weighting mechanism to enhance the sensitivity of community partitioning toward high-risk nodes. In practical circuits, logic units such as XOR gates, multiplexers, and flip-flops are frequently employed for trojan triggering or payload implementation. However, traditional community detection algorithms assign identical weights to all nodes, making it difficult to highlight such critical structures. To address this, during the directed weighted graph modeling phase, this paper assigns differentiated weights to different logic gate types by adjusting edge weights A_{ij} or introducing node semantic weights. This approach increases the influence of relevant nodes on community partitioning results during modularity optimization. Without altering the overall greedy framework of the Louvain algorithm, this semantic weighting mechanism achieves collaborative modeling of structural information and functional semantics, resulting in communities that better align with the actual distribution characteristics of Hardware Trojans. In DW-Louvain, edge weights $w[i][j]$ in the weighted adjacency matrix are modulated by gate type semantics. Specifically, edges incident to XOR/XNOR/MUX/comparator-related nodes receive higher multiplicative weights ($w[i][j] \times 2.0$), AND/OR/NAND/NOR edges receive medium weights ($w[i][j] \times 1.5$), and NOT/BUF edges retain base weights ($w[i][j] \times 1.0$). These semantic weighting factors are applied to the adjacency matrix prior to community detection (Algorithm 1, line 1) and are determined based on circuit design knowledge rather than tuned using detection labels. The method is empirically robust to moderate variations in these factors, and a fixed weighting scheme is used throughout all experiments.

Computational Complexity Analysis: The directed and resolution-adaptive Louvain algorithm exhibits near-linear time complexity $O(|E| \log |V|)$ with respect to the number of edges $|E|$ in practice, where $|V|$ represents the number of gates. The introduction of resolution parameters and semantic weighting does not alter the dominant complexity term, as these are incorporated as constant-time adjustments during the modularity optimization phase. For sparse gate-level netlists where $|E| \approx O(|V|)$, this results in effectively linear scaling with circuit size.

3.4. Feature Extraction

Traditional subgraph feature extraction methods currently focus solely on conventional metrics such as the distribution of node in-degree and out-degree, degree centrality, graph density, and community size. This approach yields suboptimal results for Trojan detection. This paper innovatively incorporates the extraction of topological features, functional features, and anomaly indicators during the feature extraction phase.

3.4.1. Topological Feature Extraction

After completing community partitioning using the improved Louvain algorithm, each community corresponds to a structurally compact subgraph, which can be approximately regarded as a local functional module or potential abnormal region in the circuit. To

characterize the structural differences between different subgraphs, this paper extracts discriminative topological features from a topological perspective, which serve as inputs to the subsequent detection model.

First, considering that Hardware Trojans often rely on specific logic units (such as XOR/NAND gates) in trigger and payload design, we calculate the proportion of the number of XOR/NAND gates in the community to the total number of nodes. Compared with the relatively uniform gate-type distribution in normal functional modules, Trojan-related subgraphs usually exhibit obvious gate-type concentration or proportional imbalance. This feature can effectively reflect potential abnormal structures.

Second, although gate-level combinational circuits can be modeled as directed acyclic graphs (DAGs), Hardware Trojans often realize hidden cyclic trigger mechanisms through sequential feedback across flip-flop (FF) boundaries. To capture this characteristic while preserving the DAG representation, this paper introduces an equivalent feedback path feature to quantify potential sequential feedback interactions within each community.

Specifically, for a node pair (i, j) in a community C , it is counted as an equivalent feedback path if the following two conditions are simultaneously satisfied:

- (1) There exists a combinational driving path $i \Rightarrow j$ (feed-forward influence);
- (2) There exists a path $j \Rightarrow \text{FF} \Rightarrow i$ that traverses one or more flip-flops (feedback influence through sequential elements).

This definition captures intentional sequential feedback preserved through register boundaries, distinguishing it from pure combinational loops that are removed during DAG construction (Section 3.2). The equivalent feedback path feature thus quantifies the density of state-machine-like dependencies within a community—precisely the cyclic trigger structures (e.g., counter states feeding back to enable Trojan payload logic) that Hardware Trojans commonly exploit.

Its quantification is defined as the number of node pairs in the community that satisfy the above two conditions.

Normal functional modules usually optimize performance or area, so their logic depth is not too large. However, some Trojans may insert deep logic paths for concealment. The maximum path length inside the subgraph is also an important indicator for measuring structural complexity and a key aspect of topological feature extraction. For a subgraph C , the maximum path depth is defined as (4):

$$d_{\max}(C) = \max_{i,j \in V(C)} \text{dist}(i, j) \quad (4)$$

where $\text{dist}(i, j)$ denotes the length of the longest directed path from node i to node j .

These topological features complement each other, enabling the detection model to perceive both local structural anomalies and global topological deviations, thereby improving the recognition ability for small-scale and highly concealed Trojans.

3.4.2. Functional Feature Extraction

The output entropy is commonly used to measure the uncertainty and information complexity of the output signal of a subgraph under a given stimulus, and the output distribution of Hardware Trojans is often abnormal. For a given subgraph, we collect the output values of the key output node set under random or specific stimulus sequences and count their probability distributions. The output entropy is defined as (5):

$$H(C) = - \sum_{x \in \mathcal{X}} P(x) \log P(x) \quad (5)$$

where $\mathcal{X}$ denotes the subgraph output space, and $P(x)$ is the occurrence probability of the corresponding output pattern. In practical implementation, the output entropy can

be calculated separately under random and specific stimuli: random stimuli are used to characterize the statistical behavior of the subgraph under normal operation, while specific stimuli target potential trigger paths or highly sensitive input configurations to amplify Trojan-related responses. The simulation stimuli are generated using random and activity-oriented input patterns without any knowledge of Trojan trigger conditions. This strategy aims to maximize signal transitions and internal propagation, thereby amplifying potential abnormal behaviors while remaining applicable to realistic detection scenarios. The output entropy of normal modules changes stably with stimuli, whereas Trojan subgraphs may exhibit sudden changes in output entropy under specific stimuli.

When activated, the triggering of Hardware Trojans usually causes abnormal local switching activity, resulting in unstable dynamic power consumption. This paper uses power consumption variance to represent this behavioral feature. Under a given input stimulus sequence, we count the instantaneous power consumption of nodes inside the subgraph and calculate its variance over the time dimension (6):

$$\sigma_P^2(C) = \frac{1}{T} \sum_{t=1}^T (P_C(t) - \bar{P}_C)^2 \quad (6)$$

Power variance is estimated using a switching-activity-based proxy derived from toggle counts during logic simulation, rather than cell library power models. This approach avoids expensive power analysis while capturing abnormal switching behavior with negligible additional simulation cost, where $P_C(t)$ is the total power consumption of the subgraph at time t , $\bar{P}_C$ is the average power consumption, and T is the number of simulation cycles. Power consumption variance helps identify logic structures with low activation probability but high disturbance.

To enhance the recognition capability based on functional features, this paper also takes the differences between the output entropy and power consumption variance features extracted under random and specific stimuli as auxiliary features to input into the model.

3.4.3. Anomaly Metric Extraction

We introduce an anomaly index based on distribution differences to quantify the degree of behavioral and structural anomalies in subgraphs. We construct a graph and perform community topological partitioning on a clean circuit (without Trojans) to obtain a “clean golden model”, which serves as the criterion for normal behavior. The KL/JS-based anomaly metrics are computed relative to a golden clean reference circuit. Thus, the proposed method assumes access to a trusted clean design, following a common threat model in Hardware Trojan detection. Suppose the empirical distribution of a feature vector in the clean model is P , and the feature distribution of the circuit under test is Q .

The community Kullback–Leibler (KL) divergence is commonly used to measure the information deviation between two probability distributions, and its definition is as follows (7):

$$D_{KL}(P \parallel Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \quad (7)$$

The KL divergence describes the amount of information lost when using P to characterize data under the assumption of distribution Q . Since the KL divergence is asymmetric, to overcome its shortcomings in numerical stability and symmetry, this paper further introduces the Jensen–Shannon (JS) distance as a supplementary anomaly index. The JS distance is defined based on the KL divergence (8):

$$D_{JS}(P \parallel Q) = \frac{1}{2} D_{KL}(P \parallel M) + \frac{1}{2} D_{KL}(Q \parallel M), \quad M = \frac{1}{2}(P + Q) \quad (8)$$

The JS distance is symmetric and measures the mutual deviation between P and Q . Meanwhile, it is insensitive to sparse samples and noise, thus exhibiting excellent robustness.

3.4.4. Community Feature Vector via GNN Embedding

After constructing community topological features, functional features, and anomaly indices, a key issue remains: although manually designed features have good interpretability, Trojans exhibit complex high-order structural patterns in gate-level netlists. Manually designed features cannot fully cover these patterns, and the potential nonlinear structural correlations within subgraphs still need to be explored. Therefore, a Graph Neural Network (GNN) is introduced to encode each community subgraph into a vector, which serves as a high-level feature input to the detection model.

For each community obtained via the improved Louvain algorithm, it is treated as an independent directed subgraph, and the corresponding GNN input is constructed. Node features can be defined as logic gate types (represented by one-hot encoding), node in-degree/out-degree, node semantic weights, and local logic depth; edge features can be defined as signal propagation directions and edge weights. This modeling method can simultaneously perceive the circuit's topological structure and signal flow semantics.

Since gate-level circuits are inherently directional, this paper adopts a GNN structure that supports directed edge modeling. In the first layer, the update form of node i can be abstracted as (9):

$$h_i^{(1)} = \text{AGG}\left(\left\{h_j^{(0)} \mid j \in \mathcal{N}_{\text{in}}(i)\right\}, \left\{h_k^{(0)} \mid k \in \mathcal{N}_{\text{out}}(i)\right\}\right) \quad (9)$$

where $\mathcal{N}_{\text{in}}(i)$ and $\mathcal{N}_{\text{out}}(i)$ denote the in-neighbor and out-neighbor sets of node i , respectively. By distinguishing incoming and outgoing information flows, the GNN can more accurately model the signal propagation relationships of the circuit.

After multi-layer GNN message passing, node-level features need to be aggregated into a community-level vector. This paper uses a community-level readout function (Readout) to generate a fixed-dimensional embedding vector; Readout can be implemented via attention pooling. This process compresses the subgraph's structural features, gate type distribution, and signal flow features into a compact vector.

Introducing GNN to generate compact vectors can enhance the generalization ability for unknown Trojans, as it does not rely on specific Trojan structures. More importantly, it can model multi-hop neighborhood dependencies and capture high-order structural patterns.

3.5. Hardware Trojan Detection Model

To fully leverage the subgraph structural information after community partitioning, this paper introduces multiple community-level detection models, including graph neural network (GNN)-based representation learning methods and density-based unsupervised anomaly detection methods. Models are selected for detection from the perspectives of different supervision assumptions and modeling capabilities.

CommunityGNN is a hierarchical representation learning method that integrates community detection and GNNs. Its core idea is to treat community subgraphs as the basic modeling units, then use GNNs to encode each community subgraph into a fixed-dimensional embedding vector. This approach can both preserve local functional characteristics and characterize the overall structure and semantic features of the subgraph. Using CommunityGNN can significantly reduce the graph scale and computational complexity, making it suitable for large-scale, modular circuit detection tasks. Training protocol:

200 epochs, Adam optimizer ($lr = 0.001$), cross-entropy loss with class weighting (1:10 for positive:Trojan) to handle imbalance, early stopping (patience = 20).

Graph-Based Detector [31]: This baseline is implemented following the method proposed by Gao et al. [31]. In this approach, each gate-level netlist is first converted into a directed graph, where logic gates are represented as nodes and signal connections as edges. A set of handcrafted graph structural features is then extracted, including node degree statistics, connectivity patterns, and path-related metrics. These graph features are used as inputs to a machine learning classifier to distinguish Trojan-infected circuits from clean ones and to localize suspicious gates. We strictly follow the feature definitions and experimental protocol described in [31] to ensure a fair and reproducible comparison. Notably, this method operates on the entire circuit graph (global-level detection) rather than community subgraphs, making it susceptible to interference from normal logic structures in large-scale netlists. Therefore, applying it to community-level detection serves as a critical baseline to isolate the performance gains attributable to community partitioning.

CommunityGAT is an attention-enhanced variant of CommunityGNN. Unlike traditional GNNs, it can learn adaptive attention weights between different nodes within a community, assigning higher aggregation weights to high-saliency signals (e.g., Trojan trigger and payload logic). This improves the modeling ability for fine-grained structural differences within the community, generating more interpretable community representations. It is typically used in scenarios where community subgraphs have uneven signal propagation intensity and highly coupled logic functions. Training protocol: 200 epochs, Adam optimizer ($lr = 0.001$), focal loss ($\gamma = 2$) for hard example mining on imbalanced Trojan samples, attention dropout 0.3, early stopping (patience = 20).

The Local Outlier Factor (LOF) is a density-based unsupervised anomaly detection algorithm that identifies outliers by comparing the local density deviation between each node and its neighboring nodes. In this paper, it is applied to the community feature space: the comprehensive feature vector of each subgraph is scored for anomalies. It is suitable for fine-grained anomaly screening after community partitioning, and serves as an auxiliary detector in unsupervised scenarios and a supplementary verification method for GNN model detection results.

4. Experimental Results and Analysis

4.1. Experimental Platform and Environment

Our experimental environment configuration is shown in the Table 2 below. We used Python scripts to preprocess gate-level netlists, and leveraged an NVIDIA GPU to accelerate model training and inference. The preprocessing steps include netlist parsing, construction of directed acyclic graphs (DAGs), and community subgraph partitioning via the improved Louvain algorithm proposed in this paper. Circuit logic processing in the feature construction phase and the simulation phase were implemented using the ABC tool and custom scripts. For the graph neural network (GNN) model training phase, CommunityGNN, Graph-Based Detector, and CommunityGAT were implemented based on Deep Graph Library (DGL) and PyTorch; the Local Outlier Factor (LOF) algorithm was implemented using scikit-learn. The CUDA framework was used to accelerate GNN-related computations.

Table 2. Experimental Environment Configuration.

Item	Configuration
CPU	Intel (R) Core (TM) i7-10750H CPU @ 2.60GHz
GPU	NVIDIA GeForce RTX 2060
Programming Language	Python 3.8
Deep Learning Framework	Torch 1.12.0+cu113
	Scikit-learn 1.3.0
	NetworkX 3.1
	Dgl-cuda113 0.9.1

To construct the experimental dataset for Hardware Trojan detection, we used the c880, s27, and s1488 benchmark circuits provided by Trust-Hub as the basis. We generated structurally diverse but functionally equivalent clean circuit samples through multiple logic optimizations and mapping configurations. Meanwhile, based on parameterized classic Trojan templates, we embedded trigger and payload logic into different positions (without significantly damaging the original functionality and synthesizability) to construct diversified Trojan circuit instances. Finally, a circuit dataset for community partitioning, feature extraction, and detection model training/evaluation was formed the partitioned circuits are shown in the Figure 2.

After modeling the circuit as a directed acyclic graph, this paper improved the traditional Louvain algorithm based on directed modularity. This method explicitly preserves signal direction information, introduces a resolution parameter to adjust community granularity, and weights nodes and edges by combining logic gate semantics—thus better adapting to the structural characteristics of gate-level circuits. Through the iterative process of greedy optimization and multi-level aggregation, the circuit is divided into several structurally compact, functionally relatively independent community subgraphs. The partitioning results shown in the Figure 3 lay the foundation for subsequent feature extraction and Hardware Trojan detection.

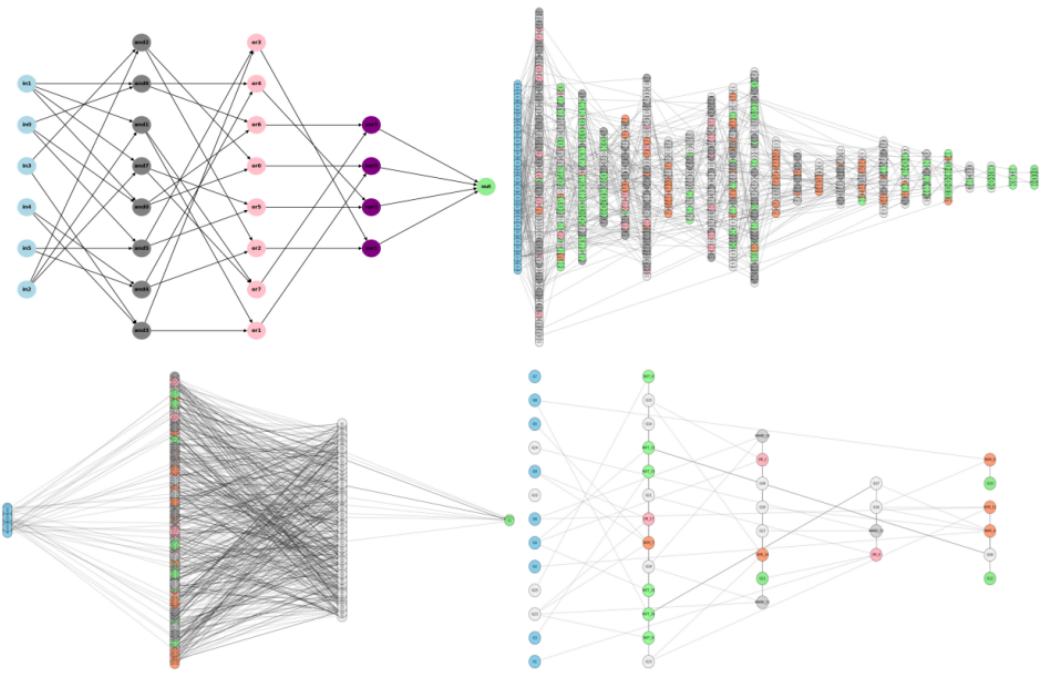


Figure 2. Overview of the Generated Circuit Variants.

4.2. Experimental Data

The dataset used in this study specifically selects three representative circuit types from Trust-Hub—combinational circuit c880, sequential circuits s27 and s1488—as the original reference designs. These circuits exhibit significant differences in scale, structural complexity, and sequential behavior, enabling systematic evaluation of the proposed method’s detection performance across various gate-level circuit types. Based on the original benchmark circuits, this paper further constructs a large-scale experimental dataset. For clean circuits, multi-configuration logic synthesis and process mapping are performed on the original design to generate gate-level netlists with diverse structures (while maintaining functional equivalence and synthesizability). For Trojan circuits, different forms of trigger mechanisms and payload logic are embedded into different functional regions of the circuit according to parameterized Trojan templates, forming Trojan instances with diverse structures and varying concealment levels. Finally, the dataset integrates the generated clean circuits and corresponding Trojan-inserted circuits, providing a sufficient and representative experimental basis for subsequent community partitioning, feature extraction, and training/evaluation of detection models. The final dataset comprises 2400 circuit instances in total, with an equal distribution between clean and Trojan-inserted variants:

- c880: 500 clean circuits and 500 Trojan-inserted circuits;
- s27: 300 clean circuits and 300 Trojan-inserted circuits;
- s1488: 400 clean circuits and 400 Trojan-inserted circuits.

For model training and evaluation, the dataset is partitioned using stratified random sampling (70% training, 15% validation, 15% testing) to maintain the distribution balance across circuit types and Trojan templates. In addition, a stricter cross-circuit evaluation protocol is adopted, where all variants derived from c880 and s27 are used for training and validation, and all variants derived from s1488 are exclusively used for testing. This setting ensures that the model is evaluated on an unseen circuit topology and prevents circuit-specific memorization.

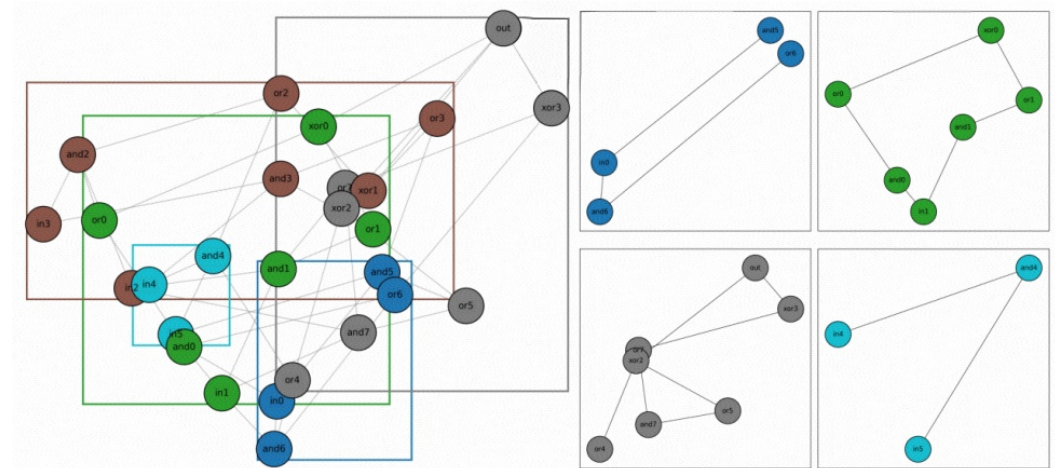


Figure 3. Partitioned Communities of the Circuit DAG.

4.3. Experimental Evaluation Metrics

To comprehensively evaluate the performance of the proposed method in Trojan detection, this paper adopts True Positive Rate (TPR), True Negative Rate (TNR), F1-score, and Average Localization Coverage (ALC) as the main experimental evaluation metrics. These metrics reflect the detection model’s ability to distinguish clean circuits and Trojan circuits from different angles.

TPR is commonly used to measure the model's ability to correctly identify Trojan circuits, defined as the ratio of correctly detected Trojans to the total number of Trojans. Its calculation formula is (10):

$$\text{TPR} = \frac{TP}{TP + FN} \quad (10)$$

where TP (True Positive) denotes the number of correctly detected Trojans, and FN (False Negative) denotes Trojan circuits misidentified as clean circuits.

TNR is used to measure the model's ability to identify clean circuits, defined as the ratio of correctly detected clean circuits to the total number of clean circuits. Its calculation formula is (11):

$$\text{TNR} = \frac{TN}{TN + FP} \quad (11)$$

where TN (True Negative) denotes the number of samples correctly detected as clean, and FP (False Positive) denotes Trojan circuit samples misjudged as clean circuits.

The F1-score is the harmonic mean of precision and recall, used to comprehensively evaluate the model's detection performance on imbalanced datasets. First, precision is defined as (12):

$$\text{Precision} = \frac{TP}{TP + FP} \quad (12)$$

Recall is defined as (13):

$$\text{Recall} = \frac{TP}{TP + FN} \quad (13)$$

Then the F1-score is (14):

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \quad (14)$$

ALC measures gate-level coverage of actual Trojan gates, prioritized over precision because security contexts require detecting all malicious logic even at the cost of inspecting additional benign gates. It calculates the average ratio of detected to actual Trojan gates (15):

$$\text{ALC} = \frac{1}{N} \sum_{i=1}^N \frac{DG_i}{AG_i} \quad (15)$$

where DG_i is the number of actual Trojan gates in detected communities for circuit i , and AG_i is the total actual Trojan gates. Due to deterministic community partitioning and fixed train/test splits, metrics are reported as mean values across the dataset; variance across random seeds is minimal (<2%) and omitted for clarity.

4.4. Experimental Results

Table 3 presents the performance of each detection model under the configuration of using the original Louvain community partitioning algorithm and only traditional basic features. Overall, the detection performance under this configuration is relatively limited: both TPR and ALC are significantly constrained, indicating that traditional features based on undirected community partitioning cannot fully characterize the structural properties of gate-level Hardware Trojans. Among the models, CommunityGAT achieves the relatively optimal results (TPR = 94.2%, F1 = 0.93), demonstrating that the attention mechanism improves feature modeling capability to a certain extent. However, its ALC remains low (73.1%), limiting the localization accuracy. In contrast, LOF performs the weakest, reflecting that density-based unsupervised methods struggle to effectively identify Trojan subgraphs in the absence of discriminative features and reasonable community structures.

As a baseline experiment, this result verifies that relying solely on traditional features and the original Louvain method is insufficient to meet the performance requirements of gate-level Hardware Trojan detection and localization.

Table 3. Detection Performance of Baseline Features with Original Louvain Community Detection.

Metric	CommunityGNN	Graph-Based Detector	CommunityGAT	Local Outlier Factor
TPR	89.1%	82.4%	94.2%	69.1%
TNR	86.9%	85.5%	92.1%	76.3%
F1	0.88	0.83	0.93	0.71
ALC	67.2%	63.3%	73.1%	52.0%

Table 4 shows that under the condition of keeping features unchanged, the proposed improved Louvain community partitioning method significantly enhances the overall performance of various detection models: both TPR and ALC metrics exhibit consistent growth. Specifically, the ALC of CommunityGNN increases from 67.2% to 72.5%, and the ALC of CommunityGAT rises to 76.8%.

This result indicates that after introducing the proposed community partitioning strategy, the obtained community structure better aligns with the actual functional module division of the circuit, thereby helping to effectively isolate and accurately identify the local structures corresponding to Trojans. It should be noted that under this experimental setup, the performance improvement originates entirely from the optimization of the community partitioning mechanism (rather than the enhancement of features themselves), which further verifies the effectiveness of the improved Louvain algorithm in gate-level netlist modeling—especially its obvious advantage in separating small-scale, structurally compact Trojan logic.

Table 4. Detection Performance of Baseline Features with Improved Louvain Community Detection.

Metric	CommunityGNN	Graph-Based Detector	CommunityGAT	Local Outlier Factor
TPR	91.8%	85.7%	95.6%	72.5%
TNR	88.4%	86.8%	93.5%	78.9%
F1	0.90	0.86	0.94	0.75
ALC	72.5%	68.1%	76.8%	56.3%

Table 5 illustrates the impact of introducing the proposed improved feature set on detection performance, under the condition of using the original Louvain community partitioning algorithm. Compared with Table 3, the performance of all detection models is significantly improved: for example, the TPR of the Graph-Based Detector increases from 82.4% to 88.3% (with its F1-score rising to 0.88), and the ALC of CommunityGNN also grows to 74.1%.

This result indicates that the proposed features can more effectively characterize the structural anomalies and behavioral characteristics related to Hardware Trojans, thereby enhancing the discriminative capability of the models. Meanwhile, since the community partitioning granularity remains relatively coarse, some structurally compact Trojan logic may be included in larger normal communities—this limits further performance improvement. Overall, this experiment verifies the independent effectiveness of the improved features in Hardware Trojan detection, while also revealing the performance bottleneck of relying solely on feature enhancement (without simultaneous optimization of the community partitioning strategy).

Table 5. Detection Performance of Enhanced Features with Original Louvain Community Detection.

Metric	CommunityGNN	Graph-Based Detector	CommunityGAT	Local Outlier Factor
TPR	92.6%	88.3%	96.1%	75.8%
TNR	87.9%	87.4%	94.0%	80.1%
F1	0.90	0.88	0.95	0.78
ALC	74.1%	70.5%	78.9%	59.7%

Table 6 presents the detection results under the complete configuration, where both the improved feature engineering and the directed Louvain community partitioning algorithm are adopted simultaneously. Overall, all models achieve optimal or near-optimal performance under this configuration. Among them, CommunityGAT exhibits the most outstanding performance: its TPR, TNR, F1-score and ALC reach 97.5%, 95.8%, 0.97 and 83.9%, respectively, demonstrating significant advantages in both detection accuracy and Trojan localization capability. This indicates that there exists an obvious synergistic enhancement effect between the direction-aware community structure and the discriminative features. Meanwhile, the performance of CommunityGNN and Graph-Based Detector is also significantly improved, which verifies that the proposed community-level modeling and feature design have good generality across different detection models. Although the absolute performance of Local Outlier Factor is still lower than that of supervised methods, it achieves the largest improvement margin in TPR and ALC. This further confirms that the community-representation-based modeling also brings significant gains to unsupervised detection.

Analysis of Unsupervised Detection Limitations. The notably weaker performance of LOF (TPR 81.4%, F1 0.82, ALC 65.4%) compared to supervised methods reveals a fundamental challenge in hardware Trojan detection: statistical camouflage by intentional design. Trojan authors deliberately select trigger conditions with low activation probability and distribute payload logic across normal functional paths, ensuring that community-level features (XOR ratio, path depth, output entropy) often fall within the distribution range of benign functional modules. Density-based methods like LOF identify anomalies by local density deviation, but hardware Trojans are engineered to avoid precisely this—exhibiting neither extreme outlier status nor obvious density sparsity in feature space. For example, a counter-based Trojan trigger may reside in a community with normal XOR density and path depth, differing only in subtle sequential dependencies that require learned discriminative representations to capture. This failure case underscores why supervised community-aware learning is essential for this task, as subtle structural anomalies cannot be reliably detected through statistical rarity alone. In summary, these results systematically demonstrate the effectiveness and robustness of the proposed method in the tasks of gate-level Hardware Trojan detection and localization.

Table 6. Detection Performance of Enhanced Features with Improved Louvain Community Detection.

Metric	CommunityGNN	Graph-Based Detector	CommunityGAT	Local Outlier Factor
TPR	95.3%	91.2%	97.5%	81.4%
TNR	90.1%	89.6%	95.8%	84.2%
F1	0.93	0.90	0.97	0.82
ALC	81.7%	76.8%	83.9%	65.4%

Table 7 compares the improvement effects of various detection models after the introduction of the method proposed in this paper, from the perspective of performance gain. The results show that the Graph-Based Detector and Local Outlier Factor achieve the most significant improvements in the TPR and ALC metrics, with the maximum increases reaching 8.8% and 13.4%, respectively. This indicates that the proposed community partitioning and feature enhancement strategies exert a notable optimization effect on traditional detection methods. In contrast, although CommunityGAT maintains a leading position in terms of absolute performance, its relative gain is relatively small. This suggests that the model already possesses strong representation capabilities under the baseline configuration, and the method proposed in this paper further enhances its ability to perceive fine-grained structural differences in circuits.

Overall, these results quantitatively verify that the method proposed in this paper can bring stable and consistent performance improvements to different types of detection models, reflecting the good generality and robustness of the framework.

Table 7. Summary of Performance Improvements under Optimal Configurations.

Method	MAX TPR ↑	MAX F1 ↑	MAX ALC ↑
CommunityGNN	95.3% (+6.2%)	0.93 (+0.05)	81.7% (+14.5%)
Graph-Based	91.2% (+8.8%)	0.90 (+0.07)	76.8% (+13.5%)
CommunityGAT	97.5% (+3.3%)	0.97 (+0.04)	83.9% (+10.8%)
LOF	81.4% (+12.3%)	0.82 (+0.11)	65.4% (+13.4%)

4.5. Comparison with State-Of-The-Art Methods

To further validate the effectiveness and advancement of the proposed method for Hardware Trojan detection and localization, this paper selects several representative state-of-the-art methods as comparative baselines and conducts systematic comparative experiments under the same dataset and evaluation criteria.

4.5.1. Selection of Comparative Methods

This paper selects several representative recent methods as comparative baselines. These methods cover mainstream technical paradigms in gate-level Hardware Trojan research: (1) Machine learning-based detection with handcrafted structural features: We adopt the method proposed by Zhao et al. (2022), which extracts gate-level structural features and applies different machine learning classifiers for Trojan detection and diagnosis. (2) Graph-theory and machine learning hybrid approach: We adopt the method proposed by Wang et al. (2023), which combines graph-theoretic features with machine learning techniques to model gate-level circuits and perform Trojan detection and localization. (3) Subgraph-based detection approach: We adopt the SubG4TJ method proposed by Hu et al. (2025), which constructs collaborative subgraph representations with multidimensional attributes for fine-grained Trojan detection. These baseline methods respectively represent three typical research directions: structural feature-based learning, graph-theoretic modeling, and subgraph-level modeling, and thus provide strong and meaningful comparisons for the proposed community-based framework.

4.5.2. Fairness Settings

To ensure fairness and reproducibility of comparisons among different methods, all experiments adopt unified datasets, data partitioning protocols, and evaluation metrics:

- **Dataset** : The gate-level circuit variant dataset constructed in this paper (based on c880, s27, and s1488) is used for all methods.
- **Data Partitioning**: The same training and test set split ratios are applied to all methods.

- **Evaluation Metrics:** True Positive Rate (TPR), True Negative Rate (TNR), F1-score, and Trojan Localization Accuracy (ALC) are consistently adopted as performance metrics.

Furthermore, for all deep learning-based methods, identical training epochs and early-stopping strategies are employed to minimize performance bias caused by differences in training parameters.

4.5.3. Experimental Results and Comparative Analysis

Table 8 presents the performance comparison between the proposed method and various baseline approaches. Overall, CommunityGAT outperforms recent state-of-the-art methods across all evaluation metrics. Compared with circuit-level and graph-level baselines, our method achieves finer localization granularity at the community level, higher detection accuracy (TPR = 91.2%, F1 = 0.91), and substantially improved localization precision (ALC = 81.7%). The performance gains indicate that adaptive community partitioning effectively mitigates feature dilution caused by small Trojan logic embedded in large netlists. Moreover, the consistent superiority across benchmarks demonstrates the robustness of the proposed framework against different circuit structures and Trojan variants.

Table 8. Average performance across all benchmarks.

Method	Localization Granularity	TPR (%)	F1	ALC (%)
Zhao et al. (2022) [32]	Coarse (circuit-level)	75.6	0.74	42.2
Wang et al. (2023) [14]	Coarse (graph-level)	86.8	0.87	58.9
SubG4TJ (Hu et al., 2025) [21]	Coarse (Subgraph-level)	89.9	0.90	65.2
CommunityGAT	Fine (community)	91.2	0.91	81.7

5. Conclusions and Future Work

To address the issues of Hardware Trojans in gate-level netlists—such as strong concealment, small structural scale, and easy obscuration by normal logic circuits—this paper proposes an improved Louvain algorithm with direction-aware enhancement, resolution adaptability, and semantic weighting. A complete detection framework of “community partitioning–multi-dimensional feature extraction–community-level detection” is constructed. For detection models, multiple models (CommunityGNN, Graph-Based Detector, CommunityGAT, and Local Outlier Factor) are selected for comparative experiments. Experimental results show that the method based on improved community partitioning and enhanced feature extraction outperforms traditional methods in multiple metrics. Among them, CommunityGAT achieves 97.5% TPR and an F1-score of 0.97 under the optimal configuration, verifying the effectiveness and robustness of the proposed methods in Hardware Trojan detection and localization tasks.

Although this paper has made progress in gate-level Hardware Trojan detection, several issues still deserve further research and expansion. In the future, we can consider jointly optimizing the community partitioning process and graph neural network training to construct an end-to-end community-aware detection model, where the community structure is dynamically adjusted during training to further improve generalization ability. Scalability validation: Future work will also validate the proposed method on modern large-scale benchmarks, specifically ITC’99 circuits (e.g., b17, b20, b22) and EPFL standard cell designs, which feature significantly larger gate counts (>100K gates) and more complex hierarchical structures than the current ISCAS benchmarks. This will demonstrate the method’s scalability to industrial-scale netlists while maintaining detection accuracy. Meanwhile, in the feature extraction phase, since a single feature cannot fully reflect the complex characteristics of Hardware Trojans, we can further explore multi-feature fusion strategies—an in-depth study of multi-feature fusion can also be part of future work.

Author Contributions: Conceptualization, Q.W. and X.L.; methodology, H.G., D.D., Y.L. and Y.L.; software, H.G., C.Z., Y.Z. and J.L.; validation, D.D., C.Z. and X.L.; investigation, Y.Z. and J.L.; resources, C.Z. and D.D.; data curation, H.G. and Y.L.; writing—original draft preparation, H.G., C.Z., Y.Z. and J.L.; writing—review and editing, H.G., D.D. and C.Z.; supervision, H.G., D.D., X.L., Y.L. and Q.W.; project administration, H.G. and Q.W.; funding acquisition, Q.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China under grants 61972302 and 62167009, and in part by the Fundamental Research Program of Shanxi Province under grant 202403021212165.

Data Availability Statement: Dataset available on request from the authors.

Acknowledgments: The authors would like to thank the editors and reviewers for their contributions to our manuscript.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

IC	Integrated Circuit
IoT	Internet of Thing
HT	Hardware Trojan
RTL	Register-transfer Level
GNN	Graph Neural Network
DAG	directed acyclic graph
FF	Flip-Flop
KL	Kullback–Leibler
JS	Jensen–Shannon
LOF	Local Outlier Factor
DGL	Deep Graph Library
TNR	True Negative Rate
TPR	True Positive Rate
ALC	Average Localization Coverage
TP	True Positive
FN	False Negative
TN	True Negative
FP	False Positive

References

1. Shakya, B.; He, T.; Salmani, H.; Forte, D.; Bhunia, S.; Tehranipoor, M. Benchmarking of Hardware Trojans and Maliciously Affected Circuits. *J. Hardw. Syst. Secur.* **2017**, *1*, 85–102.
2. Ma, P.; Li, J.; Liu, H.; Shi, J.; Zhang, S.; Pan, W.; Hao, Y. Hardware Trojan Detection Methods for Gate-Level Netlists Based on Graph Neural Networks. *IEEE Trans. Comput.* **2025**, *in press*.
3. Chen, W.; Bai, Z.; Pan, G.; Wang, J. A Fast Modularity Hardware Trojan Detection Technique for Large Scale Gate-Level Netlists. *Comput. Secur.* **2025**, *148*, 104111.
4. Tiempo, A.J.; Jeong, Y.-J. FP-GNN: A Graph Neural Network for Hardware Trojan Detection in Gate-Level Netlist. *IEICE Trans. Inf. Syst.* **2024**, *in press*.
5. Hasegawa, K.; Yanagisawa, M.; Togawa, N. Trojan-Feature Extraction at Gate-Level Netlists and Its Application to Hardware-Trojan Detection Using Random Forest Classifier. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.
6. Nahiyani, A.; Sadi, M.; Vittal, R.; Contreras, G.; Forte, D.; Tehranipoor, M. Hardware Trojan Detection through Information Flow Security Verification. In Proceedings of the 2017 IEEE International Test Conference (ITC), Anaheim, CA, USA, 15–19 October 2017; pp. 1–10.

7. Banga, M.; Hsiao, M.S. A Region Based Approach for the Identification of Hardware Trojans. In Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust, Anaheim, CA, USA, 13 June 2008; pp. 40–47.
8. Jin, Y.; Makris, Y. Hardware Trojan Detection Using Path Delay Fingerprint. In Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust, Anaheim, CA, USA, 13 June 2008; pp. 51–57.
9. Negishi, R.; Kurihara, T.; Togawa, N. Hardware-Trojan Detection at Gate-Level Netlists Using Gradient Boosting Decision Tree Models. In Proceedings of the 2022 IEEE 12th International Conference on Consumer Electronics (ICCE-Berlin), Berlin, Germany, 5–7 September 2022; pp. 1–6.
10. Mohanraj, P.; Paramasivam, S. A Multiobjective Approach for Side-Channel Based Hardware Trojan Detection Using Power Traces. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2024**, *107*, 825–835.
11. Waksman, A.; Suozzo, M.; Sethumadhavan, S. FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 697–708.
12. Hu, W.; Mao, B.; Oberg, J.; Kastner, R. Detecting Hardware Trojans with Gate-Level Information-Flow Tracking. *Computer* **2016**, *49*, 44–52.
13. Rathmair, M.; Schupfer, F.; Krieg, C. Applied Formal Methods for Hardware Trojan Detection. In Proceedings of the 2014 IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne, Australia, 1–5 June 2014; pp. 169–172.
14. Wang, J.; Zhai, G.; Gao, H.; Xu, L.; Li, X.; Li, Z.; Huang, Z.; Xie, C. A Hardware Trojan Detection and Diagnosis Method for Gate-Level Netlists Based on Machine Learning and Graph Theory. *Electronics* **2023**, *13*, 59.
15. Hashemi, M.; Momeni, A.; Pashrashid, A.; Mohammadi, S. Graph Centrality Algorithms for Hardware Trojan Detection at Gate-Level Netlists. *Int. J. Eng.* **2022**, *35*, 1375–1387.
16. Chen, F.; Liu, Q. Single-Triggered Hardware Trojan Identification Based on Gate-Level Circuit Structural Characteristics. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.
17. He, J.; Zhao, Y.; Guo, X.; Jin, Y. Hardware Trojan Detection through Chip-Free Electromagnetic Side-Channel Statistical Analysis. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 2939–2948.
18. Bhunia, S.; Hsiao, M.S.; Banga, M.; Narasimhan, S. Hardware Trojan Attacks: Threat Analysis and Countermeasures. *Proc. IEEE* **2014**, *102*, 1229–1247.
19. Yasaei, R.; Chen, L.; Yu, S.-Y.; Al Faruque, M.A. Hardware Trojan Detection Using Graph Neural Networks. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *44*, 25–38.
20. Chen, L.; Dong, C.; Wu, Q.; Liu, X.; Guo, X.; Chen, Z.; Zhang, H.; Yang, Y. Gnn4ht: A Two-Stage GNN Based Approach for Hardware Trojan Multifunctional Classification. *IEEE Trans. -Comput.-Aided Des. Integr. Circuits Syst.* **2024**, in press.
21. Hu, X.; Zhang, Y.; Liu, S.; Chen, X.; Wang, Y.; Zhao, Z.; Guo, Y.; Li, K. SubG4TJ: A Collaborative Subgraph Classification Method Based on Multidimensional Attributes for Hardware Trojan Detection. *Expert Syst. Appl.* **2025**, *297*, 129355.
22. Blondel, V.D.; Guillaume, J.-L.; Lambiotte, R.; Lefebvre, E. Fast Unfolding of Communities in Large Networks. *J. Stat. Mech. Theory Exp.* **2008**, *2008*, P10008.
23. Tiempo, A.J.; Jeong, Y.-J. Implementing Region-Based Segmentation for Hardware Trojan Detection in FPGAs Cell-Level Netlist. *IEICE Trans. Inf. Syst.* **2023**, *106*, 1926–1929.
24. Fortunato, S.; Barthélemy, M. Resolution Limit in Community Detection. *Proc. Natl. Acad. Sci. USA* **2007**, *104*, 36–41.
25. Yasaei, R.; Faezi, S.; Al Faruque, M.A. Golden Reference-Free Hardware Trojan Localization Using Graph Convolutional Network. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2022**, *30*, 1401–1411.
26. Tehranipoor, M.; Koushanfar, F. A Survey of Hardware Trojan Taxonomy and Detection. *IEEE Des. Test Comput.* **2010**, *27*, 10–25.
27. Mao, J.; Jiang, X.; Liu, D.; Chen, J.; Huang, K. A Hardware Trojan-Detection Technique Based on Suspicious Circuit Block Partition. *Electronics* **2022**, *11*, 4138.
28. Xiao, K.; Forte, D.; Jin, Y.; Karri, R.; Bhunia, S.; Tehranipoor, M. Hardware Trojans: Lessons Learned After One Decade of Research. *ACM Trans. Des. Autom. Electron. Syst. (TODAES)* **2016**, *22*, 1–23.
29. Rad, R.M.; Wang, X.; Tehranipoor, M.; Plusquellic, J. Power Supply Signal Calibration Techniques for Improving Detection Resolution to Hardware Trojans. In Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, USA, 10–13 November 2008; pp. 632–639.
30. Rama, E.; Ayache, M.; Buchty, R.; Bauer, B.; Korb, M.; Berekovic, M.; Mulhem, S. Trustworthy Integrated Circuits: From Safety to Security and Beyond. *IEEE Access* **2024**, *12*, 69603–69632.

31. Gao, H.X.; Zhai, G.X.; Li, Z.Y.; Zhou, J.; Li, X.; Wang, Q. A Hardware Trojan Diagnosis Method for Gate-Level Netlists Based on Graph Theory. *Electronics* **2024**, *13*, 2400.
32. Li, Z.; Yang, P.; Huang, Z.; Wang, Q. A Hardware Trojan Detection and Diagnosis Method for Gate-Level Netlists Based on Different Machine Learning Algorithms. *J. Circuits, Syst. Comput.* **2022**, *31*, 2250135:1–2250135:25.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.