electronics

Article

An Efficient Framework with Node Filtering and Load Expansion for Machine-Learning-Based Hardware Trojan Detection

Meng Dong [1], Weitao Pan [1,*], Zhiliang Qiu [1], Yiming Gao [1], Xiaoxin Qi [1] and Ling Zheng [2]

[1] State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China; mdong@stu.xidian.edu.cn (M.D.); zlqiu@mail.xidian.edu.cn (Z.Q.); 20011210526@stu.xidian.edu.cn (Y.G.); xxqi@stu.xidian.edu.cn (X.Q.)
[2] School of Communication and Information Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, China; lingzheng@xupt.edu.cn
* Correspondence: wtpan@mail.xidian.edu.cn

Abstract: The globalization of the integrated circuit (IC) industry has raised concerns about hardware Trojans (HT), and there is an urgent need for efficient HT-detection methods of gate-level netlists. Machine learning (ML) is a powerful tool for this purpose. A Trojan-detection framework is proposed in this paper to solve the data imbalance and low accuracy problems of existing ML-based HT-detection algorithms. To solve the problem of data imbalance, we propose the node-filtering algorithm, which extracts structure templates from HT circuits and removes most normal nodes based on them. To enhance the identification of unknown HT payload, we propose the load-expansion algorithm, which expands the identified HT nodes based on their fanout features. We evaluate the framework using different ML algorithms. The results show that the framework significantly improves the Trojan-detection rate of the original algorithms, and achieves a 10% improvement in true positive rate compared to the original algorithms.

Keywords: hardware Trojan detection; machine learning; node filtering; load expansion; gate-level netlists

check for updates

Citation: Dong, M.; Pan, W.; Qiu, Z.; Gao, Y.; Qi, X.; Zheng, L. An Efficient Framework with Node Filtering and Load Expansion for Machine-Learning-Based Hardware Trojan Detection. *Electronics* **2022**, *11*, 2054. https://doi.org/10.3390/electronics11132054

Academic Editor: Vijayakumar Varadarajan

Received: 11 May 2022
Accepted: 28 June 2022
Published: 30 June 2022

1. Introduction

Integrated circuits (ICs) have been widely used in various industries, and the IC design process has become more complex. The production chain of modern circuits is shown in Figure 1. The presilicon contains three stages: specification, RTL design, and netlist. The initial stage of pre-silicon translates the specification into RTL design with a Hardware Design Language (HDL). Then, the RTL design is transformed into a gate-level design. To meet the requirement of time-to-market (TTM), IC designers must use third-party intellectual properties (3PIP) and outsource parts of their products to third-party hardware design companies [1].

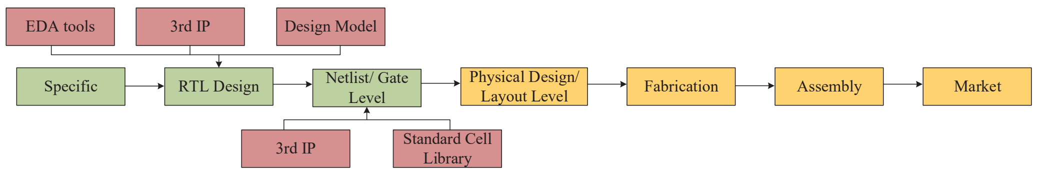


Figure 1. Design flow of IC.

However, both 3PIP and third-party vendors are associated with a risk of compromised chip security [2]. One threat is Hardware Trojans (HT), which can destroy functions, steal secrets, reduce system reliability, and even invalidate chips [3]. An HT consists of two components called the Trojan trigger and Trojan payload. The Trojan trigger is responsible for monitoring the running status of the circuit. If certain conditions are fulfilled, the HT is activated and the Trojan payload will perform specific malicious operations. If a chip with malicious HTs is used in key fields such as finance, communication, and national defense, serious security problems and huge economic losses may ensue.

The dangers of HT are well-understood and their detection methods have been widely researched [4,5]. To avoid numerous simulations, some HT-detection methods based on features are proposed and show good performance [6]. In these methods, features are extracted from gate-level netlists and used in machine learning (ML)-based detection algorithms, such as support vector machines (SVMs), artificial neural networks (ANNs), or recurrent neural networks (RNNs) [7,8].

However, feature-based ML methods have limitations in the balance of datasets and the identification of HT's payload. On the one hand, compared to the whole netlist, the number of HTs is tiny, so the dataset is extremely imbalanced. Data imbalance leads to overfitting of the learning model, reducing the prediction accuracy. On the other hand, due to the complex function, HT's payload is difficult to identify. The selected features are found from known HTs by heuristic approaches, which are valid for existing HTs. The trigger circuits of various HTs have the commonality of rare activation conditions, but the payload does not. Therefore, it is unrealistic to rely on a heuristic approach to detect every unknown HT's payload.

In order to identify all HTs in presilicon design and overcome limitations of feature-based HT detection, this paper focuses on the HT detection in the gate-level netlist and proposes a Trojan-detection framework that can integrate various existing machine learning algorithms. The framework consists of four stages: netlist modeling, node filtering, machine learning, and load expansion. In the netlist modeling stage, the input netlist is converted to a technology-independent netlist to eliminate functionally duplicated cells and reduce the problem size. Then, the netlist is modeled as a directed graph, with its nodes being the element of the circuit and its edge being wires. In the node-filtering stage, structure template is constructed to reflect the HT circuit structure. Then, subgraphs are divided from the directed graph. By matching the subgraphs with the structure template, suspicious nodes are identified. The dataset is then formed by the suspicious nodes, which alleviates the dataset imbalance problem. In the machine-learning stage, the node features are determined, which are used to train an ML algorithm to classify the nodes as HT or normal nodes. Note that the specific machine learning algorithm is not the focus of this paper. Finally, in order to improve the poor HT payload identification ability of existing ML algorithms, the already identified HT nodes are expanded in the load expansion stage, which means that some neighboring nodes of the already identified HT nodes are taken as HT nodes. This helps to further identify other HT payload and get a complete HT structure. The experiments show that compared with pure HT-detection methods based on RNN and SVM, by integrating these methods into the proposed framework, the true positive rate achieves 10% and 9% improvements, respectively. The paper makes the following contributions:

1. We propose a Trojan-detection framework which can support most ML-based HT-detection algorithms;
2. A node filtering algorithm is proposed as a pre-processing stage of ML-based HT detection algorithms. Suspicious nodes are extracted to obtain a balanced dataset;
3. A load expansion algorithm is proposed as a post-processing stage of ML-based HT detection algorithms. HT nodes are expanded in this stage to strengthen Trojan payload detection capabilities.

This paper first introduces the related work of HT-detection algorithms and describes their strengths and weaknesses (Section 2). A Trojan-detection framework is designed to improve detection accuracy and strengthen payload identification capability, which consists of a netlist modeling stage, node-filtering stage, machine-learning stage, and load expansion stage (Section 3). Experiments are conducted to show the effectiveness of the proposed framework in Section 4. Conclusions are drawn and further work is outlined in Section 5.

2. Background

Existing HT-detection techniques can be broadly divided into three categories: side-channel analysis, logic testing, and circuit analysis. Side-channel analysis focuses on the difference in side-channel signatures between the expected (golden specification) and actual (Trojan-inserted implementation) values [9,10]. These methods capture Trojans via power consumption [11], current activity [12], delay [13], thermal distribution [14], optical imaging [15], and some other detectable information. A major drawback of side-channel analysis is that it is difficult to detect the negligible side-channel difference caused by a tiny Trojan since the difference can easily hide in process variation and environmental noise. In addition, it requires a "golden model" for parameter comparison.

Logic testing is a reliable method that is independent of process variations, which is robust against process variations and noise margins [16,17]. It activates HTs by applying test vectors and compares the responses with the correct results. Previous studies [18,19] have improved the possibility of observing the impact of Trojans from the primary output by developing test pattern-generation algorithms. However, due to the existence of many logic states in the circuit, it is a fundamental challenge to activate an extremely rare trigger without trying all possible input sequences. The exponential input space complexity makes it unsuitable for detecting Trojans in large designs using traditional logic testing.

Circuit analysis technology is a fast and reliable method. It identifies HTs by analyzing the HT circuit structure and comparing it with the normal circuit. Machine-learning algorithms can analyze existing data sets, discover inherent patterns, and predict future data, and have been widely used in circuit analysis. Depending on the detection method, circuit analysis can be divided into three categories: feature-based, testability-based, and structure-learning-based.

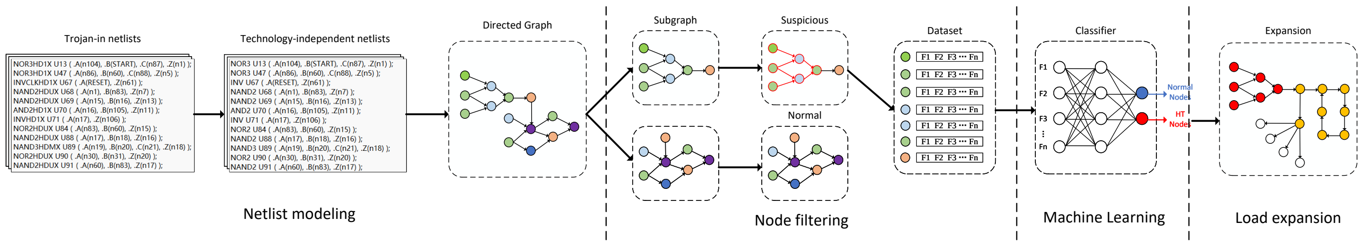
Feature-based. The authors in [6] obtained the features of the circuit structure via heuristic approaches, and first used a feature-based method to detects HTs. Ever since that work, features have been widely used in HT-detection methods. For instance, SVM [20,21], neural networks [8,22,23], and random forest [24] are all based on features, and achieve relatively high detection performance. However, the features obtained by heuristic approaches need to be continuously updated, and the extraction process is time-consuming when using large designs.

Testability-based. Using combinational testability measures as Trojan features to detect HT is proposed in [25], achieving better accuracy than previous methods. The measures are calculated using SCOAP in [26], an algorithm to evaluate the observability and testability of circuits. Based on SCOAP, a previous study [27] clustered the nets into two groups with the k-means method, used the intercluster distance as the major feature, and trained a support vector machine classifier to distinguish the Trojan circuits. However, for HTs with normal triggering probability, methods with SCOAP will reduce detection accuracy.

Structure-learning-based. To overcome the limitations of feature-based and testability-based HT-detection methods, structure-learning-based methods are introduced. Grams-Det [28] uses natural language processing (NLP) technology to solve the HT-detection problems, and uses a recurrent neural network as the machine-learning model. GNN4TJ [29] uses Graph Neural Network (GNN) to extract features from data flow graphs at the resister transfer level, learns the circuit's behavior, and identifies whether HTs exist in the design. Another study [30] proposed a node-wise HT-detection method at the gate level based on graph learning, and compared the performance of different classifiers, including GNN, GAT, and MPNN. The structure-learning-based method has good generalization ability on hardware circuits. However, it has poor ability to deal with data imbalances, which is one of the focuses of this paper.

3. Trojan-Detection Framework

Figure 2 shows the proposed Trojan-detection framework, which consists of four main stages: netlist modeling, node filtering, machine learning, and load expansion. The netlist is integrated, mapped, and modeled as a directed graph structure in the netlist modeling stage. In the node-filtering stage, which is the pre-processing of the machine learning stage, subgraphs are divided from the directed graph and matched with templates, and the dangerous nodes are filtered in order to balance the dataset. In the machine-learning stage, the nodes are classified by an ML-based HT-detection method. The features and learning models can be configured by the user flexibly. Finally, the identified Trojan nodes are expanded to obtain a complete Trojan structure in the load expansion stage, which is the post-processing of the machine learning stage. Note that the proposed framework aims to enhance the performance of existing ML methods through the pre-processing and post-processing stages, and the specific ML method is not the focus of this paper.



```
NOR3HD1X U13 ( .A(n104), .B(START), .C(n87), .Z(n1) );
NOR3HD1X U47 ( .A(n86), .B(n60), .C(n88), .Z(n5) );
INVCLKHD1X U67 ( .A(RESET), .Z(n61) );
NAND2HDUX U68 ( .A(n1), .B(n83), .Z(n7) );
NAND2HDUX U69 ( .A(n15), .B(n16), .Z(n13) );
AND2HD1X U70 ( .A(n16), .B(n105), .Z(n11) );
INVHD1X U71 ( .A(n17), .Z(n106) );
NOR2HDUX U84 ( .A(n83), .B(n60), .Z(n15) );
NAND2HDUX U88 ( .A(n17), .B(n18), .Z(n16) );
NAND3HDMX U89 ( .A(n19), .B(n20), .C(n21), .Z(n18) );
NOR2HDUX U90 ( .A(n30), .B(n31), .Z(n20) );
NAND2HDUX U91 ( .A(n60), .B(n83), .Z(n17) );
```

```
NOR3 U13 ( .A(n104), .B(START), .C(n87), .Z(n1) );
NOR3 U47 ( .A(n86), .B(n60), .C(n88), .Z(n5) );
INV U67 ( .A(RESET), .Z(n61) );
NAND2 U68 ( .A(n1), .B(n83), .Z(n7) );
NAND2 U69 ( .A(n15), .B(n16), .Z(n13) );
AND2 U70 ( .A(n16), .B(n105), .Z(n11) );
INV U71 ( .A(n17), .Z(n106) );
NOR2 U84 ( .A(n83), .B(n60), .Z(n15) );
NAND2 U88 ( .A(n17), .B(n18), .Z(n16) );
NAND3 U89 ( .A(n19), .B(n20), .C(n21), .Z(n18) );
NOR2 U90 ( .A(n30), .B(n31), .Z(n20) );
NAND2 U91 ( .A(n60), .B(n83), .Z(n17) );
```

Figure 2. An overview of the Trojan-detection framework. The input netlists are converted to technology-independent netlists, and then modeled as directed graphs. Subgraphs divided from the directed graphs are matched with templates to access their dangers, and the features of nodes in suspicious subgraphs are used as datasets. The nodes are classified as HT or normal nodes by ML-based HT detection based on their features. Finally, the HT nodes are expanded to obtain unknown HT payload and a complete HT structure.

3.1. Netlist Modeling

In IC design, cells with the same logical function usually have different descriptions because they vary in timing, area, and power, which makes it more difficult to analyze the netlist. In hardware Trojan detection, we only focus on circuit structure and logic gate function. Therefore, in order to reduce the complexity of the algorithm, we first learn the existing technology libraries such as SMIC and TSMC, and build a technology-independent cell model. Then, the cells in the original netlist are replaced by cells in the technology-independent cell model with the same function to obtain the technology-independent netlist.

The netlist is a description of the circuit connection relationship, including gates, wires, and the relationships between them, which is consistent with the directed graph structure. A gate-level netlist can be represented as a graph structure by translating the elements of the circuit into nodes and the wires into edges. The mapped technology-independent netlist is modeled as a directed graph $G = (V, E)$, where V is the set of vertices in the directed graph and E is the set of edges in the directed graph. We define $V = \{v_1, v_2, \ldots, v_j\}$, where v_j denotes a logic gate description such as XOR2, AND2, or OR2. We define $E = \{e_{ij}\}$, where e_{ij} equals 1 if logic gate v_i is a fan-in of logic gate v_j, and 0 otherwise.

3.2. Node Filtering

Since the Trojan circuit always accounts for less than one-thousandth of the whole circuit, the dataset is severely imbalanced and the accuracy of the model is degraded in ML-based detection methods. Oversampling and undersampling are typically used to address data imbalance. However, the new samples synthesized by oversampling are highly correlated with the original samples and tend to overfit during training. In undersampling, it is difficult to select removed samples, and random removal tends to introduce uncertainty. In our framework, considering the inherent characteristics of logic gates and the low trigger probability of Trojan circuits, undersampling is used to filter data. Firstly, according to expert experience and characteristics of HT trigger circuits, a structure template is proposed. Then, subgraphs are extracted from the directed graph modeled by the gate-level netlist. Lastly, all the subgraphs are matched with the structure template, and dangerous nodes are extracted to form a balanced dataset.

Structure template. Trojan circuits are difficult to activate and are triggered only when the input is a specific combination. To illustrate this, the HT structure of the s35932-T100 circuit in the Trust_Hub benchmark is shown in Figure 3. It can be seen that the trigger circuit of the Trojan contains many cascade structures composed of AND and NOR, which makes the probability of the Trojan being driven by random excitation extremely low.

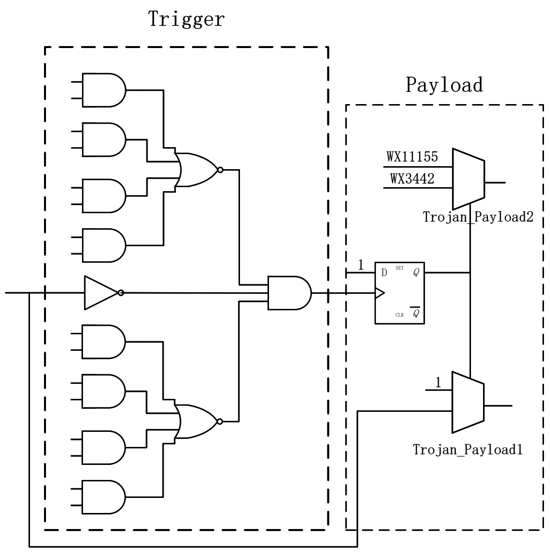


Figure 3. HT circuit in s35932-T100.

To better analyze the structure in the circuit, we calculated the output probabilities $P_{gate}(0)$ and $P_{gate}(1)$ of each logic gate separately, assuming they have the same probability of input 0 and 1. For example, for a three-input AND gate, the output is 1 only if the input sequence is '111', so $P_{gate}(1)$ is 0.125 and $P_{gate}(0)$ is 0.875. Logic gates are classified according to their output probabilities. Trojan trigger circuits usually consist of multiple low-probability logic gates of the same type. The structure template is proposed based on the above characteristics, as shown in Table 1, where m and n are integers greater than 1, indicating the number of input ports of the logic gate, and p is a natural number.

Table 1. HT structure template.

Low Probability of 0	Low Probability of 1
ORm $\rightarrow$ (2p)INV $\rightarrow$ ORn	NORm $\rightarrow$ (2p)INV $\rightarrow$ NORn
ORm $\rightarrow$ (2p)INV $\rightarrow$ NANDn	NORm $\rightarrow$ (2p)INV $\rightarrow$ ANDn
NANDm $\rightarrow$ (2p)INV $\rightarrow$ ORn	ANDm $\rightarrow$ (2p)INV $\rightarrow$ NORn
NANDm $\rightarrow$ (2p)INV $\rightarrow$ NANDn	ANDm $\rightarrow$ (2p)INV $\rightarrow$ ANDn
ANDm $\rightarrow$ (2p+1)INV $\rightarrow$ ORn	ORm $\rightarrow$ (2p+1)INV $\rightarrow$ NORn
ANDm $\rightarrow$ (2p+1)INV $\rightarrow$ NANDn	ORm $\rightarrow$ (2p+1)INV $\rightarrow$ ANDn
NORm $\rightarrow$ (2p+1)INV $\rightarrow$ ORn	NANDm $\rightarrow$ (2p+1)INV $\rightarrow$ NORn
NORm $\rightarrow$ (2p+1)INV $\rightarrow$ NANDn	NANDm $\rightarrow$ (2p+1)INV $\rightarrow$ ANDn

Subgraph matching. For large-scale netlists, it is difficult to find Trojan-triggered structures by directly analyzing the entire structure. In order to reduce the complexity of structure template matching, the backward breadth first search (BBFS) algorithm is used to divide the netlist graph structure into subgraphs. All subgraphs are matched with the structure template proposed in Table 1. It is observed that the number of matched structures of Trojan-triggered circuits is much larger than that of normal circuits. Therefore, we propose a subgraph classification algorithm as shown in Algorithm 1. The algorithm counts the number of structures that match the template, and if greater than the classification threshold T_i, the subgraph will be identified as suspicious and all its nodes will be marked as suspicious Trojan nodes.

Algorithm 1: Subgraph matching

```
Data: Subgraph $G = (V, E)$, Template Set $S_T$, Predecessor Set $S_P$, Classification
      Threshold $T_i$
Result: Suspicious HT Nodes $HT_{nodes}$
for $node \in V$ do
    if $node \in S_T$ then
        $node\_set$.add($node$)
    end
end
for $node1 \in node\_set$ do
    for $node2 \in node\_set$ do
        if exist path from $node1$ to $node2$ then
            $path\_set$.add($path$)
        end
    end
end
for $path \in path\_set$ do
    if $path \in S_T$ then
        $match\_num$ += 1
    end
end
if $match\_num \geq T_i$ then $HT_{nodes} \leftarrow V$ ;
else $HT_{nodes} \leftarrow None$ ;
```

3.3. Machine Learning

After the node-filtering stage, suspicious Trojan nodes are extracted. Features are extracted from these suspicious nodes to form the dataset used by the machine-learning stage. Different machine-learning algorithms can be applied in this stage to detect HT nodes in the dataset. We now discuss how the features are defined and how different learning models are trained based on the features.

Feature Library. Given the set of suspicious nodes, features are extracted to train the machine-learning model. Note that different models may use different features. On the basis of previous work and our analysis of existing HT, we create a HT feature library to support different ML models, which consists of static, statistical, and structural features.

Static features are extracted from the netlist based on fan-in, adjacent circuit elements, and minimum distance to a specific circuit element. We count the number of various logic gates near the target node and calculate the distance from the node to the input and output ports. All the static features are shown in Table 2, where x is a positive integer in the range 1 to 5.

Table 2. Static features.

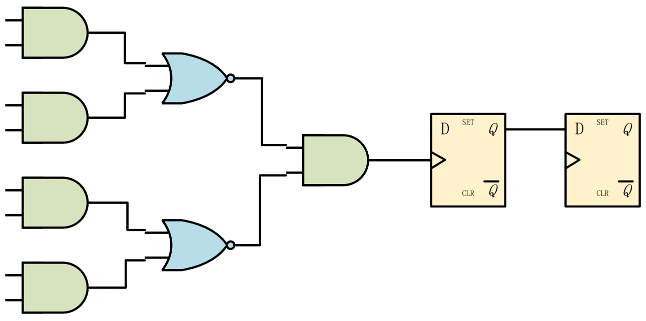
Feature	Specification
fan_in(x)	No. of fan-ins at the distance x-level from the net input
fan_out(x)	No. of fanouts at the distance x-level from the net input
in_dff(x)	No. of flip-flops at the distance x-level from the net input
out_dff(x)	No. of flip-flops at the distance x-level from the net output
in_gate(x)	No. of gates at the distance x-level from the net input
out_gate(x)	No. of gates at the distance x-level from the net output
in_loop(x)	No. of loops at the distance x-level from the net input
out_loop(x)	No. of loops at the distance x-level from the net output

Table 2. *Cont.*

Feature	Specification
in_const(x)	No. of constants at the distance x-level from the net input
out_const(x)	No. of constants at the distance x-level from the net output
nearest_pin	Min. level to any primary input
nearest_pout	Min. level to any primary output
nearest_dff	Min. level to any flip-flops from the net output
nearest_mux	Min. level to any multiplexer from the net output

Statistical features are proposed in SCOAP and utilized for evaluating the testability of a circuit. There has been research to prove that the SCOAP values are reasonable for HT detection, which include combinational controllability-0 (CC0), combinational controllability-1 (CC1), combinational observability (CO), sequential controllability-0 (SC0), sequential controllability-1 (SC1), and sequential observability (SO).

The structural feature is obtained by extracting netlist structural information near the target node, such as that on node types and connection relationships. In this paper, a depth-first search method with a maximum depth of 2 is used to sample each gate to obtain the connection relationship, and the co-occurrence matrices ($COM_{|D|\times|D|}$) are constructed to abstract cascade structural information, where D is the number of gate types. $COM_{|D|\times|D|}$ indicates the forward and reverse connection relationship of various gates in the subgraph extended with a node as the starting point. Figure 4 is an example of the co-occurrence matrix where the right part is the $COM_{6\times6}$ constructed for the AND gate in the center of the left part.



	AND	NOR	INV	DFF	MUX	OTHER
AND	0	0	0	1	0	0
NOR	2	0	0	0	0	0
INV	0	0	0	0	0	0
DFF	0	0	0	1	0	0
MUX	0	0	0	0	0	0
OTHER	0	0	0	0	0	0

Figure 4. Example of the co-occurrence matrix.

Learning Model. Our framework is not limited to a specific machine-learning algorithm. Generally, three categories of learning models are supported, namely support vector machine (SVM), multilayer perceptron (MLP), and recurrent neural network (RNN).

SVM aims to solve the problem of determining the parameters of the optimal classification hyperplane. In essence, the process of determining the parameters is a quadratic optimization problem, whose geometric meaning is to find the maximum classification interval under the constraints. For nonlinear problems in the input space, kernel functions are used to transform them into a linear classification problems of the feature space.

An MLP consists of an input layer, one or more hidden layers and an output layer, as shown in Figure 5. The multi-dimensional feature vector enters the model from the input layer. Linear operations and nonlinear activation are performed at the neurons in each hidden layer. Finally a one-dimensional result is generated by the output layer. The numbers of neurons in the input layer and output layer are equal to the dimension of the feature vector and 1, respectively. The number of hidden layers and neurons in each hidden layer can be dynamically adjusted.

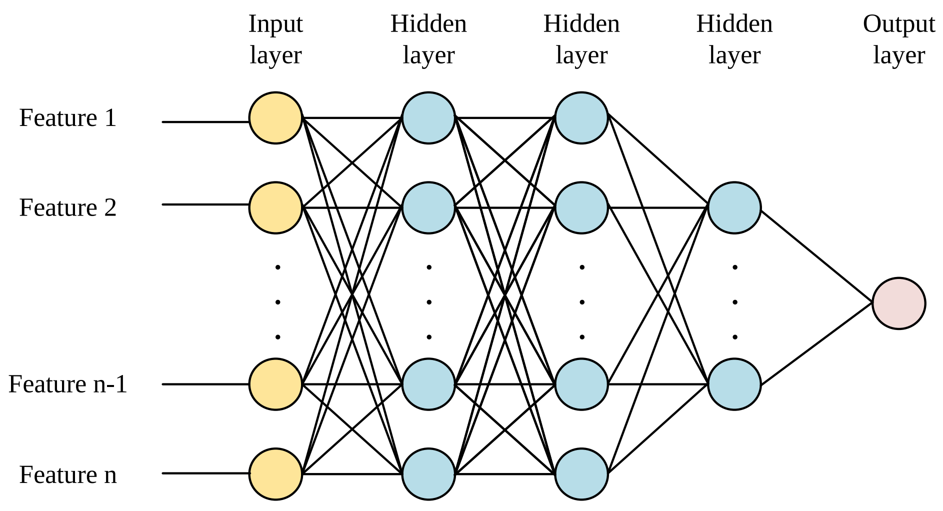


Figure 5. Structure of the multilayer perceptron.

An RNN model is typically used for time-series processing and predicting problems. GramsDet [28] demonstrated the feasibility of using a long short-term memory (LSTM) network in HT detection, and proposed a stacked LSTM network, as shown in Figure 6. The row vectors of each gate in *COM* are sent to different LSTMs for training, and the output of the last layer of LSTMs is a single vector, which is converted to a probability value through the Sigmoid function.

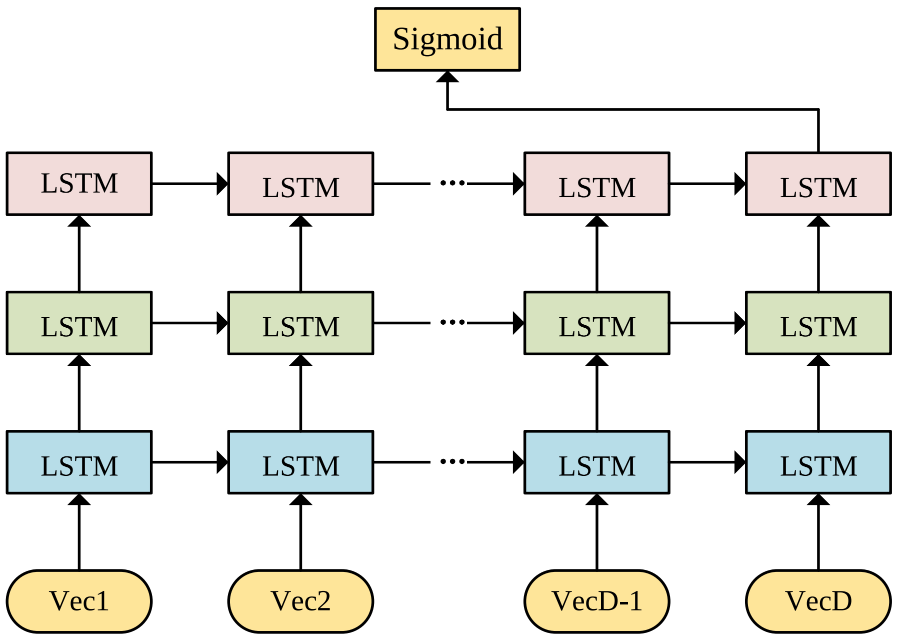


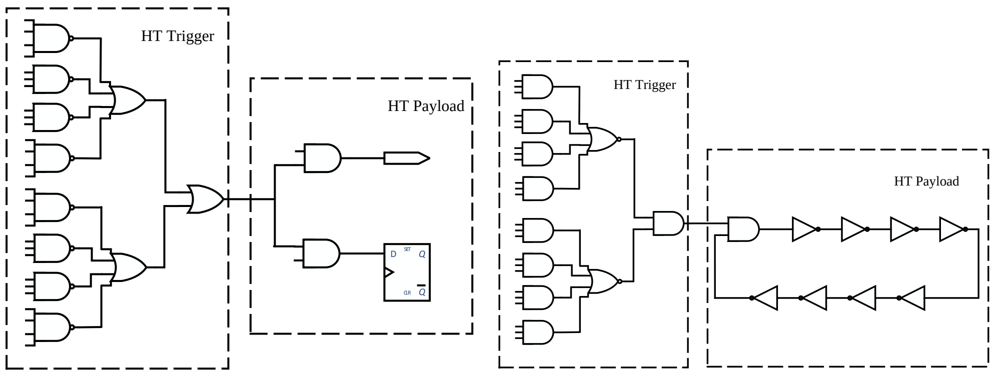
Figure 6. Structure of a stacked LSTM network.

Given the learning model, proper features should be selected from the feature library to train the model. Specifically, for SVM and MLP, the features can be selected from the static and statistical features; for RNN, only the structural feature can be used.

3.4. Load Expansion

After the machine-learning stage, the nodes extracted in the node-filtering stage are labeled as either HT or normal nodes. However, the node-filtering stage may have excluded some HT payload nodes, and the machine-learning stage may misclassify some HT nodes as normal nodes. In order to further identify such HT nodes, we proposed the load expansion mechanism. Before discussing the proposed algorithm, we first inspect the HT load structure.

The load structure of Trojans with different functions varies greatly, which poses challenge for the identification of hardware Trojans. Via analysis of the Trojan load circuits in gate-level netlists, we found that the HTs can be divided into two categories: (1) The load circuit is similar to the normal circuit, as shown in Figure 7a. Such Trojans can be classified into two categories according to their functions: leaking information and violating functions. If the function of the HT is to leak information, its load is close to the netlist output. If the function of the HT is to violate the function of the circuit, its load is generally located in a critical position in the circuit. For these HTs, the size of the load circuit is small, i.e., the fanout of the Trojan trigger circuit is small. When applying machine-learning algorithms to identify Trojan nodes, such HT nodes are usually regarded as normal nodes, resulting in low recognition accuracy. (2) The load circuit contains a ring oscillator, as shown in Figure 7b. Since this structure rarely appears in both normal circuits and Trojan circuits, it is hard for a machine-learning algorithm to identify all such HT nodes.



(**a**) (**b**)

Figure 7. Two categories of Trojan load circuits. (**a**) The load circuit is similar to the normal circuit (**b**) The load circuit contains structure that rarely appeared in both normal circuits and Trojan circuits.

Through the above analysis, it can be seen that compared with normal circuits, HT load circuits have lower fanout. This is because Trojan designers prefer to use Trojans to attack a small number of key components in the circuit. This can lead to poor detection ability of machine-learning-based algorithms. Based on the above observations, we propose a load expansion algorithm to expand the HT nodes identified in the machine-learning stage, as shown in Algorithm 2. To cope with situations where the HT load contains rare circuits such as ring oscillators, a fanout threshold T_{gate} for each node. Only when the number of fanouts of the node is less than T_{gate}, the node is expanded. In addition, considering the small scale of the Trojan load circuits, a total fanout threshold T_{total} is set. When the fanout of all expanded nodes reaches T_{total}, the expansion terminates.

Algorithm 2: Load expansion

```
Data: HT nodes $HT_{nodes}$, Total Fanout Threshold $T_{total}$, Gate Fanout Threshold
      $T_{gate}$, Successor Node Set $S_S$
Result: Expanded Nodes $HT_{expand}$
for $node \in node\_set$ do
    $Successor\_Nodes \leftarrow S_S$ ;
    if $len\ of\ Successor\_Nodes \geq T_{gate}$ then Break;
    else $HT_{expand} \leftarrow Successor\_Nodes$;
    while $True$ do
        if $current\_fanout \geq T_{total}$ then
            Break
        end
        for $suc\_node \in Successor\_Nodes$ do
            if $suc\_node \in HT_{expand}$ then
                Continue
            end
            if $len\ of\ suc\_node \geq T_{gate}$ then
                Continue
            end
            $current\_fanout$ += len of $suc\_node$;
            $HT_{expand}$.add($suc\_node$)
        end
    end
end
```

4. Experiment and Analysis

To verify the effectiveness of each stage and the overall framework, we first perform experiments on the node-filtering algorithm and the load expansion algorithm, and then apply existing machine-learning algorithms in the machine-learning stage to evaluate the overall framework. The algorithms proposed in this paper are written in Python language. The test cases include 15 benchmark test circuits selected on TrustHub. The descriptions of the HT benchmarks are listed in Table 3. The machine-learning algorithms selected to evaluate the performance of the framework are recurrent neural networks and support vector machines, which are implemented through an open-source program in Keras.

Table 3. Description of HT benchmarks.

Netlists	Scale	No. of Normal Nodes	No. of Trojan Nodes	No. of Trigger Nodes	Function of HTs
RS232-T1000	215	202	13	10	Change circuit function
RS232-T1100	216	204	12	11	Change circuit function
RS232-T1200	216	202	14	13	Change circuit function
RS232-T1300	213	204	9	7	Change circuit function
RS232-T1400	215	202	13	12	Change circuit function
RS232-T1500	216	202	14	11	Change circuit function
RS232-T1600	214	202	12	10	Change circuit function
S15850-T100	2182	2156	26	22	Denial of Service, Change circuit function
S35932-T100	5441	5427	14	11	Leak information, Change circuit function
S35932-T200	5436	5422	14	12	Denial of Service
S35932-T300	5462	5426	36	12	Reduce performance, Denial of Service
S38417-T100	5341	5329	12	11	Denial of Service, Change circuit function
S38417-T200	5344	5329	15	11	Denial of Service, Change circuit function
S38417-T300	5373	5329	44	11	Denial of Service, Change circuit function
S38584-T100	6482	6473	9	8	Denial of Service, Change circuit function

4.1. Node-Filtering Evaluation

In our experiment, we generate technology-independent netlists by netlist mapping and model them as directed graphs. Then, the node-filtering algorithm is applied to extract suspicious nodes. The extracted subgraph depth is 6 and the threshold T_i is 8.

The numbers of different types of nodes before and after node filtering are shown in Table 4. The average retention rate of the total nodes of the netlist is 20.6%, and the retention rates of most netlists are less than 50%. For netlist S15850-T100, its retention rate of the total nodes reaches 59%, because many low trigger structures also appear in its normal circuits.

Table 4. Result of node filtering.

Netlists	**Node Number before Filtering**			**Node Number after Filtering**		
	Total	**HT**	**Trigger**	**Total**	**HT**	**Trigger**
RS232-T1000	215	13	10	108(50%)	13(100%)	10(100%)
RS232-T1100	216	12	11	99(46%)	12(100%)	11(100%)
RS232-T1200	216	14	13	103(48%)	14(100%)	13(100%)
RS232-T1300	213	9	7	100(47%)	9(100%)	7(100%)
RS232-T1400	215	13	12	110(51%)	13(100%)	12(100%)
RS232-T1500	216	14	11	108(50%)	14(100%)	11(100%)
RS232-T1600	214	12	10	92(43%)	12(100%)	10(100%)
S15850-T100	2182	26	22	1290(59%)	24(92%)	22(100%)
S35932-T100	5441	14	11	52(1%)	12(86%)	11(100%)
S35932-T200	5436	14	12	48(1%)	14(100%)	12(100%)
S35932-T300	5462	36	12	54(1%)	16(44%)	12(100%)
S38417-T100	5341	12	11	1710(32%)	12(100%)	11(100%)
S38417-T200	5344	15	11	1653(31%)	15(100%)	11(100%)
S38417-T300	5373	44	11	1665(31%)	14(32%)	11(100%)
S38584-T100	6482	9	8	1613(25%)	9(100%)	8(100%)

From Table 4, it can be seen that all Trojan-triggered circuit nodes of the netlist are completely retained, as expected. In addition, most Trojan payload nodes are preserved due to the fact that all nodes of the danger subgraphs are preserved. Only when the Trojan load is too long, as in the case of netlists S35932-T300 and S38417-T300, will part of the Trojan load be truncated and not recognized. However, as shown later, such Trojan load will be fully recognized in the load expansion stage and will not affect the overall Trojan recognition accuracy. This experiment demonstrates the ability of the node-filtering algorithm to balance the dataset by extracting suspicious nodes, which filters out a large proportion of the normal nodes.

4.2. Load Expansion Evaluation

The HT trigger nodes are extracted using the node-filtering algorithm, and then, they are expanded by Algorithm 2 to obtain the Trojan load nodes. The gate fanout threshold T_{gate} is set to 2, and the total fanout threshold T_{total} is set to 40. The purpose of the load expansion is to extract as many HT payload nodes as possible with less error. Therefore, we evaluate the algorithm by the number of expanded HT nodes and misclassified normal nodes. The experimental results of the load expansion algorithm are shown in Table 5. For each netlist, with the given trigger nodes, the algorithm expands all the Trojan load nodes and misclassifies less than four normal nodes. This experiment shows that, by further including the neighboring nodes of the detected HT nodes in the machine-learning stage, the load expansion algorithm can accurately obtain the Trojan load structure.

Table 5. Result of load expansion.

Netlists	Number of Input Triggers	Number of Expanded HT Nodes	Recognition Rate of HT Nodes	Misjudged Normal Nodes
RS232-T1000	10	3	100%	1
RS232-T1100	11	1	100%	1
RS232-T1200	13	1	100%	0
RS232-T1300	7	2	100%	0
RS232-T1400	12	1	100%	0
RS232-T1500	11	3	100%	1
RS232-T1600	10	2	100%	0
S15850-T100	22	4	100%	0
S35932-T100	11	3	100%	1
S35932-T200	12	2	100%	3
S35932-T300	12	24	100%	0
S38417-T100	11	1	100%	1
S38417-T200	11	4	100%	3
S38417-T300	11	33	100%	0
S38584-T100	8	1	100%	1

4.3. Trojan-Detection-Framework Evaluation

In order to verify the effectiveness of the overall Trojan-detection framework, we apply two different ML methods, i.e., RNN and SVM, respectively, in the machine-learning stage of the framework.

When implementing the RNN-based Trojan-detection algorithm, we extracted the connection relationships of gates in the netlist and established the order-sensitive coevolution matrix (OSCOM) as features according to the algorithm of a previous study [28]. The constructed recurrent neural network model contains three LSTM layers and the sigmoid function is used as the output layer activation function. When implementing the SVM-based Trojan-detection algorithm, we selected static features of the hardware Trojan, including the total number of level-2 and level-3 logic gate fan-ins from the target node, the minimum number of levels from the target node to any primary input and output, and the minimum number of levels from the target node to the trigger input. A radial basis function is used as the kernel function of the SVM, and a Relu function is used as the activation function.

We use the cross-validation method in the training phase. That is, when testing a particular netlist, all the remaining netlists are used as the training set. For each of the above two algorithms, we first train the model using node features in the whole training set, and adjust the parameters to obtain the best classifier and test the netlist. Then, we apply the algorithm in our framework, i.e., we train the best model with the features of the extracted nodes, classify the nodes of the tested netlist, and further identify HT nodes by load expansion. The subgraph depth is set to 6 and the threshold T_i is set to 8 in node filtering; the gate fanout threshold T_{gate} is set to 2 and total fanout threshold T_{fanout} is set to 40 in load expansion.

True positive rate (TPR) and true negative rate (TNR) are used to evaluate the effectiveness of the methods. The experimental results are shown in Table 6. It can be seen that after applying the proposed framework, the average TPR of the Trojan-detection algorithms based on RNN and SVM is improved by 10% and 9%, respectively. That is, the HT-detection capability of the original algorithms is effectively improved. The average TNR of both algorithms is reduced by 1%, as the load expansion stage enhances Trojan identification accuracy at the expense of introducing a few misclassifications of normal circuits. However, a small decrease in TNR is acceptable compared to the large increase in TPR. In Trojan detection, the true positive rate of identifying the Trojan circuit is much more important than the overall accuracy of detection. This experiment shows that, by preprocessing the circuit to extract suspicious nodes and postprocessing the detection result of the machine-learning algorithm, the proposed framework can significantly improve the TPR of the HT detection.

Table 6. Comparison of the original work and our work.

Netlists	RNN [28]		RNN Applied in the Framework		SVM [21]		SVM Applied in the Framework	
	TPR	TNR	TPR	TNR	TPR	TNR	TPR	TNR
RS232-T1000	100%	95%	100%	92%	100%	97%	100%	97%
RS232-T1100	97%	96%	100%	94%	100%	97%	100%	98%
RS232-T1200	87%	97%	100%	95%	100%	96%	100%	95%
RS232-T1300	95%	96%	100%	93%	88%	97%	100%	94%
RS232-T1400	97%	95%	100%	91%	91%	97%	91%	95%
RS232-T1500	73%	95%	82%	92%	92%	96%	92%	94%
RS232-T1600	97%	95%	100%	95%	90%	96%	100%	97%
S15850-T100	73%	98%	92%	94%	96%	95%	96%	95%
S35932-T100	89%	95%	89%	90%	92%	100%	100%	99%
S35932-T200	50%	87%	67%	89%	100%	99%	100%	98%
S35932-T300	80%	93%	91%	95%	37%	100%	97%	98%
S38417-T100	77%	98%	82%	97%	92%	97%	92%	96%
S38417-T200	78%	99%	88%	98%	86%	95%	94%	96%
S38417-T300	40%	97%	86%	97%	30%	91%	86%	92%
S38584-T100	70%	98%	78%	96%	87%	86%	87%	91%
Average	80%	95%	90%	94%	86%	96%	95%	95%

5. Summary and Conclusions

To address the problems of data imbalance and poor identification of HT payload in machine-learning-based Trojan-detection algorithms, we propose a gate-level Trojan-detection framework in this paper. The framework improves the accuracy of feature-based HT detection by the node-filtering mechanism and strengthens payload identification capability by the load expansion mechanism. Different ML models can be supported by the proposed framework. Experiment results show that compared with pure HT-detection methods based on RNN and SVM, by integrating these methods into the proposed framework, the true positive rate achieves 10% and 9% improvement, respectively. In the future, we aim to further reduce the misclassification of the framework and support more advanced HT-detection algorithms such as GNN.

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

Funding: This research was funded by the National Natural Science Foundation of China under Grant 62102314, the Natural Science Basic Research Program of Shaanxi Province under Grant 2021JQ-708 and the Scientific Research Program Funded by Shaanxi Provincial Education Department under Grant 20JK0923.

Data Availability Statement: Publicly available datasets were analyzed in this study. These data can be found here: https://trust-hub.org/#/benchmarks/chip-level-trojan (accessed on 1 January 2020).

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

References

1. Bao, C.; Forte, D.; Srivastava, A. On Reverse Engineering-Based Hardware Trojan Detection. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2015**, *35*, 49–57. [CrossRef]
2. Bhunia, S.; Hsiao, M.S.; Banga, M. Hardware Trojan Attacks: Threat Analysis and Countermeasures. *Proc. IEEE* **2014**, *102*, 1229–1247. [CrossRef]
3. Tehranipoor, M.; Koushanfar, F. A survey of hardware trojan taxonomy and detection. *IEEE Des. Test Comput.* **2010**, *27*, 10–25. [CrossRef]
4. Dhavlle, A.; Hassan, R.; Mittapalli, M.; Dinakarrao, S.M.P. Design of hardware trojans and its impact on cps systems: A comprehensive survey. In Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Korea, 22–28 May 2021; pp. 1–5.
5. Yang, K.; Hicks, M.; Dong, Q. A2: Analog malicious hardware. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 18–37.

6. Oya, M.; Shi, Y.; Yanagisawa, M. A score-based classification method for identifying hardware-trojans at gate-level netlists. In 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2015; pp. 465–470.
7. Huang, Z.; Wang, Q.; Chen, Y. A survey on machine learning against hardware trojan attacks: Recent advances and challenges. *IEEE Access* **2020**, *8*, 10796–10826. [CrossRef]
8. Kundu, S.; Meng, X.; Basu, K. Application of machine learning in hardware trojan detection. In Proceedings of the 2021 22nd International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 7–9 April 2021; pp. 414–419.
9. Lyu, Y.; Mishra, P. Maxsense: Side-channel sensitivity maximization for trojan detection using statistical test patterns. *ACM Trans. Des. Autom. Electron. Syst. TODAES* **2021**, *26*, 1–21. [CrossRef]
10. Pan, Z.; Sheldon, J.; Mishra, P. Test generation using reinforcement learning for delay-based side-channel analysis. In Proceedings of the 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD), San Diego, CA, USA, 2–5 November 2020; pp. 1–7.
11. Salmani, H.; Tehranipoor, M.; Plusquellic, J. A layout-aware approach for improving localized switching to detect hardware Trojans in integrated circuits. In Proceedings of the 2010 IEEE International Workshop on Information Forensics and Security, Seattle, WA, USA, 12–15 December 2010; pp. 1–6.
12. Cao, Y.; Chang, C.H.; Chen, S. Cluster-based distributed active current timer for hardware Trojan detection. In Proceedings of the 2013 IEEE International Symposium on Circuits and Systems (ISCAS), Beijing, China, 19–23 May 2013; pp. 1010–1013.
13. 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, 9 June 2008; pp. 51–57.
14. Forte, D.; Bao, C.; Srivastava, A. Temperature tracking: An innovative run-time approach for hardware Trojan detection. In Proceedings of the 2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, 18–21 November 2013; pp. 532–539.
15. Zhou, B.; Adato, R.; Zangeneh, M. Detecting hardware trojans using backside optical imaging of embedded watermarks. In Proceedings of the 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 8–12 June 2015; pp. 1–6.
16. Ahmed, A.; Farahmandi, F.; Iskander, Y. Scalable hardware trojan activation by interleaving concrete simulation and symbolic execution. In Proceedings of the 2018 IEEE International Test Conference (ITC), Phoenix, AZ, USA, 29 October–1 November 2018; pp. 1–10.
17. Pan, Z.; Mishra, P. Automated test generation for hardware trojan detection using reinforcement learning. In Proceedings of the 26th Asia and South Pacific Design Automation Conference, Tokyo, Japan, 18–21 January 2021; pp. 408–413.
18. Banga, M.; Hsiao, M.S. A novel sustained vector technique for the detection of hardware Trojans. In Proceedings of the 2009 22nd International Conference on VLSI Design, New Delhi, India, 5–9 January 2009; pp. 327–332.
19. Chakraborty, R.S.; Bhunia, S. Security against hardware Trojan through a novel application of design obfuscation. In Proceedings of the 2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers, San Jose, CA, USA, 2–5 November 2009; pp. 113–116.
20. Hasegawa, K.; Oya, M.; Yanagisawa, M. Hardware Trojans classification for gate-level netlists based on machine learning. In Proceedings of the 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS), Sant Feliu de Guixols, Spain, 4–6 July 2016; pp. 203–206.
21. Liangjun, G.; Jinxing, Y.; Xin, C.; Yingchun, L.; Maoxiang, Y. Hardware Trojan Detection Method Based on Feature Extraction and SVM. *Microelectronics* **2020**, *6*, 914–919.
22. Inoue, T.; Hasegawa, K.; Kobayashi, Y. Designing subspecies of hardware Trojans and their detection using neural network approach. In Proceedings of the 2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin), Berlin, Germany, 2–5 September 2018; pp. 1–4.
23. Kurihara, T.; Hasegawa, K.; Togawa, N. Evaluation on hardware-Trojan detection at gate-level IP cores utilizing machine learning methods. In Proceedings of the 2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS), Napoli, Italy, 13–15 July 2020; pp. 1–4.
24. 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.
25. Salmani, H. COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist. *IEEE Trans. Inf. Forensics Secur.* **2016**, *12*, 338–350. [CrossRef]
26. Goldstein, L.H.; Thigpen, E.L. SCOAP: Sandia controllability/observability analysis program. In Proceedings of the 17th Design Automation Conference, Minneapolis, MN, USA, 23–25 June 1980; pp. 190–196.
27. Xie, X.; Sun, Y.; Chen, H. Hardware Trojans classification based on controllability and observability in gate-level netlist. *Ieice Electron. Express* **2017**, *14*, 20170682. [CrossRef]
28. Lu, R.; Shen, H.; Su, Y. Gramsdet: Hardware trojan detection based on recurrent neural network. In Proceedings of the 2019 IEEE 28th Asian Test Symposium (ATS), Kolkata, India, 10–13 December 2019; pp. 111–1115.
29. Yasaei, R.; Yu, S.Y.; Al Faruque, M.A. Gnn4tj: Graph neural networks for hardware trojan detection at register transfer level. In Proceedings of the 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 1–5 February 2021; pp. 1504–1509.
30. Hasegawa, K.; Yamashita, K.; Hidano, S. Node-wise Hardware Trojan Detection Based on Graph Learning. *arXiv* **2021**, arXiv:2112.02213.