IEICE Electronics Express, Vol.VV, No.NN, 1–6

LETTER

A Unioned Graph Neural Network Based Hardware Trojan Node Detection

Weitao Pan[1a)], Meng Dong[1], Cong Wen[2], Hongjin Liu[3], Shaolin Zhang[3], Bo Shi[3], Zhixiong Di[4], Zhiliang Qiu[1], Yiming Gao[1], and Ling Zheng[5]

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. In this work, we propose an approach to detect Trojan-nodes at the gate level, based on graph learning. The proposed method does not require any golden model and can be easily integrated into the integrated circuits design flow. In addition, we further design a unioned GNN network to combine information from the input side, output side, and neighbor side of the directed graph to generate representative node embeddings. The experimental results show that it could achieve 93.4% in recall, 91.4% in F-measure, and 90.7% in precision on average across different designs, which outperforms the state-of-the-art HT detection methods.

key words: Hardware Trojan Detection, Graph Neural Network, Golden Reference-Free, Gate-Level Netlist.
Classification: Integrated circuits

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 Fig. 1. The pre-silicon 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, 2].

[1]State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China
[2]School of Microelectronics, Xidian University, Xi'an 710071, China
[3]Beijing Sunwise Space Technology Ltd, Beijing 100010, China
[4]School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China
[5]School of Communication and Information Engineering, Xi'an University of Posts and Telecommunications, Xi'an 710121, China
a) wtpan@mail.xidian.edu.cn

DOI: 10.1587/elex.20.20230204
Received May 06, 2023
Accepted May 16, 2023
Publicized May 23, 2023

However, the third party may not be trusted or potentially

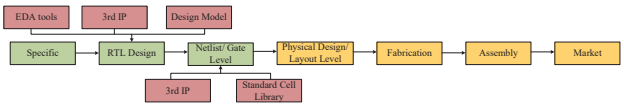

Fig. 1. Design flow of IC.

malicious, and will increases the security risk of the system. An adversary may take advantage of this process to implement some malicious functionality, referred to as HTs. These HTs may cause information leakage [3, 4], function change, degradation of chip performance, and other destructive consequences [5, 6]. HTs have drawn increasing attention in both academia and industry because of their significant potential threat.

There are many promising research efforts for HT detection in the pre-silicon stage. These approaches can be broadly classified into three categories: functional tests, formal validation, and circuit analysis. Functional test technology is a reliable method that is independent of process variations. It activates HTs by applying test vectors and compares the responses with the correct results. [7] improved the possibility of observing the impact of Trojan from the primary output by developing a new test pattern generation algorithm. However, due to the existence of many logic states in the circuit, the functional test cannot achieve 100% coverage and even suffers from state explosion problem. In addition, formal validation detects the HTs by checking whether the design satisfies a predefined set of security properties[8, 9]. Therefore, it cannot detect other unknown features introduced by the Trojan. In addition, due to the opacity of 3PIP, it is impractical to detect HT by using the golden model of 3PIP [10].

Circuit analysis technology is a fast and reliable method. It identifies HTs by analyzing the HTs circuit structure and comparing it with the normal circuit. Depending on the detection method, circuit analysis can be divided into three categories: feature-based, testability-based, and structure learning-based. The first feature-based method was proposed in [11], which successfully detects HTs by using the structural features manually extracted from the benchmark netlists. Since then, structural features have been widely used

in HT detection methods, such as support vector machines (SVM) [12], neural networks [13, 14, 15, 16], or random forest[17]. In the past five years, feature-based HT detection methods show high detection performance and can achieve 90% or more accuracy. To ensure detection accuracy, the features used in these methods are obtained by heuristic approaches and need to be continuously updated. However, as the circuit scale increases, it is very time-consuming to obtain the HT features by heuristic methods. Therefore, it is unrealistic to detect HT only using feature-based methods. Since HT is activated only in extremely rare cases and is inserted in areas with poor controllability and observability, the SCOAP[18] is used in the HT detection method. [19] first added the combinational testability measures to the Trojan features, and achieved better accuracy than previous methods. Based on the SCOAP, the nets are clustered into two groups with k-means method[20], and the inter-cluster distance is used as the major feature. Finally, it trained a support vector machine classifier to distinguish the Trojan circuits. To further improve detection performance, [21] extracted the SCOAP values without a limited threshold and used Bagged Trees as the classifier. However, for the HTs with normal triggering probability, the methods with SCOAP will reduce detection accuracy.

To overcome these limitations of feature-based and testability-based HT detection methods, structure learning-based methods are introduced. GramsDet [22] introduces the natural language processing (NLP) technology to the HT detection problems. It first models the circuit using the n-gram circuit segmentation technique and implements the "gate-embedding" by the order-sensitive co-occurrence matrix. Then the HT detection model is built by a recurrent neural network. However, the GramsDet has the problem of the explosion of parameter space and the performance is not good enough. Graph learning is now becoming an active research area in machine learning. Since a circuit can be represented as a graph structure[23], HT detection based graph learning is a promising approach. GNN4TJ [24] first translates register transfer level circuits into a data flow graph, and then trains the model by graph neural networks (GNN). It achieves better classification results by taking advantage of the excellent performance of GNN in non-Euclidean space. Unfortunately, it is only applicable to the RTL design stage. In addition, GNN4TJ only identifies whether the HT exists in the design and does not point out the location of the HT. [25] uses graph convolution neural network to build the detection model of HT trigger, and adds contrastive learning to enhance the recognition ability of GCN, which can accurately identify HT trigger. However, the contrastive learning method is equivalent to oversampling node characteristics, which will seriously affect the generalization ability of its model, and the trigger circuit is easier to be identified than the payload circuit with variable types and unclear HT characteristics. In [32] proposes NHTD-GL, a node-wise HT detection method based on GNN. Graph structure and logic gate behavior using as the node feature, NHTD-GL achieves 0.998 detection accuuracy. This method uses undirected graphs to construct netlists. Although it uses edge features to supplement the information loss of undirected graphs, this method cannot correctly restore the logic level transfer in the circuit.

Therefore, we propose an efficient and effective pre-silicon HT detection method at the gate-level based on the graph neural network. By mapping the netlist to a directed graph, the HT detection problem can be translated into a node classification problem, so this method does not need a golden model as a reference. To maintain the direction of data flow in the netlist, we model the netlist as a directed graph and automatically extract nodes' features based on the graph. In addition, to overcome the problem that the nodes connected to the opposite side and the neighbor side are ignored in the GNN model within a directed graph, we propose a unioned GNN network. With the experiments of different circuit benchmarks, we believe that it is a practical and promising HT detection method. The paper makes the following contributions:

1) The proposed method is golden reference-free and can precisely locate the HTs in netlist.
2) We construct a unioned GNN for HT detection in gate-level netlist. Forward GNN learns the characteristics of gate level information flow, and reverse GNN supplements node neighborhood information.
3) The performance of proposed method is analyzed and compared with the advanced HT detection models.

2. Methodology

Inspired by the machine learning methods for HT detection, our methodology is built on the premise of the existence of a feed-forward function f that determines whether the gate node n in gate-level netlist is a Trojan node or not. The workflow of the proposed HT detection method is shown in Fig. 2.

To improve the scalability of the proposed method, we first translate the input Trojan-in netlist into the technology-independent netlist.Then, the netlist is translated as a directed graph and a transposed graph of it, and the features of nodes are automatically extracted.Thirdly, the directed graph and transposed graph are fed to two different GNN models with the same node features.Finally, the nodes are classified by the aggregated node features.

2.1 Directed Graph Generation

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 HT 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

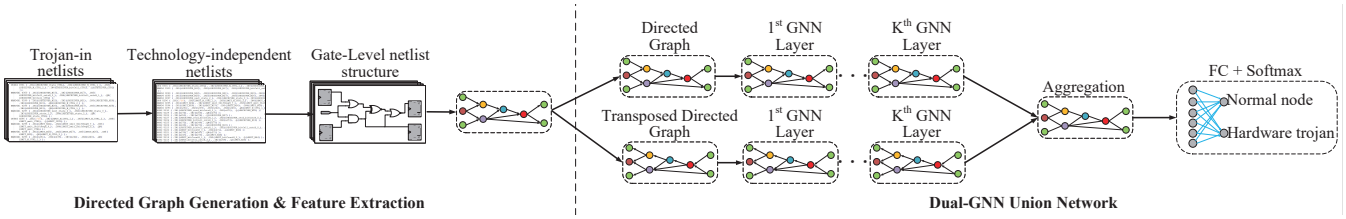

Fig. 2. An overview of the proposed HTs detection method.

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$G = (V, E, F)$, where V is the set of vertices in the directed graph , E is the set of edges in the directed graph and F is the set of nodes' attributes. 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. We define $F = \{f_1, f_2, \ldots, f_j\}$, where f_j is the eigenvector of v_j. Meanwhile, the composition of F will be introduced in the following sections. A processed gate-level netlist and its directed graph are exemplified in Fig. 3.

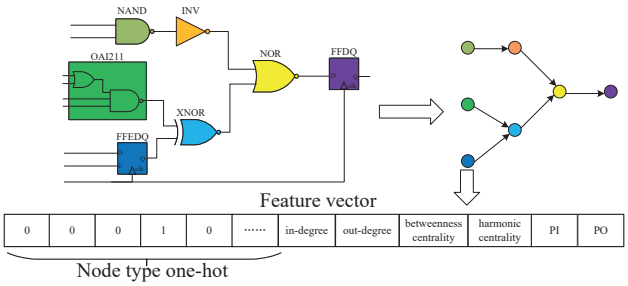

Fig. 3. Translating technology independent netlist as a graph. The node color in the graph represents the cell type. Feature vector extraction and representation for the node in the netlist sub-graph.

2.2 Features Extraction

In [12], 51 structure features are employed for HT detection in total.It extracts features from each net in a netlist from the viewpoint of fan-in, fan-out, minimum levels to the primary node. Based on the analysis of several existing HT detection methods, we divide the features into the following four categories from the viewpoint of a directed graph. (a) Cell type is represented by a one-hot encoding vector whose dimension is associated with the technology-independent cell model built before. (b) Degree is the number of incoming and outgoing neighbors. (c) Primary input is the minimum distance between the node and input. (d) Primary output is the minimum distance between the node and output. To improve the accuracy of HT detection, we add some extra features in addition to the above four features as follows.

- **Betweenness centrality** of a node v is the sum of the fraction of all-pairs shortest paths that pass through the v [26].
- **Harmonic centrality** of a node v is the reciprocal sum of the shortest path distance from all other nodes to the v [27].

Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest paths between any two nodes.Nodes with high betweenness centrality usually have more influence over the information flowing between other nodes.Since Trojan nodes in a design usually have large fan-in logic, the betweenness centrality is expected to be higher for the Trojan nodes than the normal nodes. Harmonic centrality represents the efficiency of node information conduction.Since HTs usually appear as a relatively independent part of the circuit, their distance from other nodes will be larger and their harmonic centrality will be lower than normal nodes'. In addition, considering the large variation in the number of nodes in different netlists, all features of the nodes are normalized to eliminate the adverse effects caused by odd sample data.

2.3 Unioned Graph Neural Network.

In this paper, we leverage GNN to learn the complex, nonlinear relationship between the features and classification of nodes. Our architecture is inspired by the GraphSAGE [28], which is based on sampling and aggregating. It can perform better on inductive inference problems when compared with original Graph Convolution Networks (GCNs)[29]. As mentioned before, to maintain the flow direction of the circuit, the netlist is modeled as a directed graph. HTs are lightweight structures in large-scale IC designs, which commonly contain two components called Trojan trigger (which performs Trojan activation) and harmful circuitry (which performs harmful functions). Therefore, in HT detection, the role of a node depends on its input side, output side, and neighbor side.

However, using the original GraphSAGE model, when aggregating the node features, only the input side and the edge direction are considered in the directed graph. It is therefore necessary to combine information from both directions to generate representative node embedding. Therefore, we propose a unioned graph neural network.

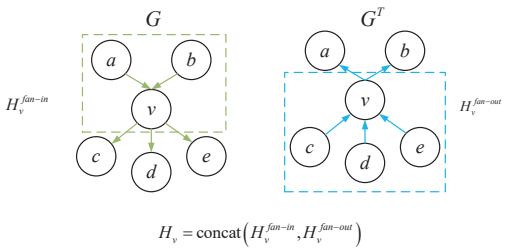

Fig. 4. Final embedding of vertex v. By training two different GraphSAGE models, the information from input side ($H_v^{f\,an-in}$) and output side ($H_v^{f\,an-out}$) is generated respectively. The final embedding H_v is a concatenation of the two representation vectors.

Firstly, we model a new directed graph denoted as $G^T = (V, E^T, F)$, where the E^T is the set of reversed flow direction between the cells. Then we train two GNN models, one for G and the other for G^T, to generate two embedding vectors $H_v^{f\,an-in}$ and $H_v^{f\,an-out}$ for each vertex to aggregate the information from the fan-in cone gates and the fan-out cone gates respectively. In a single GraphSAGE model, the update function of k_{th} GraphSAGE layer messaging is as follows:

$$H_v^k = \sigma\left(\boldsymbol{W^k} \cdot \text{AGG}\left(\{H_v^{k-1}\} \cup \{H_u^{k-1}, \forall u \in N(v)\}\right)\right) \quad (1)$$

where σ is the activation function, $\boldsymbol{W^k}$ is the matrix of a linear transformation on the k_{th} layer for weight learning of GraphSAGE and $N(v)$ is the set of the 1-hop fan-in nodes of node v. AGG is the message aggregation functions, which aggregates the information of node v and the information of $N(v)$. In a word, GNN implements the aggregation and update of the features on the graph and its own node features through the above message passing paradigm.

Fig. 4 explains how the final embedding of each vertex is generated of the proposed model. Thus, the final embedding of each vertex is given by the combination of $H_v^{f\,an-in}$ and $H_v^{f\,an-out}$:

$$H_v = \text{concat}\left(H_v^{f\,an-in}, H_v^{f\,an-out}\right) \quad (2)$$

Finally, the embedded features of nodes are fed to a fully connected layer (FC) with a softmax activation function. The FC layer converts the n-dimensional node embeddings into two-dimensional vectors, and provides the measurement of two kinds of possibility (i.e. [normal node, HT node]) through the softmax function.

3. Experiment and Evaluation

In this section, we evaluate the performance of the proposed HT method and compare it with two baselines: XGBoost[30] and RF [31]. We conduct GNN model training and inference experiments on a computer with an NVIDIA 2080Ti GPU, an Intel(R) Core(TM) i7-10700, and 16-GB memory. The proposed method and our baseline GramsDet and RF both run on the same netlist.

3.1 Dataset and Model

Benchmarks used in this paper are listed in Table I, which contain 17 gate-level netlists from Trust-HUB. The types of HTs covered in the data set include combinational logic, ring inverter and sequential logic. We use python and regex to parse these netlists and transform them into directed graphs. In addition, the proposed method consists of 3 GraphSAGE layers, and an FC layer with softmax activation after aggregation, and Table II defines the GNN model as well as the training architecture.

Table I. Benchmarks overview, sorted by design name.

Design	No. Gates	No. Trojans	Design	No. Gates	No. Trojans
RS232-T1000	215	13	RS232-T1100	216	12
RS232-T1200	216	14	RS232-T1300	222	9
RS232-T1400	215	13	RS232-T1500	216	14
RS232-T1600	214	12	S15850-T100	2182	27
S35932-T100	5456	15	S35932-T200	5438	16
S35932-T300	5462	36	S38417-T100	5341	12
S38417-T200	5344	15	S38417-T300	5373	44
S38584-T100	6472	9	S38584-T200	6556	83
S38584-T300	7204	730	Total	56342	1074

Table II. GNN configuration and training details

Architecture		**Training Setings**	
input-layer	[n, 24]	Optimizer	Adam
hidden-layer units	128	No. of hidden-layer	3
MLP-layer	[128,2]	dropout	0.1
Activation	RELU	learning rate	0.005
Classification	Softmax	n-epochs	200
aggregator type	lstm	loss-fuction	cross-entropy

3.2 HT Detection Results

To evaluate the performance, we perform a leave-one-out cross-validation. GNN models are trained on 15 of the 16 total netlists, leaving 1 netlist outside of the training data set to be the test set data. In this way, we can verify the transferability of the proposed architecture on all 16 circuits.We use true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) to calculate the evaluation metrics. The evaluation metrics used in this paper are recall, true negative rate (TNR), and F-measure. The calculation formulas are expressed as that: Recall is defined as TP/(TP + FN), TNR is defined as TN/(TN + FP), Precision is defined as TP / (TP + FP) ,F-measure is defined as 2*Recall*Precision / (Recall + Precision).

The recall of 100% signifies that all correct Trojan nodes were included in the set of identified Trojan nodes, while the precision of 100% indicates that the set of identified Trojan nodes only includes correct Trojan nodes.And the F-measure is the weighted harmonic mean of recall and precision.

Dataset plays an important role in the performance of our model. As shown in Fig 5, proposed method has higher detection for RS232 because there has more data instances in dataset. Similarly, the performance for other circuits can be enhanced by gathering a more comprehensive dataset with similar types of circuits under detection.

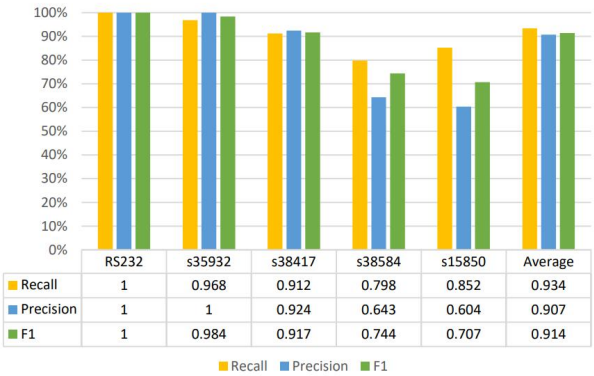

Fig. 5. The performance of unioned GNN in HT detection.

3.3 Unioned GNN Evaluation

In this section, we firstly verify the correctness and effectiveness of proposed unioned GNN model. As shown in Table III, the unioned GNN model outperforms the GraphSAGE model in most of the benchmarks.

To futher illustrate the advantages of the proposed unioned GNN model, we compare the performance of proposed with the existing methods from the perspectives of Recall, TNR and F1-score. Table IV summarizes the comparison results.

Table III. Detection results of benchmarks for the proposed method and GraphSAGE.

Design	Recall		TNR	
	GraphSAGE	Proposed	GraphSAGE	Proposed
RS232-T1000	100%	**100%**	100%	**100%**
RS232-T1100	100%	**100%**	100%	**100%**
RS232-T1200	100%	**100%**	100%	**100%**
RS232-T1300	100%	**100%**	100%	**100%**
RS232-T1400	100%	**100%**	100%	**100%**
RS232-T1500	100%	**100%**	100%	**100%**
RS232-T1600	100%	**100%**	100%	**100%**
S15850-T100	85.2%	**85.2%**	96.7%	**99.3%**
S35932-T100	93.3%	**93.3%**	100%	**100%**
S35932-T200	75.0%	**100%**	100%	**100%**
S35932-T300	94.4%	**97.2%**	100%	**100%**
S38417-T100	50.0%	**91.7%**	100%	**100%**
S38417-T200	99.9%	**100%**	98.8%	**100%**
S38417-T300	81.8%	**81.8%**	99.3%	**100%**
S38584-T100	44.4%	**44.4%**	99.0%	**99.8%**
S38584-T200	59.8%	**96.4%**	100%	**100%**
S38584-T300	85.9%	**98.6%**	99.7%	**99.7%**

As shown in Table IV, the proposed method can achieve an average of 93.4% in recall and 90.7% in precision, and performance better than XGBoost[30],RF[31] and NHTD-GL[32]. Because the dataset in [30] is a subset of dataset in [31] and [32], we chose it as the evaluated design in order to conduct a relatively fair result evaluation. Generally, the number of normal nodes is much larger than that of Trojan nodes in a circuit. For example, in the s38584-T100 benchmark, the ratio of normal nodes to Trojan nodes is 386:1. Therefore, it should be noted that both the recall and F-measure are more important than other metrics.Compared with the SOTA method[32], ours has improved 7.4% on recall and 0.7% on F1. Compared with the baseline, the unioned GNN has greatly improved in precision, which increased by 17.3%, thus driving the improvement of F1. In short, the detection method proposed by this paper can achieve a high detection accuracy and a better trade-off between the recall and precision.

Table IV. Average performance comparison with baselines.

Method	Recall	TNR	PRE	F1
Proposed	**93.4%**	99.9%	90.7%	**91.4%**
baseline	91.6%	99.7%	73.3%	81.6%
NDHL-GL[32]	86.5%	**99.9%**	97.7%	90.7%
XGboost[30]	66.7%	99.9%	96.0%	79.0%
RF[31]	57.7%	**99.9%**	**98.6%**	61.8%

4. Conclusion

In this paper, we proposed a new HT detection method that can capture suspicious circuit structures using GNN.Experimental results indicate that the proposed method can detect HT nodes quickly, and it does not require the "Golden model".In future work, we will further improve the F-measure and reduce the complexity of the proposed architecture and apply the improved algorithm to the complete dataset in [31] .

References

[1] C. Bao, D. Forte, and A. Srivastava, "On reverse engineering-based hardware trojan detection," IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 35, no. 1, pp. 49–57, 2016, doi:10.1109/TCAD.2015.2488495.

[2] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An overview of hardware security and trust: Threats, countermeasures, and design tools," IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., vol. 40, no. 6, pp. 1010–1038, 2021, doi:10.1109/TCAD.2020.3047976.

[3] Z. Huang, Q. Wang, Y. Chen, and X. Jiang, "A survey on machine learning against hardware trojan attacks: Recent advances and challenges," IEEE Access, vol. 8, pp. 10 796–10 826, 2020, doi:10.1109/ACCESS.2020.2965016.

[4] A. Jain and U. Guin, "A novel tampering attack on aes cores with hardware trojans," in 2020 IEEE International Test Conference in Asia (ITC-Asia), 2020, pp. 77–82, doi:10.1109/ITC-Asia51099.2020.00025.

[5] H. Salmani, Salmani, and Glaser, Trusted Digital Circuits. Springer, 2018, doi:10.1007/978-3-319-79081-7.

[6] S. Bhunia and M. Tehranipoor, "The hardware trojan war," Cham,, Switzerland: Springer, 2018, doi:10.1007/978-3-319-68511-3.

[7] R. S. Chakraborty and S. Bhunia, "Security against hardware trojan through a novel application of design obfuscation," in 2009 IEEE/ACM International Conference on Computer-Aided Design-Digest of Technical Papers. IEEE, 2009, pp. 113–116, doi:10.1145/1687399.1687424.

[8] J. Rajendran, V. Vedula, and R. Karri, "Detecting malicious modifications of data in third-party intellectual property cores," in 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC). IEEE, 2015, pp. 1–6, doi:10.1145/2744769.2744823.

[9] X. Guo, R. G. Dutta, Y. Jin, F. Farahmandi, and P. Mishra, "Pre-silicon security verification and validation: A formal perspective," in Proceedings of the 52nd annual design automation conference, 2015, pp. 1–6, doi:10.1145/2744769.2747939.

[10] Q. Shi, N. Vashistha, H. Lu, H. Shen, B. Tehranipoor, D. L. Woodard, and N. Asadizanjani, "Golden gates: A new hybrid approach for rapid hardware trojan detection using testing and imaging," in 2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, 2019, pp. 61–71, doi:10.1109/hst.2019.8741031.

[11] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, "A score-based classification method for identifying hardware-trojans at gate-level netlists," in 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2015, pp. 465–470, doi:10.7873/date.2015.0352.

[12] K. Hasegawa, M. Oya, M. Yanagisawa, and N. Togawa, "Hardware trojans classification for gate-level netlists based on machine learning," in 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS). IEEE, 2016, pp. 203–206, doi:10.1109/iolts.2016.7604700.

[13] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware trojans classification for gate-level netlists using multi-layer neural networks," in 2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS). IEEE, 2017, pp. 227–232, doi:10.1109/iolts.2017.8046227.

[14] T. Inoue, K. Hasegawa, Y. Kobayashi, M. Yanagisawa, and N. Togawa, "Designing subspecies of hardware trojans and their detection using neural network approach," in 2018 IEEE 8th International Conference on Consumer Electronics-Berlin (ICCE-Berlin). IEEE, 2018, pp. 1–4, doi:10.1109/icce-berlin.2018.8576247.

[15] T. Kurihara, K. Hasegawa, and N. Togawa, "Evaluation on hardware-trojan detection at gate-level ip cores utilizing machine learning methods," in 2020 IEEE 26th International Symposium on On-Line Testing and Robust System Design (IOLTS). IEEE, 2020, pp. 1–4, doi:10.1109/iolts50870.2020.9159740.

[16] S. Kundu, X. Meng, and K. Basu, "Application of machine learning in hardware trojan detection," in 2021 22nd International Symposium on Quality Electronic Design (ISQED). IEEE, 2021, pp. 414–419, doi:10.1109/isqed51717.2021.9424362.

[17] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-trojan detection using random forest classifier," in 2017 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2017, pp. 1–4, doi:10.1109/iscas.2017.8050827.

[18] L. H. Goldstein and E. L. Thigpen, "Scoap: Sandia controllability/observability analysis program," in Proceedings of the 17th Design Automation Conference, 1980, pp. 190–196, doi:10.1145/800139.804528.

[19] H. Salmani, "Cotd: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist," IEEE Transactions on Information Forensics and Security, vol. 12, no. 2, pp. 338–350, 2016, 10.1109/tifs.2016.2613842.

[20] X. Xie, Y. Sun, H. Chen, and Y. Ding, "Hardware trojans classification based on controllability and observability in gate-level netlist," IEICE Electronics Express, vol. 14, no. 18, pp. 20 170 682–20 170 682, 2017, doi:10.1587/elex.14.20170682.

[21] C. H. Kok, C. Y. Ooi, M. Moghbel, N. Ismail, H. S. Choo, and M. Inoue, "Classification of trojan nets based on scoap values using supervised learning," in 2019 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2019, pp. 1–5, doi:10.1109/iscas.2019.8702462.

[22] R. Lu, H. Shen, Y. Su, H. Li, and X. Li, "Gramsdet: Hardware trojan detection based on recurrent neural network," in 2019 IEEE 28th Asian Test Symposium (ATS). IEEE, 2019, pp. 111–1115, doi:10.1109/ats47505.2019.00021.

[23] S.-Y. Yu, R. Yasaei, Q. Zhou, T. Nguyen, and M. A. Al Faruque, "Hw2vec: A graph learning tool for automating hardware security," in 2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, 2021, pp. 13–23, doi:10.1109/host49136.2021.9702281.

[24] R. Yasaei, S.-Y. Yu, and M. A. Al Faruque, "Gnn4tj: Graph neural networks for hardware trojan detection at register transfer level," in 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021, pp. 1504–1509, doi:10.23919/date51398.2021.9474174.

[25] N. Muralidhar, A. Zubair, N. Weidler, R. Gerdes, and N. Ramakrishnan, "Contrastive graph convolutional networks for hardware trojan detection in third party ip cores," in 2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). IEEE, 2021, pp. 181–191, doi:10.1109/host49136.2021.9702276.

[26] U. Brandes, "On variants of shortest-path betweenness centrality and their generic computation," Social networks, vol. 30, no. 2, pp. 136–145, 2008, doi:10.1016/j.socnet.2007.11.001.

[27] M. Kitti, "Axioms for centrality scoring with principal eigenvectors," Social choice and welfare, vol. 46, no. 3, pp. 639–653, 2016, doi:10.1007/s00355-015-0931-2.

[28] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," Advances in neural information processing systems, vol. 30, 2017.

[29] W. L. Hamilton, "Graph representation learning," Synthesis Lectures on Artifical Intelligence and Machine Learning, vol. 14, no. 3, pp. 1–159, 2020, doi:10.32657/10356/155039.

[30] M. Hashemi, A. Momeni, A. Pashrashid, and S. Mohammadi, "Graph centrality algorithms for hardware trojan detection at gate-level netlists," International Journal of Engineering, vol. 35, no. 7, pp. 1375–1387, 2022, doi:10.5829/ije.2022.35.07a.16.

[31] T. Kurihara and N. Togawa, "Hardware-trojan classification based on the structure of trigger circuits utilizing random forests," in 2021 IEEE 27th International Symposium on On-Line Testing and Robust System Design (IOLTS). IEEE, 2021, pp. 1–4, doi:10.1109/iolts52814.2021.9486700.

[32] K. Hasegawa, K. Yamashita, S. Hidano, etal , "Node-wise hardware trojan detection based on graph learning," ArXiv, vol. abs/2112.02213, 2021.