26th IEEE International Symposium on On-Line Testing and Robust System Design
(IOLTS)

Evaluation on Hardware-Trojan Detection at Gate-Level IP Cores Utilizing Machine Learning Methods

Tatsuki Kurihara, Kento Hasegawa, Nozomu Togawa
Department of Computer Science and Communications Engineering, Waseda University
Email: {tatsuki.kurihara, kento.hasegawa}@togawa.cs.waseda.ac.jp, ntogawa@waseda.jp

***Abstract*—Recently, with the spread of Internet of Things (IoT) devices, embedded hardware devices have been used in a variety of everyday electrical items. Due to the increased demand for embedded hardware devices, some of the IC design and manufacturing steps have been outsourced to third-party vendors. Since malicious third-party vendors may insert hardware Trojans into their products, developing an effective hardware Trojan detection method is strongly required. In this paper, we evaluate hardware Trojan detection methods using neural networks and random forests at gate-level intellectual property (IP) cores that contain more than 10,000 nets. First, we extract 11 features for each net in a given netlist, and learn them with neural networks and random forests. Then, we classify the nets in an unknown netlist into a set of normal nets and Trojan nets based on the learned classifiers. The experimental results demonstrate that the average true positive rate becomes 84.6% and the average true negative rate becomes 95.1%, which is sufficiently high accuracy compared to existing evaluations.**

***Index Terms*—hardware Trojan, gate-level netlist, machine learning, neural network, random forest**

I. Introduction

Recently, as Internet of Things (IoT) devices become widespread, the demand for embedded hardware devices has been increasing. In order to produce embedded hardware devices more inexpensively, the manufacturing bases have been internationalized, and several processes in the IC design and manufacturing steps have been outsourced to third-party vendors. Under the circumstances, a hardware Trojan, which is a malicious function circuit inserted into a hardware device, may be inserted into IC products by the malicious third-party vendors, and therefore the risk of hardware Trojans has arisen. Hardware Trojans cause system malfunction, performance degradation, and leakage of confidential information. Since there are multiple processes in the IC design and manufacturing step, hardware Trojans may be inserted anywhere in the processes. In particular, attackers can easily tamper the hardware design data such as register-transfer level (RTL) or gate-level netlists by network intrusion, and thus the IC design step involves higher risk than the manufacturing step.

In consideration of this situation, the research on hardware Trojan detection has been actively conducted [1]. However, it has become a problem that a new hardware Trojan is developed immediately after a new hardware-Trojan detection method is proposed. To detect effectively unknown hardware Trojans, a hardware-Trojan detection method using machine learning has been proposed [1]. By effectively utilizing machine learning, we can expect that machine-learning-based hardware-Trojan detection methods efficiently learn the features of hardware Trojans, and detect unknown hardware Trojans. In this paper, we focus on the hardware-Trojan detection method in the IC design step utilizing machine learning methods.

Several hardware-Trojan detection methods utilizing machine learning based on the design information of IC design step have been proposed so far. Ref. [2] proposes a hardware-Trojan detection method using a probabilistic neural network based on the control flow graph of RTL designs. The 16 hardware-Trojan trigger circuits out of the 17 RTL designs can be detected, which is effective for a circuit whose control flow graph consists of 1,000 nodes and edges. Ref. [3] proposes a hardware-Trojan detection method using a neural network based on gate-level netlists. In [3], they successfully distinguish hardware-Trojan infected nets from normal nets. In [3], although an average of 85% of Trojan nets are successfully detected in 17 netlists, the number of nets in a netlist used for performance evaluation is at most 9,000. Recently, due to performance enhancement and multi-functionalization, hardware devices has become complicated and large-scaled. Therefore, in order to show the applicability of the machine-learning-based hardware-Trojan detection methods to practical use, it is quite necessary to examine the effectiveness of machine-learning-based hardware-Trojan detection methods for intellectual property (IP) cores which contain 10,000 or more nets.

In this paper, we use gate-level netlists including seven IP cores which contain 10,000 or more nets and classify them into Trojan nets and normal ones using machine learning. We evaluate the classification performance of machine-learning-based hardware-Trojan detection methods. The experimental results demonstrate that the hardware-Trojan detection methods utilizing machine learning can be effectively applied to gate-level IP cores. With the detection method using the neural networks [3], we have obtained 84.6% as the average true positive rate (TPR) and 95.1% as the average true negative rate (TNR). In the detection method using the random forests [4], we have obtained the average TPR of 48.9% and the average TNR of 100.0%.

The contributions of this paper are summarized as follows:

1) We apply machine-learning-based hardware-Trojan de-

978-1-7281-8187-5/20/$31.00 ©2020 IEEE

TABLE I
THE TRUST-HUB BENCHMARKS USED IN THE EXPERIMENTS.

Benchmark [5]	# of normal net	# of Trojan net
RS232-T1000	309	10
RS232-T1100	309	11
RS232-T1200	310	13
RS232-T1300	309	7
RS232-T1400	306	12
RS232-T1500	311	11
RS232-T1600	311	10
s15850-T100	2,420	26
s35932-T100	6,408	14
s35932-T200	6,405	12
s35932-T300	6,405	37
s38417-T100	5,799	11
s38417-T200	5,802	11
s38417-T300	5,801	44
s38584-T100	7,343	19
s38584-T200	7,373	97
s38584-T300	7,615	873
EthernetMAC10GE-T700	102,969	12
EthernetMAC10GE-T710	102,969	12
EthernetMAC10GE-T720	102,969	12
EthernetMAC10GE-T730	102,969	12
B19-T100	70,649	96
B19-T200	70,649	96
wb_conmax-T100	22,186	11

tection methods using neural networks and random forests to 24 gate-level netlists including the seven IP cores which are not evaluated in the existing methods [3], [4], and evaluate the classification performance.

2) The experimental results demonstrate that the method using neural networks achieves the average TPR of 84.6%, and that using random forests achieves the average TNR of 100.0%. The experimental results also demonstrate that the machine-learning-based hardware-Trojan detection methods can be effectively applied to gate-level IP cores.
3) Examining the experimental results, we discuss what purposes the machine-learning-based hardware-Trojan detection methods using neural networks and random forests can be useful for, and identify the future problems.

II. MACHINE-LEARNING-BASED HARDWARE-TROJAN DETECTION METHOD FOR GATE-LEVEL IP CORES

In this section, we elaborate the gate-level IP cores used in this paper, and the machine-learning-based hardware-Trojan detection methods at gate-level netlists.

A. Gate-Level IP Cores

Refs. [3], [4] evaluate the performance of the methods using netlists with hardware Trojans published at Trust-HUB [5]. Their methods realize the best values among existing machine-learning-based hardware-Trojan detection methods. They utilize small gate-level netlists which contain at most 9,000 nets. In order to put the hardware-Trojan detection method into practical use, evaluation with the netlists that contain 10,000 or more nets is required. Accordingly, we apply these methods to the gate-level IP cores, and evaluate the classification performance. In this section, we use seven gate-level IP cores from [5].

TABLE II
11 TROJAN FEATURES [4].

#	Features	Description
1	fan_in_4	The number of logic-gate fanins up to 4-level away from the input side of the net.
2	fan_in_5	The number of logic-gate fanins up to 5-level away from the input side of the net.
3	in_flipflop_4	The number of flip-flops up to 4-level away from the input side of the net.
4	out_flipflop_3	The number of flip-flops up to 3-level away from the output side of the net.
5	out_flipflop_4	The number of flip-flops up to 4-level away from the output side of the net.
6	in_loop_4	The number of up to 4-level loops on the input side.
7	out_loop_5	The number of up to 5-level loops on the output side.
8	in_nearest_pin	The minimum level to the primary input from the input side of the net.
9	out_nearest_pout	The minimum level to the primary output from the output side of the net.
10	out_nearest_flipflop	The minimum level to any flip-flop from the output side of the net.
11	out_nearest_multiplexer	The minimum level to any multiplexer from the output side of the net.

Table I summarizes the 24 benchmarks used in the experiments. In this paper, we add the seven Trust-HUB benchmarks to the 17 benchmarks used in [3]. In Table I, the top 17 benchmarks are used in [3]. The remaining seven benchmarks are the gate-level IP cores.

EthernetMAC10GE-{T700, T710, T720, T730} are benchmarks for the 10Gbps Ethernet Media Access Controller, which is used to interface to physical layer devices in a 10Gbps Ethernet system. B19-{T100, T200} are included in the ITC'99 benchmarks [6], which are composites of multiple VIPER processors and Intel 386 processors. wb_conmax-T100 is one of the IWLS 2005 benchmarks [7], and is a WISHBONE interconnect matrix IP core. These payload circuits are activated by a specific 8-bit or 16-bit signal vector and counter, and has the function of changing the original functionality by modifying the internal signal.

Each IP core consists of 20,000 to 100,000 gates, as shown in Table I. Among them, the Trojan nets are less than 100, and the proportion of the Trojan nets in one IP core is less than 0.14%. It must be difficult to detect such a small number of Trojan nets from a large-scale IP core, since the hardware Trojans are configured on a very small-scale circuit. In this paper, we apply the machine-learning-based hardware-Trojan detection methods to the gate-level netlists including the above seven IP cores, and evaluate the classification performance.

B. Machine-Learning-Based Hardware-Trojan Detection Method

The hardware-Trojan detection method is divided into two flows: the learning flow and the classification flow.

In the learning flow, we learn known Trojan nets and normal nets, where each net is labeled as either a normal net or a Trojan net, by using the 11 Trojan features listed in Table II [4]. First, we extract the 11 features from each net in the known netlists. After that, we learn the classifier using the extracted 11 features.

In the classification flow, we classify a given unknown netlist into a set of Trojan nets and a set of normal nets using the learned classifier. First, we extract the 11 features from every net in the unknown netlist. After that, we identify each net in the unknown netlist to be a Trojan net or a normal net by using the learned classifier.

In this paper, we utilize neural networks and random forests as classifiers. The parameters and conditions of each classifier are determined as follows.

1) The hardware-Trojan detection method using neural networks: The machine-learning-based hardware-Trojan detection method using neural networks [3] classifies a set of nets into Trojan nets and normal ones based on the features of gate-level netlists utilizing neural networks. Table II shows the 11 Trojan features [4] which characterizes the Trojan net to extract from the gate-level netlists.

Here we use neural networks as a classifier. There are 11 units in the input layer which is the same as the number of the features. The hidden layer is composed of three layers. The numbers of units in the first, second, and third layer are 50, 100, and 200, respectively [3]. The number of units in the output layer is the same as the number of classes to be classified. Here, we set the number of units in the output layer to two: the first unit corresponds to a normal net, and the second unit corresponds to a Trojan net. Each net is classified into the class whose unit outputs a larger value than the other.

The number of Trojan nets is very small compared to the number of normal nets as shown in Table I. In the experiments, after excluding learning data with overlapping feature values, the number of normal nets and that of Trojan nets are weighted so as to be equal, and Trojan nets are learned multiple times.

2) The hardware-Trojan detection method using random forests: The machine-learning-based hardware-Trojan detection method using random forests [4] classifies a set of nets into Trojan nets and normal ones based on the features of gate-level netlists utilizing random forests.

Random forests are one of the machine learning methods. They are an ensemble learning method which decides a prediction class by majority decision from the prediction of many decision tree classifiers. Random forests can obtain more accurate results than a single classifier by integrating prediction results of many decision tree classifiers.

The detection method classifies a set of nets into Trojan nets and normal ones based on the features of gate-level netlists utilizing random forests. First, we calculate the 11 features as shown in Table II for each net in the netlists. Then, the classifier learns the 11 features extracted from the netlists using random forests, and classifies each net in unknown netlist into normal nets and Trojan nets.

The hyper parameters are as follows: the number of estimators is 200, the measures of impurity is entropy, and the maximum depth of the tree is 13. The estimators based on decision tree classifiers are used in random forests, and therefore weighting is not applied.

III. Evaluation Experiments

A. Experimental Results

In this section, we apply the hardware-Trojan detection methods discussed in Section II to the 24 gate-level netlists in the Trust-HUB benchmarks [5] listed in Table I. We use the Python machine learning library scikit-learn for neural network and random forest classifiers.

TABLE III
THE RESULTS OF NEURAL NETWORKS CLASSIFICATION.

Benchmark [5]	TN	FP	FN	TP	TPR	TNR
RS232-T1000	299	10	0	10	100.0%	96.8%
RS232-T1100	298	11	0	11	100.0%	96.4%
RS232-T1200	301	9	0	13	100.0%	97.1%
RS232-T1300	290	19	0	7	100.0%	93.9%
RS232-T1400	300	6	0	12	100.0%	98.0%
RS232-T1500	295	16	0	11	100.0%	94.9%
RS232-T1600	301	10	0	10	100.0%	96.8%
s15850-T100	2,240	180	5	21	80.8%	92.6%
s35932-T100	6,308	100	6	8	57.1%	98.4%
s35932-T200	6,050	355	7	5	41.7%	94.5%
s35932-T300	6,367	38	0	37	100.0%	99.4%
s38417-T100	5,712	87	2	9	81.8%	98.5%
s38417-T200	5,491	311	1	10	90.9%	94.6%
s38417-T300	5,482	319	0	44	100.0%	94.5%
s38584-T100	7,269	74	16	3	15.8%	99.0%
s38584-T200	6,253	850	39	58	59.8%	88.0%
s38584-T300	7,220	395	95	579	85.9%	94.8%
EthernetMAC10GE-T700	102,366	603	4	8	66.7%	99.4%
EthernetMAC10GE-T710	102,524	445	1	11	91.7%	99.6%
EthernetMAC10GE-T720	102,463	506	0	12	100.0%	99.5%
EthernetMAC10GE-T730	101,212	1,757	5	7	58.3%	98.3%
B19-T100	63,275	7,374	0	96	100.0%	89.6%
B19-T200	63,700	6,949	0	96	100.0%	90.2%
wb_conmax-T100	17,188	4,998	0	11	100.0%	77.5%
IP cores average	–	–	–	–	88.1%	93.4%
Total average	–	–	–	–	84.6%	95.1%

TABLE IV
THE RESULTS OF RANDOM FORESTS CLASSIFICATION.

Benchmark [5]	TN	FP	FN	TP	TPR	TNR
RS232-T1000	309	0	0	10	100.0%	100.0%
RS232-T1100	309	0	0	11	100.0%	100.0%
RS232-T1200	310	0	3	10	76.9%	100.0%
RS232-T1300	309	0	2	5	71.4%	100.0%
RS232-T1400	306	0	0	12	100.0%	100.0%
RS232-T1500	311	0	0	11	100.0%	100.0%
RS232-T1600	311	0	6	4	40.0%	100.0%
s15850-T100	2,420	0	24	2	7.7%	100.0%
s35932-T100	6,408	0	14	0	0.0%	100.0%
s35932-T200	6,405	0	11	1	8.3%	100.0%
s35932-T300	6,405	0	24	13	35.1%	100.0%
s38417-T100	5,799	0	11	0	0.0%	100.0%
s38417-T200	5,802	0	11	0	0.0%	100.0%
s38417-T300	5,801	0	13	31	70.5%	100.0%
s38584-T100	7,343	0	19	0	0.0%	100.0%
s38584-T200	7,362	3	85	12	12.4%	100.0%
s38584-T300	7,615	0	870	3	0.3%	100.0%
EthernetMAC10GE-T700	102,969	0	3	9	75.0%	100.0%
EthernetMAC10GE-T710	102,969	0	2	10	83.3%	100.0%
EthernetMAC10GE-T720	102,969	0	2	10	83.3%	100.0%
EthernetMAC10GE-T730	102,969	0	11	1	8.3%	100.0%
B19-T100	70,968	1	0	96	100.0%	100.0%
B19-T200	70,968	1	0	96	100.0%	100.0%
wb_conmax-T100	21,162	3	11	0	0.0%	100.0%
IP cores average	–	–	–	–	64.3%	100.0%
Total average	–	–	–	–	48.9%	100.0%

The experiments are carried out according to the discussion above. In order to evaluate the classification performance of a classifier, we perform leave-one-out cross validation for each netlist, and obtain the average scores. In the leave-one-out cross validation, one of the netlists in Table I is considered to be an unknown netlist and the others are considered to be known netlists.

Table III shows the classification results using neural networks, and Table IV shows the classification results using random forests. In case of binary classification, there are four types of values to evaluate the classification results: the true negative value (TN), the false positive value (FP), the true positive value (TP) and the false negative value (FN).

TABLE V
COMPARISON BETWEEN THE TWO RESULTS IN TERMS OF TPR AND TNR.

Benchmarks	Average TPR		Average TNR	
	NN	RF	NN	RF
IP cores average	88.1%	64.3%	93.4%	100.0%
Average excluding IP cores	83.2%	42.5%	95.8%	100.0%
Total average	84.6%	48.9%	95.1%	100.0%

We have two values to evaluate the classification results: the true positive rate (TPR) and the true negative rate (TNR). TPR is defined as $\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$ and TNR is defined as $\mathrm{TN}/(\mathrm{TN}+\mathrm{FP})$.

B. Comparison

Table V compares the average TPR and TNR of the 24 benchmarks listed in Table I, the seven gate-level IP cores, and the 17 benchmarks excluding the seven gate-level IP cores. "NN" shows the results of neural networks and "RF" shows the results of random forests.

For the seven gate-level IP cores, the detection method using neural networks achieved the average TPR of 88.1% and the average TNR of 93.4%, and the detection method using random forests achieved the average TPR of 64.3% and the average TNR of 100.0%. The results with neural networks show that the average TPR of the seven IP core benchmarks is 4.9 points higher than that of the 17 benchmarks and 3.5 points higher than the total average TPR. The average TNR of the seven IP core benchmarks is 1.7 points lower than the total average TPR. The results with random forests show that the average TPR of the seven IP core benchmarks is 21.8 points higher than that of the 17 benchmarks and 15.4 points higher than the total average TPR. The average TNR of the seven IP core benchmarks is the same as the total average TNR.

These results demonstrate that we successfully detect the Trojan nets in the seven gate-level IP cores, which is almost the same or better than the results of the 17 benchmarks. The comparison results indicate that the machine-learning-based hardware-Trojan detection methods can be effectively applied to the gate-level IP cores.

C. Discussion

We compare the experimental results of the detection methods using neural networks and random forests as shown in Tables III and IV. The detection method using the neural networks tends to achieve higher TPR and the detection method using the random forests tends to achieve higher TNR. The detection method using the neural networks results obtains few FNs and this means that it classifies a small number of Trojan nets as normal nets mistakenly. The detection method using the random forests obtains few FPs and this means that it classifies a small number of normal nets as Trojan nets mistakenly. However, from the viewpoint of security, classifying Trojan nets into normal nets mistakenly should be avoided. The detection method using only random forests may cause a problem in this point.

Since the experimental results demonstrate that the detection method using the neural networks tends to achieve higher TPR, it can detect effectively the hardware-Trojan infected nets. In addition, the nets classified as Trojan nets by the neural networks include definitely Trojan nets. The more effective hardware-Trojan detection is expected by combining the detection method using the neural networks with the existing hardware-Trojan detection method. The detection method using the random forests obtains few FPs, which demonstrates that the nets classified as Trojan nets by the random forests are Trojan nets with high probability. Therefore, the detection method using the random forests is useful for determining whether or not the hardware-Trojan is inserted in the circuit. In this way, the detection methods using the neural networks and the random forests have the strong and weak points, and thus the more effective hardware-Trojan detection is expected to be constructed by combining the respective advantages of the neural networks and the random forests. This is our important future work.

IV. CONCLUSIONS

In this paper, we discuss the hardware-Trojan detection methods utilizing neural networks and random forests, and evaluate them with the 24 benchmarks including the gate-level IP cores. The experimental results demonstrate that the method using the neural networks achieved the average TPR of 84.6% and the average TNR of 95.1%, and the method using the random forests achieved the average TPR of 48.9% and the average TNR of 100.0%. The experimental results also demonstrate that the machine-learning-based hardware-Trojan detection methods can be effectively applied to the gate-level IP cores that contain 10,000 or more nets.

In the future, we will consider how to effectively combine the machine-learning-based hardware-Trojan detection methods so as to improve TPR and TNR. We also consider the machine-learning-based hardware-Trojan detection methods other than neural networks and random forests.

ACKNOWLEDGMENT

This paper was supported in part by Grant-in-Aid for Scientific Research (No. 19H04080).

REFERENCES

[1] R. Elnaggar and K. Chakrabarty, "Machine learning for hardware security: opportunities and risks," *Journal of Electronic Testing*, vol. 34, no. 2, pp. 183–201, 2018.

[2] F. Demrozi, R. Zucchelli, and G. Pravadelli, "Exploiting sub-graph isomorphism and probabilistic neural networks for the detection of hardware Trojans at RTL," in *Proc. IEEE International High Level Design Validation and Test Workshop (HLDVT)*, 2017, pp. 67–73.

[3] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Hardware Trojans classification for gate-level netlists using multi-layer neural networks," in *Proc. IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2017, pp. 227–232.

[4] ——, "Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier," in *Proc. IEEE International Symposium on Circuits and Systems*, 2017, pp. 1–4.

[5] "Trust-HUB," https://www.trust-hub.org/.

[6] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Design and Test of Computers*, vol. 17, no. 3, pp. 44–52, 2000.

[7] "IWLS 2005 Benchmarks," https://iwls.org/iwls2005/benchmarks.html.