Trojan-Net Classification for Gate-Level Hardware Design Utilizing Boundary Net Structures

Kento HASEGAWA¹,a, Member, Masao YANAGISAWA¹, Fellow, and Nozomu TOGAWA¹, Senior Member

SUMMARY Cybersecurity has become a serious concern in our daily lives. The malicious functions inserted into hardware devices have been well known as hardware Trojans. In this letter, we propose a hardware-Trojan classification method at gate-level netlists utilizing boundary net structures. We first use a machine-learning-based hardware-Trojan detection method and classify the nets in a given netlist into a set of normal nets and a set of Trojan nets. Based on the classification results, we investigate the net structures around the boundary between normal nets and Trojan nets, and extract the features of the nets mistakenly identified to be normal nets or Trojan nets. Finally, based on the extracted features of the boundary nets, we again classify the nets in a given netlist into a set of normal nets and a set of Trojan nets. The experimental results demonstrate that our proposed method outperforms an existing machine-learning-based hardware-Trojan detection method in terms of its true positive rate.

key words: hardware Trojan, gate-level netlist, Trojan feature, boundary nets, hardware design

1. Introduction

Since the prevalence of these hardware devices has been increased, cybersecurity has become a serious concern in our daily lives. Because hardware vendors are required to offer numerous hardware products at a low price to their consumers all over the world, they often outsource several parts of their products to third-party vendors including overseas vendors. Under the circumstances, the security of hardware devices is often disregarded. Now we must enhance the reliability of hardware devices. In order to tackle the problem, hardware-Trojan detection methods have recently been proposed [1], [2]. There are several categories in hardware-Trojan detection approaches [3]. To defeat the threats as early as possible, we focus on the ‘IP Trust Verification’ approach to address the threats in the hardware design step. Several hardware-Trojan detection methods focused on the hardware design step have been proposed. A hardware-Trojan detection method at gate-level netlists based on the structure of a circuit has been proposed [4]. In [4], several features that describe the characteristics of Trojan nets (the nets consist of a hardware Trojan) are extracted. This method suggests that Trojan nets have several peculiar features and we can effectively detect hardware Trojans by using them. However, in [4], the features and scores are manually determined by referencing several benchmarks, and therefore it is overly specific to the referenced benchmarks. Machine-learning-based hardware-Trojan detection methods have also been proposed in recent years [5]. In [6] and [7], hardware-Trojan detection methods based on the features of hardware designs have been proposed. In [6] and [7], Trojan features which can be best applied to machine-learning-based hardware-Trojan detection methods are extracted. Although their methods with the features successfully detect hardware Trojan nets, a small number of Trojan nets are mistakenly identified to be normal nets.

In this letter, we propose a hardware-Trojan detection method utilizing boundary net structures based on machine-learning-based hardware-Trojan detection methods**, and aim to enhance the classification performance. First, we obtain hardware-Trojan classification results leveraging an existing machine-learning-based hardware-Trojan detection method and investigate nearby nets connected to/from the nets that are mistakenly identified to be Trojan nets or normal nets. Based on the investigation above, we extract the structures of the mistakenly-identified nets. Finally, we propose a hardware-Trojan detection method based on the structure of the boundary nets.

The contributions of our proposed method are summarized as follows: 1) We investigate the classification results obtained by a machine-learning-based hardware-Trojan method, and newly extract the features of boundary nets between Trojan nets and normal nets. Based on the features, even if the boundary Trojan nets are mistakenly identified to be normal nets by a machine-learning-based hardware-Trojan detection method, we can successfully correct the classification results to Trojan nets; 2) The experimental results demonstrate that our proposed method successfully identifies most of the Trojan nets to be Trojan nets correctly compared to an existing method. Furthermore, the extracted boundary net structures can be successfully applied to unknown netlists that are not included in the investigated netlists.

2. Related Works

2.1 Hardware-Trojan Detection Using Machine Learning Techniques

In this letter, we focus on the hardware-Trojan detection in the hardware design step. Attackers can be categorized into two types: an untrusted third-party IP vendor and an outside attacker. The untrusted third-party IP vendor may pro-

**The preliminary version of this letter appeared in [8].
provide a hardware-Trojan-infected IP to the hardware vendor. The outside attacker may attack hardware design information which is stored as electrical information through the Internet. Our proposed method is intended to be performed in the hardware design step to detect hardware Trojans.

To begin with, we define several evaluation indicators used in machine-learning-based hardware-Trojan detection. Trojan nets correctly identified to be Trojan nets are called true positives. TP shows the number of true positives. Trojan nets mistakenly identified to be normal nets are called false negatives. FN shows the number of false negatives. TN and FP are defined similarly. Based on the definitions above, the true positive rate (TPR) is defined by TP/(TP + FN), and the true negative rate (TNR) is defined by TN/(TN + FP). The accuracy is calculated by (TP + TN)/(TP + FN + TN + FP).

Based on the Trojan net features proposed in [4], a machine-learning-based hardware-Trojan detection method has been proposed in [7]. In [7], 51 Trojan features are first extracted from the target net n in a gate-level netlist. The 51 Trojan features are related to the number of fan-ins and nearby flip-flops, multiplexers, and constants. Next, 11 Trojan features are selected using a random forest classifier with maximizing F-measures. The experimental results demonstrate that the average TPR, average TNR, and average accuracy become 68.3%, 99.7%, and 99.1%, respectively.

In the random-forest-based approach, though the average TPR is low, the average TNR becomes almost 100% and the average accuracy becomes 99%. In order to further enhance the performance of hardware-Trojan detection, re-investigating the results of the random-forest-based approach is effective because the average accuracy is 99%, which means that almost all of the nets are correctly classified as either Trojan nets or normal nets.

2.2 Hardware-Trojan Detection Based on the Random Forest Classifier and Its Limitation

Based on the discussion above, the random-forest-based approach is applicable to hardware-Trojan detection. Table 1 shows the classification results based on the random-forest-based approach [7]. In Table 1, both FP and FN in all of the benchmarks are less than 20, and this number is much smaller than the number of normal nets (TN + TP). In particular, in RS232-T1300, because both FP and FN are 0, the classifier completely classifies all of the nets in the netlist into a set of normal nets and Trojan nets. On the contrary, in RS232-T1200, several Trojan nets are mistakenly identified as normal nets, whereas all of the normal nets are correctly identified as normal nets. In hardware-Trojan detection, maximizing TPR is the first priority. In this letter, we focus on RS232 series benchmarks (RS232-T1000–RS232-T1600) to carefully investigate the netlists, and aim to increase TPR.

3. Hardware Trojan Detection Based on Boundary Net Structures

3.1 Boundary Nets and Boundary Net Structures

In this letter, we aim to correct the mistakenly-identified nets by a machine-learning-based hardware-Trojan detection method.

Here, we define boundary nets as the nets between normal nets and Trojan nets. Figure 1 illustrates the model of a hardware Trojan and boundary nets. A hardware Trojan typically consists of a trigger circuit and a payload circuit. The trigger circuit judges whether the trigger condition is satisfied or not. The payload circuit is the malicious function for the hardware Trojan. Both of the trigger circuit and payload circuit are composed of multiple gates. Thus, a hardware Trojan circuit contains multiple gates that are located together. Now, we define the boundary net as the nets between normal nets and Trojan nets. The red nets and the green net in Fig. 1 are the boundary nets. Then, we also define the boundary net structure as a set of nets which contains boundary nets and their adjacent nets. The nets included in the boundary net structure are shown as blue lines in Fig. 1.

We assume that the nets in a netlist are once classified into a set of normal nets and that of Trojan nets by a machine-learning-based hardware-Trojan detection method. Then, as shown in Fig. 1, the green net is identified to be a normal net mistakenly, and the red and blue nets are correctly identified to be Trojan nets. By just judging based on the classification result, the green net is identified to be a normal net. However, when we inspect the boundary net...
In this section, we analyze the classification results obtained in [7], and extract the features of mistakenly identified nets and their nearby nets. We have focused on the RS232 series benchmarks, which can be obtained in [9], listed in the top seven benchmarks of Table 1. Here, we investigate the Trojan nets mistakenly identified as normal nets (we define these nets as “FN nets”).

Table 2 shows the details of the 27 FN nets in [7], and the number of Trojan nets and normal nets connected to/from up to two-level away from the net. In Table 2, the “Trojan net” means the net identified to be a Trojan net by the classifier, and the “normal net” means the net identified to be a normal net by the classifier. As shown in Table 2, most of the nets have no Trojan nets two-level away to them. This means that their nets are placed near the boundary between Trojan nets and normal nets.

In order to simplify the analysis, we adopt the netlists whose FNs are the first and second largest in Table 2: RS232-T1100 and RS232-T1200.

Figure 2 shows the FN nets in RS232-T1100. The hardware Trojan inserted into RS232-T1100 has a three-logic-level combinational circuit. In the classification result, the classifier could not identify the nets around the boundary between Trojan nets and normal nets, as Trojan nets. However, their succeeding nets are correctly identified to be Trojan nets. We can identify the FN nets to be Trojan nets correctly if we investigate their succeeding nets.

Figure 3 shows the FN nets in RS232-T1200. As shown in Fig. 3, all the FN nets in RS232-T1200 are placed around the boundary between Trojan nets and normal nets. The three FN nets are connected to the trigger circuit of the hardware-Trojan circuit. The other FN net is connected to a scan-input signal of the flip-flop in the Trojan nets. The nets connected to/from the flip-flop, which is connected from the net iRECEIVER_rec.datH, are identified as Trojan nets except for the net iRECEIVER_rec.datH, and thus we can guess that the net iRECEIVER_rec.datH is a Trojan net. The NAND gate connected to the nets iXMIT_next_state_2.Next, iXMIT_state_1.Next, and iXMIT_state_0.Next has a fan-out that is identified to be a Trojan net, and it has four fan-ins, and therefore we can guess that the nets iXMIT_next_state_2.Next, iXMIT_state_1.Next, and iXMIT_state_0.Next are the trigger-condition signals for the hardware Trojan. These features can be found in the net iRECEIVER_rec.datSyncH of RS232-T1400 and the net iXMIT_bitCell_cntrH_1.Next of RS232-T1500. The above discussion can be applied to other
FN nets in RS232-T1500 and RS232-T1600.

3.3 Boundary Net Structures for Hardware-Trojan Detection

Essentially, the nets that consist of a hardware Trojan circuit are concentrated at one position in a normal circuit. As discussed in Sect. 3.1, if we can identify most of the nets in a boundary net structure as Trojan nets correctly, we can recognize that the normal nets close to the Trojan nets should be Trojan nets. In this sense, considering the boundary net structure is helpful to correct classification results. Therefore, we check if the boundary nets are Trojan or not once again. Based on the classification results obtained by [7], we correct the classification focusing on the following structures:

1. **Trigger circuit structure**: If the target normal net $n$ is connected to a logic gate $g$ that has at least one Trojan net as a fan-in or fan-out, we identify the normal net $n$ to be a Trojan net. For example, in Fig. 4 a), the net $n$ will be identified to be a Trojan net even if $n$ is mistakenly identified to be a normal net by [7];

2. **Flip-flop structure near a Trojan net**: If the target normal net $n$ is connected to a flip-flop $f$, and the flip-flop is also connected only to a logic gate $g$, that has four or more fan-ins and its fan-out is a Trojan net, we identify the normal net $n$ to be a Trojan net. For example, in Fig. 4 b), the net $n$ is identified to be a Trojan net even if $n$ is mistakenly identified to be a normal net by [7].

By utilizing the boundary net structures listed above, the classification procedure is carried out as follows:

1. **Machine-learning-based hardware-Trojan classification**: At first, an existing machine-learning-based hardware-Trojan classification method classifies the nets in several benchmark netlists on [9]. After the classification, we obtain the classification results (we call them “preliminary results”).

2. **Correction of preliminary results**: After we obtain the preliminary results, we apply the boundary net structures defined above to the preliminary results. Finally, we obtain the corrected classification results.

4. Experimental Results

4.1 Setup

In order to obtain the preliminary results, we used the results of [7]. Based on the preliminary results, we correct the classification results utilizing the extracted boundary net structures. The algorithm of the correction step is implemented using Python 3 with Pyverilog [10].

4.2 Experimental Results Using Benchmarks

1. **Correction of the reference-included benchmarks**

   First, we adopted several benchmarks that are used to extract the boundary net structures. As mentioned in Sect. 3.2, we refer to the RS232 series benchmarks, which are listed in the seven rows from the top of Table 1, to extract the boundary net structures. Table 3 shows the results on the RS232 series benchmarks. The experimental results demonstrate that four benchmarks obtained 100% TPRs. All of the accuracies become more than 93%. In hardware-Trojan detection, maximizing the TPR is important. In this sense, our method increases the TPR, which certainly leads to a good result.

   Table 4 shows the comparison between our proposed method and [7]. Our proposed method decreases FNs, and hence the average TPR is increased. These results indicate that our proposed method effectively and correctly identifies most of the Trojan nets as Trojan nets.

2. **Correction of the other benchmarks than reference-included ones**

   In order to evaluate the effectiveness and generality of the proposed method, we focus on the benchmarks that are not referenced when extracting the boundary net structures. The benchmarks are listed in the lower eight rows in Table 1. Table 5 shows the classification results for these benchmarks.
Table 3: Experimental results for each benchmark.

| Benchmark   | TN  | FP  | FN  | TP  | TPR  | TNR  | Accuracy |
|-------------|-----|-----|-----|-----|------|------|----------|
| RS232-T1000 | 278 | 5   | 0   | 36  | 100.0% | 98.2% | 98.4% |
| RS232-T1100 | 275 | 9   | 11  | 25  | 69.4%  | 96.8% | 93.8% |
| RS232-T1200 | 277 | 12  | 0   | 34  | 100.0% | 95.8% | 96.3% |
| RS232-T1300 | 286 | 1   | 0   | 29  | 100.0% | 99.7% | 99.7% |
| RS232-T1400 | 265 | 8   | 0   | 45  | 100.0% | 97.1% | 97.5% |
| RS232-T1500 | 276 | 7   | 1   | 38  | 97.4%  | 97.5% | 97.5% |
| RS232-T1600 | 287 | 5   | 1   | 28  | 96.6%  | 98.3% | 98.1% |

Table 4: Comparison between our method and [7] on the RS232 series benchmarks.

| Method               | Ave. TPR | Ave. TNR | Ave. Accuracy |
|----------------------|----------|----------|---------------|
| [7]                  | 89.1%    | 99.4%    | 98.3%         |
| Ours                 | 94.8%    | 97.6%    | 97.5%         |

Table 5: Experimental results for the benchmarks other than reference-included ones.

| Benchmark   | TN  | FP  | FN  | TP  | TPR  | TNR  | Accuracy |
|-------------|-----|-----|-----|-----|------|------|----------|
| s15850-T1100 | 2,418 | 1  | 2   | 25  | 92.6%  | 100.0% | 99.9% |
| s35932-T1100 | 6,407 | 0  | 3   | 12  | 80.0%  | 100.0% | 100.0% |
| s35932-T200  | 6,405 | 0  | 11  | 1   | 8.3%  | 100.0% | 99.9%  |
| s35932-T300  | 6,403 | 2  | 5   | 32  | 86.5%  | 100.0% | 99.9%  |
| s38417-T1100 | 5,798 | 0  | 7   | 5   | 41.7%  | 100.0% | 99.9%  |
| s38417-T200  | 5,798 | 0  | 6   | 9   | 60.0%  | 100.0% | 99.9%  |
| s38417-T300  | 5,800 | 1  | 11  | 33  | 75.0%  | 100.0% | 99.8%  |
| s35854-T1000 | 7,341 | 2  | 18  | 1   | 5.3%   | 100.0% | 99.7%  |

Table 6: Comparison between our method and [7] on all the benchmarks.

| Method               | Ave. TPR | Ave. TNR | Ave. Accuracy |
|----------------------|----------|----------|---------------|
| [7]                  | 68.3%    | 99.7%    | 99.1%         |
| Ours                 | 74.2%    | 98.9%    | 98.7%         |

Note that the benchmarks whose FPs become non-zero obtain 100.0% TNR and accuracy values because of rounding. The experimental results demonstrate that four out of eight benchmarks become greater than 75% TPR. All of the benchmarks obtained higher or equal TPRs compared to the results in [7].

Table 6 shows the comparison between the proposed method and [7] on all the benchmarks in [7] listed in Table 1. In Table 6, each value means the average value of all the benchmarks. As shown in Table 6, the average TPR in our proposed method is larger than that in [7]. Increasing the TPR is the most important goal because it is preferred for us to detect all parts of Trojan nets in hardware-Trojan detection. Even though several normal nets are mistakenly identified as Trojan nets (in other words, TNR is increased), we can screen them by only analyzing a few of the Trojan-identified nets, which costs less time compared to analyzing all the nets.

As described in Sect. 3.3, the benchmarks listed in Table 6 are not investigated when we extract the boundary net structures. However, the experimental results here demonstrate that the proposed method is effectively applied to the non-referenced benchmarks. This is because existing machine learning methods often misclassify the boundary nets between normal ones and Trojan ones, and our proposed method successfully corrects the misclassification.

5. Conclusions

In this letter, we proposed a hardware-Trojan detection method utilizing boundary net structures. The experimental results demonstrate that our method outperforms an existing machine-learning-based hardware-Trojan detection method in terms of TPR. For the practical use of hardware-Trojan detection, increasing TNR is also important, and this remains for our future work. In addition, if an intruder would be able to know the classifier, several attack schemes against machine learning can be applied. For example, an adversarial example attack is a famous attack scheme. An adversarial example generation method against hardware Trojan detection has been proposed very recently [11]. Defeating such attacks is also one of our future works.

Acknowledgments

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

References

[1] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, “Hardware Trojans: lessons learned after one decade of research,” ACM Trans. Design Automation of Electronic Systems, vol.22, no.1, pp.1–23, 2016.
[2] M. Rostami, F. Koushanfar, and R. Karri, “A primer on hardware security: models, methods, and metrics,” Proc. IEEE, vol.102, no.8, pp.1283–1295, Aug. 2014.
[3] S. Bhunia, M.S. Hsiao, M. Banga, and S. Narasimhan, “Hardware Trojan attacks: threat analysis and countermeasures,” Proc. IEEE, vol.102, no.8, pp.1229–1247, Aug. 2014.
[4] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, “A score-based classification method for identifying hardware-trojans at gate-level netlists,” Proc. Design, Automation and Test in Europe (DATE), pp.465–470, 2015.
[5] R. Elnaggar and K. Chakrabarty, “Machine learning for hardware security: opportunities and risks,” J. Electronic Testing, vol.34, no.2, pp.183–201, 2018.
[6] K. Hasegawa, M. Yanagisawa, and N. Togawa, “A hardware-Trojan classification method using machine learning at gate-level netlists based on Trojan features,” IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences, vol.E100-A, no.7, pp.1427–1438, 2017.
[7] 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,” Proc. International Symposium on Circuits and Systems, pp.2154–2157, 2017.
[8] K. Hasegawa, M. Yanagisawa, and N. Togawa, “A hardware-Trojan classification method utilizing boundary net structures,” IEEE Int. Conf. Consum. Electron. (ICCE), pp.103–106, 2018.
[9] “Trust-HUB.” http://www.trust-hub.org
[10] S. Takamaeda-Yamazaki, “Pyverilog: a python-based hardware design processing toolkit for Verilog HDL,” Proc. International Symposium on Applied Reconfigurable Computing, pp.451–460, 2015.
[11] K. Nozawa, K. Hasegawa, S. Hidano, S. Kiyomoto, K. Hashimoto, and N. Togawa, “Adversarial examples for hardware-Trojan detection at gate-Level netlists,” Proc. International Workshop on Attacks and Defenses for Internet-of-Things (ADfIoT), pp.1–18, 2019.