Hardware Trojan Detection Method for Inspecting Integrated Circuits Based on Machine Learning

Yuze Wang  
College of Information Science and Electronic Engineering  
Zhejiang University  
Hangzhou, China  
ORCID: 0000-0002-8575-3508

Xiaoxia Han  
College of Information Science and Electronic Engineering  
Zhejiang University  
Hangzhou, China

Peng Liu  
College of Information Science and Electronic Engineering  
Zhejiang University  
Hangzhou, China  
ORCID: 0000-0001-9107-6673

Yingtao Jiang  
Department of Electrical and Computer Engineering  
University of Nevada Las Vegas  
Las Vegas, USA

Abstract—Nowadays malicious vendors can easily insert hardware Trojans into integrated circuit chips as the entire integrated chip supply chain involves numerous design houses and manufacturers on a global scale. It is thereby becoming a necessity to expose any possible hardware Trojans, if they ever exist in a chip. A typical Trojan circuit is made of a trigger and a payload that are interconnected with a trigger net. As trigger net can be viewed as the signature of a hardware Trojan, in this paper, we propose a gate-level hardware Trojan detection method and model that can be applied to screen the entire chip for trigger nets. In specific, we extract the trigger-net features for each net from known netlists and use the machine learning method to train multiple detection models according to the trigger modes. The detection models are used to identify suspicious trigger nets from the netlist of the integrated circuit under detection, and score each net in terms of suspiciousness value. By flagging the top 2% suspicious nets with the highest suspiciousness values, we shall be able to detect majority hardware Trojans, with an average accuracy rate of 96%.

Index Terms—hardware Trojan detection, trigger net, detection model, gate-level, machine learning

I. INTRODUCTION

With the globalization of integrated circuit (IC) design and manufacturing, the real-world chip design and manufacturing process requires the participation of designers and suppliers from multiple parties. During this period, it may be threatened by hardware Trojans inserted from any part [1]. Due to the high complexity of real-world chip circuits in terms of scale and structure, it is necessary to study specified hardware Trojan detection methods. At the gate-level of IC design, gate-level hardware Trojans may be inserted by designers or intellectual property vendors or tool vendors. Such hardware Trojans may complicate functional verification, code inspection at the register transfer level. Moreover, it is too difficult to extract the circuit structure and function information at the layout level for the purpose of hardware Trojan detection. As a result, it is more preferable that hardware Trojan detection needs to be carried out at the gate-level.

The main difficulties in detecting the hardware Trojans in a real IC chip design come from the large circuit size and complexity of the circuit structure. There may be a circuit structure with high similarity to the Trojan circuit, which can easily be misjudged as a Trojan circuit. In addition, there may be various hardware Trojans that may be inserted in real-world chips, and there is no golden reference model for different chip designs. To solve the problem, in this paper we propose a trigger-net-based gate-level hardware Trojan detection method and model for inspecting chip circuits.

For the problem of similar structural characteristics between normal circuits and hardware Trojan circuits in a real-world chip circuit, we choose the trigger net in hardware Trojans as detection target to locate potential Trojan circuits. The trigger net is used to interconnect the trigger circuit (Trigger) and the load circuit in a hardware Trojan circuit, and is activated when the unique trigger condition is met. Trigger net has more distinctive structure and functional characteristics than other nets in Trojan circuits, which can be further distinguished from normal circuit features and reduce misjudgments. In order to create a general detection model for different types of hardware Trojans, improving the accuracy and coverage of detection, we formed a detection method for two types of hardware Trojans respectively triggered by combinational logic and sequential logic, and constructed the corresponding detection models using machine learning algorithms.

Thus, we firstly analyze the peculiarity of trigger nets on structure and extract five trigger-net features from each net in each training netlist. Next, we construct two sets of training data consisting of five feature values and labels, which flag whether a net is a trigger net or not, of each net from two types of known netlists, respectively. Then, we train these two sets of training data using the ensemble learning method to obtain two trained detection models. Finally, input the feature values of each net in a real-world chip netlist into these two detection models. The detection models are expected to give suspiciousness values of each net suspected of being a
trigger net respectively, and the suspicious trigger nets can be identified by synthesizing the results of these two models.

The rest of this paper is organized as follows. Section II shows the related work. Section III describes the gate-level hardware Trojan detection method for real-world chip circuits. Section IV demonstrates the experimental results. Section V concludes this paper.

II. RELATED WORK

In the IC design phase, hardware Trojans can be detected by performing static analysis on hardware description language (HDL) source code of the circuit. Like software virus detection technique, static structure analysis methods detect hardware Trojans by analyzing the circuit structure characteristics. Although the static structure analysis is an effective hardware Trojan detection approach, it can only detect known types of hardware Trojans. According to the idea of static analysis approach, since intrinsic differences exist between Trojan logics and normal circuit, statistical feature analysis approaches can be used to detect potential hardware Trojans in circuits under detection. At the gate-level, it is a classification problem to identify the Trojan structure in the netlist. Based on the static structure analysis and statistical feature analysis, the machine learning algorithms are effective to solve the classification problem. Hasegawa et al. detected the possible Trojan components by machine learning methods to classify nets in a netlist into a set of Trojan nets which belong to the Trojan structure and a set of normal nets. An unsupervised-learning-based hardware Trojan detection approach was proposed recently to detect abnormal objects that do not appear in the training set.

However, these works only consider the detection scale of module circuits, and lack the detection scheme and experimental demonstration for the real-world chip scenarios. Thus, we propose a trigger-net-based hardware Trojan detection method using multiple types of detection models to inspect chip circuits.

III. GATE-LEVEL HARDWARE TROJAN DETECTION METHOD FOR REAL-WORLD CHIP CIRCUITS

A. Overall Flow of the Detection Method

We have established a gate-level hardware Trojan detection method and model construction method. In order to effectively identify the hardware Trojans from the sea of digital gates, we select the trigger net with the most distinctive features in the Trojan circuit as the detection target. In order to form a universal detection model and effectively detect a variety of unknown hardware Trojans, we collect samples of multiple types of Trojan circuits and train them to construct detection models through machine learning algorithms, so that the detection models own better generalization capability for different circuits and different types of Trojans. As shown in Fig. 1, the overall detection flow is divided into two phases, the training phase and the detection phase.

During the training phase, the first step is to classify by model. We collect sets of Trojan-infected netlist samples, and by analyzing the structure and functional characteristics of the hardware Trojans in these netlists, we divide them to obtain multi-class sample sets for building multiple detection models. The second step is netlist parsing and feature extraction. Through the analysis of the structure and function characteristics of the Trojan circuits and trigger nets, we select multiple features. Then for each netlist in the sample sets, we parse the netlist to obtain the information of the circuit structure, and then extract the corresponding feature vector for each signal net according to the selected features. The third step is machine learning training. First, select the appropriate machine learning algorithm. In each type of model, the feature vector of each signal net in all netlists and the corresponding label of whether it is a trigger net are used for training, then multiple detection models can be obtained.

After the training is completed, in the detection phase, we parse the netlist of the chip circuit under detection to obtain circuit structure information. The feature values of each net in the netlist under detection are parsed and input into each detection model. The detection models are expected to give results of how a net is suspected to be a trigger net. By comprehensively analyzing the results of all detection models, detection results of suspicious trigger nets can be obtained.

B. Trigger-Net Detection and Classified Models

Trigger net is an interconnection between the trigger and the payload in a Trojan circuit, as shown in Fig. 2. It is activated when a specifically unique trigger condition is met. That means the trigger net plays a role as a significant confluence of pre-stage signals, representing a tapered fan-in structure. After the activation, the trigger net drives the payload to realize malicious functions, which are generally some modifications on primary outputs or internal registered values. Based on these characteristics, the trigger net is expected to show a remarkable distinction from other signal nets on structural features, more suitable as the detection target to reduce false positive.

The forms of hardware Trojans that may be inserted in real-world chips can be various, and it is difficult to achieve comprehensive detection for multiple types of hardware Trojans by relying on only one type of detection model. We adopt the method of collaborative detection and comprehensive analysis of multiple detection models. By dividing the samples into multiple sets of classified samples under various models, detection models can be constructed using the corresponding sets of samples to realize the detection for various types of hardware Trojans. Since we use trigger net as the detection target, considering the influence of different trigger modes on the structural characteristics of trigger net, we classify the sample circuits according to their trigger modes. We use 17 gate-level netlists in the Trust-HUB benchmarks as the sample circuits to train detection models for inspecting chip circuits. According to their trigger modes, they can be divided into two categories: triggered by combinational logic and triggered by sequential logic. Among them, s15850-T100, s35932-T100, s38584-T200, and s38584-T300 circuits
are sequential logic types, and the rest are combinational logic types.

C. Netlist Parsing and Feature Extraction

In order to extract feature values of circuits for model training and hardware Trojan detection, it is necessary to parse the netlist to obtain a digital representation of the netlist. First, we collect the cell library information of the netlist by traversing the netlist, including the name of each gate cell and its input and output pin names. Then we parse the netlist according to the cell library information, including the module’s input and output, pin assignment, gate cell information, signal net information, and the connections among gate cells and signal nets.

After parsing the netlist, feature values can be extracted based on the structure information. Since we choose the trigger nets as the detected target and use the machine learning algorithms, it is required to select features which are expected to represent the structural particularities of the trigger nets. We select following features which are strongly related to the trigger nets [15].

1) $LGFi_{x}$ (x-level logic fan-ins): Since the trigger net is activated only when a specifically unique trigger condition is met by a special combination of the primary inputs or internal states, it is expected to be a significant confluence of pre-stage signals, which means the transitive fan-ins of the trigger net are large enough. Based on this, we use $LGFi_{x}$, which means the number of the inputs of the logic gates x-level away from the target net, and choose $LGFi_1$, $LGFi_2$, $LGFi_3$, and $LGFi_4$ as features to represent the fan-in characteristic.

2) $min\{FFi, PO\}$: Since the trigger net is activated to drive the malicious payload which is generally designed to register the internal state values or give particular outputs, it is expected to be located near the post-stage Flip-Flops or primary outputs. We define $FFi$ to be the minimum gate-level to any Flip-Flop inputs from a target net, and $PO$ to be the minimum gate-level to any primary outputs from the target net. Then we define $min\{FFi, PO\}$ as one feature representing the minimum value between $FFi$ and $PO$, which denotes the characteristic of driving a Flip-Flop or a primary output.

A simple example is shown in Fig. 3. For the target net in Fig. 3, $LGFi_1 = 2$, $LGFi_2 = 4$, $LGFi_3 = 8$, and $min\{FFi, PO\} = 1$. In summary, we choose five features: $LGFi_1$, $LGFi_2$, $LGFi_3$, and $LGFi_4$, and $min\{FFi, PO\}$.

D. Machine Learning Algorithm and Data Preprocessing

After the extraction of features from the netlists, a detection model is required to be constructed to distinguish the trigger net from others in a detected netlist based on the machine learning method. The ensemble learning method is effective to help improve machine learning results by combining several predictive models. And the XGBoost model [16] implemented on this basis has been widely recognized, since it uses a more regularized-model formalization to control over-fitting, which gives it better performance. The algorithm of XGBoost model used in our method is shown in Algorithm 1 [15][17].
Algorithm 1 XGBoost.

**Input**: Data set \((x_i, y_i);\) Loss function \(L;\) The number of iterations \(M;\) The number of leaf nodes \(T;\)

**Output**: \( f(x) = f^{(M)}(x) = \sum_{m=0}^{M} f_m(x) \)

1. Initialize \( f^{(0)}(x) = f_0(x) = 0 = \arg \min_{\theta} \sum_{i=1}^{n} L(y_i, \theta) \)
2. for \( m = 1 \) to \( M \) do
   3. partial derivatives for loss functions/
   4. leaf node \( f_i \)/
   5. Determine tree structure \( R^T \) by allocating \( \{G_{jm}, H_{jm}\} \) to maximize \( \text{Gain}/\)
   6. New tree with structure \( R^T \) and leaf weights \( \omega_{jm} \)/
   7. Add the new tree to the model/\)
   8. \( f_m(x) = \sum_{m=1}^{M} \omega_{jm} \cdot f_{jm}(x) \)
   9. \( f^{(m)}(x) = f^{(m-1)}(x) + f_m(x) \)
10. end for

XGBoost trains multiple weak classifiers and combines them to obtain a strong classifier. Each new classifier is generated based on the original classifiers along the negative gradient direction of the loss function.

A set of data from the trigger nets and a set from other nets are expected to be input into the XGBoost model for training. However, there is only one trigger net in a netlist. Hence the training data set of the trigger nets may be much smaller than that of other nets, causing an imbalance in the training phase. Two data preprocessing operations are introduced to balance the training data sets. First, we search out the nets whose feature values are identical with each other, and keep one of them with its feature values and delete the rest. Next, count the total number of remaining other nets as \( N_o \) and the trigger nets as \( N_t \) by the feature data of the training nets \( [N_o/N_t-1] \) times to shape the size of the training data set similar to that of other nets. Thus, a new balanced data set is formed for the training.

IV. EVALUATION

A. Experimental Setup

In the experiment, we validate the effectiveness of the proposed detection method and model by detecting hardware Trojans inserted into a 32-bit RISC processor design \([18][19]\) compatible with MIPS 4Kc. We use 17 Trust-HUB benchmarks \([14]\) listed in Table I as the training samples. Each of these sample circuits contains a Trojan-injected circuit with a trigger net. To build chip circuits under detection, we first use the Design Compiler to synthesize the processor register transfer level source code into a gate-level netlist under the TSMC 90 nm CMOS process. Then each of the 17 Trust-HUB Trojans is inserted into the RISC processor to build the final circuit netlist, while the other 16 netlists are classified and trained to obtain the two detection models.

We use an Intel Xeon E5-2660 V3 computer environment with 128 GB memory. An application for netlist parsing and feature extraction is programmed in Python, and we use a Python interface of the XGBoost tool library \([16]\) for the machine learning.

The XGBoost model is used in the training phase, and the key parameters of the model are set as follows. The \( \text{max depth} \) is set as 7 by varying its value from 3 to 10. The \( \text{min child weight} \) is set as 20 by varying from 1 to 30. The \( \gamma \) is decided by varying its value from 0 to 0.5 and set as 0.3 finally. We divide the training data into a training set and a verification set according to the ratio of 7:3 and input them into the XGBoost model for learning. After 300 rounds of iteration, the model converges and achieves high accuracy on both the training set and the verification set.

B. Experimental Results

After the training phase, two types of detection models are obtained. We then inspect the chip netlist by inputting the feature data of each net into the detection models. The detection target is the trigger net, and we define other signal nets as ordinary nets. Each detection model is expected to give suspiciousness values of each net suspected of being a trigger net. Detection results of suspicious trigger nets are synthesized by flagging the top \( n\% \) suspicious nets from both of two detection models according to the suspiciousness values.

Four values are used to count the results: the true positive value (TP), the true negative value (TN), the false positive value (FP), and the false negative value (FN). TP means the number of trigger nets correctly identified to be trigger nets. TN means the number of ordinary nets correctly identified to be ordinary nets. FP means the number of ordinary nets falsely identified to be trigger nets. FN means the number of trigger nets falsely identified to be ordinary nets. Based on these, three popular evaluation metrics are applied to justify the experimental results: the true positive rate (TPR), the true negative rate (TNR), and accuracy. TPR is defined by \( TP/(TP+FN) \), meaning the ratio of the number of correctly identified trigger nets to the number of total trigger nets. TNR is defined by \( TN/(TN+FP) \), meaning the ratio of the number of correctly identified ordinary nets to the number of total ordinary nets. Accuracy is defined by \( (TP+TN)/(TP+TN+FP+FN) \), meaning the ratio of the number of correctly identified nets to the number of total nets. For the identification of the suspicious trigger nets, we use the TPR, TNR, and accuracy as figures of merit.

1) **Suspiciousness Ranking of the True Trigger Net**: We implement experiments for each of the 17 Trojan-inserted netlists under detection. Table I shows the suspiciousness ranking of the true trigger net given by both of two detection models, and we calculate the suspiciousness ranking rate according to the higher suspiciousness ranking between two models and the number of total nets, which is, Suspiciousness ranking rate\(=\frac{\text{Higher suspiciousness ranking}}{\text{Number of total nets}} \times 100\%\).

2) **TPR and TNR**: Since there is one true trigger net in each detected netlist, the TPR is 100% if the true trigger net is correctly flagged, while the value is 0% if the true trigger net is not flagged. Hence the average TPR of all 17
TABLE I
THE SUSPICIOUSNESS RESULTS OF THE TRUE TRIGGER NETS

<table>
<thead>
<tr>
<th>Chip circuit</th>
<th>Inverted Trojan</th>
<th>Number of total nets</th>
<th>Suspiciousness ranking model 1</th>
<th>Suspiciousness ranking model 2</th>
<th>Suspiciousness ranking rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS232-T1000</td>
<td>51493</td>
<td>1</td>
<td>23835</td>
<td>0.0019%</td>
<td></td>
</tr>
<tr>
<td>RS232-T1100</td>
<td>51494</td>
<td>2</td>
<td>25388</td>
<td>0.0319%</td>
<td></td>
</tr>
<tr>
<td>RS232-T1200</td>
<td>51497</td>
<td>2</td>
<td>23920</td>
<td>0.2214%</td>
<td></td>
</tr>
<tr>
<td>RS232-T1300</td>
<td>51499</td>
<td>2</td>
<td>23903</td>
<td>0.4887%</td>
<td></td>
</tr>
<tr>
<td>RS232-T1400</td>
<td>51493</td>
<td>1</td>
<td>23827</td>
<td>0.0019%</td>
<td></td>
</tr>
<tr>
<td>RS232-T1500</td>
<td>51496</td>
<td>1</td>
<td>23838</td>
<td>0.0019%</td>
<td></td>
</tr>
<tr>
<td>RS232-T1600</td>
<td>51489</td>
<td>2</td>
<td>24789</td>
<td>0.9711%</td>
<td></td>
</tr>
<tr>
<td>RS232-T1700</td>
<td>51499</td>
<td>2</td>
<td>25000</td>
<td>0.0019%</td>
<td></td>
</tr>
<tr>
<td>RS232-T1800</td>
<td>51499</td>
<td>2</td>
<td>25000</td>
<td>0.0019%</td>
<td></td>
</tr>
<tr>
<td>RS232-T1900</td>
<td>51499</td>
<td>2</td>
<td>25000</td>
<td>0.0019%</td>
<td></td>
</tr>
<tr>
<td>RISC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s35932-T1000</td>
<td>57090</td>
<td>1</td>
<td>57090</td>
<td>15%</td>
<td></td>
</tr>
<tr>
<td>s35932-T1100</td>
<td>57090</td>
<td>1</td>
<td>57090</td>
<td>15%</td>
<td></td>
</tr>
<tr>
<td>s35932-T1200</td>
<td>57112</td>
<td>2</td>
<td>34850</td>
<td>7.8328%</td>
<td></td>
</tr>
<tr>
<td>s35932-T1300</td>
<td>56959</td>
<td>2</td>
<td>34850</td>
<td>7.8328%</td>
<td></td>
</tr>
<tr>
<td>s35932-T1400</td>
<td>56937</td>
<td>2</td>
<td>34850</td>
<td>7.8328%</td>
<td></td>
</tr>
<tr>
<td>s35932-T1500</td>
<td>56124</td>
<td>2</td>
<td>34850</td>
<td>7.8328%</td>
<td></td>
</tr>
<tr>
<td>s35932-T1600</td>
<td>56124</td>
<td>2</td>
<td>34850</td>
<td>7.8328%</td>
<td></td>
</tr>
</tbody>
</table>
| TABLE II
AVERAGE TPR, TNR, AND ACCURACY RESULTS

<table>
<thead>
<tr>
<th>Detection results under n%</th>
<th>n%</th>
<th>1%</th>
<th>2%</th>
<th>5%</th>
<th>10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average TPR</td>
<td></td>
<td>58%</td>
<td>88%</td>
<td>94%</td>
<td></td>
</tr>
<tr>
<td>Average TNR</td>
<td></td>
<td>98%</td>
<td>96%</td>
<td>90%</td>
<td>80%</td>
</tr>
<tr>
<td>Average accuracy</td>
<td></td>
<td>98%</td>
<td>96%</td>
<td>90%</td>
<td>80%</td>
</tr>
</tbody>
</table>

netlists under detection is the ratio of the number of detected netlists whose true trigger net is correctly flagged to the total number 17. In addition, TP and FN can only be 0 or 1, thus the accuracy is almost equal to the TNR. When we flag the top n% suspicious nets of each detection model as the suspicious trigger nets, with the increment of n%, the number of benchmarks whose true trigger net is correctly flagged increases, increasing the average TPR, and the number of ordinary nets falsely identified to be trigger nets increases as well, decreasing the TNR and the accuracy. We set n% respectively as 1%, 2%, 5%, 10%. The average TPR, TNR, and accuracy are calculated and listed in Table II. It can be seen from the table that we set n% as 2%, the best detection results will be obtained in our experiments where the average TPR is 88%, the average TNR is 96%, and the average accuracy is 96%. Our method shows a satisfactory capability and accuracy on detecting the trigger nets, while misjudging few ordinary nets as trigger nets.

V. CONCLUSION

In this paper, we have proposed a trigger-net-based gate-level hardware Trojan detection method and model to locate possible hardware Trojans, should they exist in an IC chip. Our focus was placed on searching for the trigger nets. In specific, each net is scored with five trigger-net feature values, and they are used to train two detection models following the ensemble learning algorithm. The detection models assign suspiciousness values to each net in the chip circuit. By finding the suspicious trigger nets based on their suspiciousness values, most hardware Trojans can be exposed, with 88% TPR, 96% TNR, and 96% accuracy.

REFERENCES


