Machine Learning-based Classifier for Hardware Trojan Detection

Yuhao xie

University of Florida yuhao.xie@ufl.edu

Yuhjia Liu

University of Florida

Yujia.liu@ufl.edu

Patrick Craig

Univerisity of Flordia

Pcraig1@ufl.edu

*Abstract—The Integrated Circuit can be inserted Hardware Trojans during the time they fabrication, which will cause a huge number of economic losses. Measuring the variation of RO(Ring Oscillator) frequency has been proven as an effective way to detect the Hardware Trojan. For this paper, we use some Machine Learning classifiers to detect the Hardware Trojan based on given RO frequency data. We explained the reason of ML model choice and collected the result data. Next, we discussed the performance of different model and showed out conclusions.*

Keywords—Integrated Circuit, Hardware Trojans, Ring Oscillator, Machine Learning

# Introduction

## Motivation and Problem Statement

Hardware Trojan attacks have emerged as a major security concern for integrated circuits (ICs) [1]–[5]. Trojan can be inserted during the IC design or fabrication time, which will cause untrust and money lost issues. Therefore, detecting the Trojan in a short time with less cost becomes an important task.

Traditional post-manufacture logic testing is not suitable for detecting hardware trojans. Due to the stealthy nature of hardware trojans and the unusually wide range of possible Trojan instances available to adversaries[6]. Sometimes, an adversary will design a Trojan that triggers a fault only under rare circuit conditions to evade detection, which means it is difficult to test all the possible combinations to see what happened. On the other hand, structural tests, such as those based on stuck-at, delay, or bridging fault models, are based solely on the original untampered netlist or model, and thus cannot guarantee Trojan detection[7].

Measuring the RO frequency has been proven as an effective way to detect Trojans. Because RO frequencies are affected by power supply and temperature fluctuations (runtime activity), capacitance, threshold voltage, etc. variations (from manufacturing process). Since wires are not perfect superconductors (they have finite resistance and inductance), Trojan switching will draw power from one part of the on-chip power network and cause a small, temporary voltage dip in another. This voltage drop causes a temporary drop in RO frequency when the Trojan is active. If the resulting drop in RO frequency can be detected, then the Trojan can be detected.

The situation in real life, however, is much more intricate. Since ROs experience random process variations and thus produce slightly different frequencies even with similar design and operating conditions, it is challenging to compare RO frequencies from different chips, even if they are both Trojan-free. There is no single frequency that functions as a pass/fail threshold for detection. Additionally, regular switching activity can result in RO frequency changes that are at least as significant as Trojans. As a result, we require a classifier to aid in the trojan detection process. Therefore, ML classifier is a very effective method to help us, which can automatically order or categorizes data into one or more of a set of “classes”.

The rest of this paper is organized as follows: Section 2

provides a brief overview of the Machine Learning. It overviews the Machine Learning model usually used for Trojan detection. The reason of choosing different classifier is discussed on Section 3, which also describes the detail about implementation. Section 4 describes the results from testing the RO frequency data set. Section 5 provides final conclusions.

# Background

## Machine learning approaches

ML approaches often entail a learning process with the goal of learning from "experience" (training data) in order to carry out a task. In ML, the input is a collection of examples. Typically, a set of properties, often referred to as features or variables, are used to characterize a specific example. A characteristic can be numeric, binary, ordinal, or nominal (enumeration, such as A+ or B-) (integer, real number. etc.). A p erformance indicator that gets better with practice over time is used to assess how well the ML model performs at a given task. After the learning process is complete, the trained model can be used to categorize, forecast, or cluster fresh instances (testing data) based on the knowledge gained.

图示

描述已自动生成

Fig. 1. A typical machine learning approach [8]

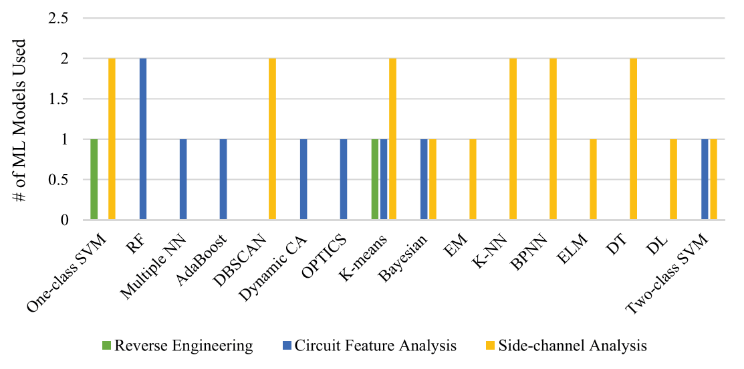
## Tasks of Learning

Depending on whether or not past knowledge is used (supervised) or not (unsupervised) with relation to the dataset at hand, ML tasks are divided into two basic groups, namely supervised and unsupervised learning.

In supervised learning, a class label, or flag designating the category to which a sample belongs, is matched with every sample in the training set. The goal is to create a general rule that accurately predicts each sample's correct label and applies to data not in the training set. Although the data in unsupervised learning is unlabeled, there is no separation between the training and test sets. In order to find hidden patterns, the learner analyses the incoming data.

## Trojan detection by ML methods

We may infer from Fig. 2 that the most popular supervised machine learning approach for recognizing Hardware Trojan-infected ICs is SVM, including one class and two-class SVM. SVM, however, makes the assumption that golden ICs are accessible for training. Furthermore, unsupervised learning techniques for Trojan detection using K-means approaches are also rather common. They are not restricted by the aforementioned criterion, though.



*Fig. 2. Frequency of ML used for HT detection[9]*

1. Methods

The objective of this project is to distinguish Trojan Free (TF) RO frequencies and Trojan inserted (TI) RO frequencies. Although using a simple threshold like what we did in previous homework is tempting, unfortunately, the random process variation between each individual chip sample makes it unrealistic. As a result, machine learning classifiers are taken as potential solutions to our Trojan detection problem since we have both golden samples (Trojan free) and problematic ones (Trojan inserted) for all 33 chips.

图示

描述已自动生成Machine learning algorithms are mainly classified into two categories, which are supervised learning algorithms and unsupervised learning algorithms. For supervised learnings, both data and labels are known before training and testing. While in unsupervised learning, there’s no label at all, which means the hidden pattern is supposed to be found by the algorithm in the training process. In this project, both types will be used and compared for better performance. The procedure workflow is shown in Fig. 3.

图示

描述已自动生成

Fig. 3. Procedure Workflow

* 1. *Data preparation*

Fig. 4. Depict the flow of data preprocessing. The first step is to randomly select samples according to the project requirements. Take training sample size 6 as an example, random.sample(indices, num) method in Python is used to get random numbers from 1 to 33. Then decide whether the data type we need is Trojan free or Trojan inserted, if it is case 1 (the supervised learning), then half of the sample will be TI and the other half will be TF; While for case 3 (unsupervised learning), the type will be random by using random.choices(['TI','TF']. The rest of the data will be all test dataset for evaluations. The whole operations described below are packaged in our self-defined train\_test\_split() function.

* 1. *Classifier Selection and Design*

*Case 1 – Supervised learning*

1. *K-Nearest Neighbors*

As we are dealing with a binary classification problem, K-Nearest Neighbors (KNN) is naturally selected first for its simplicity in implementation. Beyond its simplicity, KNN also has a good performance in classification problems with irregular boundary for being a non-parametric method. We think it will be a good match to our problem since the RO frequencies are affected by multiple factors like the power supply, temperature variations, capacitance, threshold voltage, etc. and have very vague boundaries between samples. However, for large datasets with multiple features, KNN has its con in the computational complexity for it will store all the training data, and because of this, the prediction may be slower compared to other supervised learning algorithms. But in our case, the feature size for each row is 8 and we only have 33 chips, the impact of size of data is relatively mild.

The basic assumption of KNN is that similar data is close to each other, as a result, in order to decide which data point among the neighboring candidates is the closest to a given query point, distances will be calculated.

The most often used distance metric is the Euclidean distance, which is defined as the following:

For our project, the n equals to 8, since each chip has 8 RO frequencies. The main hyper-parameter in KNN is K, namely the number of neighboring data of a given point. We use sklearn packages in Python to implement KNN classification and change K from 2 to 10 with step size 1 to search for the K yielding the best training performance. Besides tuning the number of neighbors, leaf\_size parameter is also tuned in range of 2 to 30 with step size 2. This parameter has effects on construction time, query time and the memory size, which will mitigate KNN’s disadvantage in computational complexity.

1. *Logistic Regression*

Logistic regression algorithm is usually used in binary classification problems, and it is suitable for our supervised learning task, to decide whether the chip is Trojan Free, or Trojan inserted. This method is based on the concept of probability. Unlike linear regression, binary logistics regression transforms its outputs to probability space by using the logistic sigmoid function.

Like KNN, logistic regression also has its advantages in easy implementation, and it also can achieve good accuracy for simple data sets which are linearly separable. What’s more, logistics regression doesn’t require high computational power which outperforms KNN in this aspect. Yet logistic regression has its main limitation in its assumption of linearity between the dependent variables and the independent variables. In our dataset, we are not sure about the correlation between 8 ring oscillators frequencies.

*Case 3 – Unsupervised learning*

1. *K-Means clustering*

K-Means is an unsupervised learning algorithm which aims to divide the input dataset to K different clusters. When a new data comes in, it will be distributed to the cluster whose center point is the closest to the new data. Then the center point will also be updated accordingly. Thus, this algorithm is very sensitive to outliers which is its biggest disadvantage. But this algorithm is still worthy of trying as a baseline for further optimization for its simple implementation in sklearn.

1. *DBSCAN*

Different from the partition-based clustering K-Means, DBSCAN is a density-based clustering method which stands for density-based spatial clustering of applications with noises. Considering the frequency of ring oscillators can be easily influenced by the environment, DBSCAN may be a good helper to eliminate the effects of these variations. To implement this algorithm, we first import the DBSCAN from the sklearn.cluster, and tune the eps (distance of neighbors) and min\_samples to get the best trained model.

Compared to K-Means, DBSCAN doesn’t need to specify the number of clusters beforehand and is robust to outliers. In our RO data, we have more trojan-inserted data than trojan-free in every sample chip (the ratio is 23:2), fortunately, DBSCAN has a great advantage in separating clusters of high density from those of low density, which makes it a candidate with large potential.

# Results

## Case 1 Results -

## Case 2 Results -

# Conclusion

Before you begin to format your paper, first write and save the content as a separate text file.

##### References

1. S. Adee, ‘‘The hunt for the kill switch,’’ IEEE Spectrum, vol. 45, no. 5, pp. 34–39, May 2008.
2. J. Kumagai, ‘‘Chip detectives,’’ IEEE Spectrum, vol. 37, no. 11, pp. 43–48, Nov. 2000.
3. Defense Advanced Research Projects Agency (DARPA), ‘‘TRUST in integrated circuits (TIC),’’ 2007. [Online]. Available: http:// www.darpa.mil/MTO/solicitations/baa07-24
4. D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, ‘‘Trojan detection using IC fingerprinting,’’ in Proc. IEEE Symp. Security Privacy, 2007, pp. 296–310.
5. M. Tehranipoor and F. Koushanfar, ‘‘A survey of hardware Trojan taxonomy and detection,’’ IEEE Design Test Comput., vol. 27, no. 1, pp. 10–25, Jan.–Feb. 2010.
6. R. S. Chakraborty, S. Narasimhan and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," 2009 IEEE International High Level Design Validation and Test Workshop, 2009, pp. 166-171, doi: 10.1109/HLDVT.2009.5340158.
7. Kelly, S., Zhang, X., Tehranipoor, M. et al. Detecting Hardware Trojans using On-chip Sensors in an ASIC Design. J Electron Test 31, 11–26 (2015). https://doi.org/10.1007/s10836-015-5504-x
8. K. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, “Machine Learning in Agriculture: A Review,” Sensors, 2018.
9. Z. Huang, Q. Wang, Y. Chen and X. Jiang, "A Survey on Machine Learning Against Hardware Trojan Attacks: Recent Advances and Challenges," in IEEE Access, vol. 8, pp. 10796-10826, 2020, doi: 10.1109/ACCESS.2020.2965016.