A Self-learning Framework to Detect the Intruded Integrated Circuits

F. K. Lodhi [1], I. Abbasi [1], F. Khalid [2], O. Hasan [1], F. Awwad [3]and S. R. Hasan [4]
[1] Sch. of Elect. Engg. and Comp. Sc., National University of Sciences and Technology (NUST), Islamabad, Pakistan
[2] Military College of Signals, National University of Sciences and Technology (NUST), Islamabad, Pakistan
[3] College of Engineering, United Arab Emirates University, Al-Ain, UAE
[4] Department of Electrical and Computer Engineering, Tennessee Technological University, Cookeville, TN, USA
{faiq.khalid, osman.hasan, 15phdiabbasi}@seecs.nust.edu.pk, 09310827@mcs.edu.pk, f_awwad@uaeu.ac.ae, shasan@tntech.edu

***Abstract* —Globalization trends in integrated circuit (IC) design using deep submicron (DSM) technologies are leading to increased vulnerability of ICs against malicious intrusions. These malicious intrusions are referred as hardware Trojans. One way to address this threat is to utilize unique electrical signatures of ICs. However, this technique requires analyzing extensive sensor data to detect the intruded integrated circuits. In order to overcome this limitation, we propose to combine the signature extraction mechanism with machine learning algorithms to develop a self–learning framework that can detect the intruded integrated circuits. The proposed approach applies the lazy, eager or probabilistic learners to generate self-learning prediction model based on the electrical signatures. In order to validate this framework, we applied it on a recently proposed signature based hardware Trojan detection technique. The cross validation comparison of these learner shows that eager learners are able to detect the intrusion with 96% accuracy and also require less amount of memory and processing power compared to other machine learning techniques.**

Keywords— Hardware Trojan; Machine Learning; Rapid Miner Studio; MSMA; Asynchronous pipeline.

I. Introduction

With the globalization of integrated-circuit chip design-process, the chances of malicious hardware design intrusion, known as hardware Trojan, have grown tremendously [1]–[3]. Hardware Trojans can lead to many unwanted activities, including leaking confidential information, changes in the timing characteristics of the circuits, malfunctioning, denial of service and counterfeiting [1][4][5]. Various techniques detecting the hardware Trojans at different design stages have been proposed in the literature [6]. Some of the prominent works include micro-architecture modification to improve triggering of the potential Trojan payload during test [7][8] and the usage of inherent error detection of quasi delay insensitive (QDI) architectures to detect malicious intrusions [9].

Generally, hardware Trojans alter the physical characteristics of the design. Trojan inserted during the design or fabrication is bound to have an effect on the power consumption and in most cases, it also affects the delay of the circuits. This behavior has been widely exploited to analyze power and delay characteristics of the design to detect hardware Trojans. This solution extracts the timing, power, current or frequency based signatures[9]–[12]. Ferraiuolo et. al. presented a novel ring oscillator network (RON) based technique to generate the frequency signature for Trojan detection [13]. The frequency of the ring oscillator is dependent on power supply and thus a small change in power supply can affect its frequency. This technique [13] leverages this dependency to detect the change in power in terms of frequency. Lodhi et. al. proposed a macro synchronous micro asynchronous (MSMA) pipeline based hardware Trojan technique, which is based on a hybrid quasi delay insensitive (QDI) pipeline architecture [14][15]. It uses the intrinsic error prevention property to generate the timing signatures with environmental uncertainties. However, the signature-based techniques require sensors to acquire the values of power, temperature, current, delay and frequency. These sensors generate an extensive amount of data, which are generally handled by conventional automation techniques and require user interaction, thus highly vulnerable to human errors.

Therefore, in this paper, we propose a generic self-learning framework to identify intruded integrated circuits during the testing phase. This framework includes the design and testing stages of integrated circuits. In the design stage, it utilizes the measured values of metrics that are attributed as golden. For example power, delay, current and frequency, of the integrated circuits are used to develop a trained model using RapidMiner studio[16], which provides software platform for machine learning. The extracted model is then used in the testing stage to classify the intruded and non-intruded integrated circuits. As a case study, we applied this approach to our previously proposed macro synchronous and micro asynchronous (MSMA) pipeline [15][17]. Furthermore, to elaborate this concept, we have used three different machine-learning approaches, i.e. Decision Trees (DT) from eager learning, Bayesian Classifiers (BC) from probabilistic learning and k-nearest neighbors (KNN) from lazy learning, to extract the trained models. The experimental results show that the extracted models from the decision trees (eager learning) and k-nearest neighbors (lazy learning) have identified the Trojans with a precision of 96%. However, the k-nearest neighbors (KNN) algorithm stores the whole dataset and generates the trained model for every testing data. Unlike KNN, decision trees store the trained model instead of whole dataset. Hence, decision trees are found to be more efficient as it requires less memory and computational resources as compare to KNN, for the same level of performance.

II. Preliminaries

For the better understanding of the paper, this section provides an overview of the MSMA pipeline architecture, RapidMiner Studio, and some general performance parameters that are frequently used in machine learning, such as class precision, class recall and class accuracy.

A. Macro Synchronous Micro Asynchronous (MSMA)

A traditional NCL pipeline architecture is primarily composed of three main blocks: NCL based registers, combinational logic block and completion detection scheme. Our earlier proposed MSMA technique, based on the NCL pipeline [15][17], is a hybrid form of the synchronous and asynchronous pipelines. In this design, there is a NCL based pipeline to replace the combinational logic block between every synchronous pipeline stage. The implementation of the MSMA pipeline, as shown in Figure 1, can be divided into two main blocks: asynchronous and synchronous. The asynchronous block includes the NCL pipeline, which consists of three sub-blocks: Dual rail Registers, Combinational Block and Completion Detection. Since NCL uses dual rail encoding, therefore, all the asynchronous blocks are implemented using the dual rail encoding. The synchronous block consists of synchronous registers at both ends of the MSMA pipeline and is interfaced via the Dual/Single Rail to Single/Dual Rail conversion blocks [15].

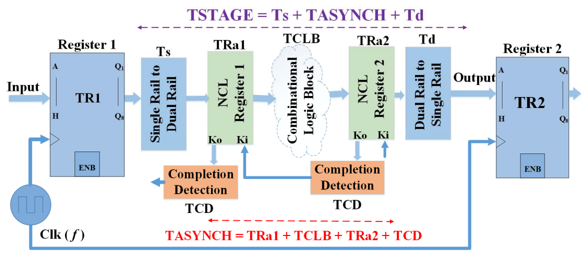

Figure 1. Macro Synchronous Micro Asynchronous Pipeline, where TR1, TR2, TRa1, TRa2, TCLB , TCD, Ts, Td represent delay in Synchronous registers 1 & 2, Asynchronous (NCL) registers 1 & 2, Combinational Logic Block, Completion Detection, Single to dual rail and Dual to single rail [15]

1) Timing Analysis of MSMA

The propagation delays of NCL register1, MUX and combinational block, NCL register 2 and completion detection blocks are denoted as T_{Ra1}, T_{CLB}, T_{Ra2} and T_{CD}, respectively, as shown in Figure 2. Equation (1) provides the latency of the NCL handshake signals (T_{ASYNCH}), which is the duration from the availability of data at register 1 to the assertion of the acknowledgement signal through the completion detection unit.

$$_{ASYNCH} = T_{Ra1} + T_{CLB} + T_{Ra2} + T_{CD} \qquad (1)$$

NCL pipeline consists of two states of data communication: data and null states. These two states have different latencies, and represented as data latency (T_{DATA}) and null latency (T_{NULL}). The total latency of the pipeline can be written as follows:

$$T_{TOTAL} = T_{DATA} + T_{NULL} \qquad (2)$$

2) Hardware Trojan Detection in MSMA pipeline

The combinational unit of MSMA is an asynchronous circuit, i.e., based on event driven handshaking signals. We took the advantage of the event driven nature of the NCL and measured one complete handshake cycle to obtain a unique timing signature. Figure 2 shows the implementation of test vector insertion to generate timing signature. The test vectors TA and TB are 2-bit dual rail signals. The two LSBs of TA [0:3] are used to select whether the circuit is in test mode or in normal operational mode.

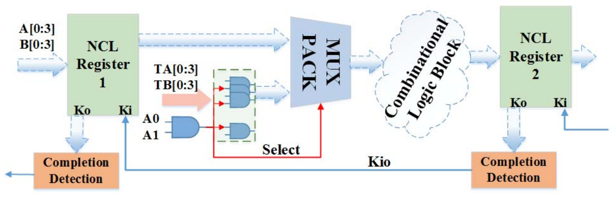

Figure 2. Test vector insertion to obtain timing signature in MSMA [14][15]

B. Performance Evaluation Parameters

Performance parameters identification is very important to evaluate machine learning algorithms for hardware Trojan detection. Accurately, recognizing Trojans and precision of the analysis is an important measure to assess the soundness of Trojan detection technique. These terms are formally defined as follows, TP, TN, FP and FN represent number of times a Trojan detection are true positives, true negatives, false positives and false negatives, respectively.

1) Correctly recognizing Trojans

It is defined as the percentage of test set tuples that are correctly classified:

$$\text{Correct\_recog\_trojans} = \frac{TN + TP}{TN + TP + FP + FN}\ \% \qquad (3)$$

2) Precision

Precision or exactness is defined as percentage of tuples that the classifier labeled as positive are actually positive:

$$Precision = \frac{TP}{TP + FP}\ \% \qquad (4)$$

C. RapidMiner Studio

RapidMiner is a tool that provides many procedures of data mining and machine learning, e.g., loading, transformation, preprocessing, post processing, visualization, modelling, evaluation, and deployment of attributes. The data mining processes can be made using arbitrarily nestable operators, which are described in XML files and created in the RapidMiner's graphical user interface (GUI).

We chose RapidMiner studio for our work, because of its ability to handle different machine learning algorithms specially the decision trees. Thus, it allows us to efficiently manage the large data set and seamlessly translate them in complete decision tree.

III. Proposed Framework

Figure 3 shows the proposed framework to detect the intruded circuits during the testing stage of the IC design process. The first step of this frame work is to obtain the measured values of attributes for a specific input vector using the existing state-of-the-art techniques. In our case study we extracted the timing signatures in MSMA using Monte-Carlo simulation. Conventionally such signatures are called golden signatures. In the next step, these extracted attributes are analyzed to remove the redundant attributes, which do not contribute to generate the prediction model. The dataset of the selected attributes is analyzed and rectified using dimensionality reduction algorithms, i.e., Principal Component Analysis (PCA), Fisher Linear Discriminant (FLD), to obtain the optimized dataset for machine learning algorithms. This data preprocessing optimizes the required resources (memory, procession time) and enhances the performance of machine leaning algorithms. Then the machine learning algorithm learns and generates the prediction model from the optimized dataset. Since this framework is generating the prediction model from the golden attributes dataset, thus, any changes in the golden attributes can be learned by the model through this framework. In the testing phase, the same input vector is applied to the integrated circuits under test (ICUT) to obtain the values for the attributes. Finally, the prediction model uses the attribute values from the testing phase to identify ICUT. In this work, we used three different machine learning algorithms, namely, Decision Trees (DT), Bayesian Classifiers (BC) and k-Nearest Neighbors (KNN).

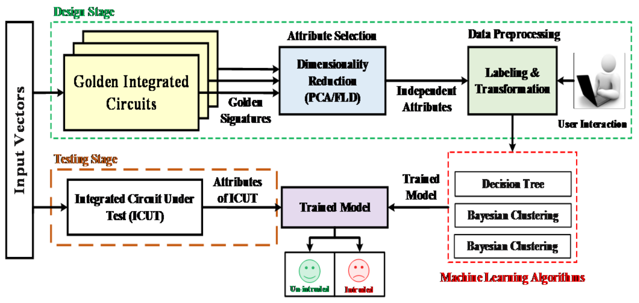

Figure 3. Proposed Framework to Identify Intruded Integrated Circuits

IV. Case Study: Intrusion Detection in MSMA Pipeline using Proposed Framework

As stated earlier, for a case study, we applied the proposed framework on our previously proposed timing signatures based hardware Trojan detection technique [14][15]. The complete procedure to identify the intruded MSMA is explained below:

A. Golden Attributes Data Acquisition

The first step of our proposed framework is to acquire the golden dataset for selected attributes as the hardware Trojan detection in MSMA only generates the timing signatures. In order to identify the intruded MSMA, we only acquire the dataset based on timing signatures. The dataset for the timing signatures is acquired by using Monte Carlo simulation with 500 iterations per input vector. In this analysis, we also incorporated the process and environmental variations by applying the Gaussian distribution on thickness oxide, threshold voltage and junction depth of nMOS and pMOS [14][15]. Table 1 presents the effects of the process and mismatch variations on the total delay of MSMA, which can serve as timing signature of the NCL based MSMA system.

B. Data Labeling and Transformation

The next step is to simplify the acquired data to reduce the storage resources for the prediction model by applying the numeric to nominal transformation. Therefore, the numeric values of the delays are transformed into the nominal values based on the rules, shown in Table 2. After data transformation, the next step is to label the data to train the prediction model. We applied the following rules to label the acquired data: If $min \leq delay \leq max$ then label the data as "No Trojan" else label it as "Trojan", where *max* and *min* are the maximum and minimum delay values from Table 1, respectively.

Table 1: Effect of Process and Mismatch Variation on delay [14]

Test Vector		Delay				Deviation
Single Rail	Dual Rail	Max. (ns)	Min (ns)	SD. (ps)	Mean (ns)	Max. (ps)
0000	**01010101**	**2.0059**	**1.9851**	**6.391**	**1.9962**	**20.8**
0001	01010110	2.0498	1.9823	8.493	2.0019	67.5
0010	01011001	2.0313	1.9879	6.578	2.0018	43.4
0011	01011010	2.0118	1.9493	7.923	2.0059	62.5
0100	01100101	2.0148	1.932	11.253	2.0076	82.8
0101	01100110	2.0221	1.9657	7.743	2.0042	56.4
0110	01101001	2.0042	1.9089	13.623	2.0078	95.3
0111	01101010	2.0163	1.961	7.653	2.0036	55.3
1000	10010101	2.038	1.9525	11.502	2.0092	85.5
1001	10010110	2.019	1.9425	9.524	2.005	76.5
1010	10011001	2.0166	1.9357	10.623	2.0012	80.9
1011	**10011010**	**2.035**	**1.9225**	**16.253**	**2.0021**	**107.5**
1100	10100101	2.0441	1.972	9.663	2.0042	72.1
1101	10100110	2.053	1.963	13.592	2.0034	90
1110	10101001	2.066	1.9753	13.894	2.0185	90.7
1111	10101010	2.0801	1.9821	14.253	2.0376	98

Table 2: Data Transformation (Numeric to Nominal) Rules

Condition	*Label*	*Condition*	*Label*
delay < 1.976	*Extremely Low*	*2.024 > delay ≥ 2.012*	*Medium*
1.988 > delay ≥ 1.976	*Very Low*	*2.036 > delay ≥ 2.024*	*High*
2.00 > delay ≥ 1.988	*Low*	*2.048 > delay ≥ 2.036*	*Slightly High*
2.012 > delay ≥ 2.00	*Slightly Low*	*delay ≥ 2.048*	*Very High*

C. Trained Model Extraction

The final step of the proposed framework is to apply the machine learning algorithms to extract the trained models from the labelled data, which are used in the testing stage to automate the intrusion detection in MSMA. Therefore, we extracted the prediction model, by applying the Decision Trees (DT), Bayesian Classifiers (BC) and k-nearest neighbors (KNN) on the labelled data in a machine learning tool, RapidMiner.

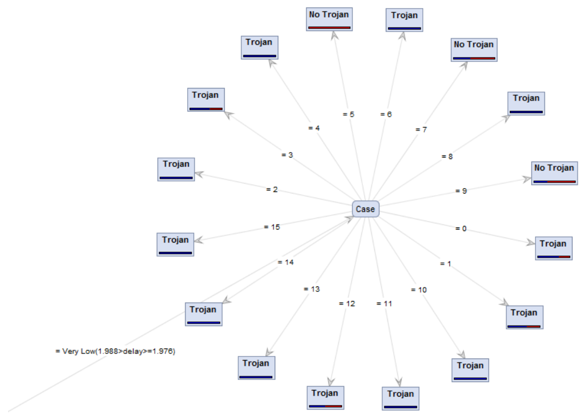

Figure 4. Decision branch when delay is very low (1.988 > delay ≥ 1.976)

Figure 4 shows the complete trained model obtained from decision tree (DT) algorithm. Each ring in Figure 4 represents the decision case for one of the conditions of Table 2. To classify the intruded and non-intruded MSMA pipeline, RapidMiner is trained based on the delay and input vectors. For example, if the delay is very low and input vector is 9 then it is highly probable that MSMA is non-intruded as shown in Figure 4. Similarly, Bayesian Classifiers (BC) and k-nearest neighbors (KNN) algorithms also classify the intruded and non-intruded MSMA.

D. Cross Validation of Trained Model

In order to validate the extracted trained model, we used the cross validation feature of the RapidMiner tool. In cross validation, the tool randomly divides the labelled data into k mutually exclusive subsets, each of approximately equal size and performs the training and testing *k* times. In each iteration, one subset is used for testing and others are used in training. Thus, each subset is used equal number of times for training and once for testing. Table 3 shows validation results of MSMA DT, BC and KNN trained models.

TABLE 3: PERFORMANCE COMPARISON (DT, BC AND KNN)

Decision Tree (DT) Correctly recognizing Trojans = 95.19%)			
	True Trojan	True No Trojan	Precision
Predicted Trojan	1307 (TP)	55 (FP)	**95.96%**
Predicted No Trojan	22 (FN)	216 (TN)	**90.76%**
Bayesian Classification (BC) (Correctly recognizing Trojans = 93.50 %)			
	True Trojan	True No Trojan	Precision
Predicted Trojan	1313 (TP)	88 (FP)	**93.72%**
Predicted No Trojan	16 (FN)	183 (TN)	**91.96%**
k-Nearest Neighbors (KNN) (Correctly recognizing Trojans = 93.12 %)			
	True Trojan	True No Trojan	Precision
Predicted Trojan	1274 (TP)	55 (FP)	**95.86%**
Predicted No Trojan	55(FN)	216 (TN)	**79.70%**

V. COMPARISON

In order to analyze the performance of the three main machine learning approaches (lazy, eager and probabilistic learning), we have obtained the confusion matrices for Decision Tree (DT), Bayesian Classifiers (BC) and k-Nearest Neighbors (KNN) algorithms, as shown in Table 3. where true Trojan and true no Trojan represent the delay values that are labelled as *Trojan* and *no Trojan,* respectively. Similarly, predicted Trojan and no Trojan means the delay values which the trained model predicted as Trojans and no Trojans, respectively. These matrices show that the trained models acquired from the decision trees and k-nearest neighbor algorithm detected the intruded integrated circuits with 96 % precision. However, k-nearest neighbor algorithm has a significantly less precision in the classification of non-intruded integrated circuits. Moreover, it is also a lazy learner and requires the complete dataset for each prediction, which requires extra storage and also makes it computationally expensive. The Bayesian Classification (BC) algorithm has less precision and accuracy as compared to Decision Tree (DT). Thus, these comparison results show that the DT algorithm is the best among these prediction algorithms in term of accuracy and precision. However, with more number of attributes, the decision trees become more complex and computationally expensive.

VI. CONCLUSION

In this paper, we propose a self-learning framework to detect the intruded integrated circuits. Unlike the traditional methods to detect the intrusion in integrated circuits, this solution uses machine learning algorithms to generate the intrusion detection models. The self-learning nature of our technique allows it to update Trojan prediction model using the typical training methods in machine learning. In future, we intend to explore the multi-parameter based signatures as attributes to generate the more accurate prediction model.

REFERENCES

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware Trojan taxonomy and detection," *IEEE Des. Test Comput.*, vol. 27, no. 1, pp. 10–25, 2010.

[2] G. Di Natale, S. Dupuis, and B. Rouzeyre, "Is side-channel analysis really reliable for detecting hardware Trojans?," in *DCIS'2012: XVII Conference on Design of Circuits and Integrated Systems*, 2012, pp. 238–242.

[3] X. Zhang, K. Xiao, M. Tehranipoor, J. Rajendran, and R. Karri, "A study on the effectiveness of Trojan detection techniques using a red team blue team approach," in *VLSI Test Symposium (VTS), 2013 IEEE 31st*, 2013, pp. 1–3.

[4] F. Koushanfar and A. Mirhoseini, "A unified framework for multimodal submodular integrated circuits Trojan detection," *IEEE Trans. Inf. Forensics Secur.*, vol. 1, no. 6, pp. 162–174, 2011.

[5] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy Hardware: Identifying and classifying hardware Trojans," *Computer (Long. Beach. Calif).*, vol. 43, no. 10, pp. 39–46, 2010.

[6] H. Li, Q. Liu, J. Zhang, and Y. Lyu, "A Survey of hardware Trojan detection, diagnosis and prevention," *Int. Conf. Comput. Des. Comput. Graph.*, pp. 173–180, 2015.

[7] M. Banga and M. S. Hsiao, "A region based approach for the identification of hardware Trojans," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, 2008, pp. 40–47.

[8] M. Banga and M. S. Hsiao, "A novel sustained vector technique for the detection of hardware Trojans," in *VLSI Design, 2009 22nd International Conference on*, 2009, pp. 327–332.

[9] C. LaFrieda and R. Manohar, "Fault detection and isolation techniques for quasi delay-insensitive circuits," *Dependable Systems and Networks, 2004 International Conference on*. pp. 41–50, 2004.

[10] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia, "MERO: A statistical approach for hardware Trojan detection," in *Cryptographic Hardware and Embedded Systems-CHES 2009*, Springer, 2009, pp. 396–410.

[11] A. Gbade-Alabi, D. Keezer, V. Mooney, A. Y. Poschmann, M. Stöttinger, and K. Divekar, "A signature based architecture for Trojan detection," in *Proceedings of the 9th Workshop on Embedded Systems Security*, 2014, pp. 1–10.

[12] H. T. Kung and S. J. Tarsa, "Partitioned compressive sensing with neighbor-weighted decoding," in *Military Communications Conference (MILCOM)*, 2011, pp. 149–156.

[13] A. Ferraiuolo, X. Zhang, and M. Tehranipoor, "Experimental analysis of a ring oscillator network for hardware trojan detection in a 90nm asic," in *Proceedings of the International Conference on Computer-Aided Design*, 2012, pp. 37–42.

[14] F. K. Lodhi, S. R. Hasan, O. Hasan, and F. Awwad, "Hardware Trojan detection in soft error tolerant Macro Synchronous Micro Asynchronous (MSMA) pipeline," in *Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on*, 2014, pp. 659–662.

[15] F. K. Lodhi, S. R. Hasan, O. Hasan, and F. Awwad, "Low power soft error tolerant Macro Synchronous Micro Asynchronous (MSMA) pipeline," *IEEE Comput. Soc. Annu. Symp. VLSI*, pp. 601–606, 2014.

[16] "RapidMiner - #1 Open source predictive analytics platform." [Online]. Available: https://rapidminer.com/. [Accessed: 05-Feb-2016].

[17] F. K. Lodhi, O. Hasan, S. R. Hasan, and F. Awwad, "Modified null convention logic pipeline to detect soft errors in both null and data phases," in *Midwest Symposium on Circuits and Systems MWSCAS*, 2012, pp. 402–405.