Power Profiling of Microcontroller's Instruction Set for Runtime Hardware Trojans Detection without Golden Circuit Models

Faiq Khalid Lodhi*, Syed Rafay Hasan†, Osman Hasan* and Falah Awwadl‡
*School of Electrical Engineering & Computer Sciences, National University of Sciences & Technology, Islamabad, Pakistan
†Department of Electrical and Computer Engineering, Tennessee Technological University, Cookeville, TN, USA
‡College of Engineering, United Arab Emirates University, Al-Ain, UAE
Email: {faiq.khalid, osman.hasan}@seecs.nust.edu.pk, shasan@tntech.edu, f_awwad@uaeu.ac.ae

***Abstract*—Globalization trends in integrated circuit (IC) design are leading to increased vulnerability of ICs against hardware Trojans (HT). Recently, several side channel parameters based techniques have been developed to detect these hardware Trojans that require golden circuit as a reference model, but due to the widespread usage of IPs, most of the system-on-chip (SoC) do not have a golden reference. Hardware Trojans in intellectual property (IP)-based SoC designs are considered as major concern for future integrated circuits. Most of the state-of-the-art runtime hardware Trojan detection techniques presume that Trojans will lead to anomaly in the SoC integration units. In this paper, we argue that an intelligent intruder may intrude the IP-based SoC without disturbing the normal SoC operation or violating any protocols. To overcome this limitation, we propose a methodology to extract the power profile of the micro-controllers instruction sets, which is in turn used to train a machine learning algorithm. In this technique, the power profile is obtained by extracting the power behavior of the micro-controllers for different assembly language instructions. This trained model is then embedded into the integrated circuits at the SoC integration level, which classifies the power profile during runtime to detect the intrusions. We applied our proposed technique on MC8051 micro-controller in VHDL, obtained the power profile of its instruction set and then applied deep learning, k-NN, decision tree and naive Bayesian based machine learning tools to train the models. The cross validation comparison of these learning algorithm, when applied to MC8051 Trojan benchmarks, shows that we can achieve 87% to 99% accuracy. To the best of our knowledge, this is the first work in which the power profile of a microprocessor's instruction set is used in conjunction with machine learning for runtime HT detection.**

Keywords: *Power Profiling, Machine Learning, Hardware Trojans, Runtime Detection, Microcontroller, Assembly Language Instructions*

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], [2]. 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], and affect the power consumption and, in most cases, the delay of the underlying circuits. Researchers exploited this behavior to extract the timing, power, current or frequency based signatures [3], [4] from the golden integrated circuits (ICs) to detect the abnormal electrical behavior.

However, the intruder may overshadow the above-mentioned hardware Trojan detection techniques, i.e., in a SoC design, some hard or firm IPs may hide Trojans depending on the aging of the chip. The possibility of detecting these Trojans during test phase is very low, and they may get activated once the chip is in use [5]. Runtime approaches, on the other hand, can monitor an IC for its entire operational lifetime, providing an important last-line of defense [6], [7]. Forte et. al. [8] and Bao et. al. [9] proposed runtime Trojan detection methodology that analyzes the behavior of built-in temperature sensors of ICs but it requires a precise calibration over the environmental changes and process variations. Zhao et al. exploited the dynamic thermal management techniques of ICs to detect Trojans at runtime [10]. A key feature of this technique is to analyze the thermal profile of the ICs to obtain the dynamic thermal/power parameters. This approach inherits the overhead of the cumbersome measurement of power from various locations of the power distribution network and requires classification algorithm and majority voting schemes and thus compromises the performance of ICs.

Generally, runtime monitoring techniques monitor the specific set of properties and parameters under the specific conditions and require precise calibration to incorporate the environmental changes. Therefore, researchers have proposed to embed a trained machine learning (ML) algorithm, like the k nearest neighbors (k-NN), naive Bayesian classification (NBC), decision tree (DT) and deep learning (DL), for runtime intrusion detection. The first step in these runtime approaches is to obtain the reference models or set of rules. Bao et al. proposed a reverse engineering approach utilizing support vector machine (SVM) to efficiently classify the intruded and un-intruded integrated circuits (ICs) at test stage [11], [12]. However, reverse engineering comes with the challenge of a large computational overhead. Therefore, Lodhi et al. proposed an adaptive methodology to cater the process variations and environmental changes by automating the classification of intruded and un-intruded integrated circuits during the test stage with the help of machine learning [13]. Iwase et al. proposed a SVM based technique to detect the hardware Trojans [14], where the golden power behavior of the electronic devices is extracted and then converted to the frequency domain to train the SVM based machine learning model. Therefore, in order to minimize the overhead due to frequency domain conversion, Kulkarni et al. [15], [16] proposed to use other supervised ML tools, i.e., k-NN and modified balanced winow (MBW) online algorithm, under

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

the assumption that IP modules are not intruded and Trojan can only be introduced through communication.

In this paper, we propose a novel methodology that is based on using the power profile of different instructions of a micro-controller for obtaining the trained machine learning algorithm to detect the hardware Trojans at runtime. The proposed methodology consists of two major steps: Firstly, we propose to extract the power profile of the assembly level instructions of micro-controllers, which is used to train the machine learning algorithm. The power profile is obtained by extracting the power behavior of the micro-controllers for different assembly language instructions, e.g., MOV, JMP, INC, JNC, ADD, SUBB in case of the MC8051 micro-controller. The second step is to obtain a trained machine learning (ML) algorithm by using different ML tools. During the SoC integration levels (which is considered trusted in our threat model), these models can be embedded in the original equipment manufacturer's (OEM) documentation. In this paper, we assume that while extracting the power profile of intellectual properties (IP), the Trojans are not activated. For illustration purposes, we implemented the MC8051 in VHDL (VHSIC Hardware Description Language) to extract the power profile of various commonly used instructions and then we applied used lazy, eager and probabilistic machine learning tools, i.e., k-NN, DT and NBC, to train the models. In addition, we also used a DL algorithm. The cross validation comparison of these learning algorithms shows that eager learners, which are least expensive in terms of computations, are able to detect the intrusions with 87% accuracy. On the other hand, with k-NN, which requires higher computation resources, the validation reaches an accuracy of about 99%. It also requires less amount of memory and processing power compared to the existing machine learning techniques.

The main contributions of this paper are as follows:

1) To the best of our knowledge, this is the first methodology to extract the power profile purely from micro-controller's assembly language instructions without requiring the golden circuits.
2) Machine learning tools are trained using the generated power profile and validated against the Trojan benchmarks for MC8051 micro-controller, provided by trsuthub.org [17].
3) A comprehensive methodology is proposed to train the ML algorithms online for runtime intrusion detection.

II. Proposed Methodology

This section explains our runtime hardware Trojan detection methodology based on power profiling. The proposed methodology consists of two major steps as shown in Fig. 1.

A. Power Profiling of Instructions

The first step of the proposed methodology is to obtain the power profile of the assembly instructions of micro-controllers by utilizing the following steps:

1) Identification of the modules that involve the basic operations of pipelines for each instruction, e.g., memory addressing, reading modules and the respective registers that would be active if an instruction is in the fetch stage.
2) Insertion of the power ports to facilitate the calculation of the respective power of each pipeline stage. For the proposed methodology, each pipeline stage must have a separate corresponding power port.
3) Application of a set of instructions to calculate the power at the respective power ports. For better performance, we propose to use different combinations of instructions, at a time to obtain a robust power profile.

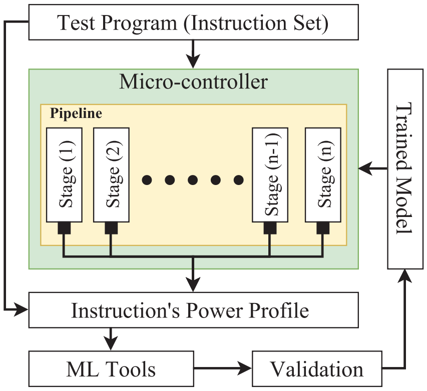

Figure 1: Proposed Methodology to detect hardware Trojan using Power Profile of the Assembly Instructions

B. ML based Trained Model Extraction

The second step of the proposed methodology is to extract the trained model from the respective power profile of the given set of instructions of micro-controller using following steps::

1) We start by obtaining the respective power profile of each set of instructions, which is followed by the labeling to differentiate the intruded and un-intruded power profiles. These labeling are in turn used to validate the trained model.
2) Next, we train the model by using different ML tools, i.e., k-NN, DT, DL and NBC, for extracting the trained model.
3) Then, the trained modeled is validated by applying the unlabeled data to find out the performance of different ML techniques to choose the best online ML based trained model.
4) Lastly, during the SoC integration level these models can be embedded in the OEM's documentation.

III. Case Study

This section illustrates the utilization and effectiveness of the proposed methodology by applying it on the MC8051 micro-controller. We first obtain the power profile of the assembly language instructions of the un-intruded MC8051. We used these power profiles for the online training of the ML models. For validation, we used the benchmark Trojans for MC8051 available on [17].

A. Pipeline Stages of MC8051

In the basic MC8051 micro-controller, each instruction follows a five stage pipeline to complete its execution. These are standard RISC pipeline stages, which include Fetch, Decode, Fetch Operands, Execute, and Store.

B. Selected Instructions of MC8051

To generate the power profile of instruction set of MC8051 micro-controller, we choose the following set of MC8051 instructions [18]. The brief explanation of the selected instructions is given below:

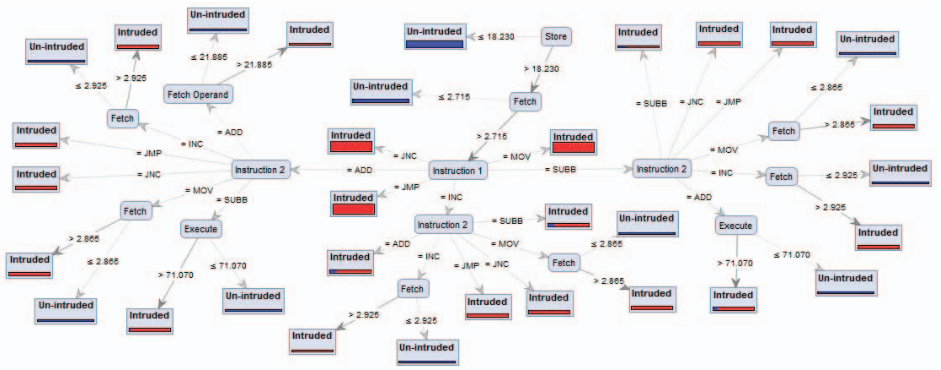

Figure 2: Decision Tree Model for two Parallel Assembly Language Instructions

1) *ADD:* It represents the *"Add Accumulator"* operation and it adds the value of the operand to the value of the accumulator and stores the resulting value in the accumulator.
2) *MOV:* It represents the *"Move Memory"* operation and it copies the value of operand2 into operand1.
3) *SUBB:* It represents the *"Subtract from Accumulator With Borrow"*, i.e., it subtracts the value of the operand from the value of the accumulator, leaving the resulting value in the accumulator.
4) *INC:* It represents the *"Increment Register"* operation and it increments the value of the given register by 1.
5) *JNC:* It represents the *"Jump if Carry Not Set"* operation and it branches to the address indicated by relative address (reladdr) if the carry bit is not set.
6) *JMP:* It represents the *"Jump to Data Pointer + Accumulator"*, it does the unconditional jump to the address represented by the sum of values of data pointer (DPTR) and the accumulator.

C. Power Profile of MC8051's Assembly Instruction

The first step of the proposed methodology is to extract the power profile of the MC8051's assembly instruction. For obtaining power profile, we implemented the MC8051 microcontroller in VHDL using the Xillinx's ISE 14.5 design suite for Spartan 6 (xc6slx45) FPGA. The power profile for each instruction is calculated by using the Xillinx power analyzing tool (Xpower analyzer). The power of each module involved in the pipeline stage is calculated separately for each instruction. In the experimental setup, we extracted the power of each pipeline stage for one and two parallel instructions. For example, Table I shows the power consumption in each pipeline stage for a particular instruction, which is extracted by executing one instruction at a time.

D. ML based trained model for MC8051

The second step of the proposed methodology is to train the ML model based on the extracted power profile. Therefore, after extracting the power profile, we analyzed and transformed it in such a way that it can be utilized for different ML techniques. This transformation involves the proper labeling of the extracted power behavior to differentiate between intruded and un-intruded behavior, and pre-pruning to reduce the redundant data set. After the transformation, we applied four different ML tools, k-NN, DT, DL and NBC, to generate the corresponding trained models. For example, Fig. 2 shows the trained model generated by applying the decision tree. This model shows the set of rules extracted from the two parallel instruction power profiles of MC8051, which are then utilized to classify the intruded and un-intruded power profiles.

Table I: Instruction Power (mW) Profile of MC8051

Instructions	Fetch	Decode	Fetch Operand	Execute	Store
MOV	3.15	8.15	11.97	50.55	19.05
ADD	3.02	8.2	22.98	73.65	31.15
SUBB	3.09	8.745	23.645	75.84	34.56
INC	3.19	8.21	20.65	61.25	23.91
JNC	2.98	8.152	16.08	22.15	18.95
JMP	2.94	8.19	15.47	22.74	20.15

E. Cross Validation of ML Algorithms

To validate the extracted trained model, we used the cross validation feature of the RapidMiner tool. In cross validation, the tool randomly divides the labeled data into k mutually exclusive subsets, where each set is approximately of the same size and performs the training and testing k times. In each iteration, one subset is used for testing and the others are used in training. Thus, each subset is used an equal number of times for training and testing. Table II shows validation results of MC8051 DT, DL, NBC and k-NN trained models.

We obtained the confusion matrices for all the mentioned ML algorithms by implementing the benchmark intrusions of MC8051 [17], as shown in Table II. In this table, each ML algorithm has two rows, namely predicted un-intruded and predicted intruded, and correspondingly two columns, namely true intruded and true un-intruded. For example, for the k-NN algorithm, the cell of the table that corresponds to predicted intruded row and true intruded column has a value of 1229. Whereas, the cell corresponding to the predicted intruded row and true un-intruded column has a value of 43. Hence, this indicates that 1229 out of 1272 times the corresponding algorithm correctly identifies the Trojan, hence making an accuracy of 96.62%. We performed the same analysis for all the rows as can be seen in Table II.

These matrices show that the trained models, acquired from the DT and k-NN algorithms, detected the intruded integrated circuits with 94.84% and 99.02% accuracy, respectively. However, the k-NN algorithm has a significantly less precision in classification of non-intruded integrated circuits. Since, it

Table II: Performance Comparison

	True Intruded	True Un-intruded	Precision
K-Nearest Neighbor (k-NN) (Accuracy = 99.02%)			
Predicted Intruded	1229	43	96.62%
Predicted Un-intruded	31	6257	99.51%
Decision Tree (DT) (Accuracy = 94.84%)			
Predicted Intruded	870	0	100.00%
Predicted Un-intruded	390	6300	94.17%
Deep Learning (DL) (Accuracy = 87.09%)			
Predicted Intruded	526	242	68.49%
Predicted Un-intruded	734	6058	89.19%
Naive Bayesian Classification (NBC) (Accuracy = 86.46%)			
Predicted Intruded	312	76	80.41%
Predicted Un-intruded	948	6224	86.78%

is a lazy learner and requires the complete dataset for each prediction, therefore, it gives more accurate results but at the expense of computational resources, area and memory overhead [19]. Table II also shows that, its accuracy is much higher as compared to the other algorithms. The Bayesian classification has less precision and accuracy as compared to DT because it is based on the probabilistic predictions and its accuracy can be improved by increasing the size of the dataset at the training stage. The DL algorithm performs less accurately as compared to the other ML algorithms but its precision can be improved by increasing the hidden layers. Due to the computational limitations, for this experimental setup, we used only 10 hidden layers. However, for a smaller dataset, when we increased the number of hidden layers beyond 50 then its accuracy approaches to approximately 93%. Thus, these comparison results show that the accurate and precise prediction decision trees provide the best option for detecting intruded hardware, among these algorithms for the given dataset.

F. Comparison with state-of-the-art Techniques

The proposed approach extracts the instruction set based power profile of the micro-controllers to train the ML algorithms for runtime Trojan detection. It is found out to be better than the other state of the art techniques in the following ways:

1) The proposed approach provide the runtime Trojan detection without utilizing the golden circuits with less off-chip computational overhead, like some other techniques [11], [12].
2) Unlike the other state-of-the-art techniques, the proposed approach can detect the Trojans that affect the both power and delay of the system [13].
3) The attributes used in the proposed methodology reflect the generic power behavior obtained by measuring at the power ports. This is contrast to some other techniques [15], [16] which only detect the Trojans that effect the communication of the SoC.

IV. Conclusion

This paper presents a power profile based methodology for runtime Trojan detection in micro-controllers. The proposed methodology consists of two major steps: The first step is to extract the assmebly language instruction based power profile of the given micro-controller, for which we propose to obtin power profile of micro-controller by utilizing its existing power ports in each pipeline stage. The second step is to train the online adaptive ML based model by using the extracted power profile of the assembly language instructions and then embed this trained model into the micro-controller for online Trojan detection. For illustration purpose, we implemented the intruded and un-intruded MC8051 in VHDL to extract the power profile and respective trained model by using some of the commonly used ML tools, i.e., k-NN, DT, DL and NBC. The experimental results show that k-NN has the highest accuracy of 99.02% because of high number of nearest neighbors (k>10) but at the expense of a large area overhead. Our technique is general enough to be applied to any pipelined microprocessor. In future, we plan to further validate our approach on various other pipelined architectures.

References

[1] M. Tehranipoor, "A survey of hardware trojan taxonomy and detection," *Design & Test Computer*, vol. 27, no. 1, pp. 10–25, 2010.

[2] H. Li, Q. Liu, J. Zhang, and Y. Lyu, "A survey of hardware trojan detection, diagnosis and prevention," in *Conference on Computer-Aided Design and Computer Graphics*, 2015, pp. 173–180.

[3] A. Gbade-Alabi, D. Keezer, V. Mooney, A. Y. Poschmann, M. Stöttinger, and K. Divekar, "A signature based architecture for trojan detection," in *Embedded Systems Security*, 2014, p. 3.

[4] 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 *Midwest Symposium on Circuits and Systems*, 2014, pp. 659–662.

[5] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. S. Hsiao, J. Plusquellic, and M. Tehranipoor, "Protection against hardware trojan attacks: Towards a comprehensive solution," *Design & Test*, vol. 30, no. 3, pp. 6–17, 2013.

[6] X. Cui, K. Ma, L. Shi, and K. Wu, "High-level synthesis for run-time hardware Trojan detection and recovery," in *Design Automation Conference*, 2014, pp. 157:1–157:6.

[7] M. Hicks, M. Finnicum, S. King, M. Martin, and J. Smith, "Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically," in *Security and Privacy*, 2010, pp. 159–172.

[8] D. Forte, C. Bao, and A. Srivastava, "Temperature tracking: An innovative run-time approach for hardware Trojan detection," in *Conference on Computer-Aided Design*, 2013, pp. 532–539.

[9] C. Bao, D. Forte, and A. Srivastava, "Temperature tracking: Toward robust run-time detection of hardware Trojans," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1577–1585, 2015.

[10] H. Zhao, K. Kwiat, C. Kamhoua, and M. Rodriguez, "Applying chaos theory for runtime hardware trojan detection," in *Computational Intelligence for Security and Defense Applications*, 2015, pp. 1–6.

[11] C. Bao, D. Forte, and A. Srivastava, "On application of one-class svm to reverse engineering-based hardware trojan detection," in *Quality Electronic Design*, 2014, pp. 47–54.

[12] ——, "On reverse engineering-based hardware trojan detection," *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 49–57, 2016.

[13] F. K. Lodhi, I. Abbasi, F. Khalid, O. Hasan, F. Awwad, and S. R. Hasan, "A self-learning framework to detect the intruded integrated circuits," in *International Symposium on Circuits and Systems*, 2016, pp. 1702–1705.

[14] T. Iwase, Y. Nozaki, M. Yoshikawa, and T. Kumaki, "Detection technique for hardware trojans using machine learning in frequency domain," in *Global Conference on Consumer Electronics*, 2015, pp. 185–186.

[15] A. Kulkarni, Y. Pino, and T. Mohsenin, "Adaptive real-time trojan detection framework through machine learning," in *Hardware Oriented Security and Trust*, 2016, pp. 120–123.

[16] ——, "Svm-based real-time hardware trojan detection for many-core platform," in *International Symposium on Quality Electronic Design*, 2016, pp. 362–367.

[17] M. Tehranipoor and H. Salamani, "trust-HUB," 2016. [Online]. Available: https://www.trust-hub.org/

[18] M. A. Mazidi, J. G. Mazidi, and R. D. McKinlay, *The 8051 microcontroller and embedded systems: using Assembly and C*. Pearson/Prentice Hall, 2006, vol. 626.

[19] F. Pernkopf, "Bayesian network classifiers versus selective k-nn classifier," *Pattern Recognition*, vol. 38, no. 1, pp. 1–10, 2005.