A Hardware Trojan Insertion Framework against Gate-Level Netlist Structural Feature-based and SCOAP-based Detection

Chi-Wei Chen*, Pei-Yu Lo*, Wei-Ting Hsu*, Chih-Wei Chen†, Chin-Wei Tien* and Sy-Yen Kuo*
*{r08921a28,r08921a29,r09921a30,cwtien,sykuo}@ntu.edu.tw †chihweichen@iii.org.tw
* Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
† Institute for Information Industry, Taipei, Taiwan

Abstract—**The division of labor in the integrated circuit (IC) industry has become more detailed. Outsourcing IC design or fabrication to third-party vendors is inevitable, which has caused a variety of hardware security issues. With the attention of hardware Trojan, most recent research has focused on detection techniques instead of attack or insertion methods. Existing detection methods for gate level can be classified into two categories: one based on circuit structure features and the other based on the Sandia Controllability/Observability Analysis Program (SCOAP) values. This paper proposes a new hardware Trojan insertion framework that can automatically construct Trojan with low SCOAP values against both structural features-based and SCOAP-based detection techniques. In our framework, we use the greedy algorithm to build a Trojan structure similar to a genuine circuit as much as possible and then reduce SCOAP values by inserting genuine nets. The experimental results demonstrated that our greedy method achieves significant improvements in structure generation than the random method. Moreover, the insertion framework can reduce the average SCOAP values by 68% after inserting three genuine nets. Finally, we evaluate state-of-the-art detection techniques using hardware Trojan samples generated by our proposed framework, which provides a high False Positive Rate (FPR)/False Negative Rate (FNR) and low accuracy.**

Index Terms—**Hardware security, Hardware Trojan, Gate-level insertion framework, Greedy algorithm, SCOAP reduction**

I. Introduction

With the globalization of the IC industry, the outsourcing of design and manufacturing will inevitably increase the risk of hardware security vulnerabilities. Hardware Trojan has attracted considerable attention among hardware security issues. Hardware Trojans usually consist of a trigger and payload. When the trigger conditions are met, the payload is triggered, which affects the function of the genuine circuit. To find out the hardware Trojan maliciously inserted by the attacker, most recent research has focused on machine learning-based detectors [11], with very little research on adversaries. The learning-based detectors built a detection model with structural feature and SCOAP values feature. However, we found that the impact of both features can be reduced by specific composition and insertion methods. In view of this, this paper proposes a new hardware Trojan generation framework to automatically generate Trojan samples that can evade the above mainstream detection methods to assist in the improvement and optimization of current detection methods.

In Fig. 1, we present an overview of the proposed insertion framework, which can separate into three primary phases. The users must provide the gate-level netlist Verilog file and input parameters defined in Table I. To ensure low trigger probability and increase the similarity between Trojan nets and genuine nets, our framework constructs a Trojan trigger with AND gate by the greedy algorithm. After that, the hardware Trojan is expected to have a high testability value with low active probability. It has been proven that it is easy to detect using SCOAP-based detection techniques. Thus, we insert the genuine net into hardware Trojan to obtain the average of Trojan nets' SCOAP values close to the average of genuine nets' SCOAP values. Finally, we used our hardware Trojan samples to evaluate the Trojan detection techniques [3] [4] [5], and the experimental results demonstrate that all three hardware Trojan detections have high FPR and FNR, which indicates a decrease in detector reliability.

Our main contributions are as follows:

- We proposed a flexible insertion framework that automatically inserts or generates hardware Trojan against both circuit structural feature-based and SCOAP-based detection.
- We then proposed the greedy algorithm to construct a Trojan structure which is better than the random method.
- We also proposed the SCOAP reduction method that can reduce hardware Trojan nets' SCOAP values effectively.
- We can guarantee the trigger probability upper bound and not affected by SCOAP reduction.

The remainder of this paper is organized as follows. In Section II, we discuss previous related studies. Section III describes the flow and algorithm of the proposed framework. Section IV presents the experimental results using state-of-the-art hardware Trojan detection. The article concludes in Section V.

II. Related Work

A. Gate-level Hardware Trojan

A hardware Trojan is a malicious modification to a circuit that affects the circuit function. In this work, we discuss gate-level hardware Trojans, which will consist of triggers and payloads. Typically, attackers will design Trojans that

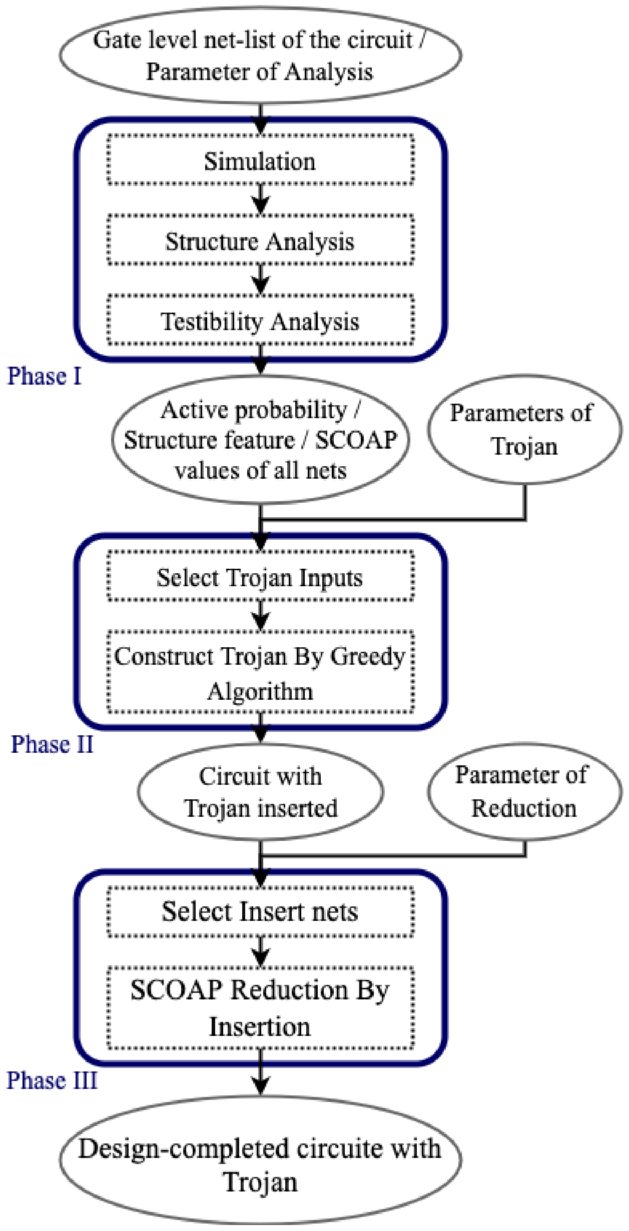
Gate level net-list of the circuit / Parameter of Analysis
Simulation
Structure Analysis
Testibility Analysis
Phase I
Active probability / Structure feature / SCOAP values of all nets
Parameters of Trojan
Select Trojan Inputs
Construct Trojan By Greedy Algorithm
Phase II
Circuit with Trojan inserted
Parameter of Reduction
Select Insert nets
SCOAP Reduction By Insertion
Phase III
Design-completed circuite with Trojan

Fig. 1. The proposed Hardware Trojan insertion framework.

are hard to trigger by low active probability nets. Once the activation condition is reached, the Trojan effect circuit is activated through the payload gate. These payload effects can be categorized as functional denial of service (DoS) or information leakage [2]. Because hardware Trojans can be easily inserted by third-party companies and are difficult to trigger, they will be a significant threat to hardware security.

B. Greedy Algorithm and SCOAP values

A greedy algorithm is an algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage. The greedy algorithm is successful in some problems, such as Huffman encoding or Dijkstra's algorithm [7]. The Sandia Controllability/Observability Analysis Program (SCOAP) [8] is a topology-based program developed to analyze digital circuit testability. Testability is related to the difficulty in controlling and observing the signals in digital circuits. In SCOAP, it first calculates 0-controllability(CC0) and 1-controllability (CC1) from primary inputs to primary outputs and then computing observability(CO) in reverse order. SCOAP values can achieve similar results faster and more effectively than the simulation.

TABLE I
USER INPUTS PARAMETERS

Parameters	Type	Description
r	Analysis	Simulation rounds
p	Trojan	Gate level Trojan Payload
t	Trojan	Threshold of Trojan inputs' probability
n	Trojan	Number of Trigger inputs
i	Reduction	SCOAP reduction interactions

C. Gate-Level Classification Methods

In previous works, HT classification methods were classified into two categories: structure-based and SCOAP-based. For structure-based methods, in [5], they were the first to use Support Vector Machine(SVM) to classify the Trojan nets. The 5 structural features were extracted from gate-level netlist and optimized the SVM by learning rate. The proposed method provides average 83.5% for TPR and 49.5% for TNR with dynamic weighting. To improve the Trojan detection results, they proposed new 51 features in [4] and selected the best 11 of 51 to maximize the F-measures. When using the random forest to identify hardware Trojan, average precision and average F-measure achieved 92.2% and 74.6%, respectively. For SCOAP-based detection, COTD [3] is the most well-known and effective detection method. Because Trojan nets often have high controllability and observability values, COTD can easily be distinguished by k-means clustering with an average 0% FPR and an average 0% FNR.

D. Gate-Level Insertion framework

Trust-Hub [1] [6] [9] provides the most frequently used hardware Trojan samples for training and testing. However, the experimental results in [3] [4] [5] show that it is easy to identify whether it is a Trojan or not. Moreover, the limited benchmarks bias the detection experiment results. Cruz et al. proposed a flexible and customized Trojan insertion framework for dynamic generating benchmarks to provide another benchmark generating technique [2]. An algorithm was developed to analyze and validate potential triggers and Trojan payloads. Commercial test tools were first used to ensure the validity of random Trojans and optimized the Trojan footprint to make it difficult to detect using a side-channel analysis-based detector.

III. PROPOSED METHOD

This section describes the detailed steps of the proposed insertion framework, including three significant phases as

TABLE II
STRUCTURE FEATURE SETS ($1 \leq x \leq 5$) [4]

Feature	Description
fan_in_x	The number of logic-gate fanins up to x-level away.
in_ff_x	The number of flip-flops up to x-level away.
in_mux_x	The number of multiplexers up to x-level away.
in_loop_x	The number of up to x-level loops.
in_const_x	The number of constants up to x-level away.
in_nearest_pin	The minimum level to the primary input from the net n.
in_nearest_ff	The minimum level to any flip-flop from the net's input.
in_nearest_mux	The minimum level to any multiplexer from the net's input.

illustrated in Fig. 1. All parameters and descriptions of the user inputs are listed in Table I.

A. Phase I: Analyzing the Trojan-free circuit

The first phase aims to analyze and calculate the active probability, structure features, and SCOAP value for all nets. This phase takes gate-level netlist v and simulation round r as input. First, we construct the hypergraph of the circuit by parsing the gate-level netlist file v. We then simulate the circuit with random inputs to obtain the active probability Pr, SCOAP values vector $\vec{s} = < CC0, CC1, CO >$ and structural features vector $\vec{f}$ shown in II for all nets. After the first phase is completed, we will have the active probability set, SCOAP values set, and structural features set as output.

B. Phase II: Constructing the hardware Trojan

The second phase aims to build the hardware Trojan by the greedy algorithm that makes its nets as similar as possible to genuine nets and finally insert it into gate-level netlist v. In order to measure the similarity between nets, first, we use the Eq. 1 to calculate the similarity value θ between each net's structural vector $\vec{f}$ and the average structural vector $\vec{f^*}$. The larger the value, the less similar the net is to the genuine nets. We then select n nets which active probability is less than parameter t but not 0 with the biggest θ as Trojan inputs.

$$\theta = \left\| \vec{f} - \vec{f^*} \right\|_2^{-1} \tag{1}$$

In this phase, the proposed method builds the Trojan trigger by only AND gate. We put all Trojan inputs into set S as candidates and start the greedy algorithm. As the greedy algorithm defines, we make the locally optimal choice at each stage. In our problem, since we hope the structural features of the Trojan nets and the genuine nets are as similar as possible, we connect any two nets in set S by AND gate and calculate the new output nets' θ. It is easy to calculate the new output net's structural feature vector $\vec{f'}$ from any two known input feature vectors $\vec{f_1}$ and $\vec{f_2}$. In each stage, we select the new net n_{new} which has the biggest θ as the optimal choice. When the stage end, we delete the input nets of n_{new} from set S and insert n_{new} in set S as the new candidate. The algorithm is completed until there is only one net leave in set S. After that, the insertion framework automatically constructs the trigger and payload into the gate-level netlist file v.

The time complexity of the proposed greedy method is $O(n^2)$, where n is the number of Trojan inputs. Because only AND gates are used, the proposed method can promise the upper bound of trigger active probability is

$$Pr(n_{trigger}) = \prod Pr(n_i), n_i \in N \tag{2}$$

,where N is the set of the Trojan inputs. A clear upper bound of trigger probability allows users to determine the threshold t easily. In Fig. 3, we display the example of $s38584\_greedy$ construct by proposed greedy method with parameters $r = 100000, v = s38584.v, p = ANDgate, t = 0.0015, n = 6$. The red nets are defined as Trojan nets and the upper bound of trigger probability is $Pr(n_{trigger}) = 1.14 * 10^{-17}$.

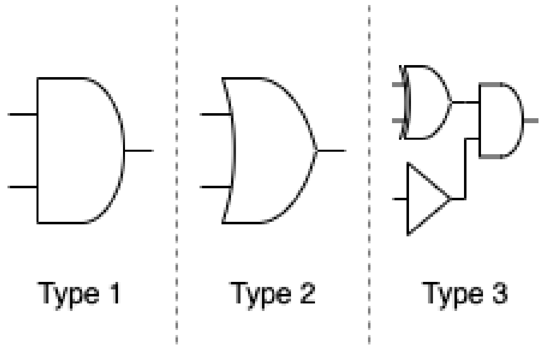

Fig. 2. Three types of the insert structure

C. Phase III: Reducing Trojan nets' SCOAP values

The purpose of the final phase is to reduce the Trojan nets' SCOAP value without raising the upper bound of trigger active probability $Pr(n_{trigger})$. After the greedy algorithm is completed, we expect to have a new Trojan inserted gate-level netlist with high testability values because of the characteristics of the AND gate. To reduce the Trojan nets' SCOAP values, we provide three types of structures shown in Fig. 2 that can effectively reduce the SCOAP value by inserting net n_i. According to calculation rules in [8], type 1 can reduce the net's CC0 value, type 2 can reduce the net's CC1 value, and type 3 can reduce both CC0 and CC1. The three insertion structures have their advantages and disadvantages. For types 1 and 2, they can make fewer changes to the hardware Trojan trigger structure and give a more significant reduction than type 3. However, for type 2, if we try not to raise the upper bound of $Pr(n_{trigger})$, we must only choose 0 logic net as the insert net. Or gate connects with 0 logic is easy to detect if the detector uses the synthesis tool to do a redundancy removal. Thus, to escape the detection of the synthesis tool, the adversary can use type 3 instead of type 2 to reduce CC1 with no redundant gate generation.

In this phase, we take the parameter i and Trojan inserted gate-level netlist v' as input. In each iteration, we first select the net n_{cc0} with lowest CC0 and n_{cc1} with lowest CC1. As we discussed above, we do not consider logic 0 and 1. For each Trojan net, we have four combinations to try : type 1 with n_{cc0}, type 2 with n_{cc1}, and type 3 with both n_{cc0} and n_{cc1}. After simulating all insert combinations for all Trojan nets, we only let the new Trojan with maximum reduction rate δ as the

TABLE III
SCOAP VALUE CHANGES BEFORE AND AFTER REDUCTION

Trojan nets	Before			After		
	CC0	CC1	CO	CC0	CC1	CO
t1	10	436	500	10	436	347
t2	11	174	762	11	174	772
t3	11	495	411	11	495	288
t4	12	670	266	4	512	271
t5	13	876	60	5	718	65
t6	9	928	8	4	62	8

next iteration input $v^{'}$. To calculate the reduction rate δ, we must first calculate the original average Euclidean distance d_o of Trojan nets' SCOAP vector $< CC0, CC1, CO >$ and the new average Euclidean distance d_{new} of the new Trojan nets' SCOAP vector, and then calculate δ as follows:

$$\delta = \frac{d_o - d_{new}}{d_o} \tag{3}$$

When the SCOAP reduction has been executed i iterations, the phase will be completed, and the file $v^{'}$ is the output gate-level netlist with a deigned-completed hardware Trojan inserted.

The example of SCOAP reduction is shown in Fig. 4. After passing $s38584\_greedy$ (Fig. 3) through this reduction phase with parameter $i = 2$, we will have six new Trojan nets($t2^*, t5^*, t7 - t10$) and six original Trojan nets ($t1 - t6$). Then we record the SCOAP values changes of original ones in Table III. We can clearly find out that CO values of $t1$ and $t3$ are reduced by 153 and 123, and cc1 values of $t4 - t5$ are reduced by 158, 158, and 866, respectively.

IV. Experiments

A. Experimental Setup

To demonstrate the effectiveness of our insertion framework, we programmed the tool according to the proposed insertion framework in C#. Our tool constructed Trojan with parameters $r = 100000, p =$ AND gate$, t = 0.0015, n = 6, i = 3$, and then inserted Trojans into four ISCAS'89 benchmarks ($s15850, s35932, s38417$, and $s38584$), which are most commonly used to train the detection. To avoid redundancy removal by the detector, we only used type 1 and type 3 structures to reduce the SCOAP value of the hardware Trojan. Then we evaluate the state-of-the-art structural feature-based

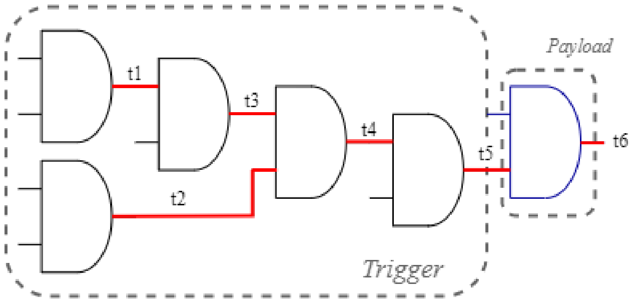

Fig. 3. s38584_greedy

and SCOAP-based Trojan detection [3] [4] [5] using the hardware Trojan samples we generated.

B. Result and Analysis

To compare the greedy and random methods, we randomly generated 20 hardware Trojan samples and a hardware Trojan sample by the greedy algorithm with the same Trojan inputs according to four ISCAS benchmarks. We sum the similarity value θ of all Trojan nets as λ representing the similarity between the overall Trojan and genuine circuit for each hardware Trojan sample. Next, we sorted the 20 random Trojans and one greedy Trojan from the same ISCAS benchmarks by their λ and then displayed the ranking results of the greedy method in the "R_g" column of Table IV. For s15850, s35932, and s38584, our proposed greedy method ranked first and achieved the second-highest value for s38417. The experimental results fully match our assumptions about the greedy algorithm that the final results are not always optimal but consistently excellent.

We evaluate structure-based detection [4] [5] using our generated greedy hardware Trojan samples and show the False positive rate (FPR) and False negative rate (FNR) from the experimental results in Table V. For [4], four test cases have an average of 85% FNR and only 4.99% FPR, which means most Trojan successfully disguises themselves as genuine nets. For [5], four test cases have an average of 55% FNR, which is much higher than those provided. The high FNR greatly decreases the usefulness of the detector. In general, our proposed greedy method causes an average of 4.99% FPR and 85% FNR for [4] and 4.87% FPR and 55% FNR for [5], respectively.

In Fig. 5, there are an average of 0.43, 0.32, and 0.19 of reducing rate δ for iterate SCOAP reduction 1,2 and 3 times. After iterating the SCOAP reduction method three times, the average Trojan SCOAP values are reduced by 69% and prove that our method has a significant effect on lowering SCOAP values. Moreover, when we evaluate COTD [3] using our generated hardware Trojan samples with three genuine nets insertion (three iterations), our hardware Trojan samples result in an average of 21.29% FPR and 57% FNR. Because COTD uses the clustering method , every time we lower the SCOAP values of Trojan nets, it increases the FPR and FNR. Therefore, we do not need to worry about inserting too many nets to make Trojan too large because inserting a few nets can make the COTD lose its functionality.

V. Conclusion

This paper presents a new hardware Trojan insertion framework against structural feature-based and SCOAP-based detection. We first generate hardware Trojan structure by the proposed greedy method and then reduce SCOAP values by inserting three types of insertion structures. The experimental results demonstrate that our proposed method can find excellent Trojan structure more efficiently than the random method and significantly reduce the SCOAP value of hardware Trojans. Both structure-based and SCOAP-based detection

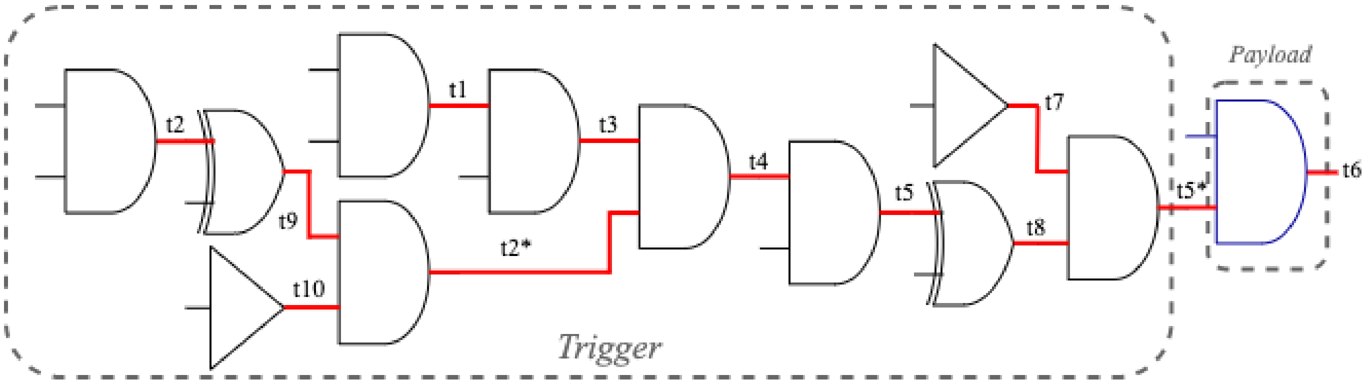

Fig. 4. s38584_greedy_2

TABLE IV
GREEDY METHOD(1) V.S. RANDOM METHOD(20)

HT samples	R_g	Greedy(1)	Random(20)		
		λ	Max λ	Min λ	Mean λ
s15850_greedy	1	0.83	0.68	0.43	0.51
s35932_greedy	1	1.69	1.69	1.30	1.48
s38417_greedy	2	0.71	0.72	0.39	0.57
s38584_greedy	1	1.21	1.11	0.57	0.83

TABLE V
EXPERIMENT RESULTS(%)

HT samples	SCOAP values		Structure Feature			
	[3]		[4]		[5]	
	FPR	FNR	FPR	FNR	FPR	FNR
s15850_greedy_3	11.78	66.67	0.86	73.33	5.57	66.67
s35932_greedy_3	37.47	26.77	17.64	73.33	3.17	46.67
s38417_greedy_3	16.64	66.67	1.06	100	6.69	53.33
s38584_greedy_3	19.28	66.67	0.43	93.33	4.1	53.33
average	21.29	57	4.99	85	4.87	55

provides high FPR/FNR and low performance when testing by our insertion framework. Moreover, propsed framework can be used to generate a large number of benchmarks that are significant for evaluating detection techniques. Since hardware Trojan benchmark generating technology is minimal, the detector will continue to improve based on the small amount of data available, which may lead to the overestimation of their experimental results. To establish more unbiased standards, the development of generation technology will be the priority now, and we believe our insertion framework can help improve it.

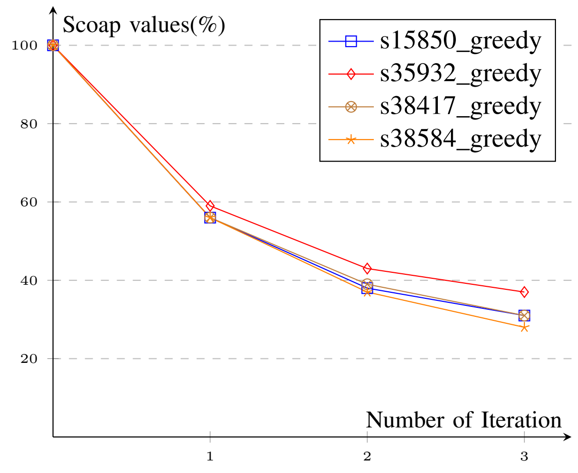

Fig. 5. SCOAP values after SCOAP Reduction

REFERENCES

[1] Trust-hub. Available on-line: https://www.trust-hub.org, 2016.
[2] J. Cruz et al., "An automated configurable Trojan insertion framework for dynamic trust benchmarks," 2018 Design, Automation Test in Europe Conference & Exhibition (DATE), 2018, pp. 1598-1603.
[3] H. Salmani, "COTD: Reference-Free Hardware Trojan Detection and Recovery Based on Controllability and Observability in Gate-Level Netlist," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 2, pp. 338-350, Feb. 2017.
[4] K. Hasegawa et al., "Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1-4
[5] K. Hasegawa et al., "Hardware Trojans classification for gate-level netlists based on machine learning," 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS), 2016, pp. 203-206.
[6] H. Salmani et al., "On design vulnerability analysis and trust benchmarks development", 2013 IEEE 31st International Conference on Computer Design (ICCD), 2013, pp. 471-474.
[7] Black, Paul E. (2 February 2005). "greedy algorithm". Dictionary of Algorithms and Data Structures. U.S. National Institute of Standards and Technology (NIST). Retrieved 17 August 2012.
[8] L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia Controllability/Observability Analysis Program," 17th Design Automation Conference, 1980, pp. 190-1
[9] B. Shakya et al., "Benchmarking of Hardware Trojans and Maliciously Affected Circuits", Journal of Hardware and Systems Security (HaSS), April 2017.
[10] Zhixin Pan and Prabhat Mishra, "Automated Test Generation for Hardware Trojan Detection using Reinforcement Learning", Proceedings of the 26th Asia and South Pacific Design Automation Conference, 2021
[11] B. Tan and R. Karri, "Challenges and New Directions for AI and Hardware Security," 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS), 2020