Journal of Electronic Testing (2021) 37:473–487
https://doi.org/10.1007/s10836-021-05960-2

SC-COTD: Hardware Trojan Detection Based on Sequential/ Combinational Testability Features using Ensemble Classifier

Mahshid Tebyanian[1] · Azadeh Mokhtarpour[1] · Alireza Shafieinejad[1]

Received: 14 January 2021 / Accepted: 19 July 2021 / Published online: 24 August 2021

Abstract
Security against Hardware Trojans (HT) is an important concern in integrated circuits (IC) design and fabrication. Most of the current HT detection methods are based on the golden model of circuit design. Further, some approaches require test pattern for HTs activation. In this paper, we propose *SC-COTD* (Sequential/Combinational Controllability and Observability features for hardware Trojan Detection), an effective hardware Trojan detection to get rid of both golden chip and test pattern limitations. *SC-COTD* uses both sequential and combinational testability measures to detect and locate HT signals by a machine learning approach. This method deploys an ensemble classifier based on k-means clustering. The clustering models have diverse variety in testability features along with size of clustering which inspect and reveal different aspects of netlist conventional for a collaborative scheme. The clustering results are filtered and then fed into a decision-making procedure based on majority voting to eliminate the limited flaws of each model. The evaluation results on TrustHUB benchmarks demonstrate that, *SC-COTD* can detect and locate HTs with 100% without any false negative, i.e., Recall = 1. Although our method has a limited number of false positive, it has the best performance in comparison to well-known previous approaches.

Keywords Controllability and observability · Hardware testability · K -means clustering · Hardware Trojan detection · Hardware security

1 Introduction

In the last decade, increasing demand for hardware designs has forced manufacturers to get help from third-party vendors to produce chips, i.e., the *distributed form of chip production* [1]. However, the manufacturing relies on third-party IP can lead to serious threats on the security and reliability of ICs such as *hardware Trojan*. Hardware Trojans (HTs), now as a major challenge in Integrated Circuit (IC) security, have raised serious concerns from industry, government, military department and other critical communities.

HTs are malicious modifications to an IC that can transform IC functionality, reveal valuable information, reduce reliability, and even incapacitate a chip. A hardware Trojan usually consists of two parts: the trigger and the payload. The former monitors signals within the chip, waiting for the occurrence of an event or a series of events to take place. The latter is responsible for the malicious behavior of the Trojan i.e. to transmit private data or to corrupt certain internal signals. When the triggering conditions are met, the trigger circuit enables the payload part in order to execute the desired malicious function.

For Trojans with both payload and trigger, the payload remains inactive most of the time, since the trigger circuit is designed to active under rare conditions to avoid being detected. Further, for Trojans in logic or memory circuits, the faulty behavior may be activated either by a certain input or a sequence of input combinations, triggering the undesirable or faulty behavior [2]. These rare triggering events are usually designed to be stealthy and undetectable during simulation and test. This secret nature of HTs makes it difficult to identify them. Therefore, hardware Trojans detection needs more and more attention.

Responsible Editor: S. Bhunia

✉ Azadeh Mokhtarpour
a.mokhtarpour@modares.ac.ir

Mahshid Tebyanian
m.tebyanian@modares.ac.ir

Alireza Shafieinejad
shafieinejad@modares.ac.ir

[1] Department of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran

HT detection can be classified into two major groups: *post-silicon detection* and *pre-silicon detection*. Post-silicon detection mainly divides into three categories: side channel analysis, reverse engineering and functional testing. Among them, side channel analysis [3], which is widely applied, compares IC characteristics such as delay, power consumption, and temperature with a golden chip, i.e., malware-free chips. Detecting these malicious events usually requires a golden reference model [4] that is assumed to be Trojan-free. The main drawback of the golden reference model is that it may be inconclusive or too complex for exhaustive verification, especially for large designs [3]. Another fundamental limitation of side channel approach is that the effect of a small enough Trojan on both of the logic and side-channel fingerprints of the circuit, can be masked by process variation and noise. Moreover, it is amplified by increasing the scale of integration and processing diversity in advanced nodes.

On the other hand, pre-silicon methods mainly apply HT detection on gate-level netlists. These methods are categorized into two types: the dynamic and static detection. Dynamic detection techniques are based on the activation of HTs parts. For example, FANCI marks gates with low activation probability as suspicious [5]. VeriTrust marks gates that are not driven by functional inputs as suspicious [6]. The implicit assumption here is that those gates are driven by Trojans, as they do not perform any computation on functional inputs. Finally, the SoC integrator manually checks for fewer suspected gates to determine whether they are a Trojan or genuine. However, HTs are rarely activated under ordinary functional verification constrains and accordingly it leads to complexity of detection [7]. By contrast, static detection techniques do not require any test pattern generation. Further, these techniques can detect various HTs with different functions by utilizing existing HTs related characteristics.

This paper proposes a golden chip free model by using ensemble clustering method based on gate-level netlists. More specifically, we discuss the following set of issues:

Use of Testability Features for Hardware Trojan Detection: In order to reduce the detectability, hardware Trojans designers will try to improve the stealth characteristics and distribution of Trojan insertion. To hide the activity of Hardware Trojans, the trigger might be connected to nets with low controllability and observability. Thus, such as prior work [8, 9], we use the controllability and observability of the circuit nets as the basic features for detection of HT signals.

Design of a Tool to Evaluate the Aforementioned Features: Due to some limitations of testability measurement tools, we design and implement a new tool to compute controllability and observability of circuit nets. We take the HT benchmarks at the gate-level netlists as the inputs, and perform the testability analysis to determine the controllability and observability values. This tool provides the computation of both sequential and combinational testability features.

Static Trojan Detection using Ensemble Classifier: Based on both corresponding testability feature and the size of clustering, we have different classifications. For final decision, we use a smart filtering and then an ensemble classifier to aggregate the results, separating HT inserted netlist from normal netlist. This ensemble classifier is based on k-means algorithm with different values for k.

1.1 Prior Work on Static Hardware Trojan Detection

In the past decade, various static hardware Trojan detection methods have been proposed. The latest of them have been integrated with machine learning based approaches. In the dominant work COTD [8], the controllability and observability analysis are used to detect hardware Trojans. Controllability and observability of a signal are two important attributes related to testability of a design. Controllability of a signal is the ability to force the value of the signal to become '0'/ '1' by applying suitable input vectors. Observability is the ability to observe the signal state at a primary output [10]. COTD uses the Sandia Controllability and Observability Analysis Program (SCOAP) [11] to compute the controllability/observability values for each signal in the netlist. The signals are clustered based on these features using a k-means clustering algorithm with $k=3$. However, COTD uses only combinational measurements, sequential testability measures are also necessary because some Trojans manipulate not only the combinational signals but also sequential signals [12].

The work in [13] proposed a Trojan net detection method by training supervised classifiers using both combinational and sequential testability values as features in gate-level netlists with considerably large circuit size. Then the model is learned based on four popular supervised machine learning algorithms for binary-class classification: Fine Tree, Weighted k-NN, Fine Gaussian SVM and Bagged Trees.

In [9], the inter-cluster distance was calculated and combined with the number of primitives, such as "AND" and "OR" gates in the circuit, as a four-dimensional vector to input to a SVM (Support Vector Machine) classifier for HT detection. Here, the primitives are all the information of inputs/outputs, DFFs, BUFs, MUXs and other gates.

Hasegawa proposed new learning structure features for Trojan detection [14]. Thus 51 features are extracted based on (1) the logic fan-in; (2) the number of flip-flops away

from the input and output of the target net; (3) the level of the nearest flip-flop to the input and output of the target net; (4) the number of multiplexers away from the input and output of the target net; (5) the level of the nearest multiplexer to the input and output of the target net; (6) the number of nets that are assigned a constant value of 0 or 1 and (7) the number of n-level-loops. The computation level is up to 5 from the input and output of the target net. After feature extraction, these 51 features reduced to most-important 11 features using random forests classifier.

Bian et al. [15]. Provides another machine learning based approach for HT detection. The machine learning algorithm is *k*-means which uses two types of clustering models, the partitioning-based and the density-based clustering.

Wang et al. [16]. Detects gate-level hardware Trojans by identifying the Trigger nets of potential Trojan-infected circuits with the use of an ensemble learning algorithm. The authors claim that the proposed algorithm detects both combinational and sequential Trojans, based on the Trigger characteristics of the Trojans.

In [17], the structural features in gate-level netlist of both the HT circuits and the host circuits are examined and combined for HT detection. This method is based on the fact that the elusiveness of HTs relies on not just HT structure design but also on the insertion positions in the host circuits, in which HTs are usually inserted at either the low controllability position or the low observability position owning the stealth characteristic.

In [18], some characteristics such as the number of DFFs, MUXs, loops, logic gate fan-ins, logic levels to primitive input/output ports are extracted as Trojan-net features. These features are feed to a neural network with two units in the output layer. One unit corresponds to the normal nets, and the other corresponds to the Trojan nets. When the output value of the former is larger than that of the latter, the net is considered as a normal net; otherwise, identified as a Trojan net. However, as previously mentioned, the common problem of most of these approaches is the requirement for a golden chip. Further, some other approaches have limitation in detecting sequential HTs. [19]. Moreover, in spite of detection the existence of hardware Trojans based on static methods, they are incapable of locating the Trojan signals [20].

1.2 Contribution of the Work

The main contributions of this paper can be summarized as follows:

- We use both sequential and combinational SCOAP features for proposed algorithm which improves the precision of our signal classification. In other words, our scheme finds the Trojan which previous methods are incapable of identifying them.
- We propose a machine learning based algorithm that classifies the Trojan signals from genuine signals based on Ensemble Classifier with:
  - Identification of Trojan-Signals: The proposed method, in addition to detecting the existence of a Trojan, specifically distinguishes Trojan signals from the genuine ones.
  - Golden chip free method: All the steps are done without a need for golden reference model.

Our proposed method basically is an extension of COTD [8] with two differences: first we use both sequential and combinational testability measures instead of one of them and second we classify normal nets from HT inserted nets based on ensemble classifier instead of a single *k*-means clustering method. Further, although the COTD claimed that achieve 100% TPR and TNR in TRUST-Hub benchmarks, it fails in some circuits such as S38417-T200 and S38417-T200 due to closeness of combinational features of Trojan and genuine nets. However, our method is capable to detect Trojans in all TRUST-Hub benchmarks.

The rest of the paper is organized as follows. Section 2 describes our hardware Trojan detection approach of using proposed tool STMT. Section 3 describes the experimental setup, analysis, and evaluation results. Finally, Section 4 concludes the paper.

2 Proposed Method for Hardware Trojan Detection

In this paper we proposed a method named *SC-COTD* (**S**equential/**C**ombinational **C**ontrollability and **O**bservability features for hardware **T**rojan **D**etection). This novel method, uses both sequential and combinational testability measures based on SCOAP to detect and locate HTs.

2.1 Definitions and Background

Hardware testing is used to ensure that each component of a system is operating as it should be. In the last decade, increasing the density of transistors at the chip surface makes the testing process difficult. Thus, vendors are tending to design and fabricate circuits in a way that the testability process are facilitated. The adoption of testable design imposes some constraints to the designer. It is a beneficial method where in the resulting digital circuit will have features to

Table 1 Combinational Controllability Calculation Formulas For Primary Logical Gates

	0-Controllability	1-Controllability
Primary Input	1	1
AND	min(CC0(a),CC0(b)) + 1	CC1(a) + CC1(b) + 1
OR	CC0(a) + CC0(b) + 1	min(CC1(a),CC1(b)) + 1
NOT	CC1(a) + 1	CC0(a) + 1
NAND	CC1(a) + CC1(b) + 1	min(CC0(a),CC0(b)) + 1
NOR	min(CC1(a),CC1(b)) + 1	CC0(a) + CC0(b) + 1
BUFFER	CC0(a) + 1	CC1(a) + 1
XOR	min(CC0(a) + CC0(b), CC1(a) + CC1(b)) + 1	min(CC0(a) + CC1(b), CC1(a) + CC0(b)) + 1
XNOR	min(CC0(a) + CC1(b), CC1(a) + CC0(b)) + 1	min(CC0(a) + CC0(b), CC1(a) + CC1(b)) + 1

facilitate the production of test sequences as well as test processes. In summary, testability is related to the difficulty of controlling and observing the logical values of internal nodes from circuit inputs and outputs, respectively. SCOAP is a well-known method to calculate the testability measurement. In SCOAP, controllability and observability are two primary metrics which can be calculated in two structural methods: Sequential (SC0, SC1 and SO) and combinational (CC0, CC1 and CO).

Controllability is the difficulty of setting and controlling intermediate/output signal to 0 or 1 by means of primary input signals. Observability is difficulty of observing changes in the input/intermediate signals at the primary outputs. A circuit with low values for signal controllability and observability is testable.

Table 1 summarizes the controllability formulation for primary logical gates. Further, Table 2, shows the rules corresponding to observability calculation for primary combinational elements. For a comprehensive manual of SCOAP computation, you can refer to [11].

Throughout this paper, we denote SCOAP measurements for combinational controllability-0/1 by CC0/CC1, combinational observability by CO, sequential controllability-0/1 by SC0/SC1 and sequential observability by SO, respectively. The higher the value of controllability/observability, the more difficult for the net to be controlled or observed. The SCOAP values for each net can be extracted according to the logic of circuit. For primary element without loop, controllability is defined for outputs while observability is defined for inputs.

The SCOAP features for each signal can be represented by both triples $<$CC0, CC1, CO$>$ and $<$SC0, SC1, SO$>$. As CC0/SC0 and CC1/SC1 introduces the same property, we can integrate them into a single feature with the Euclidian distance. More specifically, the CC/SC features for each signal s are defined by:

$$CC(s) = \sqrt{CC0(s)^2 + CC1(s)^2} \tag{1}$$

$$SC(s) = \sqrt{SC0(s)^2 + SC1(s)^2} \tag{2}$$

The above definitions allow us to represent the SCOAP features by the pairs $<$CC, CO$>$ and $<$SC, SO$>$ instead of triples $<$CC0, CC1, CO$>$ and $<$SC0, SC1, SO$>$. This leads to reduction of the dimension of algorithm and accordingly its complexity.

2.2 SC-COTD Basic Architecture

Testability measures are useful for design for test because they are initially developed to analyze the testing difficulty and estimate the fault coverage. In normal circuit, a design with good testability will keep the testability measures of each net as low as possible, so that a fault could be easily tested. HT signals are always stealthy and the triggering condition is a very rare event. Such design makes the Trojan nets difficult to be controlled and observed and thus Trojan nets tend to have a much higher average of testability measures

Table 2 Combinational Observability calculation formulas

	Input "a" observability	Input "b" observability
Primary Output	0	0
AND/NAND	CO(z) + CC1(b) + 1	CO(z) + CC1(a) + 1
OR/NOR	CO(z) + CC0(b) + 1	CO(z) + CC0(a) + 1
NOT/BUFFER	CO(z) + 1	CO(z) + 1
XOR/XNOR	CO(z) + min(CC0(b),CC1(b)) + 1	CO(z) + min(CC0(a),CC1(a)) + 1

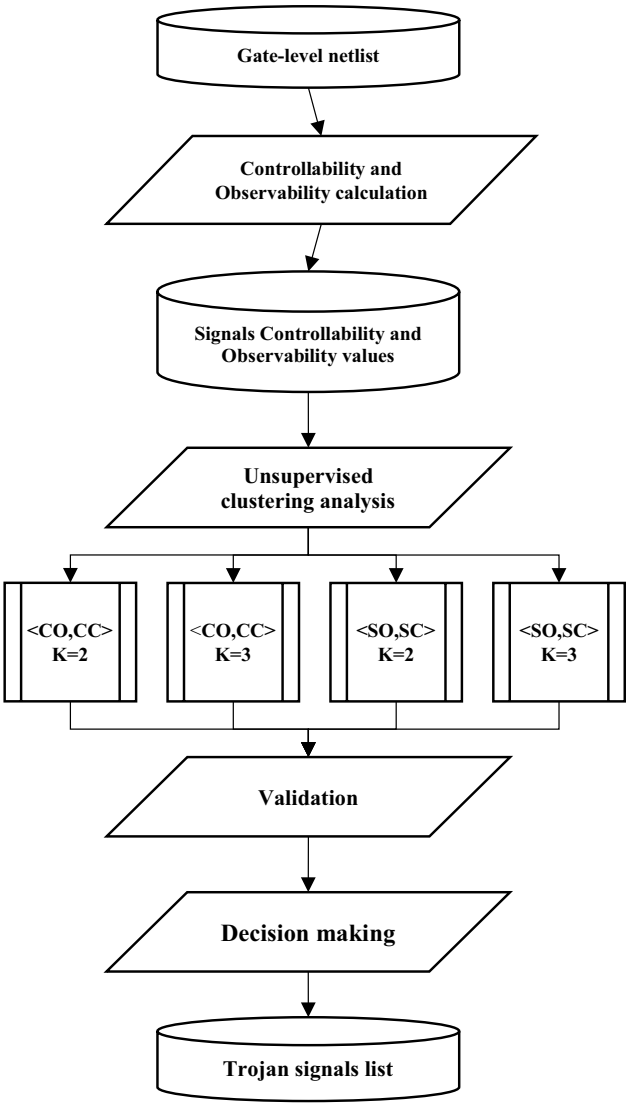


Fig. 1 The overview of *SC-COTD* approach

than the normal nets. Figure 1, shows the basic steps of the proposed method. In the first step, the controllability and observability values of each signal base on SCOAP method are calculated. In the second step, the signals are clustered by the *k*-means algorithm using the extracted features. Based on the size of clustering and the sequential/combinational SCOAP values, we will have four classifications. Finally, the clusters are filtered using a smart method and are aggregated to result the final decision, i.e., labelling each signal as genuine or Trojan.

2.3 Feature Extraction

As explained earlier, to hide Trojans, it is necessary to insert them in somewhere that is not activated during the test phase. Therefore, in the design of the Trojan's trigger circuit, signals are used that cannot be easily controlled by the primary inputs. The Trojan's payload affects the parts of the circuit that are not easily observed at the primary outputs. Therefore, one of the most important features of HT signals is the low value of testability measures. The COTD method uses only combinational controllability/observability features to detect HTs. This issue raises a problem for detecting Trojan whose trigger-part is a sequential circuit, i.e., activating the Trojan is subject to the system's arrival in a particular state. Thus, in *SC-COTD*, in addition to the combinational features, sequential features are considered. We represent the SCOAP features in two pairs $<$CC, CO$>$ and $<$SC, SO$>$ by the Eqs. (1) and (2).

2.4 Classification

By collecting the required features, the classification step can start. To analyze the features, we use unsupervised classification method with *k*-means clustering. The input of the clustering is either two-dimensional vector $<$CC, CO$>$ or $<$SC, SO$>$. The first challenge is the selection of the number of clusters, namely *k,* for the algorithm. The second challenge is to decide whether a clustering is valid or not valid for involving it in final decision.

In the following two sub-sections, the initial parameters of the *k*-means algorithm and the decision criterion are discussed.

2.4.1 Number of clusters

Primary, we consider different number of clusters for classification to achieve high precision. Empirically, we found that choosing $k=4$ and more does not lead to meaningful clustering. Let INF denotes the infinity value. As we expect that large value for either controllability or observability implies Trojan signal, we predict that the clusters near to (0, INF), (INF,0) and (INF,INF)are likely to be Trojan signals while the ones near to (0,0) is likely to be genuine signals. Thus, it is likely to converge the clustering algorithm to four different zones around the points (0,0), (0,INF), (INF,0) and (INF,INF). The experimental results not only show that $k>4$ is not a suitable choice but also implies that $k=4$ does not lead to a meaningful clustering. Thus, we ignore the case $k\geq 4$ and limit the options to $k=2$ and $k=3$. Consequently, regarding with the sequential/combinational features, we will have four clustering models:

- Clustering $<$CC, CO$>$ with $k=2$
- Clustering $<$CC, CO$>$ with $k=3$
- Clustering $<$SC, SO$>$ with $k=2$
- Clustering $<$SC, SO$>$ with $k=3$

Throughout this paper, we denote these clusterings as Γ_{C2}, Γ_{C3}, Γ_{S2} and Γ_{S3}, respectively. Further, the resulting clusters for Γ_{C2} and Γ_{S2} are denoting by $\sigma_{(0,0)}$ and $\sigma_{(I,I)}$. While the clusters which are given by either Γ_{C3} or Γ_{S3} are denoting by $\sigma_{(0,0)}$, $\sigma_{(0,I)}$ and $\sigma_{(I,0)}$. It is worth noting that COTD which use $k=3$ with <CC, CO> for clustering, is the same as our second clustering.

2.4.2 Determining the Initial centroid of the Cluster

The *k*-means clustering method is an exploratory method, and accordingly the correct determination of the initial points of centers plays an important role in the resulted clusters. In each of the four clustering models proposed above, the cluster of genuine signals is supposed to be formed near the origin, i.e., the primary centroid of the genuine cluster is set to (0,0) for.

each of the four models. However, as there is no negative value in SCOAP features, the center of the genuine cluster is converged to a point different than the origin.

The remaining cluster(s) which likely contain the Trojan signals, are expected to contain the large testability values either for controllability or observability.

Signals that are related to the Trojan trigger are less controllable and are expected to be located around the (INF,0) while the payload signals are mostly less observable and are expected to be placed around the (0,INF). Thus, for $k=3$ clustering, we set the initial point of centroids to (0,0), (0,INF) and (INF,0) respectively. Further, for $k=2$ clustering, we set the initial point of centroids to (0,0) and (INF,INF) respectively. We select (INF,INF) as centroid of Trojan cluster since it close to both (0,INF) and (INF,0) signals. It is worth mentioning that INF is finally replaced with the maximum amount of controllability/observability measurement are given for each experiment. Therefore, the cluster centers are different for each circuit and for each clustering model.

2.5 Decision Making

This step, decision making phase, starts with filtering of the clustering models and continues with voting. We discuss them in the next two subsections.

2.5.1 Filtering

At this step, the validity of each clustering model is examined. Since the Trojan's circuit should occupy only a small portion of the original circuit in order to remain hidden, it is expected that after the clustering, there is a single large cluster corresponds to genuine signals and one/two small cluster(s) corresponds to Trojan signals. Let α denotes the maximum acceptable value for the percentage of the Trojan signals to total signals. The filtering procedure is described in Algorithm 1.

Algorithm 1 Filtering

input: Γ : Clustering, k : Clustering size
α : Validity threshold
output: IsValid, S_G, S_T, R_G, R_T

1. $N\leftarrow$ number of signals in netlist
2. Let the clusters of Γ denoted by $\{\sigma_{(0,0)},\sigma_{(I,I)}\}$ when k=2 and $\{\sigma_{(0,0)},\sigma_{(0,I)}, \sigma_{(I,0)}\}$ when k=3
3. $S_G\leftarrow \sigma_{(0,0)}$
4. **if** (k=2) **then**
5. $S_T\leftarrow \sigma_{(I,I)}$
6. **else**
7. **for** each σ in $\{\sigma_{(0,I)}, \sigma_{(I,0)}\}$ **do begin**
8. **if** ($|\sigma| \geq \alpha.N$) **then**
9. Insert signals of σ into S_G
10. **else**
11. Insert signals of σ into S_T
12. **end if**
13. **end for**
14. **end if**
15. **if** ($S_T \neq \emptyset$ and $|S_T| \leq \alpha.N$) **then**
16. IsValid ←True
17. *else*
18. IsValid ←False
19. go to line 32
20. **end if**
21. $DIST_G = DIST_T \leftarrow 0.0$
22. **for** each signal *s* in netlist **do begin**
23. *cent* ←centroid of the cluster where in *s* is located
24. **if** (s belongs to a cluster in S_G)
25. $DIST_G \leftarrow DIST_G + |pos(s) - cent|$
26. **else**
27. $DIST_T \leftarrow DIST_T + |pos(s) - cent|$
28. **end if**
29. **end for**
30. $R_T \leftarrow DIST_T / |S_T|$
31. $R_G \leftarrow DIST_G / |S_G|$
32. return (IsValid, S_G, S_T, R_G, R_T)

The primary candidate for genuine signals is the cluster around the centroid (0,0), namely $\sigma_{(0,0)}$. Further, when $k=2$ (for clusterings Γ_{C2} and Γ_{S2}), the set of Trojan signals is initialized by the cluster around the centroid (INF,INF), namely $\sigma_{(I,I)}$. The case for $k=2$ (for clusterings Γ_{C3} and Γ_{S3}) has a little different since there are two candidates for Trojan signals, the clusters $\sigma_{(0,I)}$ and $\sigma_{(I,0)}$. We insert each of them into set of Trojan signals if the number of signals is less than α-percentage of total signals. Otherwise, we insert each of them into set of genuine signals.

After categorization signals into either S_G and S_T, we check the Trojan set. If it is either empty ($S_T \neq \emptyset$) or larger than the α-percentage of entire signals ($|S_T| > \alpha.N$), the clustering is set to invalid and accordingly is filtered from the decision-making procedure. Otherwise, it is accepted and entered into decision-making. Further, we compute two additional parameters R_G and R_T which respectively denote the average distance of genuine and Trojan signals from the cluster centroid. Since the Euclidean distance is used for clustering metric, we further involve it in average-distance measurement. The average distance is used during decision-making procedure when the Trojan and genuine

votes of a specific signal becomes equal. Finally, the five-tuple (*IsValid,* S_G, S_T, R_G, R_T) is returned by the algorithm.

2.5.2 Final Decision

At this step, four clustering models are fed as the inputs into the algorithm. At the beginning loop, each of the clusterings Γ_{C2}, Γ_{C3}, Γ_{S2} and Γ_{S3} is verified by Algorithm 1 to select the valid of them. If none of them is valid, the whole circuit is labeled as Trojan-free and algorithm ends. Otherwise, the algorithm entered into majority-voting loop by S_{VALID} containing the valid clusterings. Each clustering model has one vote per sample, and each vote has the same weight.

Algorithm 2 Majority Voting Procedure

input: Γ_{C2}, Γ_{C3}, Γ_{S2}, Γ_{S3}
output: Signal labels

```
1.  S_VALID ← ∅
2.  for each Γ in (Γ_C2, Γ_C3, Γ_S2, Γ_S3) do begin
3.     (IsValid_i, S_G, S_T, R_G, R_T) ←Filtering of Γ using Algorithm 1
4.     if (IsValid)
5.        Insert (S_G, S_T, R_G, R_T) into S_VALID
6.     end if
7.  end for
8.  if (S_VALID= ∅) then
9.     The circuit is TROJAN-FREE
10.    return;
11. end if
12. for each signal s in netlist do begin
13.    int vt = vg ← 0;
14.    float dt = dg ← 0.0;
15.    for each (S_G, S_T, R_G, R_T) in S_VALID do begin
16.       cent ←centroid of the cluster in which s is located;
17.       if (s belongs to a cluster in S_G) then
18.          Vg++
19.          dg ← dg + |pos(s) − cent|/R_G
20.       else
21.          vt++
22.          dt ← dt + |pos(s) − cent|/R_T
23.       end if
24.    end for
25.    if (vg > vt) or (vg =vt and dg ≤ dt)
26.       Mark s as GENUINE;
27.    else
28.       Mark s as TROJAN;
29.    end if
30. end for
```

For each signal s of the netlist, by inspecting the valid clustering in S_{VALID}, we compute four variables vt, vg, dt and dg which respectively denote the vote(s) for Trojan, the vote(s) for genuine, the normalized Trojan-distance and the normalized genuine-distance. The distance is normalized by dividing the distance of s from its cluster centroid to either R_G or R_T acquired by the filtering algorithm. Particularly, if the signal s is located in a cluster in genuine section S_G, vg is incremented by one. Otherwise, the Trojan-vote counter vt increments by one. At the same time, the normalized genuine-distance dg and Trojan-distance dt corresponding to signal s is updated at each iteration.

In the case of $vt \neq vg$, s is simply marked as either Trojan or genuine signal based on majority voting. Precisely, when $vt > vg$, s is labeled as Trojan and vice versa is happened when $vt < vg$.

On the contrary, the case of equal votes ($vt = vg$) is a challenge. It can be occurred when the number of valid clusters is even, i.e., $vt = vg = 1$ or $vt = vg = 2$. This challenge is resolved by regarding the weight for each vote. As mentioned above, the weight is chosen as the normalized distance from the cluster centroid. Thus, if the sum of normalized genuine-distance is less than the sum of Trojan-distance, i.e., $dg \leq dt$, it is marked as genuine signal. Otherwise it is marked as Trojan.

3 Evaluations

In this section, the precision of our HT detection scheme is compared with other well-known schemes. First, we introduce the STMT [21], our self-designed tool for computing testability measurements of a netlist, its architecture and basic algorithms. Next, we explained the Datasets and parameters regulation. Finally, the evaluation results along with comparison with previous work are presented.

3.1 Design a Tool for Evaluating SCOAP Features

As mentioned before, most testability-based HT detection schemes used Synopsys's TetraMAX tool to calculate SCOAP values. Although TetraMAX is a well-known powerful testability tool, it has some basic limitations.

Firstly, it has a threshold value of 254 for SCOAP values calculations. Indeed, it is specialized for ATPG and generated SCOAP values with it are just indicate roughly the testability of each net. Thus, the corresponding value larger than 254 will be replaced by the symbol of asterisk (*). However, it is very likely even for the medium-size design that net's SCOAP values cross this threshold and accordingly using TetraMAX for computing will decline the accuracy of the scheme. Secondly, TetraMAX can only compute the sequential SCOAP values (SC0/SC1) while we need both sequential and combinational values.

Furthermore, there is some other tools, such as "Testability Measurement Tool [22] which is free from the above limitations but does no support Verilog format of netlist. Indeed, it supports a special non-standard format of netlist and correspondingly does not cover the Trust-HUB benchmarks.

Therefore, the above restrictions triggered us to develop our self-designed tool, namely STMT, acronym of Static Testability Measurement Tool. We implement this tool as a C++ program. The benefits of STMT are:

- There is no limitation on upper bound of SCOAP value and accordingly we can set its infinity to desired value suitable for test, leading to reduce false positive classification and give more precise results.
- The computation of both combinational and sequential SCOAP measurements are simultaneously achieved in a single traversing the netlist graph, i.e. taking the benefits of *computation overlapping*.
- Although the primary goal of STMT is the computation of SCOAP values, its modular architecture allows us to easily extending the calculation in order to support non-SCOAP testability measurements such as FANCI [5].

The details about STMT architecture along with its algorithms are explained in Appendix A.

3.2 Datasets

For evaluation, we use Trust-HUB Benchmarks [23], which have been investigated in prior work. Trust-hub netlists are based on both 90 and 180 nm standard library. There is a diverse variety in both size and functionality of the Trojan infected circuits.

3.3 Parameters regulation (selection of α)

The parameter α has a crucial role in Filtering algorithm. Thus, correct adjustment of α is important. The low value of α causes the clusters corresponds to Trojan signal are rejected and accordingly leads to false negative. Further, the high value for α results in acceptance of large cluster as Trojan which in its turn produces false positive. By inspection 50% of Trust-HUB benchmarks, we found that appropriate value for α is between 0.03 and 0.15. The lower bound 0.03 is originated from the fact that the benchmark such as RS-232 contains a Trojan circuit that is about 2.5% of original circuit. Against, the upper bound 0.15 is stem from the large circuit such as Ethernet that has a continuous distribution of testability features. Thus, clustering of this circuit usually tends to produce approximately balanced parts and accordingly generation of false positive. Thus, from the acceptable range, we empirically select 0.075 for α.

3.4 Evaluation Results

The clustering along with majority voting is implemented in MATLAB, a well-known and flexible tool. As mentioned earlier, the input of clustering is coming from the STMT in the CSV format. Further, both Algorithm 1 and Algorithm 2 are implemented as m-files in MATLAB.

The results of implementing SC-COTD on Trust-HUB benchmarks are shown in Tables 3 and 5. The former shows the filtering results while the latter illustrates the majority voting procedure.

The first column of Table 3 indicates the target benchmark. The four preceding columns respectively show the maximum testability value of CC, CO, SC and SO computed by the STMT. We can see that the maximum value for all the benchmarks except the Ethernet ones are less that 1000. For Ethernet group, the maximum testability measures are close to10000.

During the adjustment and configuration of the *k*-means algorithm, we set INF to the maximum values obtained from STMT, indicating the initial centroid of the intended clusters. Furthermore, Euclidean distance is chosen for measurement metric. The detail of clustering is described in the columns entitled by Γ_{C2}, Γ_{S2}, Γ_{C3} and Γ_{S3}, respectively. For instance, Γ_{C2} which denotes clustering on $<$CC,CO$>$ with $k=2$ has two columns that shows the size of its clusters, $\boldsymbol{\sigma_{(0,0)}}$ and $\boldsymbol{\sigma_{(I,I)}}$. The invalid clusters are identified in red color. We can see that four Γ_{C2}, one Γ_{S2}, one Γ_{C3} and one Γ_{S3} are identified as invalid clustering and accordingly excluded from majority voting procedure. The last group of columns indicates the set of Trojan signals for each clustering. We see that for Γ_{C2} and Γ_{S2} the set of Trojans is always equal to $\sigma_{(I,I)}$. Further, for Γ_{S3}, the set of Trojans is union of the clusters $\sigma_{(0,I)}$ and $\sigma_{(I,0)}$. However, for Γ_{C3}, the set of Trojans is equal to $\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$ for six benchmarks and $\sigma_{(I,0)}$ for other non-filtered benchmarks. More specifically, in the benchmarks RS232-T1200, RS232-T1300, RS232-T1600 and Ethernet-T720, the cluster $\sigma_{(0,I)}$ is larger than α-percentage of total signals and accordingly is considered as genuine signals.

The detail of majority voting for identification of Trojan signals is shown in Table 4. For each benchmark, the total number of signals along with the real Trojan signals are illustrated in the first group of columns. The next column shows the number of valid clusterings that identifies total votes for each benchmark.

In the seven leading columns, distribution of voting results for the signals labeled as Trojan are shown respectively. Each column corresponds to a acceptable-value of pair (*vt*, *vg*). For example, the column "4:0" indicates the number of signals confirmed to be as Trojan by all of the clustering models or equally by 4-votes for Trojan and 0-vote for genuine. The columns "2:2" and "1:1" show the case of equal votes for Trojan/genuine where in the decision of Trojan-marking is made by the normalized distance.

Similarly, the five preceding columns show the distribution of voting results for the signals labeled as genuine. Further, the columns "2:2" show the case $vt=vg=2$ where in the decision of genuine-marking is made by the normalized distance.

Table 3 Results of SC-COTD clustering and filtering on Trust-Hub benchmarks (Red cells identifies filtered clusterings)

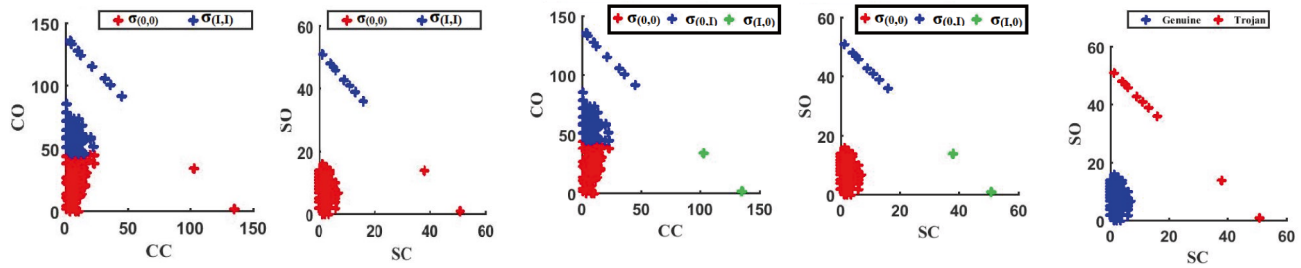
Benchmarks	Max Value				Γ_{C2}		Γ_{S2}		Γ_{C3}			Γ_{S3}			S_T (Set of Trojan signals)			
	CC	**CO**	**SC**	**SO**	$\lvert\sigma_{(0,0)}\rvert$	$\lvert\sigma_{(I,I)}\rvert$	$\lvert\sigma_{(0,0)}\rvert$	$\lvert\sigma_{(I,I)}\rvert$	$\lvert\sigma_{(0,0)}\rvert$	$\lvert\sigma_{(0,I)}\rvert$	$\lvert\sigma_{(I,0)}\rvert$	$\lvert\sigma_{(0,0)}\rvert$	$\lvert\sigma_{(0,I)}\rvert$	$\lvert\sigma_{(I,0)}\rvert$	$\boldsymbol{\Gamma_{C2}}$	$\boldsymbol{\Gamma_{S2}}$	$\boldsymbol{\Gamma_{C3}}$	$\boldsymbol{\Gamma_{S3}}$
RS232-T1000	223	233	72	74	247	9	246	10	244	9	3	244	9	3	$\sigma_{(I,I)}$	$\sigma_{(I,I)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
RS232-T1100	223	233	72	74	247	9	246	10	244	9	3	244	9	3	$\sigma_{(I,I)}$	$\sigma_{(I,I)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
RS232-T1200	135	136	51	51	171	87	247	11	164	92	2	245	11	2	x	$\sigma_{(I,I)}$	$\sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
RS232-T1300	130	121	48	43	249	4	244	9	146	103	4	244	5	4	$\sigma_{(I,I)}$	$\sigma_{(I,I)}$	$\sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
RS232-T1400	260	257	84	83	246	11	246	11	244	10	3	244	11	2	$\sigma_{(I,I)}$	$\sigma_{(I,I)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
RS232-T1500	227	222	72	71	248	10	248	10	245	10	3	245	10	3	$\sigma_{(I,I)}$	$\sigma_{(I,I)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
RS232-T1600	123	122	44	43	153	103	244	12	154	98	4	244	8	4	x	$\sigma_{(I,I)}$	$\sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
S15850-t100	958	849	764	576	2404	40	2420	24	2404	35	5	2418	23	3	$\sigma_{(I,I)}$	$\sigma_{(I,I)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
S35932-t200	295	310	18	23	6404	11	2318	4099	6405	10	2	3692	672	2053	$\sigma_{(I,I)}$	x	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$	x
S38417-t100	996	1000	466	381	4984	827	5800	11	4646	709	456	5799	9	3	x	$\sigma_{(I,I)}$	x	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
S38417-t200	987	1000	401	415	4973	849	5803	19	4700	735	387	5802	18	2	x	$\sigma_{(I,I)}$	$\sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
S38584-t100	611	996	255	344	7342	8	7345	5	6044	8	1298	7345	4	1	$\sigma_{(I,I)}$	$\sigma_{(I,I)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
Ethernet-T700	9068	5874	9065	2342	102,742	15	102,742	15	102,742	12	3	102,742	11	4	$\sigma_{(I,I)}$	$\sigma_{(I,I)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
Ethernet-T710	9061	5874	8436	8580	102,742	15	102,594	163	102,742	12	3	102,408	188	161	$\sigma_{(I,I)}$	$\sigma_{(I,I)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$
Ethernet-T720	9061	7860	9066	2342	102,742	15	102,742	15	91,968	10,774	15	102,742	11	4	$\sigma_{(I,I)}$	$\sigma_{(I,I)}$	$\sigma_{(I,0)}$	$\sigma_{(0,I)} \bigcup \sigma_{(I,0)}$

Table 4 SC-COTD majority voting and final results on trust-hub benchmarks

Benchmarks	Total signals	Trojan signals	Total votes	(*vt:vg*) *for marking as Trojan*							(*vt: vg*) *for marking as Genuine*						Detected Trojans	FP	FN	Precision	Recall
				4:00	**3:00**	**2:00**	**3:01**	**2:01**	**2:02**	**1:01**	**2:02**	**1:03**	**1:2**	**0:02**	**0:03**	**0:04**					
RS232-T1000	256	10	4	9			1		2							244	12	2	0	0.83	1
RS232-T1100	256	10	4	9			1		2							244	12	2	0	0.83	1
RS232-T1200	258	11	3					13							245		13	2	0	0.85	1
RS232-T1300	253	9	4	4					5							244	9	0	0	1	1
RS232-T1400	257	11	4	11					2							244	13	2	0	0.85	1
RS232-T1500	258	10	4	10					3							245	13	3	0	0.77	1
RS232-T1600	256	12	3		4			8							244		12	0	0	1	1
S15850-t100	2444	26	4	24			2		14							2404	40	14	0	0.65	1
S35932-t200	6415	11	2			11				1				6404			12	1	0	0.92	1
S38417-t100	5811	11	2			11				1				5799			12	1	0	0.92	1
S38417-t200	5822	11	3				20						386		5416		20	9	0	0.55	1
S38584-t100	7350	8	4	5					3							7342	8	0	0	1	1
Ethernet-T700	102,757	15	4	15												102,742	15	0	0	1	1
Ethernet-T710	102,757	15	4	6			9				157	177				102,408	15	0	0	1	1
Ethernet-T720	102,757	15	4	15												102,742	15	0	0	1	1
Average				**44%**	**2%**	**13%**	**13%**	**12%**	**14%**	**1%**			**0.40%**	**13.30%**	**19.50%**	**66.40%**				**88%**	**100%**

Table 5 Summary of comparison between SC-COTD and other methods

Benchmarks	Trojan type	Trojan features	COTD*		*SC-COTD*		COTD [8]		[14]		[9]
		Insertion/physical characteristic/Effect/activation	*FN*	*FP*	FN	*FP*	FN	*FP*	*FN*	*FP*	*Detected*
RS232-T1000	**combinational**	design/functional/change-functionality/physical-condition-based	0	2	0	2	0	0	0	3	/
RS232-T1100	**sequential**	design/functional/change-functionality/physical-condition-based	0	2	0	2	0	0	18	5	/
RS232-T1200	**sequential**	design/functional/change-functionality/physical-condition-based	9	0	0	2	0	0	4	0	/
RS232-T1300	**combinational**	design/functional/change-functionality/physical-condition-based	5	0	0	0	0	0	0	0	//
RS232-T1400	**sequential**	design/functional/change-functionality/physical-condition-based	0	2	0	2	0	0	1	0	/
RS232-T1500	**sequential**	design/functional/change-functionality/physical-condition-based	0	3	0	3	0	0	2	1	/
RS232-T1600	**sequential**	design/functional/change-functionality/physical-condition-based	8	0	0	0	0	0	2	3	/
S15850-T100	**sequential**	design/functional/denial-of-service, change-functionality/time-based	0	14	0	14	0	0	6	1	/
S35932-T200	**combinational**	design/functional/ leak-information, change-functionality/time-based	0	1	0	1	-	-	7	1	/
S38417-T100	**combinational**	design/functional/denial-of-service, change-functionality/time-based	10	455	0	1	0	0	8	0	/
S38417-T200	**combinational**	design/functional/denial-of-service, change-functionality/time-based	10	386	0	9	0	0	8	0	/
S38584-T100	**combinational**	design/functional/denial-of-service, change-functionality/time-based	0	0	0	0	-	-	18	2	/
Ethernet-T700	**sequential**	design/functional/change-functionality/physical-condition-based	0	0	0	0	0	0	-	-	/
Ethernet-T710	**sequential**	design/functional/change-functionality/physical-condition-based	0	0	0	0	0	0	-	-	/



a. Clustering graphs for ($k = 2$) **b.** Clustering graphs for ($k = 3$) **c.** Final decision

Fig. 2 Evaluations for RS232-T1200

In the last four columns, the number of Trojan detected signals, False Positive (FP), False Negative (FN), Precision and Recall are shown respectively. Note that Recall and Precision are defined as [24]:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{4}$$

where TP identifies the True Positive, i.e., the number of Trojan signals correctly detected as Trojan.

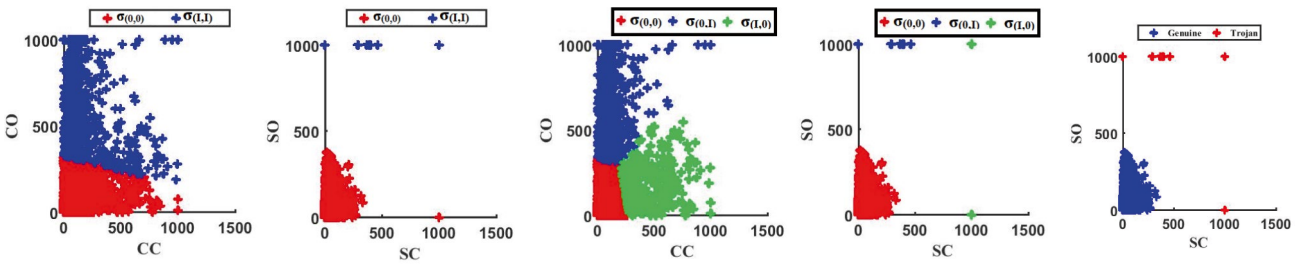
The results show that the filtering has crucial role in the final decision. Among the 15 experiments, four of them have two invalid clusterings and two of them have a single invalid clustering. This implies that a single clustering is incapable of producing the valid clusters to detect Trojan signals. In contrast, the collaborative approach by deploying different clustering models allows us to filter likely invalid cluster(s) and make the correct decision by majority voting procedure. By exploring voting-result distribution, we can see that the higher rates of Trojan detection are respectively corresponding to "4:0", "2:2","2:0", "3:1" and "2:1" decisions with 44%, 14%, 13%, 13% and 12%. Further, for genuine detection, the higher rates are obtained for "0:4", "0:2" and "0:3" decisions with 66%, 20% and 13%.

In spite of invalid clustering models that most of them are taken place for Γ_{C2} and Γ_{C3}, the rest of the valid models are successfully worked together in decision-making phase and accordingly SC-COTD reaches to Recall = 100% for Trust-Hub benchmarks. However, there are few numbers of FP which are appearing in the final decision. As shown in Table 4, the average precision of our scheme is equal to 88%. We can see a large set of true negative for each benchmark, i.e., correctly genuine detection rate. This makes the other indices to be hold in good ranges. More specifically, FPR (False Positive Rate), Accuracy and CSI (Critical Success Index) [24] are respectively equal to 0.3%, 99.7% and 2%.

3.5 Comparison with Prior Work

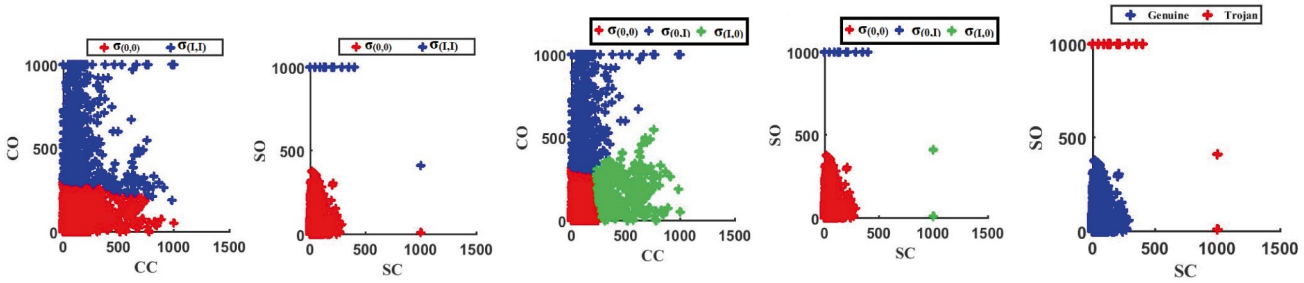
Table 5, summarizes the comparison between the proposed method and other methods such as *COTD* [14], *and* [9]. It should be noted that *COTD** (column 1) is our implementation of original *COTD* (column 3). As shown in the Table 5, *COTD** and *SC-COTD* have similar performance except for five benchmarks in which *SC-COTD* outperforms *COTD* with higher accuracy and significantly smaller number of false alarms.

First of all, we focus on the RS232-T1200 circuit. We can see that SC-*COTD* finds all HT signals while *COTD** has 9 false negatives which is relatively high in comparison to



a. Clustering graphs for ($k = 2$) **b.** Clustering graphs for ($k = 3$) **c.** Final decision

Fig. 3 Evaluations for S38417-T100



a. Clustering graphs for ($k = 2$) **b.** Clustering graphs for ($k = 3$) **c.** Final decision

Fig. 4 Evaluations for S38417-T200

all 11 Trojan signals. Figure 2, shows the details of clustering graphs. We find that among the four possible clustering models, the ones correspond to $k=2$ on $<$CC, CO$>$ is invalid. This is due to fact that this clustering generates two clusters greater than the threshold (0.1). For *SC-COTD*, this cluster is filtered and the final decision is made by doing the majority-voting among the three remained clustering models. It is worth noting that the clustering corresponds to $k=3$ on $<$CC, CO$>$ also generates two clusters greater than threshold (identified by blue and red) and one clusters (green) below threshold. Thus, according to Algorithm 1 the first and second clusters (red and blue) are aggregated to form the genuine clusters while third cluster (denoted by green) is set as Trojan signals. As indicated in Fig. 2(c), *SC-COTD* produces the correct result, detecting all the 11 Trojan signals from the genuine ones. All of the Trojan decisions are made by a result of 2 votes for Trojan and 1 vote for genuine. However, the detection algorithm leaves two false alarms, identified two genuine signals as Trojan.

At the other hand, *COTD* also claims that successfully detects the Trojan signals for RS232-T1200 benchmark. This is while that, the authors of *COTD* throughout the paper asserted they employed the clustering on $<$CC, CO$>$ with $k=3$ which we found that it has some false negative in HT detection. Actually COTD*, our implementation of *COTD*, shows slightly different results.

It is basically originated from the testability measures which in COTD are evaluated by Synopsys TetraMAX while in COTD* the evaluation is made by our developed tool, STMT.

The first difference is the limitation of TetraMAX for setting the infinity value which is bounded to 254 [8]. Our tool has not such limitation. Further, we have some challenges with Synopsys TetraMAX to extract SCOAP measures. For example, the combinational metrics reported by TetraMAX, i.e., CC0, CC1 and CO, are always zero. Moreover, the reported sequential metrics (SC0, SC1 and SO), which are the only non-zero measures, are different than the accepting results in the literature even for small-scale benchmarks. Finally, since the exact computation of SCOAP metrics is a NP-Complete problem [25] it is usual for different tools to report different metrics for a specific large digital circuit with complex feedback.

In Appendix B, we bring some details of challenge of SCOAP computations with Synopsys TetraMAX even for small-scale circuit. Getting back to our discussion, [14] could not find HT signals in the first three benchmarks. Furthermore, its FP and FN alarms in the other benchmarks are relatively high and accordingly is ineligible. The work in [9] only reported detection of Trojan-infected circuits. Actually, it does not localize the detection, namely to identify Trojan signals, Thus, a precise comparison with this method is impossible.

Now, we focus on two other specific benchmarks which *COTD* fails to detect Trojan signals. The first is S38417-T100 and the second is S38417-T200. The details of HT detection for S38417-T100 are shown in Fig. 3(a) and Fig. 3(b). We can see that clustering on $<$CC, CO$>$ for both $k=2$ and $k=3$ becomes invalid since it generates two clusters larger than threshold. In contrast, clustering on $<$SC, SO$>$ for both $k=2$ and $k=3$ leads to valid clustering that detects Trojan signals without any false positive nor false negative. The final decision for this circuit is shown in Fig. 3(c). It is worth noting that the authors of *COTD* admitted the restriction of their algorithm for this circuit in [8].

The second example that *SC-COTD* outperforms *COTD* is S38417-T200. The details of the detection process are shown in Fig. 4(a) and (b). Similar to S38417-T100, we can see that the clustering on $<$CC, CO$>$ with $k=2$ produces invalid clusters and thus it is filtered. In contrast to S38417-T100, the clustering on $<$CC, CO$>$ for $k=3$ generates valid clusters, $\sigma(I,0)$ for Trojan signals and $\sigma(0,0)\bigcup\sigma(0,I)$ for genuine signals. But it is not useful sine it contains a lot of false positive in $\sigma_{(I,0)}$. Actually, in spite the validation of clustering on $<$CC, CO$>$ with $k=3$, it produces a large number of false positive for Trojan signals. It is the way that *COTD* follows and thus reports a lot of false alarms (386 FP). But, as we seen, *SC-COTD* has a brilliant performance employing the

clustering on < SC,SO > for both $k=2$ and $k=3$. For these suspicious signals, the false alarms of the < CC, CO > with $k=3$ are eliminated by the correct detections of clustering < SC,SO > for both $k=2$ and $k=3$. As we seen, the final decision of Trojan signals is made the result of 2 votes for Trojan and 1-vote for genuine. Further, the incorrect detection of Trojan signals by < CC, CO > with $k=3$, is cleared by two genuine-votes given by < SC,SO > for both $k=2$ and $k=3$.

Furthermore, our action in the case of equal votes, leads to correct detection of Trojan signals in some benchmarks such as S38584-t100 and RS232-T1300. In the former, three of Trojan signals have 2-vote for Trojan along with 2-vote for genuine which the decision is made by examining the normalized distance from the cluster-centroids. In the latter circuit, among the nine Trojan detected signals, five of them face with equal-vote state which finally detected as Trojan by checking the normalized distances. Although, this method leads to correct detection in above mentioned benchmarks, it causes some false positives in benchmarks such as S15850-t100.

3.6 Types of Trojans

SC-COTD successfully identifies both combinational and sequential hardware Trojans. It is basically originated from the fact that the attacker tries to insert Trojan in signals with high controllability and observability features to realize its stealthy. Further, if Trojan circuit uses sequential components, the controllability and observability of signals in their fan-out will be increased. Since inserting flip-flops into a Trojan circuitry would considerably increase the controllability and observability, *SC-COTD* can identify Trojan signals easily.

Finally, evaluation result also approves our claim. As show in Table 5, the evaluated benchmarks consist of both combinational and sequential Trojan triggering. For example, RS232-T1000 has combinational Trojan trigger while RS232-T1200 has sequential one. We can see that the proposed method is able to find hardware Trojans for both sequential and combinational type.

4 Conclusion

In this paper, we proposed a static method for detecting hardware Trojans based on controllability and observability features named *SC-COTD*. Further, we introduced a new tool for calculation of testability measures. Existing tools face limitations such as infinite amount and format of the input file. Proposed tool receives the netlist at the gate-level as input and calculates the controllability and observability values of all signals in a short time. By feeding testability measurement, *SC-COTD* uses both sequential and combinational controllability and observability for HTs detection. We applied an ensemble classifier based on *k*-means clustering with specified initial parameters over testability measures. Results show that *SC-COTD* can find Trojan inserted netlist without any need for HT free chip. Furthermore, proposed method can localize HTs which are not detected by previous scheme like *COTD*.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10836-021-05960-2.

Data Availability The datasets analysed during the current study are available in the TRUST-HUB repository, available at https://trust-hub.org/.

Declarations

Research Involving Human and Animal Participants This article does not contain any studies with human participants or animals performed by any of the authors.

Conflict of Interest Mahshid Tebyanian declares that she has no conflict of interest. Azadeh Mokhtarpour declares that she has no conflict of interests. Further, Alireza Shafieinejad declares that he/she has no conflict of interest.

References

1. Rostami M, Koushanfar F, Karri R (2014) A primer on hardware security: Models, methods, and metrics. IEEE 102(8):1283–1295
2. Bhunia S, Hsiao M, Banga M, Narasimhan S (2014) Hardware Trojan attacks: threat analysis and countermeasures. IEEE 102(8):1229–1247
3. Wang, C, Li J, Yu M, Wang J (2013) An intelligent classification method for Trojan detection based on side-channel analysis. in IEICE Electron Express
4. Narasimhan S, Wang X, Du D, Chakraborty R, Bhunia S (2011) TeSR: A robust temporal self-referencing approach for hardware Trojan detection. in IEEE International Symposium on Hardware-Oriented Security and Trust
5. Waksman A, Suozzo M, Sethumadhavan S (2013) FANCI: identification of stealthy malicious logic using boolean functional analysis. in ACM SIGSAC conference on Computer & communications security
6. Zhang J, Yuan F, Wei L (2015) VeriTrust: Verification for hardware trust. IEEE Trans Comput Aided Des Integr Circuits Syst 34(7):1148–1161
7. Oya M, Shi Y, Yamashita N, Okamura T, Tsunoo Y, Goto S, Yanagisawa M, Togawa N (2015) A hardware-Trojans identifying method based on Trojan net scoring at gate-level netlists. IEICE Transactions on Fundamentals of Electronics, Commucations and computer sciences 18(12):2537–2546
8. Salmani H (2017) COTD: Reference-free hardware trojan detection and recovery based on controllability and observability in gate-level netlist. IEEE Trans Inf Forensics Secur 12(2):338–350
9. Xie X, Sun Y, Chen H, Ding Y (2017) Hardware Trojans classification based on controllability and observability in gate-level netlist. IEICE Electronics Express. vol. 14, no. 18
10. Bushnell ML, Agrawal VD (2002) Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits. Springer Series on Frontiers in Electronic Testing, vol. 17

11. Goldstein LH, Thigpen E (1980) SCOAP: Sandia controllability/observability analysis program. in 17th Design Automation Conference
12. Dharmadhikari P, Raju A, Vemuri R (2018) Detection of Sequential Trojans in Embedded System Designs Without Scan Chains. in IEEE Computer Society Annual Symposium on VLSI (ISVLSI)
13. Kok C, Ooi C, Moghbel M, Ismail N, Choo H, Inoue M (2019) Classification of Trojan Nets Based on SCOAP Values using Supervised Learning. in IEEE International Symposium on Circuits and Systems (ISCAS)
14. Hasegawa K, Yanagisawa M, Togawa N (2017) Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classified. in IEEE International Symposium on Circuits and Systems (ISCAS)
15. Bian R, Xue M, Wang J (2018) A novel golden models-free hardware Trojan detection technique using unsupervised clustering analysis. in International Conference on Cloud Computing and Security
16. Wang Y, Han T, Han X, Liu P (2019) Ensemble-Learning-Based Hardware Trojans Detection Method by Detecting the Trigger Nets. in IEEE International Symposium on Circuits and Systems (ISCAS)
17. Liu Q, Zhao P, Chen F (2019) A Hardware Trojan Detection Method Based on Structural Features of Trojan and Host Circuits. IEEE Access 7:44632–44644
18. Hasegawa K, Yanagisawa M, Togawa N (2017) Hardware Trojans classification for gate-level netlists using multi-layer neural networks. in IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)
19. Cruz J, Farahmandi F, Ahmed A Mishra P (2018) Hardware Trojan detection using ATPG and model checking. in 31st International Conference on VLSI Design and 17th International Conference on Embedded Systems (VLSID)
20. Farahmandi F, Huang Y, Mishra P (2017) Trojan localization using symbolic algebra. in 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)
21. Tebianian M, Mokhtarpour A, Shafieinejad A STMT : SCOAP Testability Measurement Tool. [Online]. Available: https://github.com/azadehmokhtarpour/SCOAP-values-computation-STMT.
22. Samimi SMS Testability Measurement Tool. Available: https://sourceforge.net/projects/testabilitymeasurementtool/.
23. Trust-Hub Benchmarks. [Online]. Available: https://trust-hub.org/.
24. Olson DL, Delen D (2008) Advanced Data Mining Techniques, 1st edition ed., Springer, p. 138
25. Agrawal VD (2006) Auburn University, ELEC7250-001 VLSI Testing, Lecture 8: Testability Measures [Online]. Available: https://www.eng.auburn.edu/~vagrawal/COURSE/FULL/lec8a.ppt. [Accessed 16 12 2019]

Mahshid Tebyanian was born in Tehran, Iran in 1988. She received the B.S. degree in computer engineering from Sharif University of Technology, Tehran, in 2016. She was also received M.Sc degree in computer engineering from Tarbiat Modares University, Tehran, Iran, in 2019. Now, she is a researcher in SE-Lab (Security Evaluation Laboratory) at Tarbiat Modares University. Her research area includes Hardware Design and Test, Network Security and Penetration Test.

Azadeh Mokhtarpour was born in Zanjan, Iran in 1990. She received the B.S. degree in Information Technology engineering from Tabriz University, in 2013. She was also received M.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2015. Since 2019, she is a Ph.D. student and researcher in SE-Lab (Security Evaluation Laboratory) at Tarbiat Modares University. Her research area includes Hardware Security, Learning Methods in Hardware and Fault tolerant systems. Her current interests include Hardware Trojan Detection and Model Checking.

Alireza Shafieinejad was born in Isfahan, Iran in 1977. He received the B.S. degree in computer engineering from the Sharif University of Technology, Tehran, in 1999. He was also received M.Sc and Ph.D degrees in electrical engineering from Isfahan University of Technology, Isfahan, Iran, respectively in 2002 and 2013. Since 2015, he has been an Assistant Professor with the Electrical and Computer Engineering Department, Tarbiat Modares University, Tehran, Iran. His research interests are in the area of Hardware Security and Trust, Network Security and 5G/6G Mobile Networks. At Tarbiat Modares University, he leads research in network security and penetration test in SE-Lab (Security Evaluation Laboratory).