Abstract—This paper presents a novel hardware Trojan detection technique in gate-level netlist based on the controllability and observability analyses. Using an unsupervised clustering analysis, the paper shows that the controllability and observability characteristics of Trojan gates present significant inter-cluster distance from those of genuine gates in a Trojan-inserted circuit, such that Trojan gates are easily distinguishable. The proposed technique does not require any golden model and can be easily integrated into the current integrated circuit design flow. Furthermore, it performs a static analysis and does not require any test pattern application for Trojan activation either partially or fully. In addition, the timing complexity of the proposed technique is an order of the number of signals in a circuit. Moreover, the proposed technique makes it possible to fully restore an inserted Trojan and to isolate its trigger and payload circuits. The technique has been applied on various types of Trojans, and all Trojans are successfully detected with 0 false positive and negative rates in less than 14 s in the worst case.

Index Terms—Hardware security, hardware Trojans, controllability, observability, unsupervised clustering.

I. INTRODUCTION

MODERN designs like system-on-chips and network-on-chips are equipped with a variety of resources such as memory blocks in different sizes and types, interfaces with different characteristics, and several processors and co-processors to realize high performance computing systems, often with tight design restrictions in terms of power and area. The high complexity of modern designs, the constraint of time-to-market window, and the cost restriction of final product highly drive the horizontal design process. The third-party intellectual properties (3PIPs) are widely used in the forms of soft, firm and hard IPs to expedite the design development process. However, using external IPs exposes a design to 3PIPs intentionally modified by inserting a hardware Trojan (HT) to tamper with the design and causes its malfunction under very rare circumstances [1], [2].

Hardware Trojans have negligible effects on circuit parameters and rarely become fully activated. While considerable amount of work has been presented on hardware Trojan detection, they can be broadly categorized into two major groups: the side-channel signal analysis and the logic value analysis. Majority of work on Trojan detection based on side-channel analysis has focused on power and delay side-channel signals [3]–[13]. Some of main challenges with side-channel Trojan detection techniques are environmental and process variations that can mask Trojan contributions into circuit power consumption and delay characteristics. The need for a set of golden circuits as a reference is another important issue. To improve the effectiveness of side-channel-based detection technique, some work has recommended design-for-hardware-trust [14], [15].

Besides side-channel based detection techniques mainly applied after design fabrication, some work have investigated hardware Trojans in early design stages [16]–[22]. High-level syntheses for security have been proposed to render a hardware Trojan ineffective [23]–[26], and several logic testing techniques have been introduced to detect hardware Trojans [27]–[29]. Some others have studied the switching activity of circuit signals and their correlation to identify a hardware Trojan in gate-level netlist [30]–[34]. Some of the main issues with these techniques are none-zero false positive and false negative rates, significant processing time and their scalability limitations. Furthermore, it has been shown they can be defeated, and a carefully designed Trojan can remain undetected [34], [35]. In addition, neither of the techniques are able to detect always-on Trojans such as a ring-oscillator Trojan that is not connected to the main circuit, and presents normal switching activities, but aims to reduce circuit reliability.

While majority of existing techniques mainly generate list of suspicious signals, it remains the responsibility of a design house to still scrutinize the netlist. To be used in practice, a HT detection technique should be easily integrable into commercial circuit design flows. Furthermore, they should present 0 false positive and false negative HT detection rates. In addition, authentication time should be small to meet time-to-market constraints.

This paper introduces the Controllability and Observability for hardware Trojan Detection (COTD) technique for hardware Trojan detection in gate-level netlist. Controllability reflects the difficulty of setting a signal line to a required logic value, and observability reflects the difficulty of propagating the logic value of the signal line to observation points. The COTD
Simulation results are presented in Section V and Section VI and motivation. Section IV introduces the COTD technique. Section II, and Section III discusses the background false negative rates. The contributions of the paper are:

1) COTD does not need any golden circuit as a reference,
2) COTD does not need any test pattern application for Trojan partial/full activation,
3) COTD presents 0 false positive and false negative Trojan detection rates,
4) COTD can be readily integrated into current commercial integrated design flows,
5) COTD presents time complexity of linear function of the number of signals in the circuit, and
6) COTD makes it possible to fully isolate an inserted Trojan.

COTD has been applied to all gate-level Trojans available on Trust-HUB [36] and introduced by DeTrust [34], always-on Trojans, and HiTCh [35], and COTD has successfully detected all of them in a fraction of a minute with 0 false positive and false negative rates.

The paper is organized as follows. Previous work is studied in Section II, and Section III discusses the background and motivation. Section IV introduces the COTD technique. Simulation results are presented in Section V and Section VI provides detailed discussions. Section VII concludes the paper.

II. PREVIOUS WORK

The unused circuitry identification (UCI) technique [37] is one of the first such techniques which distinguishes minimally used logic from the other parts of the circuit. First, UCI creates a data-flow graph for a circuit. Nodes of graph are signals (wires) and state elements and its edges indicate data flow between the nodes. Based on this data-flow graph, UCI generates a list of all direct and indirect signal pairs where data flows from a source signal to a sink signal. In the following, UCI simulates the HDL code using design verification tests to find the set of data-flow pairs where intermediate logic does not affect the data that flows between the source and sink signals. While UCI centers on the fact that the HT circuitry mostly remains inactive within a design, and hence such minimally used logic can be distinguished from the other parts of the circuit, some later works such as [32] showed how to design HTs which can defeat the UCI detection scheme.

VeriTrust [32] flags suspicious circuitries by identifying potential trigger inputs used in HTs, based on the observation that these inputs keep dormant under non-trigger condition and hence are redundant to the normal logic function of the circuit. In order to detect the redundant inputs, it first performs functional testing and records the activation history of the inputs in the form of sums-of-products (SOP) and product-of-sums (POS). Then it further analyzes these inactivated SOPs and POSs to find the redundant inputs. However, because of the functional verification constraints, VeriTrust can see several inactivated SOPs and POSs and thus regard the circuit to be potentially infected resulting in false positives.

FANCI [33] applies Boolean function analysis to flag suspicious wires in a design which have weak input-to-output dependency. For each input in the combinational logic cone of an output wire, a control value (CV), which represents the percentage impact of changing an input on the output, is computed. If the mean of all the CVs is lower than a threshold, then the resulting output wire is considered malicious. This is a probabilistic method where the threshold is computed with some heuristic to achieve a balance between security and the false positive rate. A very low threshold may result in a high false positive rate by considering most of the wires (even non-malicious ones) as malicious, whereas a high threshold may actually result in false negatives by considering a HT related (malicious) wire to be not malicious.

Both VeriTrust and FANCI only monitor the combinational logic between two registers (i.e. one sequential stage). Knowing this limitation, DeTrust [34] designs new HTs whose circuitries are distributed over several sequential stages such that FANCI/VeriTrust, while observing a single sequential stage, would consider them non-malicious.

In [30], an information-theoretic approach for Trojan detection has been proposed. It basically estimates the statistical correlation between signals in a circuit for Trojan detection with the use of OPTICS clustering algorithm. To study the correlation between the signals, inputs patterns are applied and a weighted graph of design created. While the technique presents full coverage for selected benchmarks, the accuracy of technique highly depends on observing enough activity on each signal to study signals correlation and presented results indicate non-zero false positive rate. Furthermore, the application of the technique for large circuits may require considerable processing time and memory usage.

In another effort, a score-based classification method is presented for identifying hardware Trojans. The proposed technique extracts Trojan characteristics introduced at Trust-HUB [36] and defines an incremental metric to isolate some of Trojan nets from the rest of circuit. While the proposed technique does not require a golden circuit, it is constructed towards the limited Trust-HUB Trojans, and it takes considerable processing time even for such small circuits.

III. BACKGROUND AND MOTIVATION

While the horizontal integrated circuit design flow is widely practiced, a circuit is susceptible to hardware Trojan insertion across circuit design flow. A hardware Trojan can be inserted in a gate level netlist provided by a malicious 3PIP vendor. Detecting such a hardware Trojan is a very challenging task for two main reasons: (1) the lack of a reference/golden circuit and (2) no pre-information about the hardware Trojan inserted into a given netlist [31].

Hardware Trojans can be typically divided into two groups from the activation mechanism perspective: always-on and triggered. Always-on Trojans are activated as soon as their hosting designs are powered-on while conditional Trojans seek specific triggers to launch. In general, a functional Trojan consists of two parts: a Trojan trigger and a Trojan payload. The Trojan trigger determines conditions under which the Trojan payload propagates erroneous values into the main circuit. While both always-on and triggered Trojans contain the
Trojan payload, the triggered type also has the Trojan trigger. The Trojan trigger might be connected to some internal nets, and the Trojan payload re-stitches some other nets [39].

To reduce Trojan detectability, the Trojan trigger might be connected to nets with low activity to create a rare triggering vector, and the Trojan payload might re-stitch nets whose deviations are not readily observable. As a result, it is expected that Trojan signals present a very low switching activity and small correlation with other signals in a circuit as considered in [30], [31], and [33]. However, the circuit may contain some genuine signals with low switching activity, and they can be wrongly identified as Trojan signals (i.e. false positive). On the other hand, a smart implementation of Trojan can realize a low-active Trojan trigger whose Trojan trigger does not match any cases studies in [31]. For example, Figure 1 shows a very low active signal, the signal \( T \), with its fan-in cone in s38414 benchmark [38], and Figure 2 shows the switching on the signal \( T \) after applying random test patterns for 725041 test cycles. While the signal \( T \) presents no switching activity, this question remains whether the signal \( T \) is a Trojan trigger signal whose fan-in cone is different from cases studied in [31] to evade Trojan detection technique. Or, it is not a Trojan signal due to its correlation with its driving signal, the signal \( In2 \), as studied in [30].

It is also possible that a Trojan has no trigger as it is an always-on Trojan, and its payload is not connected to any net in the circuit. For example, an inverter-based ring-oscillator Trojan only consists of an odd number of inverters. This Trojan is not connected to any internal net and is to reduce design reliability over the time. Such a Trojan can easily evade the Trojan detection techniques presented in [30] and [31] (i.e. false negative).

Testability is a relative measure of the effort or cost of testing a logic circuit, and it can be used to identify nets with poor testability [40]. Testability considers the circuit implementation and gates’ interconnections. While a signal such as the signal \( T \) in Figure 1 may have a very low switching activity, the signal may have high testability. This fact emphasizes that applying random patterns and merely analyzing switching activity of signals do not necessarily determine whether the signal is a Trojan signal or not and can result in false negative and false positive conclusions.

Low-testability signals more infrequently determine circuit primary outputs and its states; therefore, their manipulation highly remains hidden. Testability analysis can be performed by calculating the controllability and observability of each signal. The Sandia Controllability/Observability Analysis Program (SCOAP) is the most popular testability program that measures testability of each signal in a circuit logic based on several numerical values including: \( CC0(s) \) - combinational 0-controllability of \( s \), \( CC1(s) \) - combinational 1-controllability of \( s \), and \( CO(s) \) - combinational observability of \( s \). These combinational testability measures roughly determine the number of signals that must be manipulated in order to control or observe \( s \) from primary inputs or at primary outputs. The values of controllability measures range between 1 and \( \infty \), while the values of observability measures range between 0 and \( \infty \). As a boundary condition, the \( CC0 \) and \( CC1 \) values of a primary input are set to 1, and the \( CO \) value of a primary output is set to 0 [40].

The SCOAP method first calculates the controllability values for all signals from primary inputs towards primary outputs. To do so, the circuit is levelized and each gate is assigned a level order. The output controllability of each gate is calculated in level order when the controllability of all its inputs have been calculated. For example, \( CC0(s) = \min \{\text{input } CC0\} + 1 \) and \( CC1(s) = \sum \{\text{input } CC1\} + 1 \) for the output of a AND gate. The rules for other types of gates are presented in [40]. After calculating controllability measures for all signals, the observability measures for all signals from primary outputs towards primary inputs are calculated. For example, the observability of one input of a AND gate with multiple inputs is \( CO(s) = \sum \{\text{output observability, } CC1 \text{ of other inputs}\} + 1 \). The rules for other types of gates are similarly presented in [40].

From the security perspective, signals with low controllability or low observability are more susceptible to be used for the Trojan trigger and the Trojan payload. While a target signal might be highly observable but hardly controllable or vice versa, the pair \( < CC, CO > \) is created for each signal in the circuit, and the magnitude of pair is defined as

\[
| < CC, CO > | = \sqrt{CC^2(s) + CO^2(s)}.
\]

While the value of \( CO \) can be directly obtained from the SCOAP program, the \( CC \) value is defined as

\[
CC(s) = \sqrt{CC0^2(s) + CC1^2(s)}.
\]

It is expected Trojan signals to have low testability to avoid frequent impact on design functionality. To study the relationship between a signal testability and its manipulation effect, the testability analysis is performed on s38417 benchmark [38], and the pair \( < CC, CO > \) for every signal is obtained. In the following, 10 signals with different \( | < CC, CO > | \) value in the benchmark are intentionally modified by passing them through one inverter. Figure 3 presents \( | < CC, CO > | \) versus the number of mismatches.
represented as to a primary output. Detection probability of a fault can be value and (ii) observing the net through propagating its value a fault on a net requires (i) controlling the net to a desired observability values and fault coverage [41]. Task of detecting testing.

If a Trojan is composed of signals with considerably low increases such that the average of NoError is about 430 when \( \alpha = 0.5 \) and \( \alpha = 0.3 \), this threshold is 55 and 100, respectively. Figure 4(a) indicates any fault whose \( t \) is considerably high, its detection probability is almost 0. Figure 4(b) studies the effect of the number of test vectors on the fault detection probability. While \( \alpha = 0.5 \), \( V \) takes the values of \( 7.2E + 11 \), \( 7.2E + 12 \), \( 7.2E + 13 \), and \( 7.2E + 14 \). Figure 4(b) highlights the inconceivable impact of increasing the number of test vector on fault detection probability. Similar to Figure 4(a), Figure 4(b) also emphasizes that a net whose \( t \) is considerably high remains undetected. Therefore, a circuit with 100% fault coverage does not contain any signals with large \( t \). As detecting a Trojan signal requires its activation and propagation to an observation point, the above analysis for probability of fault detection is applicable to Trojan signal detection. Complement of Figure 3, Figure 4 also indicates Trojan signals should have considerably low testability to remain hidden during circuit authentication.

IV. THE COTD FLOW

Figure 5 presents the COTD flow. It takes a gate-level netlist as the input, and the Controllability and Observability Analyses step performs the controllability and observability analyses to determine controllability and observability values, i.e. CC0, CC1, and CO values. Afterwards, the Unsupervised Clustering Analysis step is performed to cluster signals based on their controllability and observability values. Two signal lists are produced: Trojan Signals List and Genuine Signals List. The Genuine Signals List only contains all genuine/original signals in the netlist. The Trojan Signal List only contains all Trojan signals if any Trojan exists. If this list is empty, the circuit is Trojan free; otherwise, the circuit is Trojan inserted.

One major issue associated with hardware Trojan detection is the lack of golden circuit as a reference. Unsupervised learning is a type of machine learning algorithm used to explore datasets consisting of input data without labeled responses. Cluster analysis or clustering is the most common unsupervised learning method used for grouping data. It groups a set of objects in such a way that objects in the same group/cluster are more similar to each other than to those in other groups/clusters. The clusters are modeled using a measure of similarity metrics such as Euclidean. Two factors determine the quality of clustering: intra-cluster distance and inter-cluster distance. While clustering, it is desired to maximize inter-cluster distances and to minimize intra-cluster

(NoError) at the primary outputs and scan flip-flops after applying random patterns for 57400 test clock cycles.

While a large \( |<CC, CO>| \) value indicates low testability of a signal due to its low controllability and observability, Figure 3 signifies that the number of mismatches considerably low when the \( |<CC, CO>| \) value is even moderately high. The signal with \( |<CC, CO>| = 101.57 \) only causes 10 mismatches over 57400 test clock cycles. On the other hand, Figure 3 also indicates that with reducing \( |<CC, CO>| \), i.e. increasing a signal’s testability, NoError considerably increases such that the average of NoError is about 430 when \( |<CC, CO>| \leq 50 \). This analysis indicates that if a Trojan is composed of signals with considerably low testability, its existence would highly remain unknown during testing.

There is direct relationship between controllability and observability values and fault coverage [41]. Task of detecting a fault on a net requires (i) controlling the net to a desired value and (ii) observing the net through propagating its value to a primary output. Detection probability of a fault can be represented as

\[
P(t) = e^{-\alpha \times t}.
\]  
(3)

where \( t \) is sum of controllability and observability values of the faulty line for SCOAP, and \( \alpha \) generally takes a value close to 1 that indicates the strong correlation of \( t \) and \( P(t) \). Based on the geometric distribution, the probability of detecting a fault after applying \( V \) test vector is

\[
1 - [1 - P(t)]^V
\]  
(4)

Figure 4 presents the relationship between the detection probability of a fault and \( t \). Figure 4(a) presents how the probability of detecting a fault is related to \( t \) for three different value of \( \alpha \) when \( V = 7.2E + 11 \). While \( V \) is considerably high, the figure shows that the detection probability is almost 1 for any fault whose \( t \) is smaller than 40 when \( \alpha = 0.9 \). The detection probability is almost 0 if the \( t \) of a net is larger. For \( \alpha = 0.5 \) and \( \alpha = 0.3 \), this threshold is 55 and 100, respectively. Figure 4(a) indicates any fault whose \( t \) is considerably high remains undetected. Therefore, a circuit with 100% fault coverage does not contain any signals with large \( t \). As detecting a Trojan signal requires its activation and propagation to an observation point, the above analysis for probability of fault detection is applicable to Trojan signal detection. Complement of Figure 3, Figure 4 also indicates Trojan signals should have considerably low testability to remain hidden during circuit authentication.

Fig. 2. The signal \( T \)'s waveform after applying random test patterns for 725041 test cycles.

Fig. 3. The relationship between signal testability and signal error for selected signals in s38417 benchmark.
Fig. 4. The relationship between detecting a fault and its $t$ defined as the sum of controllability and observability values.

One of the most common clustering algorithms is $k$-means clustering that strives to meet both goals at the same time. $k$-means clustering aims to partition $n$ observations into $k$ clusters in which each observation belongs to the cluster with the nearest mean, serving as the center (or centroid) of the cluster. Given a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation is a $d$-dimensional real vector, $k$-means clustering aims to partition the $n$ observations into $k$ sets $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares. In other words, its objective is to find:

$$\arg \min_{\mu_1, \ldots, \mu_k} \sum_{i=1}^{k} \sum_{x \in S_i} \| x - \mu_i \|^2$$

Using the Equation 2, the COTD technique obtains the pairs $< CC, CO >$ for every signal in a circuit. Then it passes them in the form of a 2-dimensional dataset to the $k$-means algorithm where $k = 3$. Signatures can be divided into three clusters ($k = 3$): (i) genuine signals whose both $CC$ and $CO$ values are small, and Trojan signals (ii) with large $CC$ values or (iii) with large $CO$ values. Therefore, the $k$ parameter is set to 3 to distinguish Trojan signals from genuine signals.

It is possible a Trojan signal to have both large $CC$ and $CO$ values. Such a Trojan signal is still distinguished and reported in Trojan Signal List.

As Trojan signals should have large $CC$ or $CO$ values to avoid their detection as opposed to genuine signals that should have small $CC$ or $CO$ values to be testable, the $k$-means algorithm effectively separates all genuine signals in one cluster and isolates all Trojan signals in different clusters with considerable inter-cluster distances from the cluster of genuine signals. Considering the small $CC$ or $CO$ values for genuine signals and the relative small size of hardware Trojans, the cluster of genuine signals is easily distinguishable as its centroid is the closest to the corner $< CC, CO >= < 0, 0 >$ compared with the centroids of Trojan clusters, and it contains the larger portion of circuit signals. Therefore, the COTD technique can effectively determine whether a circuit is Trojan inserted without need of any golden circuit.

To perform controllability and observability analyses, we use Synopsys’s TetraMAX tool [43]. Using a simple TCL program, COTD obtains $CC_0, CC_1$, and $CO$ values for each signal in a gate-level netlist. In the following, COTD determines $CC$ value for each signal using Equation 2 and stores $CC$ and $CO$ values for all signals in a $n \times 2$ array where $n$ is the number of signals in the circuit. To perform clustering analysis, the array is passed to another simple program in R to perform the unsupervised $k$-means cluster analysis with $k = 3$ [44].

Upon existence of any Trojan signals in the circuit, they are separately clustered with considerable inter-cluster distances from the cluster of genuine signals.

### A. Complexity: COTD vs. Existing Techniques

Usability of any HT detection technique strongly depends on its time complexity. In this subsection, the time complexity of COTD is compared with HaTCh, VeriTrustX, and FANCIX techniques [35]. Both VeriTrustX and FANCIX techniques require monitoring of all sequential stages of design. VeriTrustX monitors the activation history of each entry in the truth table instead of terms in SOP/POS form of the Boolean function. Assuming $m$ the number of circuit inputs, the time complexity of VeriTrustX is $O(2^m)$; i.e. an exponential computation [35]. To compute the control value of each signal in a circuit, FANCIX needs to go through the each entry of developed truth tables and for all primary input combinations.

Fig. 5. The COTD flow.
This leads to the time complexity $O(m^2 d^m)$ [35]. Assuming $n$ represents the total number of wires in the circuit and $d$ the worst case trigger signal dimension, HaTCh has the time complexity of $O(2(n^2) d^m)$ [35]. $n$ and $d$ may have a linear relationship and a logarithmic relationship with $m$, respectively. On the other hand, COTD consists of two steps in sequence: Step-1 determining controllability and observability values, and Step-2 executing $k$-mean clustering. Step-1 has the time complexity $O(n)$, and Step-2 has the time complexity of $O(nkd)$ using the Lloyd’s algorithm where $n$ is the number of wires, $d$ is the number of data dimensions, $k$ the number of clusters which is set to 3 for COTD, and $i$ the number of iterations needed until convergence. On data that do have a clustering structure, the number of iterations until convergence is often small, and results only improve slightly after the first dozen iterations. Therefore, the Lloyd’s algorithm is often considered to be of “linear” time complexity in practice ($O(n)$). In total, the time complexity of COTD is $O(n) + O(n) = O(n)$.

Figure 6 shows a comparison of time complexity of different countermeasures. HaTCh-2in and HaTCh-4in show the complexities for the $m$-input AND gate circuit implementation with 2-input and 4-input AND gates, respectively. While HaTCh is sub-exponential in $m$ and not exponential in $m$ as for FANCIX and VeriTrustX, COTD outperforms and presents a linear relationship with $m$. The inconsiderable time complexity of COTD makes it a perfect choice for hardware Trojan detection in a gate-level netlist.

The COTD technique can be easily integrated into current integrated design flow as it is based on widely used commercial tools. Furthermore, the COTD technique does not require any pattern application; therefore, it outperforms the current existing techniques in terms of performance. As the COTD technique is based on controllability and observability analyses, any Trojan circuit not connected to the main circuit would have infinite CC or CO values. As a result, they are separately clustered with high inter-cluster distance and can be identified easily. In addition, using the unsupervised clustering technique eliminates the need for golden circuit as Trojan signals are clustered as signals with considerably low testability and with considerable inter-cluster distance from genuine signals.

V. Simulation Results

The COST technique is applied to gate-level Trojan-inserted netlist provided on Trust-HUB [36] and some other circuits infected by Trojans proposed by DeTrust [34], always-on Trojans and HaTCh [35]. In two consecutive steps, COTD firstly obtains $CC0(s)$, $CC1(s)$, and $CO(s)$ for every signal $s$ in a circuit using Synopsys TetraMAX [43], and then it executes unsupervised $k$-mean clustering with $k = 3$ to identify Trojan signals. Synopsys TetraMAX can report the SCOAP controllability and observability numbers in a set of numbers “CC0-CC1-CO”. When each of these values exceeds the 254 limit that the program can track, it reports “*” for that value. In this work, “*” is replaced with 254 to execute COTD.

Table I presents the results of applying COTD to gate-level Trust-HUB benchmarks. While the first column indicates the name of benchmark, the second and third columns present the size of trigger and payload of inserted Trojan in terms of the number of gates. The next set of two columns presents the inter-cluster distances between the genuine-signal cluster and two Trojan-signals clusters. The next set of two columns indicates the amount of time required to obtain controllability and observability values and to perform clustering and identifying Trojan signals. The last two columns show the number of signals identified as genuine signals (No Genuine Signals) and the number of signals identified as Trojan signals (No Trojan Signals). Further, false negative (FN) rate, the number of signals wrongly identified as genuine signals, and false positive (FP) rate, the number of signals wrongly identified as Trojan signals, are reported.

While the circuits have different sizes, and they contain Trojans with different sizes and functionality, all Trojans are successfully detected. For all benchmarks, signals are grouped in three clusters. All genuine signals are isolated in one cluster (Gen Clst), and all Trojan signals are divided into two clusters one with high CC values (Tj. Clst1) and the other with high CO values (Tj. Clst2). The results in Table I highlight there is considerable inter-cluster distance between genuine signals and Trojan signals such that they can be easily distinguished. The average inter-cluster distance of Gen Clst and Tj. Clst1 and Gen Clst and Tj. Clst2 are about 328 and 245, respectively.

The processing timing analysis shows time to determine a circuit contains a Trojan has taken less than 14 seconds in the worst case. This results support that the timing complexity of COTD is a linear order of the number of signals in the circuits as discussed in subsection IV-A. Table I also presents the number of genuine and Trojan signals for each benchmark, and it shows both FN and FP rates are 0 for all cases. The zero rates indicate that there is no signal that is wrongly identified as a genuine signal while it is truly a Trojan signal and vice versa.

The results signify that realizing a stealthy hardware Trojan getting rarely activated would result in Trojans whose signals have significantly high controllability and observability values, i.e. very difficult to control and observe. Otherwise, they are easily detectable. Interestingly, two gate-level Trojans in
TABLE I
DETECTING HARDWARE TROJANS IN TRUST-HUB GATE-LEVEL NETLIST USING COTD

<table>
<thead>
<tr>
<th>Trojan</th>
<th>Trigger Size</th>
<th>Payload Size</th>
<th>Inter-cluster Distance</th>
<th>Processing Time (min sec)</th>
<th>No Genuine Signals (FN)</th>
<th>No Trojan Signals (FP)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Gen Clst-Tj.</td>
<td>Clst 1</td>
<td>Gen Clst-Tj.</td>
<td>Clst 2</td>
</tr>
<tr>
<td>s38417—T100</td>
<td>11</td>
<td>1</td>
<td>316.97</td>
<td>253.69</td>
<td>00:03:47</td>
<td>00:00:08</td>
</tr>
<tr>
<td>s38417—T200</td>
<td>11</td>
<td>4</td>
<td>246.66</td>
<td>247.41</td>
<td>00:02:89</td>
<td>00:00:06</td>
</tr>
<tr>
<td>s38417—T300</td>
<td>15</td>
<td>29</td>
<td>431.11</td>
<td>246.87</td>
<td>00:02:66</td>
<td>00:00:07</td>
</tr>
<tr>
<td>Ethernet—T700</td>
<td>12</td>
<td>1</td>
<td>298.14</td>
<td>247.86</td>
<td>00:11:97</td>
<td>00:1:04</td>
</tr>
<tr>
<td>Ethernet—T710</td>
<td>12</td>
<td>1</td>
<td>283.50</td>
<td>247.45</td>
<td>00:09:96</td>
<td>00:1:10</td>
</tr>
<tr>
<td>Ethernet—T720</td>
<td>12</td>
<td>1</td>
<td>298.14</td>
<td>247.80</td>
<td>00:08:98</td>
<td>00:1:10</td>
</tr>
<tr>
<td>Ethernet—T730</td>
<td>12</td>
<td>1</td>
<td>298.14</td>
<td>247.75</td>
<td>00:09:75</td>
<td>00:1:08</td>
</tr>
<tr>
<td>RS232—T1000</td>
<td>10</td>
<td>5</td>
<td>335.72</td>
<td>251.23</td>
<td>00:05:53</td>
<td>00:00:01</td>
</tr>
<tr>
<td>RS232—T100</td>
<td>11</td>
<td>1</td>
<td>345.72</td>
<td>251.20</td>
<td>00:06:69</td>
<td>00:00:00</td>
</tr>
<tr>
<td>RS232—T200</td>
<td>12</td>
<td>1</td>
<td>375.98</td>
<td>251.21</td>
<td>00:09:96</td>
<td>00:00:02</td>
</tr>
<tr>
<td>RS232—T300</td>
<td>6</td>
<td>2</td>
<td>294.32</td>
<td>251.23</td>
<td>00:01:24</td>
<td>00:00:00</td>
</tr>
<tr>
<td>RS232—T400</td>
<td>12</td>
<td>1</td>
<td>345.53</td>
<td>251.15</td>
<td>00:08:81</td>
<td>00:00:01</td>
</tr>
<tr>
<td>RS232—T500</td>
<td>12</td>
<td>3</td>
<td>346.33</td>
<td>251.23</td>
<td>00:09:91</td>
<td>00:00:02</td>
</tr>
<tr>
<td>RS232—T600</td>
<td>9</td>
<td>2</td>
<td>362.77</td>
<td>251.23</td>
<td>00:09:96</td>
<td>00:00:00</td>
</tr>
<tr>
<td>s35850—T100</td>
<td>26</td>
<td>1</td>
<td>331.02</td>
<td>250.80</td>
<td>00:08:83</td>
<td>00:00:03</td>
</tr>
<tr>
<td>s35932—T100</td>
<td>13</td>
<td>2</td>
<td>435.86</td>
<td>250.52</td>
<td>00:08:80</td>
<td>00:00:07</td>
</tr>
<tr>
<td>s35932—T300</td>
<td>12</td>
<td>24</td>
<td>250.51</td>
<td>251.03</td>
<td>00:09:95</td>
<td>00:00:06</td>
</tr>
<tr>
<td>s38584—T200</td>
<td>126</td>
<td>1</td>
<td>431.49</td>
<td>252.14</td>
<td>00:09:97</td>
<td>00:00:08</td>
</tr>
<tr>
<td>s38584—T300</td>
<td>1142</td>
<td>1</td>
<td>434.46</td>
<td>249.01</td>
<td>00:01:05</td>
<td>00:00:09</td>
</tr>
<tr>
<td>vga_in—T100</td>
<td>4</td>
<td>1</td>
<td>248.12</td>
<td>246.60</td>
<td>00:07:08</td>
<td>00:00:71</td>
</tr>
<tr>
<td>wb.conmax—T100</td>
<td>11</td>
<td>4</td>
<td>184.42</td>
<td>153.41</td>
<td>00:02:02</td>
<td>00:00:24</td>
</tr>
</tbody>
</table>

Trust-HUB benchmarks present low controllability and observability values, and they are frequently activated by only applying few hundred test patterns, shown in Table II.

Shown in Table II, s35932—T200 benchmark contains a Trojan whose trigger has 11 gates and its payload one gate. The analysis indicates that the maximum CC and CO for the genuine circuit and the Trojan circuit are $<8.48, 12>$ and $<17.02, 20>$, respectively. Comparing with Trojans in Table I, the Trojan in s35932—T200 benchmark has very low controllability and observability values; therefore, we observe the Trojan is activated 42 times by only applying random test patterns for 4261 test clock cycles. s38584—T100 benchmark also contains a small Trojan with low controllability and observability values. For this case, the maximum CC and CO values for the Trojan circuit are even less than those for the genuine circuit. Table II indicates that this Trojan is activated 21 times within only 3286 test clock cycles. Results in Table II emphasize that the controllability and observability values of Trojan signals cannot be small and close to those of genuine signals; otherwise Trojans frequently interfere the normal operation of a circuit.

The controllability and observability values of Trojan signals should be significantly high so that the Trojan remains hidden during circuit authentication. Table III presents the maximum and minimum of $<CC,CO>$ for signals in the genuine circuit and Trojan circuit for Trust-HUB gate-level netlist, respectively. The results clearly indicate even the minimum magnitude of $|<CC,CO>|$ for Trojan signals is much higher than the maximum magnitude of $|<CC,CO>|$ for genuine signals. The larger this difference is, the stealthier a Trojan behaves. Almost all Trojans have a minimum $|<CC,CO>|$ value above 254 while the maximum $|<CC,CO>|$ value for genuine signals ranges between 12 and 101.

Figure 7 depicts the results of unsupervised $k$-mean clustering with $k=3$ for selected gate-level Trojans in Table III. The figure shows genuine signals are localized in the bottom left corner of graph while Trojan signals are mainly located in three other corners of graph where $CC, CO$, or both are high. Furthermore, Figure 7 depicts the significant inter-cluster distance between Trojan signals and genuine signals.

Among Trojans in Table III, the maximum $|<CC,CO>|$ value for genuine signals in wb_conmax—T100 benchmark is relatively close to the minimum $|<CC,CO>|$ value for the Trojan inserted in this benchmark. While this Trojan is correctly detected, the low value of minimum $|<CC,CO>|$ value for the Trojan compare to the other Trojans may expose its existence during the authentication phase. We have applied random test patterns for 3779 test clock cycles to wb_conmax—T100 benchmark, and the Trojan becomes activated 5 times. While the Trojan in wb_conmax—T100 benchmark is stealthier than Trojans in s35932—T200 and s38584—T100 benchmarks analyzed in Table II, having Trojan signals with low $|<CC,CO>|$ values reveals the Trojan
existence in a short time. This fact signifies that the minimum $\langle CC, CO \rangle$ values for Trojan signals should be considerably high to avoid Trojan activation during authentication time. On the other hand, COTD can catch Trojan signals as they stand in a cluster with considerably high inter-cluster distance from genuine signals.

While the COTD technique has correctly identified all gate-level Trojans in Trust-HUB with zero false negative and false positive rates without need for a golden model, we have applied COTD to some other Trojans introduced by DeTrust [34] and HaTCh [35].

DeTrust effectively creates a multi-layer wall of sequential elements (e.g. flip-flops) to divide the trigger cone of Trojan into several combinational blocks so as to eliminate direct reachability of a Trojan payload to its triggering inputs. This technique has effectively defeated FANCI [33] and VeriTrust [32]. While FANCI identifies rare trigger inputs based on the large fan-in cone of Trojan payload, DeTrust breaks this large fan-in cone into several smaller ones; therefore, the size of cone directly driving the Trojan payload is effectively reduced, and the Trojan resides in the false negative zone. VeriTrust identifies un-activated sum-of-product (SOP) and product-of-sum (POS) terms in a circuit and then isolates redundant terms by setting the un-activated SOP/POS terms to don’t cares. The redundant terms are flagged as suspicious Trojan triggers. In a similar manner, DeTrust indeed breaks the SOP and POS terms by raising a multi-layer wall of sequential elements between Trojan payload and its triggers. However, there is an unexplored assumption with DeTrust.

The inserted sequential elements behave as directly controllable and observable elements in confronting with FANCI and serve as present states (pseudo primary inputs) in creating SOP and POS terms in studying VeriTrust. On the other hand, COTD identifies Trojan signals based on their controllability and observability values. Even if a Trojan uses sequential elements, the controllability and observability values of signals in their fan-out cone is not reduced. According to the SCOAP program $\langle CC0, CC1, CO \rangle$ for the output $q$ and input $d$ of D flip-flop is:

$$CC0(q) = \min \{CC0(d) + CC0(CK) + CC1(CK) + CC0(r), CC1(r) + CC0(CK)\}$$

$$CC1(q) = CC1(d) + CC0(CK) + CC1(CK) + CC0(r)$$

$$CO(d) = CO(q) + CC0(CK) + CC1(CK) + CC0(r)$$

$$\text{(6)}$$

where $CK$ and $r$ are clock and reset inputs, respectively [40]. As Equation 6 highlights inserting flip-flops does not reduce the controllability and observability values of their input and output. Therefore, inserting multi-layer wall of sequential elements in Trojan fan-in cone by DeTrust to hide Trojan circuits would render ineffective in dealing with COTD. It should be noted that it is not expected that a Trojan contains scan flip-flops as this would modify the existing scan architecture and increase Trojan activation probability by serving as pseudo primary inputs.

To study Trojans designed with DeTrust in mind, Trust-HUB gate-level Trojans are updated by inserting one flip-flop at the output of each gate in their trigger circuitry. Figure 8 presents $\langle CC, CO \rangle$ values for Trojan trigger signals for original Trust-HUB Trojan (labeled Trust-HUB) and for the same signals and new introduced signals after applying DeTrust technique (labeled DeTrusted).
Figure 8 highlights how inserting flip-flops indeed increase $|<CC,CO>|$ values. For example, the original Ethernet-T700/T701/T702/T703 Trojans have the median $|<CC,CO>|$ values of about 254, and the maximum $|<CC,CO>|$ values are about 439, 359, and 439, respectively. After inserting flip-flops, the DeTrusted $|<CC,CO>|$ values are significantly increased with the median value of about 439. Or, the median $|<CC,CO>|$ value for wb_conmax-T100 before inserting flip-flops is about 178. On the other hand, this value is considerably increased to about 439 after applying the DeTrust technique.

Table IV presents inter-cluster distances between the clusters of Trojan signals and the cluster of genuine signals for DeTrusted gate-level Trojans on Trust-HUB. As it is expected based on Equation 6, there is increase in inter-cluster distances from original Trojans to DeTrusted Trojans. The average inter-cluster distances of original Trojan in Table I is 286.85 and that of DeTrusted Trojans is 312.66. Therefore, DeTrusted Trojans are even more easily distinguishable by COTD.

Always-on Trojans: A Trojan can be always-on such that it does not require any input from a circuit to operate. An inverter-based ring-oscillator Trojan that has no input control and is to reduce circuit reliability is an example of such. As such a Trojan is not connected to any internal net and its internal signals are continuously changing, the Trojan looks like a legitimate module with no rarely active state. As majority of detection techniques are to localize Trojans based on their rare activity, an always-on Trojan can be missed. As such a Trojan’s trigger is not connected to any internal signals, its internal signals are not controllable and observable; therefore, Trojan signals would have the maximum $CC_0$, $CC_1$, and $CO$ values. Hence, COTD can easily detect Trojan Always-on Trojans.
SALMANI: COTD: REFERENCE-FREE HARDWARE TROJAN DETECTION AND RECOVERY 347

Fig. 10. The 4-XOR-LFSR Trojan to leak a secret key, introduced in [35].

s38417-T300 benchmark contains a triggered Trojan whose payload is a ring-oscillator with 29 stages. To realize an always-on Trojan, Trojan trigger is removed. COTD reports that all Trojan signals (outputs of 29 Trojan inverters) have $CC_0 = 254$, $CC_1 = 254$, and $CO = 254$.

HaTCh has investigated the introduction of implicit malicious behavior to increase the stealthiness of hardware Trojans by creating a possibility of having a false negative under logic testing based techniques. To do so, HaTCh has introduced $k$-XOR-LFSR Trojans whose general structure is shown in Figure 9. Using a linear-feedback-shift-register (LFSR) brings a semi-nondeterministic behavior for a Trojan, such that it would be difficult to characterize its behavior. Such a characteristic makes Trojan detection difficult using existing detection technique. On the other hand, COTD is able to detect such a Trojan as the LFSR circuit is not connected to the main circuit. That is, a $k$-XOR-LFSR Trojan is not controllable and its controllability value is infinite.

To evaluate the effectiveness of COST, the 4-XOR-LFSR Trojan to leak a secret key described in [35] is implemented as shown in Figure 10(a). Upon reset the LFSR $<L_3L_2L_1L_0>$ is loaded with “1010”, and the Trojan leaks a secret information when $L_1 = '0'$ and $L_3L_2L_0 = '111'$. Figure 10(b) shows $CC_0$, $CC_1$, and $CO$ for Trojan signals. As the results show either controllability value or observability value of Trojan signals has the maximum value. Therefore, the $k$-XOR-LFSR Trojans can be easily detected by COTD.

Hardware Trojan Recovery: It is so valuable to fully recover an inserted Trojan in a circuit to understand the purpose of adversary. Identifying Trojan trigger and Trojan payload circuitry present detailed information about Trojan implementation. Further, it would also make it possible to determine (i) which signals are being used as inputs for the Trojan trigger, and (ii) which signals are targeted by the Trojan payload.

After isolating Trojan signals using COTD, it is possible to fully recover a hardware Trojan inserted in a gate-level netlist. Figure 11 presents the proposed hardware Trojan recovery flow in a gate-level netlist. Trojan signals identified by COTD and Trojan-inserted circuit are used as inputs. The Trojan Gates Identification step extracts Trojan gates, and their input and output pins. With this information, it is possible to execute the Trojan Reconstruction step. In this step, Trojan trigger and Trojan payload circuitry are restored. Signals connected to Trojan gates’ pins are obtained, and the interconnection between Trojan gates is reconstructed. Further, it is determined which signals from the main circuit are being used as Trojan triggering signals, and which signals in the main circuit are attacked by the Trojan payload. Any signal that drives a Trojan gate and is not driven by any Trojan gate is identified as a Trojan triggering signal. Any signal that is not a Trojan signal but passing through a Trojan gate is identified as a Trojan payload signal. Any gate whose one of inputs is a payload signal composes the Trojan payload circuitry. The remaining Trojan gates compose the Trojan trigger circuitry. The hardware Trojan recovery flow is implemented in Synopsys’ design compiler and only consists of about 100 lines, and its complexity is an order of the number of Trojan signals. The flow is being applied to all gate-level netlist on Trust-HUB, and all Trojans are successfully recovered.

As a sample, Figure 12 presents the output of flow for the s38417-T100 Trojan on Trust-HUB. While Figure 12 shows...
Comparing the report in Figure 12 with the Trojan inserted in s38417-T100 circuit shows the proposed hardware Trojan recovery flow can perfectly extract the inserted Trojan. While a Trojan circuit can be completely isolated, it is also possible to clean up the Trojan-inserted circuit.

VI. DISCUSSION

The results indicate the COTD technique has successfully detected all gate-level Trust-Hub Trojans and Trojans designed based on DeTrust and HaTCh techniques in less than 14 seconds with zero false positive and negative rates without need for a golden model. COTD is based on SCOAP controllability and observability numbers which are approximate measurements of circuit testability. As SCOAP does not consider the dependency of signals, reconvergent signals may result in the increase of controllability and observability values.

As SCOAP controllability and observability values are approximate and COTD distinguishes Trojan signals based on their significantly high controllability and observability values, two important questions may arise: Question 1 - whether it is possible to realize a Trojan signal with low controllability and observability numbers such that it remains inactive during testing (false negative); Question 2 - whether an original signal can be recognized as a Trojan signal (false positive).

False Negative: Inaccuracy of SCOAP is attributed to the existence of reconvergent signals and ignoring dependency between signals. As a result, the controllability and observability values of some signals might be overestimated. To ensure a circuit is fully testable, all signals of the circuit should be controllable and observable. As a result, any signal situated in the zone of controllable and observable signals including Trojan signals would experience switching activity. Hence, it is not possible to implement a Trojan whose signals have controllability and observability in the range of genuine signals in a hosting circuit but remain inactive during circuit testing. Such a behavior flags them as untestable signals, and it would be followed with their thorough analyses.

To evaluate switching activity of signals, Figure 13 shows the frequency of switching activity in each original circuit in Table III after applying random test vectors for 500 test cycles. While the number of transitions are sorted ascending in each circuit, the horizontal axis indicates the signal index and the vertical axis presents the number of transitions from ‘0’ to ‘1’ or ‘1’ to ‘0’. The axes are presented in logarithmic scales to provide a finer look at signals with low switching activities. Using the signal index, it is possible to observe how many signals exist with switching activities below a certain number. For example, Ethernet benchmark has four signals whose switching activity is two. The circuits have different sizes with different functionality, and larger circuits like Ethernet benchmark have larger number of signals with low switching activities in comparison with smaller circuits like RS232 benchmark. While the circuits contain signals with wide range of switching activities, there is zero signal with zero switching activity even after applying a small number of random test patterns.

The results emphasize any signal whose the controllability and observability values even if overestimated is in the range of controllability and observability values for the original signals experiences switching activity. This also holds true for a Trojan signal has controllability and observability values less than or close to the maximum $|CC, CO|$ of genuine signals. This fact is already discussed in Table II where two Trojans consisting of signals whose maximum CC and CO are slightly larger and smaller than the maximum CC and CO of genuine signals in hosting original circuits. Table II shows considerable switching activities of Trojan signals after applying few thousands test patterns.
It should be noted that there might be some signals in the original circuit that are undetectable meaning either not controllable or not observable. A signal can be undetectable if it is unused, blocked, or redundant. Undetectable signals are distinguishable using ATPGA tools; therefore, a Trojan signal cannot be implemented as an undetectable signal.

**False Positive:** To be incorrectly recognized as a Trojan signal by COTD, a genuine signal should present a considerable inter-cluster distance from other genuine signals. Table V presents the inter-cluster distances between signals after executing unsupervised $k$-mean clustering with $k = 3$ for Trojan-free gate-level benchmarks on Trust-HUB. Compared with the inter-cluster distance of Trojan signal clusters and genuine signal clusters in Table III, the inter-cluster distance of genuine signal clusters in Table V is significantly smaller. The average inter-cluster distance for Trojan-inserted circuits in Table I is about 286.85 and the average inter-cluster distance for Trojan-free circuits in Table V is about 18.45 that is about 15 times smaller. Furthermore, Figure 14 depicts the three clusters for each Trojan-free gate-level benchmarks on Trust-HUB.

### Table V

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Inter-cluster Distance</th>
<th>Clst1 - Clst2</th>
<th>Clst1 - Clst3</th>
<th>Clst2 - Clst3</th>
</tr>
</thead>
<tbody>
<tr>
<td>s38417</td>
<td>45.12</td>
<td>31.88</td>
<td>45.87</td>
<td></td>
</tr>
<tr>
<td>Ethernet</td>
<td>13.79</td>
<td>41.38</td>
<td>28.32</td>
<td></td>
</tr>
<tr>
<td>RS232</td>
<td>4.71</td>
<td>3.56</td>
<td>5.32</td>
<td></td>
</tr>
<tr>
<td>s15850</td>
<td>16.78</td>
<td>14.85</td>
<td>16.38</td>
<td></td>
</tr>
<tr>
<td>s35932</td>
<td>6.14</td>
<td>6.68</td>
<td>3.42</td>
<td></td>
</tr>
<tr>
<td>s35844</td>
<td>4.58</td>
<td>9.41</td>
<td>13.60</td>
<td></td>
</tr>
<tr>
<td>vga_lcd</td>
<td>34.16</td>
<td>23.73</td>
<td>23.36</td>
<td></td>
</tr>
<tr>
<td>wb_commax</td>
<td>12.60</td>
<td>13.09</td>
<td>23.90</td>
<td></td>
</tr>
</tbody>
</table>

VII. CONCLUSIONS

Considering the extensive practice of integrated circuit horizontal design flow and demand for untrusted third party intellectual properties, the COTD technique, a novel Trojan detection technique at the gate level, was presented. Based on controllability and observability analyses, COTD isolated Trojan signals using unsupervised $k$-mean clustering with $k = 3$. Without need for a golden model, COTD was able to detect various Trojans with zero false positive and false negative rates, and the maximum reported execution time has been a fraction of minute. While majority of Trojan detection techniques are based in test pattern application, COTD is a static technique does not require any test pattern generation and application. In addition, COTD made it possible to fully isolate a Trojan circuit, if it exists. Finally, COTD can be easily integrated into current commercial integrated design flow.

REFERENCES


...
学霸图书馆
www.xuebalib.com

本文献由“学霸图书馆-文献云下载”收集自网络，仅供学习交流使用。

学霸图书馆（www.xuebalib.com）是一个“整合众多图书馆数据库资源，提供一站式文献检索和下载服务”的24小时在线不限IP图书馆。

图书馆致力于便利、促进学习与科研，提供最强文献下载服务。

图书馆导航：
图书馆首页 文献云下载 图书馆入口 外文数据库大全 疑难文献辅助工具