A Score-Based Classification Method for Identifying Hardware-Trojans at Gate-Level Netlists

Masaru Oya†, Youhua Shi‡, Masao Yanagisawa†, Nozomu Togawa†
†Department of Computer Science and Communications Engineering, Waseda University
‡Waseda Institute for Advanced Study, Waseda University

***Abstract*—Recently, digital ICs are often designed by outside vendors to reduce design costs in semiconductor industry, which may introduce severe risks that malicious attackers implement *Hardware Trojans (HTs)* on them. Since IC design phase generates only a single design result, an RT-level or gate-level *netlist* for example, we cannot assume an HT-free netlist or a *Golden netlist* and then it is too difficult to identify whether a generated netlist is HT-free or HT-inserted. In this paper, we propose a score-based classification method for identifying HT-free or HT-inserted gate-level netlists without using a Golden netlist. Our proposed method does not directly detect HTs themselves in a gate-level netlist but a net included in HTs, which is called *Trojan net*, instead. Firstly, we observe Trojan nets from several HT-inserted benchmarks and extract several their features. Secondly, we give *scores* to extracted Trojan net features and sum up them for each net in benchmarks. Then we can find out a *score threshold* to classify HT-free and HT-inserted netlists. Based on these scores, we can successfully classify HT-free and HT-inserted netlists in all the Trust-HUB gate-level benchmarks. Experimental results demonstrate that our method successfully identify all the HT-inserted gate-level benchmarks to be "HT-inserted" and all the HT-free gate-level benchmarks to be "HT-free" in approximately three hours for each benchmark.**

***Index Terms*—hardware Trojans, golden-IC free, classification, identification, gate-level netlist**

I. Introduction

Recently, outside vendors often design and fabricate digital ICs to reduce IC costs in semiconductor industry where we face severe risks that malicious attackers may implement Hardware Trojans (HTs) on them. HTs can be inserted both in design phase and fabrication phase. In fabrication phase, many IC chips are fabricated and malicious attackers can insert HTs into some of them. In this case, we have both HT-inserted IC chips and HT-free IC chips. HT-free IC chips are called *Golden ICs* and we can effectively identify HTs by observing the difference between HT-inserted ICs and Golden ICs. However, since design phase generates only a single design result, an RT-level or gate-level *netlist* for example, we cannot assume a *Golden netlist*. Even in fabrication phase, [4] introduces HTs which do not affect side channel too little. An HT detection method for netlists without using Golden netlists is strongly required. Now we focus on an HT detection method for gate-level netlists and discuss what is required for it.

When we use an HT detection method, we want to *identify* whether a given netlist is HT-free or HT-inserted. Then an HT detection for gate-level netlists have to solve the two problems below:

(P1) **Detection**: Can an HT detection method just detect HTs in an HT-inserted gate-level netlist?
(P2) **Identification**: Can an HT detection method identify whether a given gate-level netlist is HT-free or HT-inserted?

The problem (P1) is included in the problem (P2) and an HT detection method is very strong if it solves both (P1) and (P2).

Next we consider information that HT detection methods can use and cannot use. Generally we cannot know any HT pre-informations in a given netlist, such as its trigger conditions, locations, and functionalities. Moreover, we do not know whether a given netlist is HT-free or HT-inserted and do not have its Golden netlist, either. Then we can set the two assumptions below:

(A1) We have no Golden netlist.
(A2) We know no HT pre-informations inserted into a given netlist.

If we have a Golden netlist, this means that we have an HT-free gate-level netlist. Any HT detection methods themselves are unnecessary. As mentioned above, we do not know pre-informations on HTs beforehand. HT detection methods have to detect and identify HTs assuming (A1) and (A2).

Existing HT detection methods for gate-level netlists are roughly classified into three methods: logic test [5], side channel analysis (SCA) [1], [3], [6], and post-verification [2], [7]. Logic test [5] identifies inserted HTs by applying input vectors which activate HTs and observes output logic values. However, we need to know HT locations in a given netlist to examine whether the HTs are activated or not. Logic test cannot assume (A2) to detect HTs. SCA [1], [3], [6] identifies whether a given netlist is HT-free or HT-inserted based on side channel information obtained by its Golden netlist. SCA definitely requires a Golden netlist and it cannot solve (P1) and (P2) assuming (A1). Post-verification [2], [7] first detects suspicious HTs by enumerating signals which do not activate during verification tests. In their experiments, non-activated signals include triggers/payloads in HTs but we cannot know which signals are triggers/payloads without knowing informations in HTs beforehand. This means that post-verification techniques may solve the problem (P1) but not solve the problem (P2) assuming (A2).

Based on the above discussions, we propose a score-based classification method for identifying HT-free/HT-inserted gate-level netlists without using Golden netlists. In our approach, we do not try to detect HTs themselves but we detect nets included in HTs, which is called a *Trojan net*. Firstly, we observe Trojan nets from several HT-inserted benchmarks and extract several their features. Secondly, we give *scores* to extracted Trojan net features and sum up them for each net in benchmarks. Then we can find out a *score threshold* to classify HT-free and HT-inserted netlists. Based on these scores, we can successfully classify HT-free and HT-inserted netlists.

Our proposed method consists of three phases: In Phase 1, we give scores to all the nets in a given gate-level netlist; In Phase 2, we give the three parameters called *Trojan factors* to every net with the highest score; In Phase 3, by comparing the three Trojan factors and the threshold values, we can detect *strong* Trojan nets. In this phase, only when a strong Trojan net is detected, we can conclude that the netlist is HT-inserted, i.e, we can identify HT-free or HT-inserted netlists.

Our method only uses information obtained from a given gate-level netlist and does not require HT pre-informations inserted. Still, our method identifies whether a given netlist is HT-free or HT-inserted. This means that our approach solves (P1) and (P2) assuming (A1) and (A2). Table I compares HT detection methods. Actually, our

978-3-9815370-4-8/DATE15/©2015 EDAA

TABLE I
COMPARISON OF HT DETECTION METHODS.

	Logic test [5]	SCA [1], [3], [6]	Post verification [2], [7]	Ours
(P1)	✓	✓	✓	✓
(P2)	✓	✓		✓
(A1)	✓		✓	✓
(A2)		✓		✓

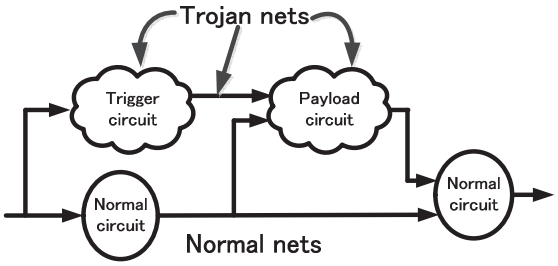


Fig. 1. HT circuit model.

proposed method successfully identify all the HT-inserted gate-level benchmarks in Trust-HUB [8] to be "HT-inserted" and all the HT-free gate-level benchmarks to be "HT-free" in approximately three hours for each benchmark.

The contributions of our score-based classification method are:

1) to clarify Trojan factors and thresholds which distinguish HT-free and HT-inserted gate-level netlists, and then
2) to successfully identify all the HT-free and HT-inserted gate-level circuits in Trust-HUB benchmarks.

This paper proposes the world-first HT-detection method which successfully identify *all* the HT-free and HT-inserted gate-level circuits in Trust-HUB benchmarks.

II. DETECTING TROJAN NETS IN HTS

Fig. 1 shows a general HT circuit model [5]. An HT circuit consists of a trigger circuit and a payload circuit. A trigger circuit generates a trigger signal to activate Trojan. A payload circuit actually activates Trojan, for example, it outputs secret keys to primary outputs. We assume that an HT circuit rarely activates Trojan, since it can be easily detected by circuit verification if it is often activated.

In order to completely eliminate the HT risks, we have to remove an HT circuit from an HT-inserted circuit. However, distinguishing an HT circuit from a normal circuit is too difficult or almost impossible, because some parts of a circuit work as both an HT circuit and a normal circuit. Rather, identifying whether a circuit is HT-free or HT-inserted is quite practical and we focus on any nets included in an HT circuit (which are called *Trojan nets*). Detecting Trojan nets is easier than detecting an entire HT circuit and then our proposed method is based on detecting some of the Trojan nets and hence identifies whether a gate-level netlist is HT-free or HT-inserted.

Now in this section, we extract several features in Trojan nets and give *scores* to them. Based on these scores, we set several *thresholds* to detect *strong* Trojan nets.

A. Trojan Net Features

Firstly, we extract the features of Trojan nets to detect them. We randomly select ten gate-level benchmarks from Trust-HUB [8].[1] Table II shows the selected gate-level benchmarks including HT-inserted ones and HT-free ones. A benchmark here is composed of several sub-modules, each of which is composed of cells such as gates, flip-flops, and adders.

[1]Benchmarks in Trust-HUB include an information on HT and then we can know beforehand the answers to the problems (P1) and (P2) for every benchmark.

TABLE II
RANDOMLY SELECTED TEN GATE-LEVEL BENCHMARKS.

Benchmark	Type	#nets
b19	HT-free	108,332
EthernetMAC10GE	HT-free	103,206
s35932	HT-free	6,423
EthernetMAC10GE-T700	HT-inserted	103,220
RS232-T1000	HT-inserted	311
s15850-T100	HT-inserted	2,456
s38417-T100	HT-inserted	5,819
s38584-T200	HT-inserted	7,580
vga_lcd-T100	HT-inserted	70,162
wb_conmax-T100	HT-inserted	22,197

Fig. 2 summarizes the extracted features of Trojan nets. The nets which have these features are called *weak classification nets*. In Fig. 2, bold signals show weak classification nets and an LSLG (low-switching logic gate) refers to either AND, NAND, OR, or NOR gate. LSLGs have low switching probabilities and hence we can consider that a trigger circuit can mainly consist of LSLGs. Trojan net features are classified into the nine cases (Case 1–Case 9) below:

Case 1 (Fig. 2(a)): When a 1st LSLG is connected to the outputs of several 2nd LSLGs and the total number of the 2nd LSLG inputs is six or more, the 1st LSLG output is a weak classification net.

Case 2 (Fig. 2(b)): When a 1st LSLG is connected to the outputs of several 2nd LSLGs and the total number of the 2nd LSLG inputs is 16 or more, or the total number of 2nd and 3rd LSLGs are 16 or more, the 1st LSLG output is a weak classification net. Note that Case 2 is a special case of Case 1. If a net matches both Case 1 and Case 2, it can be a Trojan net with high probability.

Case 3 (Fig. 2(c)): A multiplexer's selection input is a weak classification net.

Case 4 (Fig. 2(d)): When ADDER output is connected to a cell ("2nd any cell") and the "2nd any cell" is further connected to another cell ("1st any cell"), all the inputs and outputs of the "1st any cell" are weak classification nets. Note that ADDER here refers to a half adder cell or a full adder cell.

Case 5 (Fig. 2(e)): A primary output for each sub-module including flip-flops inside is a weak classification net.

Case 6 (Fig. 2(f)): When 0 or 1 is directly inputted as a logic value into a flip-flop for each sub-module, its outputs and clock signal are weak classification nets.

Case 7 (Fig. 2(g)): When the flip-flop's clock is a weak classification net of Case 2, its input and outputs, and clock signal are weak classification nets.

Case 8 (Fig. 2(h)): Consider the case where a sub-module has more than 22000 nets. In this case, if one of the cell inputs is a sub-module primary input and another one is a feature net of Case 2, then this feature net can be a Trojan net with high probability. Thus, if this case holds, we give more feature points to the feature net as Case 8.

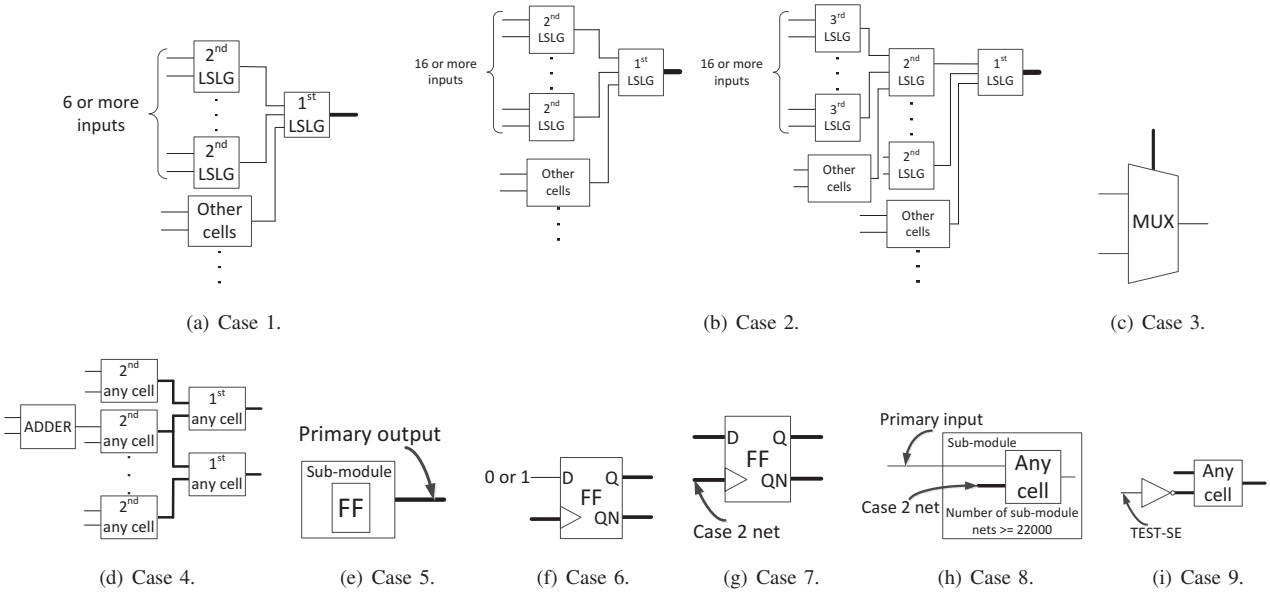
Case 9 (Fig. 2(i)): If the inverted TEST-SE is connected to a cell, its inputs and outputs are weak classification nets, where TEST-SE refers to a test scan enable signal.

B. Scores

Trojan nets have the features as shown in Fig. 2 but there are many nets which match the Cases 1–9. Next, we try to extract only Trojan net by appropriately scoring them.

We carefully analyzed weak classification nets to optimize scores. Weak classification nets are classified into two types: one type includes both Trojan nets and normal nets and the other type includes only Trojan nets. The formers are weak classification nets of (Case 1), (Case 2), (Case 3), (Case 4), and (Case 5) and the latters are weak classification nets of (Case 6), (Case 7), (Case 8), and (Case 9). Then we give scores to every weak classification net as shown in Table III.

Based on the scores on weak classification nets, we calculated scores on every net in each benchmark. If a net belongs to two or more weak classification net cases, we summed up their scores.



(a) Case 1. (b) Case 2. (c) Case 3.

(d) Case 4. (e) Case 5. (f) Case 6. (g) Case 7. (h) Case 8. (i) Case 9.

Fig. 2. Weak classification nets.

TABLE III
SCORES ON WEAK CLASSIFICATION NETS.

Weak classification net	Score
Case 1	1
Case 2	1
Case 3	1
Case 4	1
Case 5	1
Case 6	2
Case 7	2
Case 8	2
Case 9	2

TABLE IV
MAX SCORE OF RANDOMLY SELECT BENCHMARKS.

Benchmark	Max score	Number of max score nets	Contents of max score nets
b19	2	77	Only normal nets
EthernetMAC10GE	2	55	Only normal nets
s35932	1	420	Only normal nets
EthernetMAC10GE-T700	6	1	Only Trojan nets
RS232-T1000	2	2	Only Trojan nets
s15850-T100	6	1	Only Trojan nets
s38417-T100	2	5	Normal nets and Trojan nets
s38584-T200	3	1	Only Trojan nets
vga_lcd-T100	2	2	Only Trojan nets
wb_conmax-T100	4	1	Only Trojan nets

Table IV shows the highest score in each benchmark. *Max score* refers to the highest score and *max score nets* refer to the nets whose score is the highest in each benchmark. According to Table IV, if a netlist includes the net whose max score is 3 or more, all the max score nets are Trojan nets.

However, even when a max score is less than three, a netlist may include Trojan nets. We further develop another threshold to distinguish normal nets and Trojan nets.

C. Max Constant Cycles

HTs are often designed to activate only when they meet some special conditions and hence most of Trojan nets are expected to hold the same value for a long time. We inputted random input vectors to primary inputs for 1M clock cycles and simulated all the max score nets obtained in the previous subsection by a logic simulator. We used VCS from Synopsys to simulate a gate-level netlist. Here, *max constant cycles* refer to the longest cycles that the net holds the same value during the simulation. For example, it is 500k cycles in Fig. 3 when we simulate a gate-level netlist in 1M cycles.

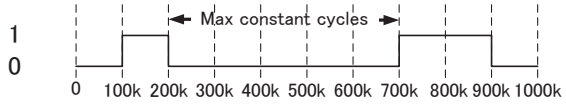


Fig. 3. Max constant cycles simulated for 1M cycles.

Table V shows the max constant cycles for Trojan nets included in max score nets whose score is less than three in Table IV. According to Table V, max constant cycles of Tj_OUT5678 are the smallest and we can see that the nets with max constant cycles of 999,996 cycles or more are expected to be Trojan nets.

Then we extracted the max score nets in Table IV whose score is less than three and whose max constant cycles are 999,996 cycles or more. Table VI shows the number of such nets.

According to Table VI, there are still 59 normal nets in b19 and 10 normal nets in EthernetMAC10GE which have the max score and their max constant cycles are 999,996 cycles or more. In particular, b19 has normal nets whose max constant cycles are 1,000,000. This is because these normal nets hold the same value throughout simulation times. Note that, both b19 and EthernetMAC10GE have the large number of nets compared with other benchmarks. Max constant cycles must be effective if a netlist is small enough but, if it is large, it can cause to increase false positives. We further develop another threshold to distinguish normal nets and Trojan nets for such nets.

D. Number of Max Score Nets

According to Table IV, the number of max score nets including Trojan nets is relatively small, while the number of max score nets including only normal nets is large.

In our scoring, the number of max score nets will decrease if the max score is higher and it will increase if the max score is lower. Therefore, gate-level netlists are expected to be HT-free if the number of max score nets is large enough. According to Table IV, the largest

TABLE V
MAX CONSTANT CYCLES FOR TROJAN NETS WHOSE SCORE IS LESS THAN THREE.

Benchmark	Trojan net name	Max constant cycles
RS232-T1000	iRECEIVER_CTRL	999,999
	iCTRL	999,999
s38417-T100	Tj_Trigger	999,997
	Tj_OUT1234	999,997
	Tj_OUT5678	**999,996**
vga_lcd-T100	Tj_Trigger	999,999
	Tj_OUT1	999,999

TABLE VI
NETS WHOSE MAX CONSTANT CYCLES ARE 999,996 CYCLES OR MORE.

Benchmark	Max score	Number of max score nets whose max const cycles are 999,996 or more	Contents
b19	2	59	Only normal nets
EthernetMAC10GE	2	10	Only normal nets
s35932	1	0	N/A
RS232-T1000	2	2	Only Trojan nets
s38417-T100	2	3	Only Trojan nets
vga_lcd-T100	2	2	Only Trojan nets

TABLE VII
IDENTIFICATION BY STRONG TROJAN NETS.

Benchmark	Number of strong Trojan nets	Contents	Identify
b19	0	N/A	HT-free
EthernetMAC10GE	0	N/A	HT-free
s35932	0	N/A	HT-free
EthernetMAC10GE-T700	1	Only Trojan net	HT-inserted
RS232-T1000	2	Only Trojan nets	HT-inserted
s15850-T100	1	Only Trojan net	HT-inserted
s38417-T100	3	Only Trojan nets	HT-inserted
s38584-T200	1	Only Trojan net	HT-inserted
vga_lcd-T100	2	Only Trojan nets	HT-inserted
wb_conmax-T100	1	Only Trojan net	HT-inserted

number of max score nets including Trojan net is five in s38417-T100. We can assume that gate-level netlists are HT-free if the number of max score nets is six or more.

By setting the threshold for the number of max score nets above, we can finally classify HT-free netlists and HT-inserted netlist in Table IV.

E. Strong Trojan Nets

In summary, every max score net n in each benchmark of Table IV is a Trojan net if n satisfies the following cases:

1) If the score of n is three or more, n is a Trojan net.
2) Otherwise, if max constant cycles of n are 999,996 or more and the number of the max score nets in the netlist is five or less, n is a Trojan net.

Nets satisfying the above cases are called *strong Trojan nets*.

Table VII shows the number of detected strong Trojan nets. In HT-free netlists, we do not find out any strong Trojan nets, whereas we find out strong Trojan nets in HT-inserted netlists. By using scores and thresholds above, we can detect strong Trojan nets and then effectively identify HTs not using Golden netlists nor HT pre-informations.

III. PROPOSED METHOD

Based on the discussions in Section II, we propose in this section a score-based classification method for identifying HT-free/HT-inserted netlists. Roughly, our method is composed of classification and identification. Classification is further composed of weak classification and strong classification. Then, our method is summarized as follows:

Phase 1: Weak classification
Phase 2: Strong classification
Phase 3: Trojan identification

Weak classification detects max score nets from all the nets in a given netlist. Max score nets can include both Trojan nets and normal nets. Then strong classification gives three *Trojan factors* to these max score nets. After that, Trojan identification extracts strong Trojan nets from max score nets and identify whether the netlist is HT-free or HT-inserted.

A. Phase 1: Weak Classification

Weak classification detects nets which have the highest score, i.e., max score nets. Max score nets are the most suspicious Trojan nets in all the nets in a given netlist.

Initially, scores of all the nets are set to be zero. For every net, if it matches one of weak classification net cases (shown in Fig.2), its score increases according to Table III. Then the nets which have the highest score become max score nets.

Calculating max scores and detecting max score nets do not need to know the details in HTs in netlists. We just check whether each net matches a weak classification net case. Weak classification satisfies the assumptions (A1) and (A2).

B. Phase 2: Strong Classification

Strong classification gives three Trojan factors below to each of max score nets:

Trojan factor 1: Max score
Trojan factor 2: Max constant cycles
Trojan factor 3: Max score net count

Trojan factor 1 is the score of the max score nets. Trojan factor 2 is the longest cycles that each max score net holds the same value as shown in Fig. 3. Trojan factor 3 is the number of max score nets.

Clearly, all of the above Trojan factor calculation satisfies the assumptions (A1) and (A2).

C. Phase 3: Trojan Identification

Trojan Identification finally identifies whether a given netlist is HT-free or HT-inserted. Fig. 4 shows a Trojan identification flowchart. Trojan factor thresholds are shown below:

Threshold of Trojan Factor 1: 3
Threshold of Trojan Factor 2: 999,996 cycles
Threshold of Trojan Factor 3: 5

If strong Trojan nets are detected, Trojan identification identifies that the netlist is Trojan-inserted. Otherwise, Trojan identification identifies that the netlist is HT-free. Both the problems (P1) and (P2) can be solved assuming (A1) and (A2).

IV. IMPLEMENTATION RESULTS

We applied our proposed method to all the HT-free and HT-inserted gate-level netlists available in Trust-HUB [8]. Table VIII shows HT-inserted gate-level netlists and Table IX shows HT-free gate-level netlist. Trust-HUB does not provide "HT-free RS232" and then we made "RS232" by manually removing HTs from RS232-T1000.

A. Weak Classification Results

Table X shows the max score and the number of max score nets for each benchmark. We took approximately 10–20 minutes to obtain the scores for each benchmark using Core i7 PC. Columns 2–10 show the number of weak classification nets (Case 1–Case 9). "#max score nets" shows the number of max score nets and "Max score" shows the max score.

TABLE X
MAX SCORE AND NUMBER OF MAX SCORE NETS.

Benchmark	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	#max score nets	Max score
b19-T100	822	43	84	16701	131	0	0	0	0	1	3
b19-T200	822	43	84	16701	131	0	0	0	0	1	3
EthernetMAC10GE-T700	347	56	2595	417	193	3	3	0	0	1	6
EthernetMAC10GE-T710	347	56	2595	417	193	3	3	0	0	1	6
EthernetMAC10GE-T720	347	56	2595	417	193	3	3	0	0	1	6
EthernetMAC10GE-T730	347	56	2595	417	193	3	3	0	0	1	6
RS232-T1000	11	2	2	0	12	0	0	0	0	2	2
RS232-T1100	11	2	2	0	12	0	0	0	0	2	2
RS232-T1200	10	2	2	0	12	0	0	0	0	2	2
RS232-T1300	9	2	2	0	12	0	0	0	0	2	2
RS232-T1400	12	3	2	0	12	0	0	0	0	3	2
RS232-T1500	11	2	2	0	12	0	0	0	0	2	2
RS232-T1600	9	2	2	0	12	0	0	0	0	2	2
s15850-T100	80	11	1	0	160	2	4	0	3	1	6
s35932-T100	3	3	1	0	420	2	2	0	4	1	8
s35932-T200	3	3	0	0	420	0	0	0	4	3	4
s35932-T300	3	3	1	0	420	0	0	0	4	1	5
s38417-T100	206	4	0	0	206	0	0	0	0	5	2
s38417-T200	203	3	0	0	206	0	0	0	0	4	2
s38417-T300	206	4	0	0	206	4	4	0	0	1	6
s38584-T100	112	3	2	0	404	0	0	0	4	2	3
s38584-T200	112	4	4	87	405	0	0	0	0	1	3
s38584-T300	123	16	3	783	405	0	0	0	0	1	3
vga_lcd-T100	19	0	1954	168	209	0	0	0	0	2	2
wb_conmax-T100	466	55	0	0	1516	0	0	1	0	1	4
b19	598	38	1108	16586	130	0	0	0	0	77	2
EthernetMAC10GE	344	55	2595	417	193	0	0	0	0	55	2
RS232	5	0	2	0	12	0	0	0	0	19	1
s15850	68	5	0	0	160	0	0	0	0	5	2
s35932	0	0	0	0	420	0	0	0	0	420	1
s38417	199	1	0	0	206	0	0	0	0	2	2
s38584	110	3	2	0	404	0	0	0	0	9	2
vga_lcd	16	0	1954	161	209	0	0	0	0	2340	1
wb_conmax	455	52	0	0	1516	0	0	0	0	48	2

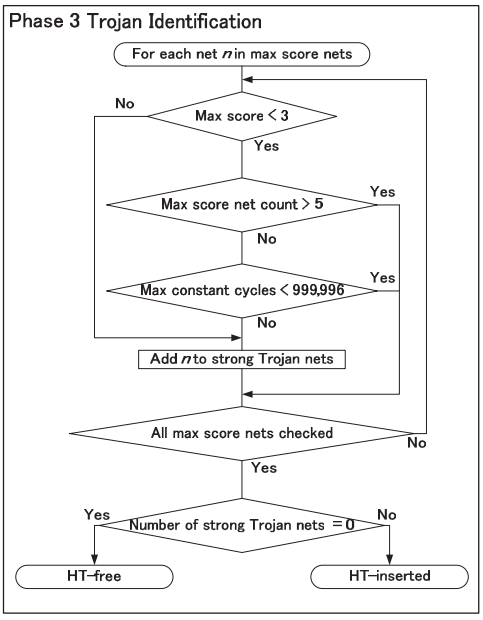


Fig. 4. Trojan identification flowchart.

TABLE VIII
HT-INSERTED NETLIST.

Benchmark	#nets
b19-T100	95,502
b19-T200	95,502
EthernetMAC10GE-T700	103,220
EthernetMAC10GE-T710	103,220
EthernetMAC10GE-T720	103,220
EthernetMAC10GE-T730	103,220
RS232-T1000	311
RS232-T1100	312
RS232-T1200	315
RS232-T1300	307
RS232-T1400	311
RS232-T1500	314
RS232-T1600	307
s15850-T100	2,456
s35932-T100	6,438
s35932-T200	6,435
s35932-T300	6,460
s38417-T100	5,819
s38417-T200	5,822
s38417-T300	5,851
s38584-T100	7,399
s38584-T200	7,580
s38584-T300	9,110
vga_lcd-T100	70,162
wb_conmax-T100	22,197

TABLE IX
HT-FREE NETLIST.

Benchmark	#nets
b19	108,332
EthernetMAC10GE	103,206
RS232	298
s15850	2,429
s35932	6,423
s38417	5,807
s38584	7,380
vga_lcd	70,157
wb_conmax	22,182

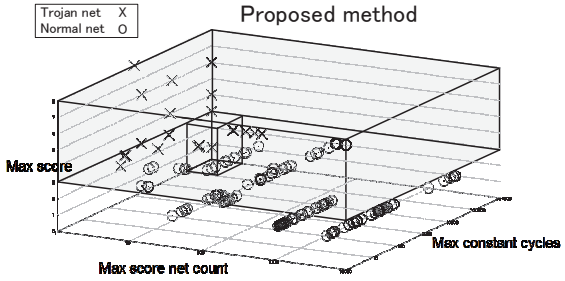


Fig. 5. 3D illustration of three Trojan factors.

B. Strong Classification Results

Strong classification gives three Trojan factors to max score nets. We used a logic simulator, Synopsys VCS, to obtain max constant cycles for every max score net. It took approximately two hours to simulate 1M cycles for each benchmark on Xeon E7-4870. Fig. 5 shows 3D illustration of three Trojan factors for max score nets in all the benchmarks. Since we know beforehand Trojan nets in Trust-HUB, X shows a Trojan net and O shows a normal, net i.e., X and O are the correct answers.

C. Trojan Identification Results

By giving the thresholds to three Trojan factors, our method can detect strong Trojan nets. Shaded rectangles in Fig. 5 demonstrate our results. Shaded rectangles only include the correct Trojan nets marked by 'X' and at least one Trojan net in each benchmark is detected by our method.

Table XI summarizes Trojan identification results. Our method successfully detects at least one strong Trojan net in every HT-inserted netlist (b19-T100–wb_conmax-T100). Then we can conclude

TABLE XI
IDENTIFY TROJAN INSERTED OR TROJAN FREE.

Benchmark	#strong Trojan nets	Contents of strong Trojan nets	Identify
b19-T100	1	Only Trojan net	Trojan inserted
b19-T200	1	Only Trojan net	Trojan inserted
EthernetMAC10GE-T700	1	Only Trojan net	Trojan inserted
EthernetMAC10GE-T710	1	Only Trojan net	Trojan inserted
EthernetMAC10GE-T720	1	Only Trojan net	Trojan inserted
EthernetMAC10GE-T730	1	Only Trojan net	Trojan inserted
RS232-T1000	2	Only Trojan nets	Trojan inserted
RS232-T1100	2	Only Trojan nets	Trojan inserted
RS232-T1200	1	Only Trojan net	Trojan inserted
RS232-T1300	1	Only Trojan net	Trojan inserted
RS232-T1400	2	Only Trojan nets	Trojan inserted
RS232-T1500	2	Only Trojan nets	Trojan inserted
RS232-T1600	1	Only Trojan net	Trojan inserted
s15850-T100	1	Only Trojan net	Trojan inserted
s35932-T100	1	Only Trojan net	Trojan inserted
s35932-T200	3	Only Trojan nets	Trojan inserted
s35932-T300	1	Only Trojan net	Trojan inserted
s38417-T100	3	Only Trojan nets	Trojan inserted
s38417-T200	2	Only Trojan nets	Trojan inserted
s38417-T300	1	Only Trojan net	Trojan inserted
s38584-T100	2	Only Trojan nets	Trojan inserted
s38584-T200	1	Only Trojan net	Trojan inserted
s38584-T300	1	Only Trojan net	Trojan inserted
vga_lcd-T100	2	Only Trojan nets	Trojan inserted
wb_conmax-T100	1	Only Trojan net	Trojan inserted
b19	0	N/A	Trojan free
EthernetMAC10GE	0	N/A	Trojan free
RS232	0	N/A	Trojan free
s15850	0	N/A	Trojan free
s35932	0	N/A	Trojan free
s38417	0	N/A	Trojan free
s38584	0	N/A	Trojan free
vga_lcd	0	N/A	Trojan free
wb_conmax	0	N/A	Trojan free

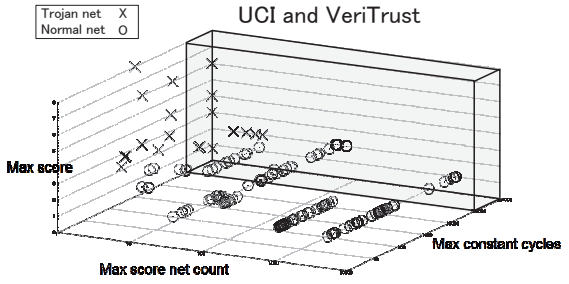


Fig. 6. Detection coverage of max score nets with using existing methods.

that these netlists are HT-inserted. Moreover, our method detects no strong Trojan nets in every HT-free netlist (b19–wb_conmax). Then we can conclude that these netlists are HT-free. "Contents of strong Trojan nets" in Table XI show whether strong Trojan nets include normal nets or not. All of our strong Trojan nets include only Trojan nets and hence Trojan identification actually identifies only HTs without false positives.

Overall, our method can successfully identify whether a given netlist is HT-free or not within three hours.

D. Proposed Method vs Existing Methods

Fig. 6 shows detection coverage of max score nets when we apply UCI [2] and VeriTrust [7] to them. UCI and VeriTrust are categorized into post-verification techniques and detect nets which are hardly activated to be Trojan nets. In other words, they detect nets which have extremely large Trojan factor 2. As in Fig. 6, the shaded rectangles include several normal nets marked as O and UCI and VeriTrust fail to detect Trojan nets.

Our simulation results show that there are Trojan nets activated frequency like normal nets. Trojan factor 2 of such Trojan nets becomes moderate and hence UCI and VeriTrust cannot detect HTs including such Trojan nets. Moreover, our simulation results show that there are normal nets activated hardly like Trojan nets in large-scale circuits. Trojan Factor 2 of such normal nets are extremely large and hence UCI and VeriTrust mistakenly detect such normal nets to be Trojan nets.

Table XII summarizes comparison of UCI, VeriTrust and our proposed method. The checkmarks for UCI and VeriTrust are derived from [7]. Our method successfully detects all the HTs but UCI and VeriTrust cannot detect HTs in several cases.

TABLE XII
COMPARISON OF HT DETECTION METHODS.

Benchmark	UCI [2]	VeriTrust [7]	Ours
b19-T100			✓
b19-T200			✓
EthernetMAC10GE-T700			✓
EthernetMAC10GE-T710			✓
EthernetMAC10GE-T720			✓
EthernetMAC10GE-T730			✓
RS232-T1000			✓
RS232-T1100			✓
RS232-T1200			✓
RS232-T1300			✓
RS232-T1400			✓
RS232-T1500			✓
RS232-T1600			✓
s15850-T100	✓	✓	✓
s35932-T100		✓	✓
s35932-T200	✓	✓	✓
s35932-T300		✓	✓
s38417-T100	✓	✓	✓
s38417-T200		✓	✓
s38417-T300	✓	✓	✓
s38584-T100			✓
s38584-T200		✓	✓
s38584-T300			✓
vga_lcd-T100			✓
wb_conmax-T100			✓

V. CONCLUSIONS

We have proposed a score-based classification method for identifying HT-free/HT-inserted netlists. Our method does not require Golden netlists nor HT pre-information but it successfully identifies whether a netlist is HT-free or not. Our method can detect all the HT-inserted gate-level benchmarks in Trust-HUB to be "HT-inserted" and all the HT-free gate-level benchmarks to be "HT-free."

Future work is to combine our score-based classification method with machine learning to have more robust HT detection.

ACKNOWLEDGEMENTS

This research was supported in part by Strategic Information and Communications R&D Promotion Programme (SCOPE) from the Ministry of Internal Affairs and Communications. The authors would like to thank Dr. Toshihiko Okamura, Dr. Ykiyasu Tsunoo, and Dr. Noritaka Yamashita of NEC and Prof. Satoshi Goto of Waseda University for their insightful comments.

REFERENCES

[1] B. Cha and S. K. Gupta, "Trojan detection via delay measurements: a new approach to select paths and vectors to maximize effectiveness and minimize cost," in *Proc. Design, Automation and Test in Europe (DATE)*, 2013, pp. 1265–1270

[2] M. Hicks, M. Finnicum, S. T. King, M. M. Martin and J. M. Smith, "Overcoming an untrusted computing base: detecting and removing malicious hardware automatically," in *Proc. Symposium on Security and Privacy (SP)*, 2010, pp. 159–172

[3] M. Potkonjak, A. Nahapetian, M. Nelson and T. Massey, "Hardware trojan horse detection using gate-level characterization," in *Proc. Design Automation Conference (DAC)*, 2009, pp. 688–693

[4] S. Wei, K. Li, F. Koushanfar and M. Potkonjak, "Hardware trojan horse benchmark via optimal creation and placement of malicious circuitry," in *Proc. Design Automation Conference (DAC)*, 2012, pp. 90–95

[5] F. Wolff, C. Papachristou, S. Bhunia and R. S. Chakraborty, "Towards trojan-free trusted ICs: problem analysis and detection scheme," in *Proc. Design, Automation and Test in Europe (DATE)*, 2008, pp. 1362–1365

[6] K. Xiao, X. Zhang and M. Tehranipoor, "A clock sweeping technique for detecting hardware trojans impacting circuits delay," *IEEE Magazines on Design and Test*, vol. 30, no. 2, pp. 26–34, 2012,

[7] J. Zhang, F. Yuan, L. Wei, Z. Sun and Q. Xu, "Veritrust: verification for hardware trust," in *Proc. Design Automation Conference (DAC)*, 2013, pp. 1–8

[8] "Trust-hub," http://www.trust-hub.org