EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

A security method of hardware Trojan detection using path tracking algorithm

Der-Chen Huang[1], Chun-Fang Hsiao[1], Tin-Wei Chang[1] and Ying-Yi Chu[2*]

*Correspondence: cyy@ncut.edu.tw

[1] Department of Computer Science and Engineering, National Chung Hsing University, Taichung, Taiwan
[2] Department of Electronic Engineering, National Chin-Yi University of Technology, Taichung, Taiwan

Abstract

Recently, the issue of malicious circuit alteration and attack draws more attention than ever before due to the globalization of IC design and manufacturing. Malicious circuits, also known as hardware Trojans, are found able to degrade the circuit performance or even leak confidential information, and accordingly it is definitely an issue of immediate concern to develop detection techniques against hardware Trojans. This paper presents a ring oscillator-based detection technique to improve the hardware Trojan detection performance. A circuit under test is divided into a great number of blocks, path assignment is optimized using a path tracking algorithm, and a high coverage is reached accordingly.

Keywords: Security, Hardware Trojans, Malicious activity, Attack detection, Algorithm

1 Introduction

Over recent years, integrated circuit (IC) operations have become more vulnerable to malicious attack than ever before as an effect of globalization on IC design and manufacturing services, and it is seen as an issue of immediate concern to develop techniques to detect maliciously inserted circuits. Currently, malicious circuits, more often referred to as hardware Trojans, can be inserted into a wide variety of commercially available chips or even devices, e.g., digital signal processors, microprocessors, network processors, microcontrollers, application-specific integrated circuits (ASICs), and many more. As a consequence, circuits may work in a way that is unexpected, and, more importantly, confidential information may be leaked accordingly. For this sake, as can be found in the literature, a continuous effort has been made to address the issue of hardware Trojan detection.

As many know, chip area, operation speed and power dissipation are the major concerns in IC design. In the early stage, chip area is the first priority concern for chip design. Subsequently, operation speed improvement turns into the most important issue, and, in recent times, the issue of power dissipation reduction draws the most attention due to power saving requirement particularly in Today's mobile devices. Low cost solutions must be found for IC designers, device manufacturers and wafer plants, due to the highly competitive market nowadays. One of the solutions is to reduce the cost using the intellectual property (IP) in the design and the manufacturing stages. Yet, there is a risk

that the designed circuit can be maliciously altered during a manufacturing process and work in an unintended way, giving rise to a security concern. Malicious circuit alterations may cause threats to the operations of household appliances, public transportation systems and financial infrastructures, or even put weapon systems in danger.

There are 3 stages involved in the development of ICs. Stage 1 is the design stage in which IPs, circuit models, design tools and designers are involved. Stage 2 is the fabrication stage involving masks, wafer and packaging plants. Stage 3 is the test stage. As many know, chip designers employ reliable computer-aided design (CAD) tools in the design stage of ASICs. Nonetheless, IPs, circuit models and standard components are seen as unreliable in the design, packaging and test stages. The fabrication stage is even unreliable, since hardware Trojans can be inserted into chips during a fabrication process. However, the fabrication and the test stages are exceptionally believed to be reliable in case they are done in semiconductor manufacturing companies or state-run institutes.

As explicitly stated in [1–3], there are mainly two approaches, i.e., IC certified protection and hardware Trojan detection, to ensure that chips are designed and manufactured in a trustworthy way. The former refers to a certified protection against malicious alteration of designed circuits, and is definitely a high-cost scheme since all the manufacturing processes must be certified. More importantly, IC manufacturing has been globalized in an attempt to keep the manufacturing cost down, making the scheme even impracticable. Hence, the latter has been developed as a "low-cost" alternative to the former, and is investigated herein. As its name indicates, hardware Trojan detection is a technique to see whether there is any inserted hardware Trojan to win clients' trust. Performance comparison between an originally designed circuit and a circuit under is conducted as a post-manufacturing detection. In recent times, the latter has been acknowledged as the mainstream detection technique against hardware Trojans, and a continuous effort has been made to improve the detection effectiveness against a wide variety of hardware Trojans.

There is a volume of publications on the precautionary measures against malicious circuit insertion and attack [4–7]. This paper presents a ring oscillator-based detection circuit, consisting of an odd number of inverters and working with an originally designed circuit. The detection circuit is configured in such a way that there is a frequency shift in the detection circuit once a hardware Trojan exists. In this paper, a combined use of a ring oscillator-based detection circuit and a path tracking algorithm is found to improve the wire and net coverages, meaning that the detection scope can be maximized.

This work is presented as five sections. Section 1 is the introduction, Sect. 2 describes hardware Trojan taxonomy and basics of Trojan detection. Section 3 gives configuration and characteristics of ring oscillators, and a detailed discussion on the presented path tracking algorithm for hardware Trojan detection and even protection against activation of malicious circuits. Section 4 presents simulation results, and Sect. 5 concludes this paper and suggests a future work.

2 Related works

This section firstly describes the taxonomy of hardware Trojans, then the various detection techniques for hardware Trojans.

2.1 Taxonomy of hardware Trojans

2.1.1 Modules

As indicated in [8–11], there are mainly two modules involved in hardware Trojans, that is, a trigger module and a payload module. The former can be triggered either internally or externally, and can remain no trigger. The latter works in the following ways: when the module is triggered, (1) messages are transmitted to hardware Trojan makers, (2) circuit operations are maliciously altered, and (3) the chip is sabotaged.

2.1.2 Types of hardware Trojans

Hardware Trojans can be classified according to the types of involved trigger and payload modules [12, 13]. There are two types of trigger modules, namely digital and analogue types. The digital module is further sorted into combinational and sequential types [14]. In the combinational case [12], a hardware Trojan is triggered on the condition that $A=0$ and $B=0$, and there is an error in node C of the payload module, as illustrated in Fig. 1a. In the sequential case, a hardware Trojan is also known as a time bomb and is triggered using a specific sequence or periodical signals. As reported in [8, 12, 15], sequential type can be further categorized into synchronous, asynchronous, mixed and rare sequential types. As in the trigger modules, payload modules can be further divided into digital and analogue types. In the digital case, logic operations, or memory contents, can be maliciously altered using the nodes in the payload module, while, in the analogue case, circuit performance, e.g., efficiency, power dissipation, noise tolerance, etc., can be degraded, using malicious alteration of parameters, say, extra path delay due to the introduction of a load capacitance, as illustrated in Fig. 1b.

2.2 Hardware Trojan detection solutions

Hardware Trojan detection techniques [12] are discussed here. A wide variety of detection techniques had been proposed, since there is no single one technique that can successfully detect all types of hardware Trojans.

2.2.1 Types of hardware Trojan detection approaches

Hardware Trojan detection approaches are mainly classified into non-destructive and destructive approaches, as illustrated in Fig. 2, and are detailed as follows.

1. Destructive detection: Metal layers are removed using chemical mechanical polishing (CMP) process, and then circuits are viewed using a scanning electron microscope (SEM).

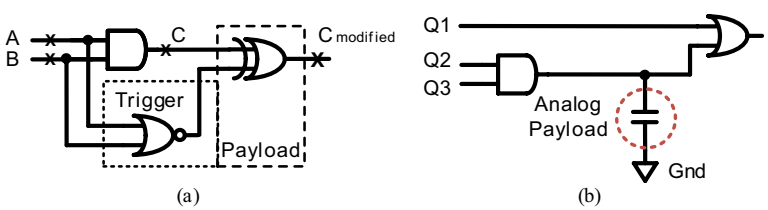


Fig. 1 Illustrations of hardware Trojans, **a** combinational case, **b** analogue case. Two examples of hardware Trojan. One is a combinational case, the other is an analogue case

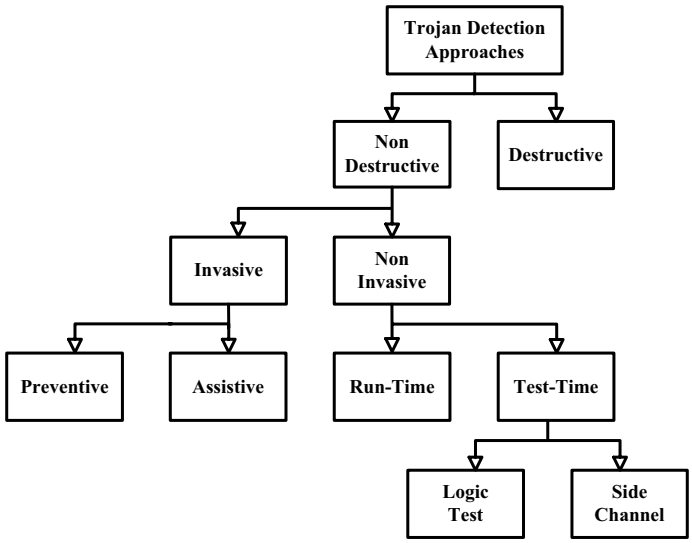


Fig. 2 A family tree of Trojan detection approaches. Hardware Trojan detection approaches can be mainly classified into non-destructive and destructive approaches

Table 1 Features of logic test detections

	Logic test detections
Advantages	Good detection performance for small-scale Trojans
	High resistance against noise in manufacturing
Disadvantages	High complexity in the production of detection information
	Poor applicability to large-scale Trojan detections

2. Nondestructive detection: It is further classified into invasive and non-invasive detections. The latter is done using a logic comparison between an originally designed IC and a one under test, and is further categorized into run-time and test-time detections. Test-time detections do not cause extra hardware cost, while a major disadvantage is that an originally designed IC is required for comparison purposes, and are further classified into logic test and side channel detections. In the former, signals are applied to the inputs of an IC under test, and then watch whether there is any unmatched output signal. If yes, the IC under test is very likely to have a hardware Trojan inside. In the latter, it is to investigate the changes of electric parameters due to inserted Trojans, e.g., transient current, power dissipation and path delay, as presented in [16]. Tables 1 and 2 give the advantages and disadvantages of logic test and side channel detections.
3. Design of hardware trust: It is an alternative way for hardware Trojan detection, and is done using built-in detection circuits of original designed ICs.

Table 2 Features of side channel detections

	Side channel detections
Advantages	Good detection performance for large-scale Trojan
	Low complexity in the production of detection information
Disadvantages	High susceptibility to noise in manufacturing
	Poor applicability to small-scale Trojan detections

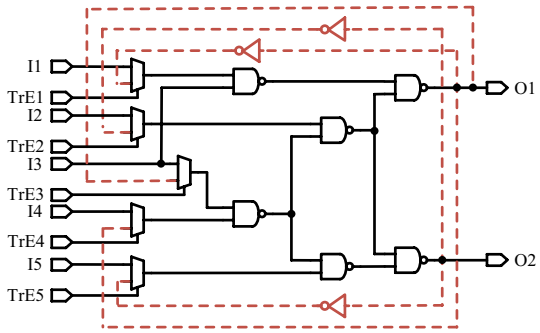


Fig. 3 A ring oscillator-based detection circuit. A ring oscillator is used to detect hardware Trojans once there is a frequency change

2.2.2 Hardware Trojan detections

According to the way Trojans are detected, side channel detections can be categorized into power-based [1, 3] and timing-based detections. The power-based detection is such a great detection technique that even non-trigger hardware Trojans can be detected. As presented in [1], hardware Trojans detection was done by a combined use of power dissipation and charge/discharge approaches. In contrast, transient power supply signals were adopted to test whether there existed any hardware Trojan [3] as a way to reduce leak current and influence on fabrication parameters. As put forward in [17], the timing-based detection is a technique that mainly measured the path delay between registers, and an IC under test is diagnosed as having Trojans if the path delay is found to be longer than a threshold.

Design for hardware trust is an alternative way of hardware Trojan detection. As referred to in [4], ring oscillators, consisting of an odd number of inverters, are introduced into an IC under test. Trojan is detected once there is a frequency change in a ring oscillator. Illustrated in Fig. 3 is an illustration of ring oscillators.

A transient-effect ring oscillator (TERO) is proposed in [30] and added to the original circuit for detection of hardware Trojan. The TERO is a non-destructive approach [31] and more sensitive to the changes of frequency, so it is efficient relative to conventional ring oscillators. Based on measuring the time delay, ring oscillators are inserted into a scan chain for hardware Trojan detection. Therefore, the design-for-testability (DFT)-based methods [32] are applied to find the inserted Trojans and an optimal trade-off between area overhead and security strength. As indicated in [33], the ring oscillators are also adopted to detect Trojan. This approach can avoid the effect of process variations and find the location of Trojan. For 3D integrated circuits, a 3D ring oscillator is presented in [34] to detect the

existence of Trojan by using delay measurements. At the same time, this approach can also be used to detect the additional delay due to process variations.

As a design for hardware trust, this work involves a combined use of the detection technique and the presented path tracking algorithm to form a ring oscillator-based detection circuit. In this manner, a Trojan can be detected before being triggered.

3 Proposed method

A ring oscillator-based detection circuit is employed herein to see whether there exists any hardware Trojans, and a path tracking algorithm and a multiplexer are presented as well. The detection performance is tested on the ISCAS85 c17 benchmark.

3.1 A combination of a ring oscillator

This work aims to present an effective hardware Trojan detection technique and to improve the wire and the net coverages using a combination of a ring oscillator, a 2-to-1 multiplexer and a path tracking algorithm. Illustrated in Fig. 4 is a ring oscillator in the ISCAS85 c17 benchmark.

A 2-to-1 multiplexer precedes a ring oscillator so as to choose a specific path for detection. With a 2-to-1 multiplexer as a controlled switch, a circuit under test can work in a way as expected, or ring oscillator can operate to detect hardware Trojans.

3.2 An alternative way for hardware Trojan detection

A hardware Trojan can be detected once there is a frequency shift in a ring oscillator. However, the frequency shift is susceptible to the way that the hardware Trojan is inserted.

Hardware Trojans are categorized according to the way they are inserted. An original detection circuit is illustrated in Fig. 5, where PATH represents a chosen path and D denotes an added circuit for Trojan detection with the frequency

$$F = \frac{1}{2 \times N \times D} \tag{1}$$

where N represents the number of inverters, and D represents the delay of a single inverter.

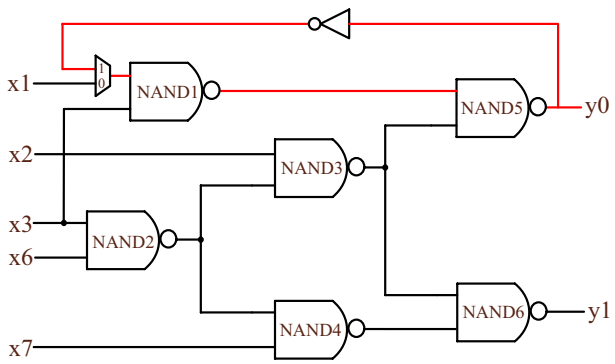


Fig. 4 A combination of the ISCAS85 c17 benchmark and a ring oscillator. A hardware Trojan detection example is illustrated for the ISCAS85 c17 benchmark circuit

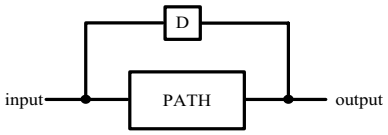


Fig. 5 An original detection circuit. A Trojan detection circuit is added to a chosen path to detect hardware Trojans

Hardware Trojans can be inserted at any position in a circuit. As categorized in Fig. 6a to 6d, it is important to understand the various ways that the Trojans are inserted. Therefore, a short circuit path is added to the original circuit to represent a frequency change in a ring oscillator. Based on the original detection circuit, Trojan detections are classified into two types according to the added path configuration. A path is on a single route in Type 1, while a path is across multiple routes in Type 2, and both types are illustrated as follows.

Path is on a single route, and Type 1 is further categorized into three modes below.

As illustrated in Fig. 6a, a path is inserted between the input and PATH. Taking into account the added path, the frequency of the detection circuit is modified as

$$F = \frac{1}{2 \times (D_{ip} + D_p + D_d)} \tag{2}$$

where D_{ip}, D_p and D_d represent the delay between the input and PATH, the delay of PATH and the delay of the added inverters, respectively.

As illustrated in Fig. 6b, a path is inserted between the output and PATH. Taking into account the added path, the frequency of the detection circuit is now modified as

$$F = \frac{1}{2 \times (D_{op} + D_p + D_d)} \tag{3}$$

where D_{op}, D_p and D_d represent the delay between the output and PATH, the delay of PATH and the delay of the added inverters, respectively.

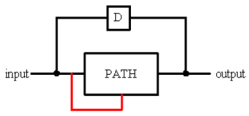
Mode 3 is further classified into five cases, and the frequency is now modified as

$$F = \frac{1}{2 \times (D_{ex} + D_p + D_d)} \tag{4}$$

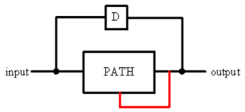
where D_{ex}, D_p and D_d represent the delay of the added path, the delay of PATH and the delay of the added inverters for detection.

As illustrated in Fig. 6c, a path is added inside PATH in case 1, is added outside PATH in case 2, is added within the detection circuit in case 3, is added between the detection circuit and input/output in case 4, and is added inside the detection circuit and PATH in case 5.

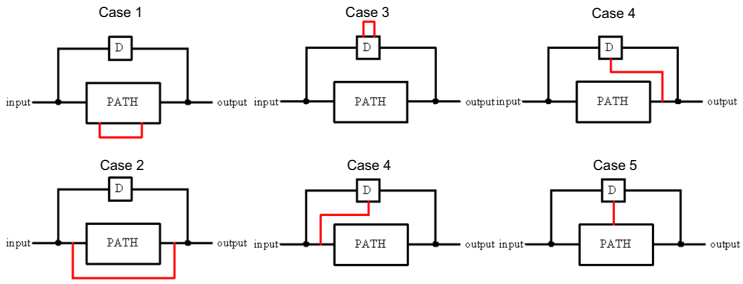
Not as in Type 1, a path is added between PATHs. As illustrated in Fig. 6d, Type 2 is further categorized into 8 cases, that is, a path is added between input 1 and input 2 in case 1, is added between output 1 and output 2 in case 2, is added between input 1 and output 2 in case 3, is added between PATH 1 and PATH 2 in case 4, is added between input 1 and PATH 2 in case 5, is added between output 1 and PATH 2 in case 6, is added



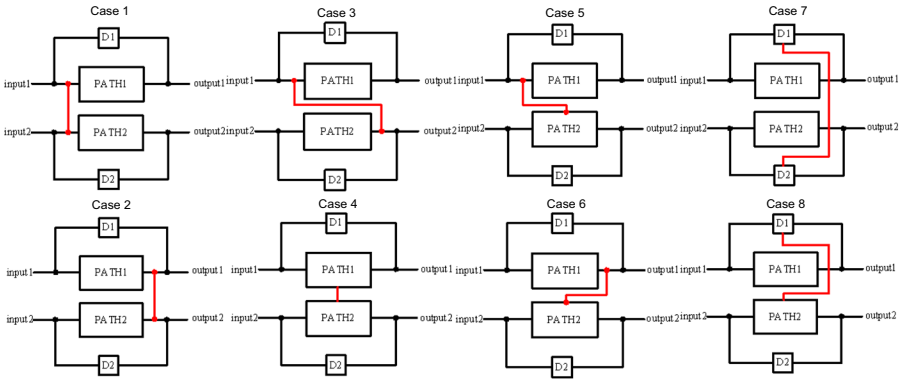
a Mode 1 Trojan path



b Mode 2 Trojan path



c Mode 3 Trojan path



d Type 2 detection cases

Fig. 6 **a** Mode 1 Trojan path, **b** Mode 2 Trojan path, **c** Mode 3 Trojan path, **d** Type 2 detection cases. Based on the original detection circuit, Trojan detections are classified into two types according to the added path configuration. A path is on a single route in Type 1, while a path is across multiple routes in Type 2. Path is on a single route, and Type 1 is further categorized into three modes

between the detection circuits D_1 and D_2 in case 7, and is added between D_1 and PATH 2 in case 8.

As explicitly stated previously, paths are added across PATHs, meaning that it is rather difficult to analyze the frequency of Type 2 cases. Besides, added paths in either type are found to demonstrate a strong effect on the frequency.

3.3 Detection circuits for series and parallel configurations

Detection circuits are categorized into two types according to the way they are configured, i.e., series, parallel and mixed configurations, and are detailed in turn as follows. Illustrated in Fig. 7 is the configuration of an original detection circuit.

Firstly illustrated is the series configuration, and is then the parallel configuration which is further categorized into two cases. A mixed configuration is finally described.

Series configuration is illustrated in Fig. 8, and all the PATHs are connected in series. A loop is formed by a detection circuit D_1 and the PATHs so as to detect a hardware Trojan.

Parallel configuration is further classified into two types according to the configuration of added paths. A path is added to a single PATH in Type 1 configuration, while is added between PATHs in Type 2 configuration.

It is assumed that a hardware Trojan is inserted into a single PATH, and a detection circuit is built across each PATH accordingly, as illustrated in Fig. 9a. However, Fig. 9a can be simplified into Fig. 9b, due to the fact that there is a frequency change once a hardware Trojan is inserted into an arbitrary PATH.

As illustrated in Fig. 10a, a hardware Trojan is inserted into a path, shown in blue, between PATHs. In this context, there is no way that the inserted Trojan can be detected using the configuration in Fig. 10a, but instead can be detected using the configuration in Fig. 10b where a detection circuit D_{ex}, shown in red, is inserted into the blue path.

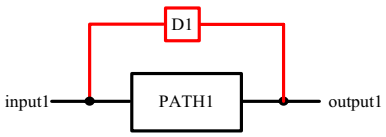


Fig. 7 A single detection circuit. Detection circuits are categorized into two types according to the way they are configured, i.e., series, parallel and mixed configurations

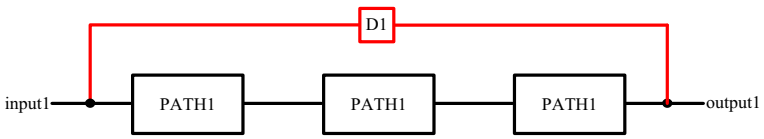


Fig. 8 A series detection circuit. All the PATHs are connected in series and a loop is formed by a detection circuit D1 to detect a hardware Trojan

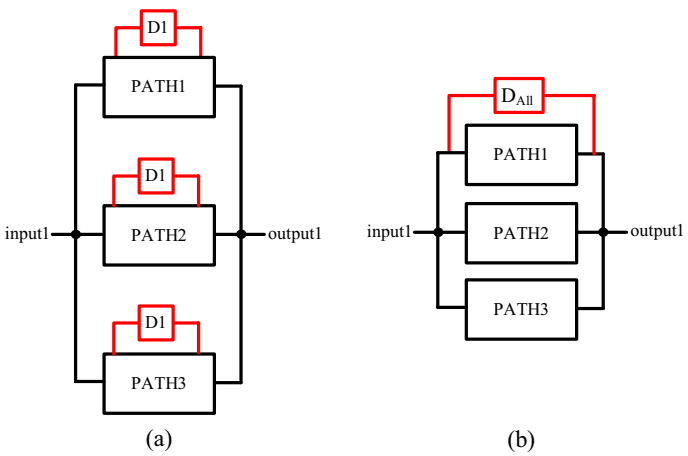


Fig. 9 Type 1 configuration, **a** a complicated detection circuit, **b** a simplified version of (**a**). In Type 1 parallel configuration, a path is added to a single PATH. A hardware Trojan is inserted into a single PATH, and a detection circuit is built across each PATH accordingly. Figure 9**b** is a simplified version of Fig. 9**a**

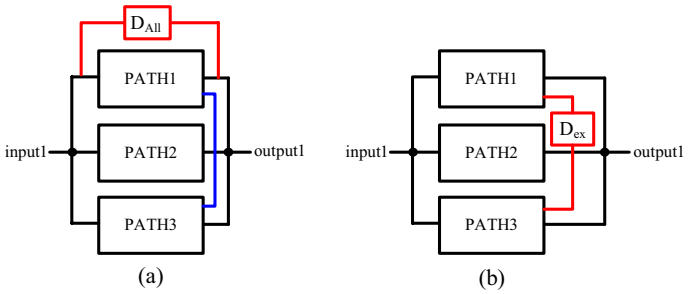


Fig. 10 Type 2 configuration, **a** a parallel detection circuit, **b** a modified version of (**a**). In Type 2 parallel configuration, a path is added between PATHs. A hardware Trojan is inserted into a path, shown in blue, between PATHs. Figure 10**b** is a modified version of Fig. 10**a**

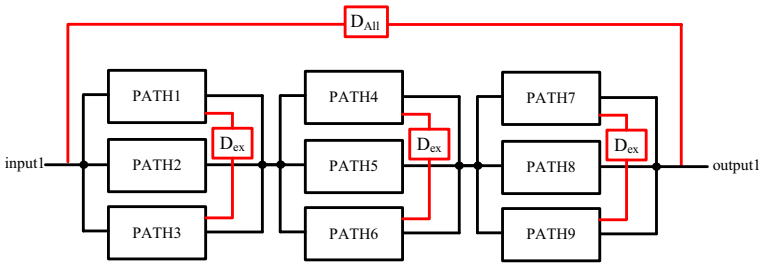


Fig. 11 A combination of parallel and series detection circuits. The mixed configuration is a mixture of series and parallel configurations to detect hardware Trojans

The mixed configuration is finally illustrated in Fig. 11, where a mixture of the above-stated series and parallel configurations is employed for hardware Trojan detection.

3.4 Algorithm

In a large-scale integrated circuit, there are tens of thousands of paths. Therefore, path selection is seen as critical for the circuit detection performance optimization.

3.4.1 A brief flowchart

As illustrated in Fig. 12, a Verilog file is first read for getting the configuration information of a circuit. Subsequently, matrices are built for path search using the Verilog file, and path match is performed to locate the optimal path and then to assign the input nodes to the chosen path. The above steps comprise the path tracking algorithm herein, while the rest is referred to as the multiplexer match algorithm aiming to improve the coverage in an attempt to reach a 100% detection.

As illustrated in Fig. 13, a Verilog file contains the information on each gate and its inputs. A total of 2 matrices are created using the Verilog file.

The first of the two matrices clearly specifies the configuration between nodes, according to which path tracking is performed until all the edges are detected. A logic 0 and a logic 1 represent the existence and non-existence of interconnection between nodes, respectively. The second matrix clearly specifies the configuration between nodes and inputs, due to which path match can be performed efficiently. A logic 0 and a logic 1 in

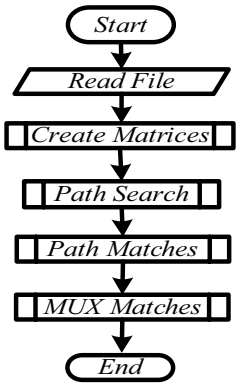


Fig. 12 Brief flowchart of a path tracking algorithm. A Verilog file is first read to build matrices of a circuit configuration. Subsequently, a path tracking and multiplexer match algorithms are presented to improve the detection coverage

```
Input    N1,N2,N3,N6,N7;

NAND2_1  (N10, N1, N3);
NAND2_2  (N11, N3, N6);
NAND2_3  (N16, N2, N11);
NAND2_4  (N19, N11, N7);
NAND2_5  (N22, N10, N16);
NAND2_6  (N23, N16, N19);
```

Fig. 13 Contents of a Verilog file. A Verilog file contains the circuit interconnection information

this matrix represent the same things as in the first matrix. The "path search" step, illustrated in Fig. 12, involves the first matrix.

3.4.2 Path tracking algorithm

Path tracking is performed as a prerequisite of the "path match" step, and is illustrated as follows.

Algorithm 1: path tracking algorithm

1. Create a matrix between nodes
2. Create a matrix between inputs and nodes
3. While (all edges are sought)
4. {
5. List the intersections of all the paths p_i and the edges e_i
6. Locate the path with the greatest number of intersections p_{max}
7. List the number of edge traversals
8. }
9. While (
10. ***All the edges are traversed*** or
11. ***All the inputs are involved*** or
12. ***Part of the edges cannot be traversed,***
13. ***using the remaining inputs***)
14. {
15. Select intersections with the largest number
16. For (i = 1; i <= n; i++)
17. If (p_i >= p_{max})
18. $p_i = p_{max}$;
19. end if
20. end for
21. Select the least number of edge traversals
22. Assign the chosen path to an input
23. Delete the chosen paths of the detection circuit
24. Delete the chosen edges of the detection circuit
25. Delete the chosen inputs of the detection circuit
26. }

Line 5 in algorithm 1 performs the intersection of a prebuilt matrix containing the information on the node connections and sought paths, as illustrated in Fig. 14, and then locates the path with the greatest number of intersections.

Line 7 indicates the number of edge traversals using the first matrix, as listed in Table 3.

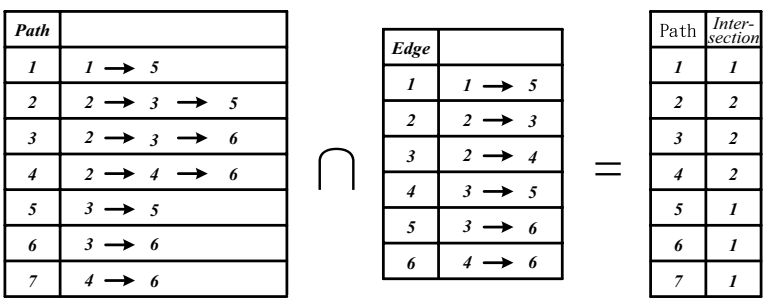


Fig. 14 Intersection of all the paths and the edges. This figure presents the information for the node connections and sought paths

Table 3 Number of edge traversals

Edge		Path Number
1	1→5	1
2	2→3	2
3	2→4	1
4	3→5	2
5	3→6	2
6	4→6	2

Subsequently, lines 15–21 perform the intersection of the node configurations and path, as illustrated in Fig. 14. Paths with the intersection containing the largest number of elements are chosen, as illustrated in Table 4.

Path tracking is optimized using Table 4, and is described as follows. Path with the least number of edge traversals is chosen for optimization, since it is likely to have an inadequate number of inputs. In this case, path 4 is chosen for the reason that merely edges 3 and 6 are traversed.

Line 22 assigns the chosen paths to an input using the second matrix, as shown in Table 5.

As illustrated in Table 4, node 2 is the first node in path 4, and a corresponding gate is chosen. In this case, two inputs are available for choice, either of which can be taken to form a path and a ring oscillator-based detection circuit accordingly.

Lines 23–25 delete part of the chosen inputs, edges and paths, constituting a detection circuit, to avoid repetition, meaning that an input is unlikely to be assigned to two different paths. For instance, deletion of path 4 and input 6 gives Tables 6, 7 and 8.

The path tracking algorithm terminates once the following conditions, i.e., those listed on line 9, are fulfilled.

Condition 1 All the edges are traversed, while a number of inputs may not get involved.

Table 4 Chosen paths

Path	Intersection	
2	2	2→3→5
3	2	2→3→6
4	2	2→4→6

Table 5 Connections between inputs and nodes

Input	Gate
1	1
2	3
3	1, 2
6	2
7	4

Table 6 Influence of deleting path 4

Path		Intersection
1	1→5	1
2	2→3→5	2
3	2→3→6	2
4	2→4→6	0
5	3→5	1
6	3→6	1
7	4→6	0

Table 7 Influence of deleting edge 3 and 6

Edge		Path Number
1	1→5	1
2	2→3	2
4	3→5	2
5	3→6	2

Table 8 Influence of deleting input 6

Input	Gate
1	1
2	3
3	1, 2
7	4

Condition 2 Contrary to condition 1, all the inputs are involved, while not all the edges are traversed.

Condition 3 Part of the edges cannot be traversed, using the remaining inputs.

Condition 1 is detailed as follows. It is that all the edges are covered by the paths, meaning that a hardware Trojan detection may not involve all the inputs. Condition 1 is further classified into two cases. In the first case, a Trojan is maliciously inserted right at the input of a circuit, and the introduction of a ring oscillator requires a multiplexer, leading to a rise in the cost. In contrast with the first case, there is no point to handle the second case.

Condition 2 is due to the shortage of inputs. It is that the inputs are all involved, while not all the edges are traversed, meaning that the path match can be performed by no means anymore. There are solutions to the hardware limitation, e.g., use of alternative multiplexers, while the issue is not addressed herein.

Condition 3, as opposed to condition 2, frequently occurs in large scale circuits, and is simply due to a shortage of edges.

The above steps are illustrated as a flowchart in Fig. 15, where path match mechanism is highlighted gray. To begin with, paths are chosen from those given in the "path

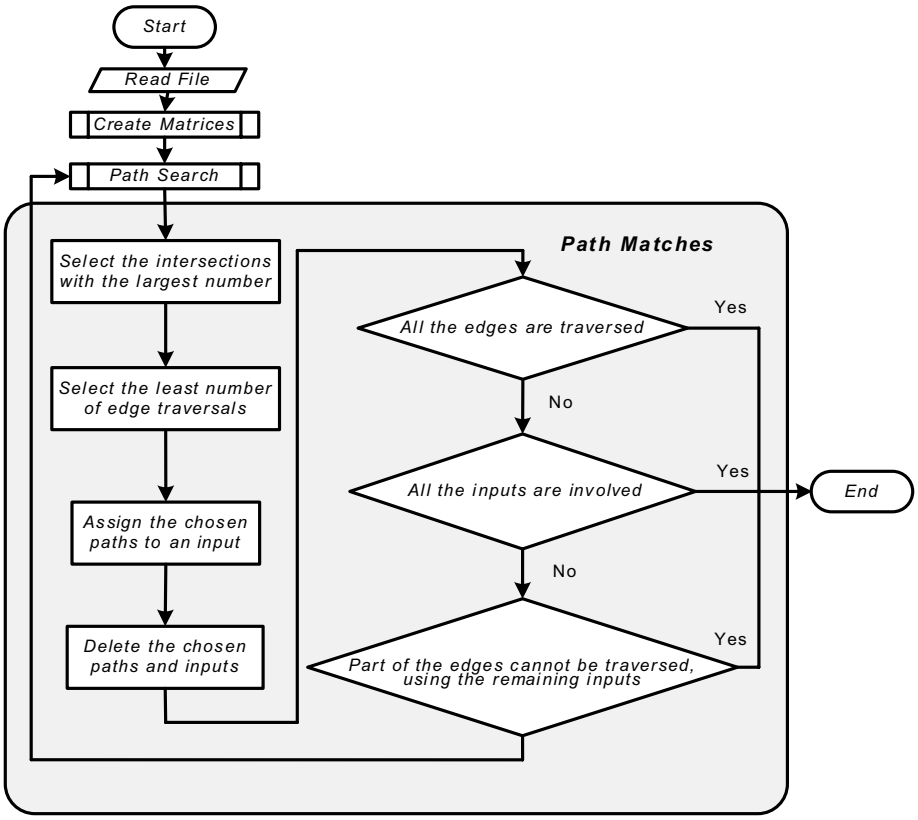


Fig. 15 A flowchart of the path matches. A path match mechanism is terminated if the three listed conditions are satisfied

search" step, and the output of the final step is determined as the path for building a ring oscillator.

3.4.3 Multiplexer match algorithm

An algorithm, designated as the multiplexer match algorithm and listed below, is developed to improve the coverage, since a 100% coverage cannot be reached by a single use of the above-stated path tracking algorithm, and the use of a multiplexer is found to be a key factor in the determination of the coverage.

Algorithm 2: *multiplexer match algorithm*

1~26. Algorithm 1: path tracking algorithm
27. UIN = the number of paths involving an input
28. {
29. $N = \lceil \log_2 UIN \rceil$
30. create 2^N to 1 MUX
31. create a 2 to 1 MUX for each unused inputs}

Line 29 gives the value of N as a function of the number of paths involving an input, and line 30 specifies the required multiplexer. For instance, an 8-to-1 multiplexer is required in the case of UIN = 7.

4 Experimental results

Computer simulations are conducted and performance is then validated using the ISCAS85 benchmark. The simulation results are presented as parts 1–4. Part 1 is the simulation results by the single use of the path tracking algorithm, part 2 is those by a combined use of the path tracking and the multiplexer match algorithms, part 3 is a performance comparison between parts 1 and 2, and finally part 4 gives the number and the types of the multiplexers employed and a delay comparison among benchmarks. Each case involves two quantities, the first of which is the wire coverage, defined as

$$\text{Wire Coverage} = \frac{\#\text{ of internal wires detected}}{\#\text{ of internal wires}} \tag{5}$$

where the numerator and the denominator represent the number of the internal wires detected and the total number of the internal wires, respectively. The second is the net coverage, defined as

$$\text{Net Coverage} = \frac{\#\text{of internal wires detected} + \#\text{of input wires detected}}{\#\text{of internal wires} + \#\text{of input wires}} \tag{6}$$

where the numerator and the denominator denote the number of the detected internal and input wires and the number of the internal and the input wires, respectively.

4.1 Single use of the path tracking algorithm

The path tracking algorithm aims to improve the wire coverage, while merely employing a 2-to-1 multiplexer. Comparisons on the wire and net coverages among benchmarks are, respectively, illustrated in Figs. 16 and 17.

4.2 Combined use of the path tracking and the multiplexer match algorithms

This combination aims to improve the wire and net coverages through a proper choice of multiplexers according to the number of paths involving an input. As in the

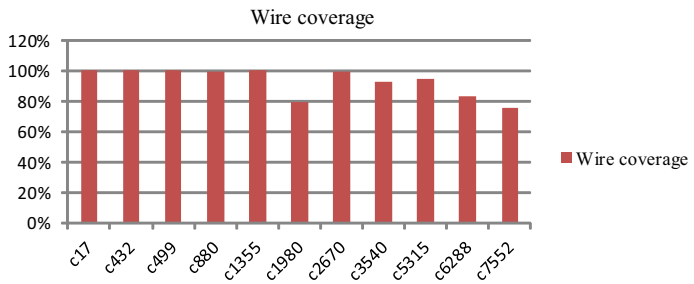


Fig. 16 Wire coverage comparison of the single algorithm among benchmarks. The path tracking algorithm aims to improve the wire coverage, while merely employing a 2-to-1 multiplexer. Comparisons for the wire coverage among benchmarks are illustrated in this Figure

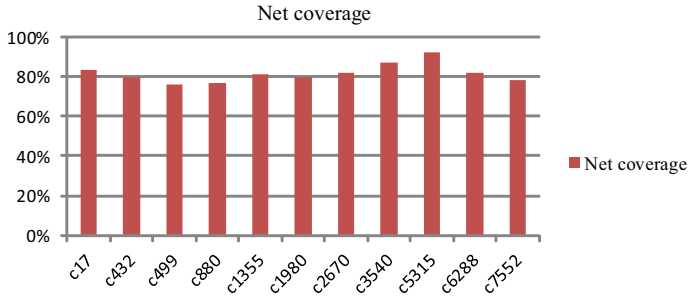


Fig. 17 Net coverage comparison of the single algorithm among benchmarks. The path tracking algorithm aims to improve the net coverage, while merely employs a 2-to-1 multiplexer. Comparisons for the net coverage among benchmarks are illustrated in this Figure

previous case, comparisons on the wire and net coverages among benchmarks are, respectively, illustrated in Figs. 18 and 19.

4.3 Performance comparison between both algorithms

Performance is compared between both algorithms in terms of the wire and the net coverages.

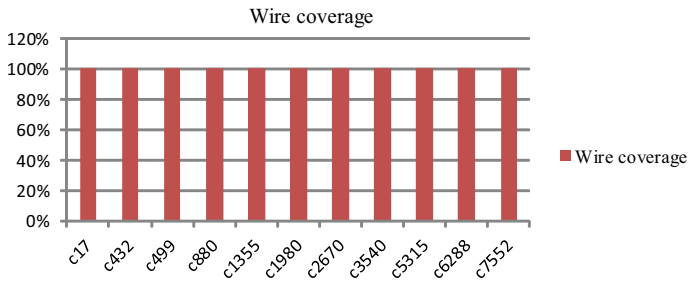


Fig. 18 Wire coverage comparison of the combined algorithms among benchmarks. A combined use of the path tracking and the multiplexer match algorithms improve the wire coverage through a proper choice of multiplexers

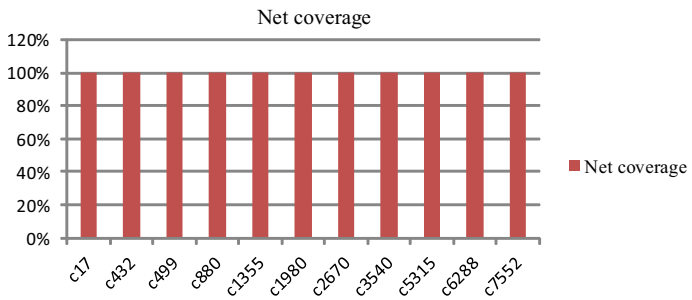


Fig. 19 Net coverage comparison of the combined algorithms among benchmarks. A combined use of the path tracking and the multiplexer match algorithms improve the net coverage through a proper choice of multiplexers

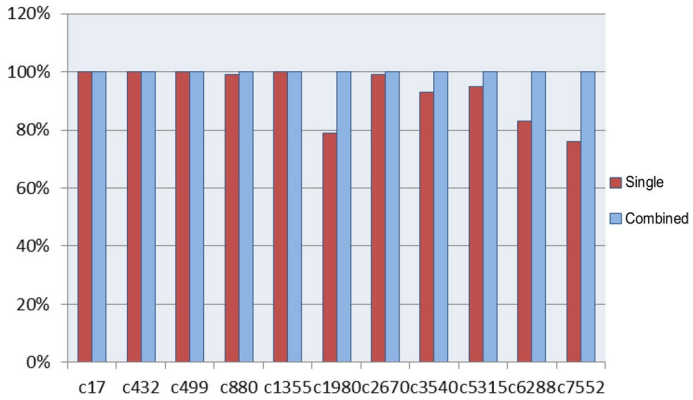


Fig. 20 Wire coverage comparison between both algorithms. By a combined use of the path tracking and multiplexer match algorithms, a 100% wire coverage can be reached in a large-scale benchmark

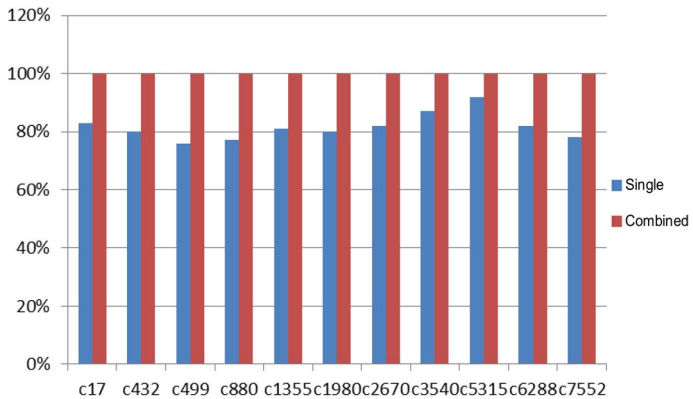


Fig. 21 Net coverage comparison between both algorithms. By a combined use of the path tracking and multiplexer match algorithms, a 100% net coverage can be reached in a large-scale benchmark

4.3.1 Wire coverage comparison

As illustrated in Fig. 20, a 100% wire coverage cannot be reached in a large-scale benchmark by a single use of the path tracking algorithm, while can be achieved in each benchmark by a combined use of the path tracking and multiplexer match algorithms.

4.3.2 Net coverage comparison

As illustrated in Fig. 21, a 100% net coverage cannot be reached in a large-scale benchmark by a single use of the path tracking algorithm, while can be achieved in each benchmark by a combined use of the path tracking and multiplexer match algorithms.

4.4 The number of multiplexers required and delay comparison among benchmarks

4.4.1 The number of multiplexers required between two coverages

By using a combination of the path tracking and multiplexer match algorithms, comparisons of the number of multiplexers required between the wire and net coverages are, respectively, illustrated in Figs. 22 and 23. It is clear that the number of

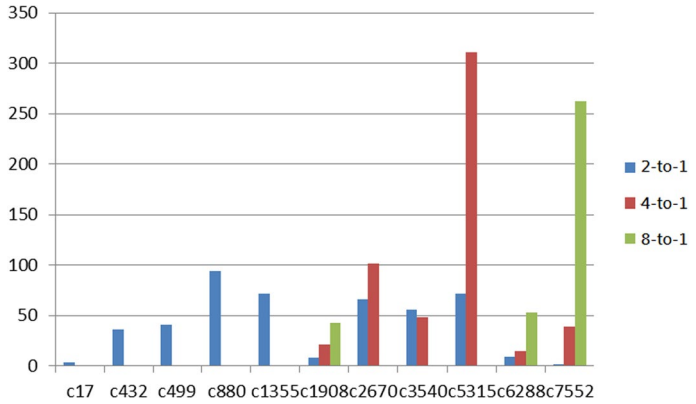


Fig. 22 Comparison of the number of multiplexers required among benchmarks for the wire coverage. By using a combination of the path tracking and multiplexer match algorithms, comparison of the number of multiplexers required for the wire coverage is illustrated in this Figure

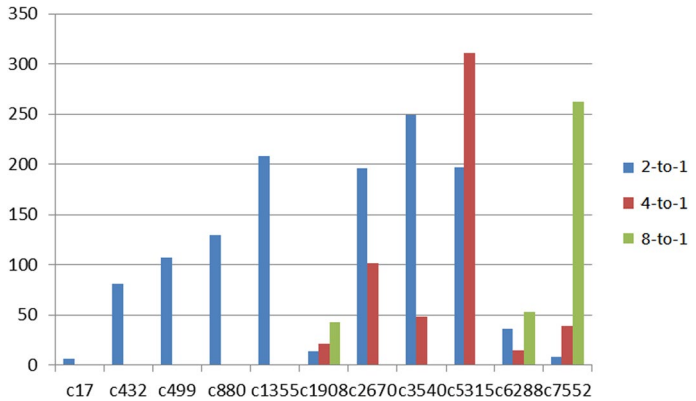


Fig. 23 Comparison of the number of multiplexers required among benchmarks for the net coverage. By using a combination of the path tracking and multiplexer match algorithms, comparison of the number of multiplexers required for the net coverage is illustrated in this Figure

multiplexers required for the net coverage is increased significantly, particularly a 2-to-1 multiplexer.

4.4.2 Delay comparison

To validate the performance of proposed algorithms, the delay is compared between two coverages. We assume that each multiplexer has one unit delay for simplicity. This approach makes this method can migrate to each different logic and process technology very easily such as static, dynamic and domino logics. The delay comparison between the wire and the net coverages is listed in Table 9 for each benchmark circuit.

5 Conclusions

This paper presents a ring oscillator-based technique to improve the wire and the net coverage. A circuit under test is divided into a great number of blocks, into each of which a ring oscillator introduced as a way to detect hardware Trojans. Furthermore, a path tracking algorithm is presented herein to optimize the path assignment. Accordingly, simulation

Table 9 Delay comparison between the wire and the net coverages

Benchmark	Wire coverage delay	Net coverage delay
c17	4	6
c432	36	81
c499	41	107
c880	95	130
c1355	72	208
c1908	72	78
c2670	168	298
c3540	104	297
c5315	383	508
c6288	77	104
c7552	303	309

results indicate a well-assigned path and a high coverage. The number of inputs is known to dominate the number of detection circuits, meaning that a proper choice of the number of inputs can improve the Trojan detection performance.

Abbreviations

IC	Integrated circuit
ASICs	Application-specific integrated circuits
IP	Intellectual property
CAD	Computer-aided design
CMP	Chemical mechanical polishing
SEM	Scanning electron microscope

Acknowledgements
The authors would like to thank the editors and reviewers for their valuable comments and suggestions.

Author contributions
All authors have contributed equally. All authors read and approved the final manuscript.

Funding
This research received no external funding.

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 3 June 2022 Accepted: 25 August 2022
Published online: 05 September 2022

References

1. M. Tehranipoor, F. Koushanfar, A survey of hardware Trojan taxonomy and detection. IEEE Des. Test Comput. **27**(1), 10–25 (2010)
2. H.E. Taheri, M. Mirhassani, A pre-activation, golden IC free, hardware Trojan detection approach. IEEE Trans. Very Large Scale Integr. Syst. **30**(3), 315–324 (2022)
3. R.M. Rad, X. Wang, M. Tehranipoor, J. Plusquellic, Power supply signal calibration techniques for improving detection resolution to hardware Trojans, in *2020 International Conference on Computer-Aided Design*, pp. 632–639 (2008)
4. J. Rajendran, V. Jyothi, O. Sinanoglu, R. Karri, Design and analysis of ring oscillator based design-for-trust technique, in *VLSI Test Symposium (VTS)*, pp. 105–110 (2011)

5. J. Rilling, D. Graziano, J. Hitchcock, T. Meyer, X. Wang et al., Circumventing a Ring Oscillator Approach to FPGA-Based Hardware Trojan Detection, in *IEEE International Conference on Computer Design (ICCD)*, pp. 289–292 (2011)
6. M. Tehranipoor, H. Salmani, X. Zhang, M. Wang, R. Karri, Trustworthy hardware Trojan detection and design-for-trust challenges. IEEE Trans. Comput. **44**(7), 66–74 (2011)
7. W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner et al., An overview of hardware security and trust: threats, countermeasures, and design tools. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **40**(6), 1010–1038 (2021)
8. D. Mukhopadhyay, R.S. Chakraborty, Testability of cryptographic hardware and detection of hardware Trojans, in *IEEE Asian Test Symposium (ATS'11)*, pp. 517–524 (2011)
9. F. Wolff, C. Papachristou, S. Bhunia, R.S. Chakraborty, Towards Trojan-free trusted ICs: problem analysis and detection scheme, in *Conference Design, Automation and Test in Europe (DATE'08)*, pp. 1362–1365 (2008)
10. L.-W. Wang, H.-W. Luo, A power analysis based approach to detect Trojan circuits, in *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE)*, pp. 380–384 (2011)
11. M. Elshamy, G.D. Natale, A. Sayed, A. Pavlidis, M.-M. Louërat et al., Digital-to-analog hardware Trojan attacks. IEEE Trans. Circuits Syst. I Regul. Pap. **69**(2), 573–586 (2022)
12. R.S. Chakraborty, S. Narasimhan, S. Bhunia, Hardware Trojan: threats and emerging solutions, in *IEEE International High Level Design Validation and Test Workshop (HLDVT'09)*, pp. 166–171 (2009)
13. A. Adamov, V. Hahanov, Security risks in hardware: implementation and detection problem, in *East-West Design and Test Symposium (EWDTS)*, pp. 425–427 (2010)
14. D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, B. Sunar, Trojan detection using IC fingerprinting, in *IEEE Symposium on Security and Privacy (SP'07)*, pp. 296–310 (2007)
15. H. Liu, H. Luo, L. Wang, Design of hardware Trojan horse based on counter, in *International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE)*, pp. 1007–1009 (2011)
16. Y. Jin, N. Kupp, Y. Makris, Experiences in hardware Trojan design and implementation, in *IEEE Int. Workshop Hardware-Oriented Security and Trust (HOST'09)*, pp. 50–57 (2009)
17. J. Li, J. Lach, At-speed delay characterization for IC authentication and Trojan horse detection, in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08)*, pp. 8–14 (2008)
18. R. Karri, J. Rajendran, K. Rosenfeld, M. Tehranipoor, Trustworthy hardware: identifying and classifying hardware Trojans. IEEE Trans. Comput. **43**(10), 39–46 (2010)
19. X. Wang, M. Tehranipoor, J. Plusquellic, Detecting malicious inclusions in secure hardware: challenges and solutions, in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08)*, pp. 15–19 (2008)
20. Zh. Chen, X. Guo, R. Nagesh, A. Reddy, M. Gora, A. Maiti, Hardware Trojan designs on BASYS FPGA board, in *Embedded System Challenge Contest in Cyber Security Awareness Week (CSAW)*, (2008)
21. Y. Jin, Y. Makris, Hardware Trojan detection using path delay fingerprint, in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08)*, pp. 51–57 (2008)
22. R.S. Chakraborty, S. Paul, S. Bhunia, On-demand transparency for improving hardware Trojan detectability, in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08)*, pp. 48–50 (2008)
23. M. Banga, M. Hsiao, A region based approach for the identification of hardware Trojans, in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08)*, pp. 40–47 (2008)
24. X. Wang, H. Salmani, M. Tehranipoor, J. Plusquellic, Hardware Trojan detection and isolation using current integration and localized current analysis, in *IEEE International Symposium on Defect and Fault Tolerance of VLSI Syst.*, pp. 87–95 (2008)
25. M. Potkonja, A. Nahapetian, M. Nelson, T. Massey, Hardware Trojan horse detection using gate-level characterization, in *IEEE Design Automation Conference*, pp. 688–693 (2009)
26. Y. Jin, Y. Makris, Hardware Trojans in wireless cryptographic ICs. IEEE Des. Test of Comput. **27**(1), 26–35 (2010)
27. Y. Alkabani, F. Koushanfar, Extended abstract designer's hardware Trojan horse, in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08)*, pp. 82–83 (2008)
28. R. Rad, J. Plusquellic, M. Tehranipoor, Sensitivity analysis to hardware Trojans using power supply transient signals, in *IEEE International Workshop on Hardware-Oriented Security and Trust (HOST'08)*, pp. 3–7 (2008)
29. Y. Alkabani,F. Koushanfar, Consistency-based characterization for IC Trojan detection, in *IEEE International Conference on Computing-Aided Design*, pp.123–127 (2009)
30. P. Kitsos, N. Sklavos, A.G. Voyiatzis, Ring oscillators and hardware Trojan detection. Hardware Security and Trust, Springer, 169–187 (2017)
31. D. S, R. S. R and N. D. M, Hardware Trojan detection using ring oscillator, in *International Conference on Communication and Electronics Systems (ICCES)*, pp. 362–368 (2021)
32. M.R.A. Kouhanjani, A.H. Jahangir, Improving hardware Trojan detection using scan chain based ring oscillators. Microprocess. Microsyst. **63**, 55–65 (2018)
33. O. Jin, L. Li, W.-T. Zhou, Hardware Trojan detection based on ring oscillator. Microelectron. Comput. **35**(11), 79–83 (2018)
34. S. Alhelaly, J. Dworak, K. Nepal, T. Manikas, P. Gui et al., 3D ring oscillator based test structures to detect a Trojan Die in a 3D die stack in the presence of process variations. IEEE Trans. Emerg. Top. Comput. **9**(2), 774–786 (2021)

Publisher's Note