Hardware Trojan detection and localization based on local detectors

Amin BAZZAZI1,*, Mohammad Taghi MANZURI SHALMANI2, Ali Mohammad Afshin HEMMATYAR2
1School of Science and Engineering, Sharif University of Technology, International Campus, Kish Island, Iran
2Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Received: 07.03.2017 • Accepted/Published Online: 18.09.2017 • Final Version: 30.05.2018

Abstract: Hardware Trojans are one of the serious threats with detrimental, irreparable effects on the functionality, security, and performance of digital integrated circuits. It is difficult to detect Trojans because of their diversity in size and performance. While the majority of current methods focus on Trojan detection during chip testing, run-time techniques can be employed to gain unique advantages. This paper proposes a method based on the online scalable detection technique, which eliminates the need for a reference chip. Involving local detectors, this technique assesses the variations in the logical values of each node to find out whether there are Trojans. This method excludes time and power measurements, which are common parameters in most conventional methods. The detectors provide Trojan-localization capability in our proposed technique. Two remarkable features of this technique are low power and low area overhead. The results are reported by simulation and implementation of common benchmarks, which show the high Trojan detection rate of the proposed method.

Key words: Hardware Trojans, run-time method, local detectors, detection, localization

1. Introduction

Hardware Trojans (HTs) pose a serious threat to the security and accuracy of electronic systems because they can manipulate the functionality of the system by adding or removing a component in the circuit. Trojans may activate in specific working or environmental circumstances, leaving destructive effects such as piracy and illegal data transfer [1–3]. Trojan detection takes place in 3 phases during the lifecycle of a chip: during design time, test time, and run time [4]. Design-time methods usually support the run-time methods. There are significantly fewer run-time methods than the others, even though they bring about the following great advantages [4]: first, if the Trojan is not active during testing, the test-time method fails in detection. Second, a chip whose Trojan has not been activated will function similar to an intact Trojan-free chip. Working perfectly and effectively, the run-time method can initially detect a Trojan, bypass it, and prevent it from penetrating into the entire chip. With respect to the characteristics of HTs and the problems found in the current detection methods, this paper intends to propose an alternative technique. The proposed method is run-time. HTs are identified by measuring the logical values of the circuit nodes. There are several outstanding advantages of the proposed method: 1- Trojan detection does not require a reference chip or Golden IC. 2- Unlike conventional methods, it does not require power and delay to be measured. 3- Process variations and noise do not affect this method. 4- It provides an online, fast, low-cost, and scalable solution. 5- It is capable of localizing Trojans. 6- The proposed method offers a high detection rate and low overhead. This paper is organized as follows: the next

*Correspondence: bazzazi@ce.sharif.edu
section reviews the relevant literature. The proposed method is described in Section 3. Section 4 provides the simulation and experimental results. Finally, the conclusions are given in Section 5.

2. Literature review
In general, Trojan detection methods can be categorized as side channel analysis, run-time, Trojan activation, and logical testing methods [2]. The proposed method falls under logical testing, which is run-time. Therefore, it is crucial to review similar studies on categorization of detection methods. A suspiciously malicious circuit activity can be revealed by monitoring the gates that have not been triggered by the circuit inputs. Known as VeriTrust, this method is implemented at design time [5]. This method is mainly characterized by its insensitivity to how Trojans are implemented. There is another method known as blue chip [6], in which a dynamic method has been developed through software–hardware integration. It can detect all Trojan-affected circuits through unused circuit identification techniques. In [7], a method was proposed to detect Trojan intrusion by extracting a Trojan netlist, without relying on a golden netlist. It provides a classification of Trojan-free and Trojan-affected netlists through scoring. In [8], Waksman checked the IP security with a tool known as FANCI. Involving static detection, this method is implemented at design time. Gates with a low probability of malicious activity are flagged as suspicious and Trojan-affected. There is another Trojan detection technique at IP level known as diversity-duplication [9]. The term ‘duplication’ refers to 2 copies of an IP (original and fake) applied in the SOC. This idea can offer more than 100% area and power consumption redundancy. In this regard, other studies focused on IP level, including [10]. There are additional methods to detect Trojans based on run-time techniques. For instance, detectors are used on the chip area. In such methods, several detectors are embedded in the circuit to measure a specific metric online. The results of measurements indicate whether or not the circuit has been exposed to a Trojan attack. Current, voltage, time, and temperature are common metrics that can be used as a scale of measurement and assessment for detectors. Although these metrics are to a large extent similar to side-channel techniques, the major difference is the absence of the Golden IC, which falls under an independent category. The following section briefly explores various types of sensors: 1) Sensors embedded in a circuit or chip that can measure current variations. These sensors were employed in [11]. In this method, Trojans are detected by measuring the local currents. 2) Embedding sensors in the circuit or chip capable of measuring time variations (delay). Such sensors were employed in [3]. In this type, certain paths of the circuit are selected and the respective delays are measured. 3) Embedding sensors in the circuit or chip capable of detecting Trojans by measuring temperature variations. This method was employed in [4]. 4) Another method revolves around the comparison of instantaneous voltage and current [12]; however, that study did not report the numerical result of success rate or overhead. There are other papers focusing on online Trojan detection, where the algorithms perform differently from that of the proposed technique. In our method, the detectors include logic gates distributed at the chip area. Logical detectors placed in the empty area of the chip are used to minimize the chances of Trojan involvement. In terms of type and diversity, these gates are quite similar to the main gates. In this scenario, if the gates are manipulated, variations can easily be discovered. This technique actually enhances the possibility of Trojan detection.

3. Proposed method
The proposed method has been implemented through the following algorithm: 1) The main circuit is divided into subcircuits (smaller segments) at the design stage. In each segment, subcircuits are built in smaller dimensions, thus leading to fewer inputs (leveling and segmentation stage in Figure 1). 2) A certain number of nodes are
selected in each segment. Moreover, the nodes are selected based on a specific idea: for example, the candidate nodes may offer low controllability and observability, or higher fan-out (selection of the desired nodes stage in Figure 1). 3) The inputs are specified in each subcircuit or segment and then the input vector is applied to the circuit. In an \( N \)-input subcircuit, \( 2^N \) vectors are applied. Every time the input vector is applied to the circuit, the candidate nodes assume logical values, which are extracted in the target nodes (forming groups containing the nodes stage and applying the input vector stages in Figure 1). 4) The values obtained for each node are arranged in 1 row of the matrix (creating a response matrix for the nodes stage in Figure 1). 5) The matrix rows indicate that nodes with maximum similarity are extracted. Given the similarity of nodes, a logical function is assigned to the candidate nodes. Meanwhile, the logical circuit corresponding to that function is implemented (extracting of a function from the matrix stage in Figure 1). 6) This function refers to the logical detector in the corresponding segment. In normal working mode, this function continuously assumes a specific value, which will not vary until the Trojan attacks the circuit function (inserting the function into the main circuit stage in Figure 1). 7) Logical detectors are created similarly for other segments. A circuit signature is defined according to detector values (defining the signature stage Figure 1). Figure 1 illustrates the proposed algorithm.

![Proposed algorithm diagram](image)

**Figure 1.** Proposed algorithm.

4. Simulation and experimental results

The simulation and implementation involves several benchmarks including ISCAS 85/89/99. There are different types of Trojans [13]. This study involves a Trojan affecting the function of the circuit. The Trojans are defined as follows. Type 1: The Trojan modifies the circuit gates, altering the circuit’s function. Type 2: The circuit paths are shortened and the gates are bypassed by the Trojans. Type 3: Trojans attack when adding 1 input to the current gate of the circuit. Type 4: A set of gates are embedded within the circuit. They are initially inactive, but are then activated after a while, penetrating the circuit and altering its function.
4.1. Input selection and segmentation

The circuits come in varying sizes and numbers of input ports. Segmentation tends to be more important in larger circuits. There are two noteworthy considerations: how to apply logical waveforms to the input ports, and how to specify the circuit segments and how to subject them to the input waveform. The first problem depends on the number of input ports in the circuit. If \( N \) represents the number of inputs, then there will be \( 2^N \) modes possible for the exhaustive test vector. A larger \( N \) involves larger test vectors, where the implementation of it on the circuit may be impossible. Therefore, the number of input ports should be split up so that an exhaustive test vector can be applied. The selection threshold for the allowable number of inputs has been assumed to be 12 in such a way that the total number of modes would not surpass 4096. The selection threshold implies that the maximum number of inputs for 1 segment can be 12, while any lower number will be fine. Basically, \( N \) in each segment should not be too large, because an excessive number of modes will demand time-consuming analysis. On the other hand, \( N \) should not be too small because it may escalate the number of segments. Since the circuits have varying numbers of input ports, there may be different modes as discussed in the following (\( N \) number of input ports): A) \( N < 12 \): the input vector can be applied directly to all ports. For example, the exhaustive test vector is applied to 128 modes when there are 7 input ports, while the circuit has 1 input segment. B) \( N > 12 \) (nearly 12): it can be divided into 2 groups since the number of modes for the exhaustive test vectors exceeds the threshold value. For example, 2 groups of 7 and 8 are suggested for 15 input ports. Nevertheless, this categorization is optional, since 5 and 10 or 9 and 6 or any other reasonable combination can be selected. C) \( N > 12 \) (not nearly 12): grouping into a larger number of segments is recommended since the number of modes for the exhaustive test vector is large. For example, 12, 12, 10, and 6 inputs can be designated for the 40-input mode. In this scenario, 4 input segments are created. As can be seen, there is a wide variety of options available to the user according to preferences. However, it is crucial to strike a balance among the number of segments, size of segments, and user-preferred detection accuracy. When the test vector is applied to the inputs of a segment, certain nodes of that segment will be triggered by the inputs and assumed logical values. Having passed through gates, the signals are propagated across a part of the circuit. The ideal nodes for selection of the next (middle) input segments of the circuit are those nodes where the propagation from the previous segment is finished. However, it is acceptable to initiate the new segments slightly ahead or behind, rather than exactly on the propagation end-nodes. This shift will affect the accuracy of detection and overall circuit overhead. Moreover, it outlines the boundaries of the zone in a circuit where the middle segments and their input nodes should be embedded. In this section of the circuit, new input ports are virtually defined so that new input vectors can be applied. The number of inputs in this section of the circuit is similar to the previous segments following the same methodology. The input applied to new input ports will lead to further signal propagation to the end of the circuit. The end-nodes of this segment can be used to define and select new segments. This progresses in the circuit until it reaches the output nodes. If the segment end-nodes are the same as circuit output ports, the segmentation is completed.

4.2. Matrix extraction

As previously mentioned, when the input is applied, the signals are transmitted in the circuit and trigger certain nodes, assuming logical values. If a node has value(s) other than 0 and 1, it would probably not be selected. All triggered nodes with desirable logical values are selected and their values are inserted into a matrix. Each matrix row represents a node, and each column indicates the logical value of that node. For instance, if a segment contains 6 inputs, there will be 64 modes in its test vector. If there are 30 nodes triggered by these vectors, then the matrix will be \( 30 \times 64 \). The circuit contains different segments whose numbers of inputs and triggered nodes vary. In fact, there are matrices with varying dimensions to be analyzed.
4.3. Extraction of the desired function

Before this step, a matrix has been constructed. Each row indicates the logical value of a node constructed by application of different inputs. Meanwhile, the rows are compared to each other to extract a logical relationship and function, based on similarity between the values of rows. It is critical to consider the accuracy and efficiency of relationships. The relationship between 2 rows barely implies desirable applicability. It is desirable that the logical values of 2 different rows be independent. It is more desirable to select 2 different paths in a segment, because the path variation in 1 of them, due to presence of Trojans, can be detected by the other one. The Trojan detection in this procedure is correlated with identification of variations. If the selected nodes are all altered exactly in the same way, the difference will not be detected and the alteration would never be noticed. In this scenario, it suffices to select nodes with minimal interdependence and define the function by adopting a greater number of nodes. For this end, a program is written to compare the matrix rows and demonstrate the similarity percentages, i.e. the position of each selected node. When the matrix rows are compared, different modes may occur: A) Two selected nodes are identical: in this case, 2 identical rows are extracted from the matrix, as shown in Figure 2a. The XOR function can be employed. The XOR value for 2 points is always 0. In this case, the similarity is 100%. B) Two selected nodes are nonidentical: in this case, 2 rows are logically opposite to one another. The inequality of the 2 rows has been shown in Figure 2b. In this scenario, the XOR function can be employed. The XOR value for 2 points is always 1. In this case, the similarity is 100%. The above 2 modes (A and B) are optimum, because they can extract accurate relationships between the nodes. C) Two selected nodes are neither identical nor nonidentical: in this case, the equation between 2 rows will be converted to a function. Nonetheless, this function performs correctly in most cases but not all. For instance, Figure 2c shows the status of 2 nodes. These nodes are identical in 12 out of 16 possible modes. Hence, the output of an XOR gate displays the equality correctly in at 75% of cases with a constant value of zero. The accuracy of this equation between the 2 nodes is therefore 75%. Although this value is always desired to be 100%, there might be cases other than 100%. In all of the above cases (A, B, and C), the function’s output is constant, i.e. invariably 0 or 1 for the user. Any variation in the logical value of the output will alter the function output, which in turn leads to Trojan detection.

A: 1111001010110001
B: 1111001010110001
(a) Two identical rows.

A: 1111001010110001
B: 0000110101001110
(b) Two opposite rows.

A: 11110101111000110
B: 1111010111111010
0000101001000001
(c) Non-relational rows.

Figure 2. Details of two rows of the matrix.

4.4. Implementation of the function

Evidently, the relationships in a segment differ from those in the next segment, which in turn diversifies the functions, i.e. not all extracted functions are necessarily similar. New functions are initially added as a code (vhdl or Verilog, in this case) and later to the main circuit at gate level. This is completed during the design stage, prior to circuit manufacturing. In this procedure, new functions are added to check the main circuit. These checking circuits are basically logical detectors locally embedded in different non-preset components of the circuit. Given that the outputs of new functions are invariably assigned to be either 0 or 1, each segment
can be associated with a specific output. It is as though each segment leaves its own signature. For example, a 10-segment circuit with each containing a checking function will assume 10 specific binary values. This circuit can be argued to have a unique 10-bit signature, which can be considered as a 10-bit register checked by the user during the circuit run-time. Needless to say, this signature may vary from 1 circuit to another, depending on the definitions of detectors in each circuit. Moreover, every signature contains a different number of bits. The signature of a circuit, however, will remain identical in its normal operation, while it changes in the presence of a Trojan in the circuit. Figure 3 displays the main circuit, divided into \( N \) segments, i.e. \( N \) self-examiner circuits (detectors) are required to detect the HT involvement.

![Diagram](image)

**Figure 3.** The deployment of proposed HT detectors with N detectors.

### 4.5. Simulation

Benchmark C499 is selected to demonstrate the simulation details. The algorithm proposed in the previous section operates as follows: 1) This circuit entails 41 inputs divided through segmentation into 5 subcircuits with fewer inputs. The desired functions cannot be extracted without segmentation, since the input vector has \( 2^{41} \) modes, which is impossible to be implemented. The inputs to the first 4 segments are the same as those of the main circuit. Inputs 1 to 10 are assigned for the 1st segment, inputs 11 to 20 to the 2nd segment, inputs 21
to 30 to the 3rd segment, and inputs 31 to 41 to the 4th one. Embedded in the middle of the main circuit, the 5th segment is selected independently of the main circuit inputs. Eight inputs are selected for the 5th segment. It can be argued that a total of 8 to 11 inputs are assigned to the 5 segments, leading to far fewer modes than 41 inputs. The five circuit benchmarks are marked in five different colors in Figure 4. The color representation helps to better specify the coverage of circuit segments across the whole circuit area. 2) When the inputs of each segment are stimulated, a number of nodes will assume acceptable, logical values in that segment. At this stage, the nodes are specified in the segment. For instance, the 1st to 5th segments will assume logical values with 12, 9, 9, 9, and 90 nodes, respectively. 3) The nodes are grouped according to the number of acceptable nodes in the previous section. All nodes are selected as 1 group because there are only a few favorable nodes in each group. This can yield more accurate results. 4) The inputs are applied to each segment in addition to matrix forms for the nodes within each group. 5) The responses are inserted in the matrix assigned to each segment. The 5th segment, the largest in the circuit, has 8 input nodes. These 8 nodes are selected because of their higher fan-out. When the input is applied to this segment, the 8 input nodes will be deemed independent of the main inputs and the connected nodes, on which all testing vector modes will be applied. Offering high output fan-out, these nodes can stimulate 90 nodes. Therefore, the 5th segment with $90 \times 256$ matrix specifies the function. 6) Based on our proposed method, a logic circuit is assigned to each matrix. The circuit defined in each segment is implemented in the same zone. Hence, five separate circuits should be defined in the original benchmark. In Figure 5, each circuit is included in the main benchmark with a distinct color to emphasis the differences between them. The outputs of the new function are constantly fixed in the Trojan-free mode. Hence, the outputs of the 5 circuits can be used as a signature to verify the circuit. As the detectors are embedded in the main circuit, the chip area and power consumption begin to vary. Table 1 displays the values before and after the involvement of detectors in 0.18-μm technology.

**Table 1.** Area and power overhead in benchmarks.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Area (μm²)</th>
<th>Power (μW)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Main</td>
<td>Modified</td>
</tr>
<tr>
<td>S298</td>
<td>1490.22</td>
<td>1497.7</td>
</tr>
<tr>
<td>C499</td>
<td>3525.98</td>
<td>3702.5</td>
</tr>
<tr>
<td>C6288</td>
<td>17,550</td>
<td>18,778.5</td>
</tr>
<tr>
<td>S38417</td>
<td>87,085</td>
<td>95,793.5</td>
</tr>
</tbody>
</table>

### 4.6. The number of segments

As shown in segmentation of C499 (Figure 4) there is a scenario where all gates may not fall in the circuit segmentation. The question at this point is whether or not the engagement of more segments will enhance the detection accuracy. This is evaluated for C499 in 2 scenarios. The 1st scenario involves 5 segments. The next scenario involves a 10 segments. Figure 6 shows the extra segmentations. The involvement of more segments will include the gates not segmented in the 1st stage. On the other hand, increasing the number of segments puts some gates in more than 1 segment. The number of detectors doubles as the number of segments doubles. By the same token, the Trojan detection rate is expected to double. Simulations suggest an increase in the Trojan detection rate, even though it is less than the expected rate. This can be explained by dependence and sharing between segments. Among all segments added, there are a few segments functioning completely independently of the others. They are called effective segments, where the embedded detectors cover a large
area of the circuit. The presence of effective segments can strengthen the detection capability. Hence, the performance of this method is not enhanced necessarily by embedding a greater number of segments. This will not only improve performance but will escalate area and power overhead.

4.7. Trojan detection accuracy

Having been designed, the detectors need to be assessed in terms of Trojan detection accuracy. For that purpose, it is essential to calculate the success threshold. The success threshold implies the number of variations in the main circuit required by the detector to detect Trojans in a segment. For instance, the variations within a segment might be detected on 1 gate in the 1st type of Trojans, where the number of gates changes randomly. The variations in the next 2 segments will be detected for 3 and 10 gates, respectively. This number varies for each segment of the circuit, which is true for other types of Trojans. The proposed equation (Eq. (1)) for detection accuracy in the new technique is as follows:

\[ P = (P_{\text{Segment}})(P_{\text{Matrix}})(P_{\text{Function}}) \]  

Here \( P_{\text{Segment}} \) is the ratio of the number of gates embedded in the circuit segmentation to the total number of gates. \( P_{\text{Matrix}} \) represents the probability of obtaining the desired function according to the matrix extracted in each segment. \( P_{\text{Function}} \): Four types of Trojans are included in the simulation. A definition is given to each of the 4 types of Trojans below. Then \( P_{\text{Function}} \) is obtained through a new equation. \( P_{T1} \): The detection
The probability of Trojan type 1 is resulted by $M$ times of simulation obtained from Eq. (2). $P_k(n)$ represents the detection probability when alteration is found in $n$ gates of the circuit.

$$P_{T1} = \frac{1}{M} \sum_{k=1}^{M} P_k(n)$$  \hspace{1cm} (2)

$P_{T2}$: The detection probability of Trojan type 2 is resulted by $M$ times of simulation obtained from Eq. (3). $P_k(s)$ is the Trojan detection probability where $s$ gates have become short-circuited.

$$P_{T2} = \frac{1}{M} \sum_{k=1}^{M} P_k(s)$$  \hspace{1cm} (3)

$P_{T3}$: The detection probability of Trojan type 3 is resulted by $M$ times of simulation obtained from Eq. (4). $P_k(r)$ represents the detection probability when alteration is found in the number of inputs in $r$ gates of the circuit.

$$P_{T3} = \frac{1}{M} \sum_{k=1}^{M} P_k(r)$$  \hspace{1cm} (4)
Figure 6. Extra segments in C499.

circuit.

\[ P_{T_3} = \frac{1}{M} \sum_{k=1}^{M} P_k(r) \]  \hspace{1cm} (4)

\( P_{T_4} \): The detection probability of Trojan type 4 is resulted by \( M \) times of simulation obtained from Eq. (5). \( P_k(t) \) represents the detection probability in a circuit to which \( t \) gates have been added within a given time.

\[ P_{T_4} = \frac{1}{M} \sum_{k=1}^{M} P_k(t) \]  \hspace{1cm} (5)

After calculating \( P_{T_1} \), \( P_{T_2} \), \( P_{T_3} \), and \( P_{T_4} \), the minimum value is selected as \( P_{\text{Function}} \) based on Eq. (6).

\[ P_{\text{Function}} = \min (P_{T_j}) \ (j = 1, 2, 3, 4) \]  \hspace{1cm} (6)

The algorithm for the above calculation is shown in Figure 7. Prior to reading the success rate, the values of \( n, s, r \), and \( t \) in the circuit should be obtained. Several tests are conducted on different segments of benchmark C499 and values of \( n = 5 \), \( s = 3 \), \( r = 3 \), and \( t = 5 \) are introduced as the success threshold. The diagrams in Figure 8 display the success rate of detecting various Trojans. Having calculated the success rates in different states, the minimum value in each segment will be selected.

4.8. Results

Considering the number of effective segments and the optimized design of detectors, a perfect simulation is conducted on the 5 benchmarks. Table 2 displays the results for 180-nm technology along with full specifications.
In conclusion, it can be argued that the success rate of the method is 80% in the worst case. Assuming the detection percentage is a weighted average, however, the detection capability will exceed 90%. The proposed method has been simulated by Mentor ModelSim and Synopsys Design Compiler, using 45-, 180-, and 250-nm technologies. The accuracy of these simulations demonstrates the scalability of the proposed method. This also includes the calculations of power and area. Simulations are then conducted by applying noise and process variations. The accuracy at this stage proves that the method is not affected by noise and PV. The hardware verification is performed on a FPGA-platform, where FPGA is employed to simulate ASIC scenarios. The advantage of this method is that Trojans with different sizes and types are checked through different benchmarks. The selected FPGA is by XILINX-SPARTAN6. The results were seen using a 100-MHz oscilloscope. Moreover, we used a LED to show the presence/absence of Trojan in the test circuit, as shown in Figure 9. Certain ports were assigned to the FPGA for sensors embedded in the circuit so that Trojan detection and localization could be completed through separate examination of each port.

### Table 2. Results of simulations.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th># of gates</th>
<th># of FFs</th>
<th># of inputs</th>
<th># of outputs</th>
<th>Power overhead</th>
<th>Area overhead</th>
<th>Probability (detection rate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S298</td>
<td>119</td>
<td>14</td>
<td>3</td>
<td>6</td>
<td>0.5%</td>
<td>0.5%</td>
<td>100%</td>
</tr>
<tr>
<td>C499</td>
<td>202</td>
<td>-</td>
<td>41</td>
<td>32</td>
<td>2%</td>
<td>5%</td>
<td>100%</td>
</tr>
<tr>
<td>C6288</td>
<td>2416</td>
<td>-</td>
<td>32</td>
<td>32</td>
<td>5%</td>
<td>7%</td>
<td>85%</td>
</tr>
<tr>
<td>S38417</td>
<td>22179</td>
<td>1636</td>
<td>28</td>
<td>106</td>
<td>8%</td>
<td>10%</td>
<td>80%</td>
</tr>
</tbody>
</table>

#: Number, FFs: flipflops.

### 5. Discussion

#### 5.1. A few features of the new technique

This section explores several features of the new technique: 1) Even when a Trojan is active in the circuit, it is crucial to detect it online as well. Effort should be made to avoid a Trojan activation or neutralize it through...
Trojan activation in a circuit is similar to unusual operations in an ALU, such as division by zero or when 2 large numbers are included, giving rise to overflow. The CPU will never stop in any of these scenarios, although this operation is incorrect. As the CPU signals a warning, the ALU makes a decision on the event as the normal operation resumes. Therefore, online detection can prevent Trojan activation and a malfunction from spreading throughout the chip. 2) One advantage of this method is localization. The SOCs often involve IPs supplied by various manufacturers. If Trojans intrude upon a specific IP, localization helps the user easily find the Trojan source. This also facilitates the convenient tracking and identifying of Trojans. Moreover, Trojans can be prevented from spreading all over the area by isolating the surface of a Trojan-affected chip. In some occasions, a given section of the chip can be switched on or off. 3) This technique will, with 100% certainty, detect any Trojan attack against the detectors embedded in the main circuit. Detectors have different logic functions and are placed in different sections of the circuit. Therefore, they are distributed similarly to other gates and cannot be easily detected. Moreover, their outputs can be defined differently. For instance, if there are several subcircuits, the correct output can be designated to be 1 for a few and 0 for the rest. The
Table 3. Area and power overhead in different methods.

<table>
<thead>
<tr>
<th>Ref</th>
<th>Year</th>
<th>Technique</th>
<th>Overhead (area/power)</th>
</tr>
</thead>
<tbody>
<tr>
<td>This work</td>
<td>2017</td>
<td>Logical detectors</td>
<td>0.5%/10% / 0.5%/8%</td>
</tr>
<tr>
<td>[10]</td>
<td>2017</td>
<td>Trust Filter</td>
<td>- / 1.7%/6.45%</td>
</tr>
<tr>
<td>[18]</td>
<td>2016</td>
<td>Weighted random pattern</td>
<td>10.26% / 9.5%</td>
</tr>
<tr>
<td>[16]</td>
<td>2016</td>
<td>TPAD (hardware Trojan prevention and detection)</td>
<td>7.4%/165% / -</td>
</tr>
<tr>
<td>[11]</td>
<td>2014</td>
<td>Current sensors</td>
<td>0.5%/8% / 0.1%/2%</td>
</tr>
<tr>
<td>[17]</td>
<td>2014</td>
<td>Insertion MUX</td>
<td>0.3%/32% / 0.6%/45%</td>
</tr>
<tr>
<td>[14]</td>
<td>2014</td>
<td>Delay fingerprint</td>
<td>19%/26% / -</td>
</tr>
<tr>
<td>[15]</td>
<td>2013</td>
<td>Camouflaging</td>
<td>5.5X / 4X</td>
</tr>
<tr>
<td>[3]</td>
<td>2012</td>
<td>Thermal sensors</td>
<td>10%/15% / -</td>
</tr>
</tbody>
</table>

15.5X implies a 5.5-fold increase of overhead, 4X implies a four-fold increase of overhead.

6. Conclusion

This paper proposes a simple and low-cost Trojan detection technique. It is based on comparing logical values of sensitive nodes of the circuit at any moment. A signature is defined for the circuit, thus enabling the users to check the signature as the circuit operates, finding out if the chip has been infected by Trojans or not. A self-authentication method, i.e. DFT, is conducted. Various types of Trojan attacks can be detected by this technique, which tends to be scalable, suitable for encountering the effects of noise and process variations. The increases in area and power overhead have been reported to be 10% and 8% in the worst case, respectively. The conducted simulations and implementations validate the proposed technique.

References


