Hindawi
Security and Communication Networks
Volume 2021, Article ID 8839222, 15 pages
https://doi.org/10.1155/2021/8839222

Research Article

Process Variation-Resistant Golden-Free Hardware Trojan Detection through a Power Side Channel

Yidong Yuan,[1,2] Yao Zhang,[3,4] Yiqiang Zhao,[1] Xige Zhang,[1,2] and Ming Tang[3,4]

[1]*School of Microelectronics, Tianjin University, Tianjin 300072, China*
[2]*Beijing Engineering Research Center of High-Reliability IC with Power Industrial Grade, Beijing Smart-Chip Microelectronics Technology Co., Ltd., Beijing 100192, China*
[3]*School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China*
[4]*Key Laboratory of Aerospace Infomation Security and Trusted Computing, Ministry of Education, Wuhan University, Wuhan 430072, China*

Correspondence should be addressed to Ming Tang; m.tang@126.com

Received 28 April 2020; Revised 5 January 2021; Accepted 16 February 2021; Published 3 March 2021

Academic Editor: Fulvio Valenza

With the globalization of the manufacturing supply chain, the malicious modification existing in the middle of distrust is becoming an important security issue on the chip. These modifications are called hardware Trojan (HT). HT is difficult to detect due to its high concealment and diversity of implementation. HT detection based on the side channel is a relatively effective detection method because it does not need to trigger the Trojan or destroy the chip. However, detection based on the side channel faces two major challenges. Firstly, the side channel detection is quite dependent on the golden model. The second one relates to the accuracy of the samples. Side channel information of the chip comes from the hardware manufacturing process and implementation, so it is obviously affected by process variation. In the existing work, many self-reference detection methods have been proposed to solve the problem of missing golden models. However, the existing methods often have special requirements for the circuit structure (such as the need for self-similar structures in the circuit). And, they can hardly resist process variation. This paper combines design and detection. We select the power consumption generated at different times and construct two self-reference 'knapsack' to detect HT. The solution proposed in this article is a kind of self-reference method, but we need neither self-similar structures nor the same state of some clocks in the circuit. Meanwhile, by constructing the 'knapsack,' we reduce the impact of process variation on detection accuracy because the process variation in the two sets of power consumption is balanced.

1. Introduction

With the development of global outsourcing manufacturing services, an emerging security problem has emerged in the field of Integrated Circuit (IC) manufacturing, that is, potential chip modification in uncontrolled chip manufacturing [1]. These modifications, maliciously and intentionally applied to the circuit, are called the Hardware Trojans [2]. The hardware Trojan can be divided into the Always-On Hardware Trojans (AHT) and the Triggered Hardware Trojans (THT) according to the different trigger mode. An Always-On Trojan can cause harm as soon as the power of the circuit is on. A Triggered Hardware Trojan contains two parts: the trigger circuit and payload circuit, as shown in Figure 1. The trigger circuit starts running after the power is on, but it does not show malicious behaviour. It only monitors some signals or a series of events in the circuit, and its output is connected to the load circuit of the Trojan. The payload circuit is usually in a silent state. Once triggered, it shows malicious behaviour. Compared with Always-On Trojans, Triggered Trojans are more dangerous because the adversary can control when the Trojan is detonated. The process of chip design and manufacture can be divided into specification, design, fabrication, testing, and assembly. It is possible to inject hardware trojans in design, fabrication, and assembly [3].

For these different Trojan injection stages, a variety of hardware Trojan detection methods have been proposed. Li et al. [4] divided detection methods into two categories.

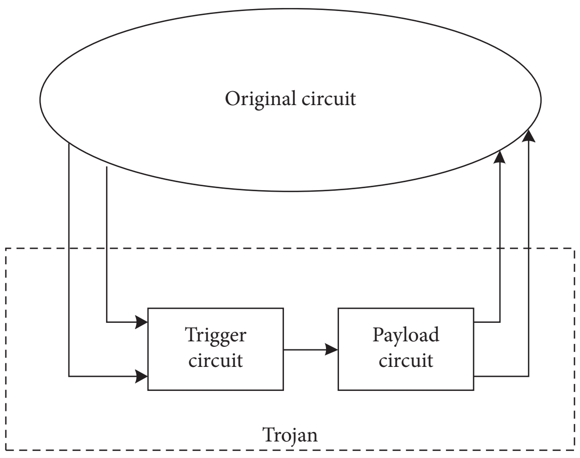

Figure 1: The structure of THT.

Detecting the HTs inserted by the EDA tools or brought in the IP cores is called pre-silicon detection and finding the HTs inserted during the assembly stage and the manufacturing stage is called post-silicon detection. The objects of pre-silicon detection are RTL code, netlist, domain, and so on. Pre-silicon detection can be further divided into (a) detections based on formal verification and functional simulation methods [5–7], (b) detections based on the information flow [8], and (c) detections based on the analysis of Trojan characteristics [9, 10]. The objects of post-silicon detection are actual IC manufactured by untrusted founders. Widely used post-silicon detection technologies include (a) logic testing [11], (b) side channel analysis [12–17], (c) detection combining logic test and side channel detection [18, 19], and (d) chip reverse engineering [20].

Existing detection techniques have great limitations. Pre-silicon detection is often used to detect the Trojan with a specific structure or function because most pre-silicon detections require prior knowledge of the Trojan. For example, the detection based on the analysis of Trojan features requires modelling in advance. Formal verification and functional simulation detection are only effective for the Trojan with specific behaviour. Among post-silicon detection, logic test and other technologies which need to activate the Trojan for detection have great limitations. Because of the rarity of Trojan triggering events, it is difficult to activate the Trojan in physical detection. Trojan detection based on side channel analysis is a widely used post-silicon method [21]. The SCA method can detect the Trojan even when it is not triggered because the trigger circuit of THT keeps running as soon as the power is on. However, most of the previous SCA approaches rely on a golden chip which is usually hard to obtain. Procuring a golden chip may require destructive reverse engineering through decapsulation, delayering, and imaging of the chip [9].

The related work of golden chip-free detection can be divided into two categories. In the first category, a golden template is simulated for detection through the netlist file or layout file. He et al. proposed a novel strategy for HT detection using electromagnetic side channel-based spectrum modelling and analysing [22]. They utilize the design data at the early stage of the IC lifecycle, and the generated spectrum can serve as the golden reference. Rad et al. proposed a method which does not need a golden chip, but a Trojan-free layout is required to serve as the trusted model [23]. The second category is self-reference hardware Trojan detection based on the spatial or time similarity of circuit parameters. Du et al. proposed a self-reference method to compare the characteristics of transient current between two circuit blocks [24]. However, the method requires a set of golden chips to effectively eliminate process variation. Hoque et al. proposed a time self-reference TeSR method [25], in which the current signature of a chip at two different time windows is compared to isolate the Trojan effect. Zheng et al. proposed an IC integrity analysis SeMIA based on spatial self-similarity [26]. SeMIA compares the side-channel signature of one block with another self-similar block on the same chip. The key idea is that different self-similar blocks (i.e., parts of an adder, comparator, memory, and logical datapath) experience different stresses due to widely varying levels of activities, or exhibit asymmetric side-channel signatures due to HT attacks.

This paper proposes a golden chip-free hardware Trojan detection scheme based on the power side channel. In order to overcome the situation that the similar structure is easily bypassed by the adversary and cannot cover the whole circuit, our scheme modifies the original design. We take advantage of the fact that the physical power consumption is proportional to the number of logic gate toggles. Under certain inputs, we construct two circuits with the same toggles for self-reference detection. It is theoretically guaranteed that the complexity of the adversary bypassing our detection scheme is $O(2^{(n/2)}\log 2^{(n/2)})$. Through simulation experiments, we demonstrate the ability to reduce process variation.

The rest of this paper is organized as follows. In Section 2, we introduce the toggle count power model, discuss our detection principle, and analyse the method to reduce process variation. In Section 3, we describe the detection scheme in detail. In Section 4, we construct both simulated and physical experiments. Section 5 concludes this paper.

2. TC-Based HT Detection

The detection technology studied in this paper is aimed at the THT injected during fabrication. The trigger part of the THT is always active, no matter whether the Trojan is triggered. Therefore, they will generate additional power consumption outside the original circuit and will be reflected on the power side channel. We can determine whether a hardware Trojan is injected in the circuit by detecting the extra power consumption. To effectively evaluate power consumption, we introduce the toggle-count power model firstly.

2.1. Toggle Count in the Digital Circuit.

2.1. Toggle Count in the Digital Circuit. The digital circuit consumes power whenever they perform computations. The total power consumption of a CMOS circuit is the sum of the power consumption of the logic cells making up the circuit. The power consumption of a logic gate is [27]

$$P_{\text{actul}} = I_{\text{leak}} \cdot V_{\text{DD}} + \alpha \cdot f \cdot \left(C_L \cdot V_{\text{DD}}^2 + V_{\text{DD}} \cdot I_{\text{peak}} \cdot t_{\text{sc}}\right), \tag{1}$$

where α indicates the number of logic gate toggles per unit time. α is related to the data and operation of the circuit. Other parameters have nothing to do with the operation or data of the circuit. They are only affected by the electrical characteristics.

Equation (1) shows that if the electrical parameters are determined, the leakage power has a direct relationship with the toggles of the logic gate. Mangard et al. mapped the toggles to simulate power consumption and successfully recovered the key from the AES encryption with mask protection [28]. This power model is called a toggle count (TC) model. The value of TC is calculated by the following equation:

$$\text{TC}(tb) = \sum_{i=1}^{m} g_i\left(t; \left(X_1, X_2\right)\right), \tag{2}$$

where t is the time point, m is the total number of logic gates in the circuit, g_i represents the TC of the ith gate during $[t, t + t']$, t' is the delay of the last toggle in the combined circuit [29], and (X_1, X_2) is the input vector pair, and we denote it as tb.

According to equation (2), TC changes with tb. The TC model only takes TC into count rather than the function of the circuit. Therefore, different circuits may have a specific tb so that their TC is the same. And, the same circuits can share the same TC under different values of tb.

Definition 1 (pair circuits). for the input vector tb_i and tb_j, if $\text{TC}_{O_1}(tb_i) = \text{TC}_{O_2}(tb_j)$, then the circuits O_1 and O_2 (O_1 and O_2 can be the same circuit) under tb_i and tb_j are *pair circuits*.

According to equation (1), when the electrical parameters are determined, the power consumption of two circuits with the same TC is equal. So, the power consumption of *pair circuits* is equal in reality.

2.2. Process Variation in TC-Based HT Detection

2.2.1. Detecting HT with TC Directly. In Section 2.1, we explained that *pair circuits* share the same power consumption. Therefore, given two tb, if the two circuits can be configured as *pair circuits*, we can achieve self-reference detection through them.

When *pair circuits* are activated by tbs which meet Definition 1, they generate equal TC. In this situation, if one Trojan is injected in the pair, the TC of the two circuits will differ from each other caused by TC of the Trojan trigger circuit unless the adversary can guess the used tb and make the Trojan generate no additional toggles. To obtain this correct tb, the adversary needs to exhaust the space of tb ($2^n * 2^n$, where n is the length of input vectors). Based on this idea, the basic detection algorithm is given in Algorithm 1.

RandomSelect. Choose two tb for two circuits separately.

CircuitExpand. Add redundant circuit into original circuit with fewer TC so that O_1 and O_2 become *pair circuits* at the design stage. After **CircuitExpand**, O_1 and O_2 can generate the same power consumption. Therefore, as long as the relationship between $P_{\text{actu}-O_1}(tb_i)$ and $P_{\text{actu}-O_2}(tb_j)$ is compared, it is possible to identify whether a hardware Trojan is injected into *pair circuits*.

TapeOut. Stand for chip manufacturing. Before **TapeOut**, we add auxiliary detection circuit. Circuit after **TapeOut** is the object of physical detection.

2.2.2. Problems Caused by Process Variation. In the physical environment, the side channel information will be affected by process variation and measurement noise during the measurement process. It means that the physical power consumption is not completely equal to the simulation power obtained by the power model. In Algorithm 1, the presence of noise will bring great challenges to the detection. It is generally believed that when the amount of data is large enough, measurement noise can be reduced by statistical methods. However, as the offset is introduced by the circuit in the production process, process variation is a fixed value even in multiple measurements and cannot be eliminated statistically.

Process variation refers to the deviation of threshold voltage and gate capacitance of the transistor [30] caused by the difference between gate length, oxide thickness, and channel doping during the production of transistors [31]. Process variation will eventually be reflected on side channel information such as power consumption and time delay. The lightweight characteristic of Trojan makes the Trojan circuit's proportion in the original circuit very small. So, using Algorithm 1 directly will make the differences caused by Trojan to be overwhelmed by noise. The detection ability of the hardware Trojan cannot be guaranteed. Let the power consumption deviation caused by logic gate g_i be $P_{pv}(g_i)$.

2.2.3. Reduce Process Variation. In order to be similar to measurement noise and statistically reduce the influence of process variation, the concept of *pair circuits* is extended to multiple groups of tb.

The new idea is shown as follows. We construct two sets of vectors, $\text{TB}_1 = (tb_1^1, tb_2^1, \ldots, tb_n^1)$ and $\text{TB}_2 = (tb_1^2, tb_2^2, \ldots, tb_n^2)$. The logic gate which toggles under TB_1 will also toggle under TB_2, and the corresponding TC is equal. We add the physical power consumption of the circuit under TB_1 and TB_2 separately and compare the overall power consumption.

Firstly, because the total TC is same, the overall power consumption is also theoretically same. Process variation caused by different processes for each logic gate may meet a certain distribution. But, when the logic gate is produced, the variation is a fixed value. For the same gate, each $P_{pv}(g_i)$ is the same even if running multiple times. We can get

```
Input: circuit design $O_1$ and $O_2$, input vector space $\mathscr{X}$
Output: whether existing Trojans in $O_1$ or $O_2$
(1)  $(tb_1, tb_2) = $ RandomSelect $(\mathscr{X})$;
(2)  $\triangle tc = |TC_{O_1}(tb_1) - TC_{O_2}(tb_2)|$;
(3)  if $TC_{O_1}(tb_1) < TC_{O_2}(tb_2)$ then
(4)      $O_1 = $ CircuitExpand $(O_1; tb_1; \triangle tc)$;
(5)  else then
(6)      $O_2 = $ CircuitExpand $(O_2; tb_2; \triangle tc)$;
(7)  end
(8)  TapeOut $(O_1, O_2)$;
(9)  if $P_{\text{actu}-O_1}(tb_i) \neq P_{\text{actu}-O_2}(tb_j)$ then
(10)         return TRUE;
(11)     else then
(12)         return FALSE;
```

Algorithm 1: TC-based HT detection.

$\sum_{\mathrm{TB}_1} P_{pv}(g_i) = \sum_{\mathrm{TB}_2} P_{pv}(g_i)$. That is, the total power consumption deviation due to process variation is same. Under this detection scheme, even process variation is considered, and the accumulated power consumptions are still equal. If a hardware Trojan is injected into the circuit, the TC generated by the Trojan trigger circuit will make the two sets of power consumption unequal.

2.2.4. Feasibility Improvement. When the number of logic gates of the circuit is extended to the order of billions or millions, it is a hard task to ensure that the TC of each gate is the same. Every time a logic gate is added to the operation, it is equivalent to an increase in the calculated dimension by one. In order to reduce the analysis complexity, we divide the circuit into several small regions, and each region is a square with side length r, as shown in Figure 2. Define each square as a *grid*, denoted as $N_i(r)$. We make the logic gates toggle the same numbers in each grid. The partition proposed in this article refers to the physical partition rather than the circuit logical segmentation and has nothing to do with the specific circuit function. Once the length r is determined, the circuit layout can be partitioned, and the time complexity is $O(1)$. The purpose of dividing the layout is to normalize the process variation of every gate in the same *grid* so that *the same TC of each gate* is converted to *the same TC of each grid*. In this way, we can reduce the complexity of the problem. Later, we discuss the rationality of such partitioning. Note that, when the circuit is segmented logically, the combination circuit between registers is generally divided into one part. However, in our method, a combination circuit may also be divided into two or more grids (when r is small enough).

Wafer is the basic material used in the manufacture of silicon semiconductor-integrated circuits. The wafer can be oxidized and etched to produce various circuit element structures. After the etching and other steps, the wafer is divided into individual die and becomes an integrated circuit product with specific electrical functions. In these circuits, intradie variations exhibit spatial correlation [32]. There are perfect correlations among the devices in the same grid, high correlations among those in close grids, and low or zero correlations in far-away grids.

Hypothesis 1. For a given grid $N_l(r)$, $\forall g_i \in N_l(r)$, $g_j \in N_l(r)$, and there is $pv(g_i) = pv(g_j)$.

Under Hypothesis 1, if the TC of two TB in one grid are the same, for this grid, the total process variation generated by two TB is the same. For the given TB_1 and TB_2, if the TC of any grid is the same, the process variation in the two power consumption is equal.

Trojan detection under Hypothesis 1 can effectively reduce the complexity of the problem and make the detection scheme feasible because we degenerate the dimensions from each gate to each grid. However, the feasibility improvement also reduces the detection accuracy. When r reaches the minimum value (as shown in Figure 2(b)), each grid contains only one logic gate. It is assumed that Hypothesis 1 is true. At this time, the highest detection accuracy can be obtained, and the effect of process variation is completely avoided. When each grid contains multiple logic gates, there are differences between each gate in the grid. But, they are highly correlated. Therefore, power consumption analysis under Hypothesis 1 can effectively reduce the impact of process variation, though it still exists in the end. The larger r is, the greater the influence of process variation will be. At the same time, the larger r makes the condition that TC of each grid reaches the same faster and easier. The balance between detection accuracy and time overhead can be dynamically adjusted according to the designer/detector.

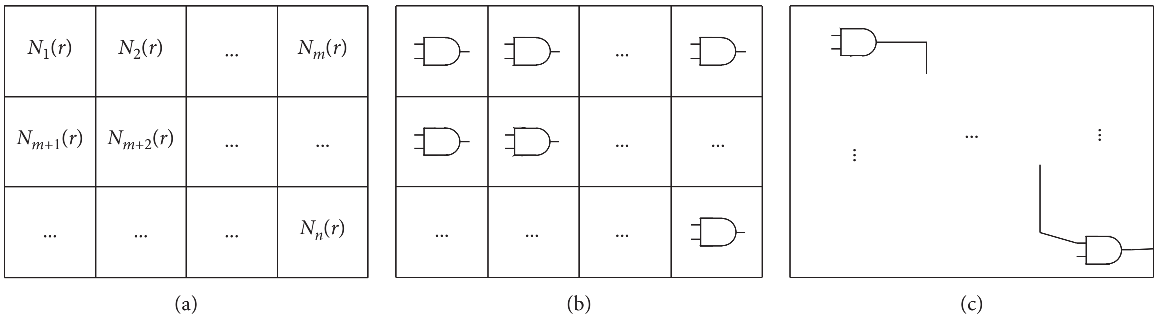

FIGURE 2: (a) Dividing the circuit by r. (b) $r = r_{\min}$. (c) $r = r_{\max}$.

2.3. Build KP-like TC

2.3.1. Pair TC Sets. For TB $= (tb_1, tb_2, \ldots, tb_n)$, $\mathbf{TC}_i = (\mathrm{TC}_{i,1}, \ldots, \mathrm{TC}_{i,\mathcal{T}})$ denotes the TC generated by the circuit under tb_i, where $\mathrm{TC}_{i,t}$ denotes the TC at the t clock and $\mathfrak{T}$ is the maximum clock cycle of the circuit. Then, the TC of the total circuit under TB can be recorded as an $n \times \tau$ matrix $\mathbf{T}$. And, the physical power leakage corresponding to $\mathbf{T}$ is a matrix $\mathbf{L}$:

$$\mathbf{T} = \begin{bmatrix} \mathbf{TC}_1 \\ \mathbf{TC}_2 \\ \cdots \\ \mathbf{TC}_n \end{bmatrix} = \begin{bmatrix} \mathrm{TC}_{1,1} & \mathrm{TC}_{1,2} & \cdots & \mathrm{TC}_{1,\tau} \\ \mathrm{TC}_{2,1} & \mathrm{TC}_{2,2} & \cdots & \mathrm{TC}_{2,\tau} \\ \cdots & \cdots & \cdots & \cdots \\ \mathrm{TC}_{n,1} & \cdots & \cdots & \mathrm{TC}_{n,\tau} \end{bmatrix},$$
$$\mathbf{L} = \begin{bmatrix} L_{1,1} & L_{1,2} & \cdots & L_{1,\tau} \\ L_{2,1} & L_{2,2} & \cdots & L_{2,\tau} \\ \cdots & \cdots & \cdots & \cdots \\ L_{n,1} & \cdots & \cdots & L_{n,\tau} \end{bmatrix}. \quad (3)$$

For any element in the matrix $\mathbf{T}$, $\mathrm{TC}_{i,j} = \sum_{k=1}^{n_g} tc_{i,j}^{(k)}$, where $tc_{i,j}^{(k)}$ represents the TC of the kth grid at the jth clock when the input is tb_i and n_g is the number of grids. If the circuit does not run for more than τ under one tb, the elements in $\mathbf{T}$ are padded with zeros.

Definition 2 (pair TC sets). For a circuit divided into n_g grids, $\forall k = 1, 2, \ldots, n_g$, and if $\sum_{\mathrm{TC}_{i,j} \in P} tc_{i,j}^{(k)} = \sum_{\mathrm{TC}_{i,j} \in Q} tc_{i,j}^{(k)}$, then P and Q are *pair TC sets* for the circuit. $P, Q \subset \mathbf{T}$, and $|P| = |Q|$.

Because for each k, there is $\sum_{\mathrm{TC}_{i,j} \in P} tc_{i,j}^{(k)} = \sum_{\mathrm{TC}_{i,j} \in Q} tc_{i,j}^{(k)}$, and the total TC under P and Q are the same. According to equation (1), activating the circuit with P and Q separately will generate the same dynamic power consumption. The static power consumption of the circuit has nothing to do with the data at runtime. Moreover, we make as many elements in P and Q as possible to guarantee the same static power. From Sections 2.2.3 and 2.2.4, the same TC of each grid ensures a balance of process variation in the overall power consumption of two sets. Therefore, if P and Q are *pair TC sets*, the sums of the physical power consumption corresponding to them are equal.

2.3.2. Build Pair TC Sets

Definition 3 (multidimensional 0/1 knapsack problem). Given an $n \times m$ matrix $\mathbf{A}$ and an m-dimensional column vector b, determine whether there is an n-dimensional binary vector $X = \{x_1, x_2, \ldots, x_n\}$ making the equation $\sum_{j=1}^{n} a_{i,j} x_j = b_i$ and $i = 1, 2, \ldots, m$ true.

Proposition 1. *Building pair TC sets can be reduced to solve the multidimensional 0/1 knapsack problem.*

Proof. The construction of pair TC sets is divided into two stages: (1) selecting elements from the matrix $\mathbf{T}$ to join the P set and (2) for a given P set, finding the paired Q set. In the second stage, for a given P, $\sum_{\mathrm{TC}_{i,j} \in P} tc_{i,j}^{(k)}$, $k = 1, 2, \ldots, n_g$ in every grid constitutes a column vector of n_g dimension, corresponding to the column vector b in the MKP problem. The remaining elements after removing the set P in the matrix $\mathbf{T}$ form a new matrix $\mathbf{T}'$:

$$T' = \left[\mathrm{TC}_1, \mathrm{TC}_2, \ldots, \mathrm{TC}_{n*-\mathcal{T}|P|}\right] = \begin{bmatrix} tc_1^{(1)} & tc_2^{(1)} & \cdots & tc_{n*\mathcal{T}-|P|}^{(1)} \\ tc_1^{(2)} & tc_2^{(2)} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ tc_1^{(n_g)} & \cdots & \cdots & tc_{n*\mathcal{T}|P|}^{(n_g)} \end{bmatrix}, \quad (4)$$

corresponding to the matrix $\mathbf{A}$ in MKP. The goal of building pair TC sets is to find a column vector in T' that satisfies $\sum_{\mathrm{TC}_{i,j} \in P} tc_{i,j}^{(k)} = \sum_{\mathrm{TC}_{i,j} \in Q} tc_{i,j}^{(k)}$. This corresponds to the binary vector solved in MKP. In the MKP problem, there is no stipulation on the number of 1 in the binary vector, and it can be the sum of any number of $a_{i,j}$ equal to b_j. However, when building pair TC sets, it is necessary to have the same number of elements of the two sets. When corresponding to the MKP problem, we add an additional constraint: the number of 1 in the binary vector is equal to the number of elements in the P set. This constraint can be transformed into an additional dimension and added to the original $\mathbf{T}'$ matrix. The new row vector in the matrix is $\underbrace{\begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix}}_{n*\mathcal{T}-|P|}$. The dimension of each column vector is increased by 1. In this way, the construction of pair TC sets can still be reduced to the MKP problem. □

3. Scheme of KP-Based HT Detection

3.1. Design of the Scheme. The detection scheme proposed in this paper associates the detector with the circuit designer and assists the detection by inserting circuits into the original design. The detection target in this article is the hardware Trojan injected in fabrication, and the Trojan is not always on but needs to be triggered.

3.1.1. Flow of the Scheme. The entire detection process is shown in Figure 3. Our scheme receives the RTL design or netlist design of the circuit and finally determines whether there is a hardware Trojan in the physical chip. The plan can be divided into three phases.

The first phase is the preprocess stage, which is used to generate the data structures required for the subsequent steps. Algorithm 2 first selects a set of input vectors X in the overall input space for testing. The selection of the input vector needs to cover the normal circuit functions according to the original circuit structure. A TB vector is then generated from X. TB is used to simulate the TC to obtain the TC matrix $\mathbf{T}$. There are many different levels of simulation of TC [33], and we use the lowest level simulation to meet the actual running state.

The second phase is the circuit design modification stage. The target of this stage is the circuit after Place and Route. The layout circuit is divided into n_g grids, and the domain length of each grid is r. In the matrix $\mathbf{T}$ generated in the first stage, n elements are randomly selected and put into the sets P and Q, respectively. For each grid, Algorithm 3 calculates the sum of the TC. If there is $\sum_{TC_{i,j}\in P} tc_{i,j}^{(k)} = \sum_{TC_{i,j}\in Q} tc_{i,j}^{(k)}$, $k = 1, 2, \ldots, n_g$, then Algorithm 3 outputs the design layout, P, Q, and TB. Otherwise, Algorithm 3 performs circuit design modification. Finally, the modified PR-level design will be output. The circuit modification strategy is described in detail in Section 3.1.2.

The third phase is the Trojan detection stage. At the end of the second phase, we consider the design of the circuit to be reliable and Trojan-free. As a potential hardware attacker, the chip manufacturer can inject a hardware Trojan in the circuit tape-out link. The target of the detection phase is a completed circuit. We run the circuit according to TB and collect the power leakage matrix $\mathbf{L}$. The elements in $\mathbf{L}$ correspond to the elements in $\mathbf{T}$ on a one-to-one basis. The elements in $\mathbf{L}$ corresponding to the elements in P and Q are accumulated, and we compare $\sum_{\{(i,j)|\mathrm{TC}_{i,j}\in P\}} L_{i,j}$ and $\sum_{\{(i,j)|\mathrm{TC}_{i,j}\in Q\}} L_{i,j}$. If the two are different, the Trojan was injected during the manufacturing process. Otherwise, the circuit is clean and secure.

3.1.2. Circuit Expansion. Section 2.3.2 proves that the core of building pair TC sets is to solve an MKP problem. At present, the knapsack problem is still an NPC problem. Horowitz and Sahni proposed a two-table algorithm using the divide-and-conquer method, with time complexity $O(2^{(n/2)}\log 2^{(n/2)})$ [34]. With the introduction of various artificial intelligence algorithms, many optimized knapsack solutions have been proposed. But, they are still essentially in exponential time complexity.

In order to make the detection scheme reach the practical and feasible time complexity, this paper combines the detection with the chip design. We make some adjustments to the original design to make it suitable for our detection scheme. As mentioned in Section 2.2.1, we insert the redundant circuit into the original circuit. Once the set P is selected, the set Q is not directly solved according to the conditions. We randomly select elements to join set Q. For each grid, calculate the distance between P and Q. By inserting redundant toggles in the original design, the TC of P and Q in each grid reaches the same. The complexity of solving the knapsack problem is transformed into the complexity of constructing redundant circuits.

In our scheme, the main function of the circuit expansion is to reduce the time overhead of the designer when building the pair TC sets. The AND gate is the most basic logic element and exists in all CMOS circuits. So, we extend redundant circuits based on an AND gate and its two fan-in gates, as shown in Figure 4. We designed five extension methods based on the different types of gate 1 and gate 2 (AND or NOT) to generate additional TC without changing the original logic function of the circuit. The extended circuit logic is shown in Table 1.

3.1.3. Overhead. The five methods mentioned in Section 3.1.2 all add two redundant logic gates, which can generate two additional TC on average. That is, the number of redundant logic gates generated by this solution is $\sum_{k=1}^{n_g} |\sum_{\mathrm{TC}_{i,j}\in P} tc_{i,j}^{(k)} - \sum_{\mathrm{TC}_{i,j}\in Q} tc_{i,j}^{(k)}| = \triangle\mathrm{TC}$.

In terms of the time complexity of Algorithm 3, as we mentioned in Section 2.2.4, **Partition** is an $O(1)$ complexity operation. Therefore, the time complexity of Algorithm 3 is related to the number of executions of **Expansion**. On average, every time the TC differs by two, we need to insert a redundant logic. The execution frequency of the algorithm is $(\Delta\mathrm{TC}/2)$. Therefore, the time complexity of Algorithm 3 is $O(\Delta\mathrm{TC})$.

3.2. Security of KP-Based HT Detection. The attacker model is as follows. The adversary can get the layout design of the circuit and the format of the input and output. In order to bypass this detection scheme, the adversary can (1) determine which grids are used in the detection by solving P and Q sets and (2) construct a special Trojan so that the injected Trojan presents the same TC for any test vector.

For the first method, the adversary cannot obtain $\mathbf{T}$ because $\mathbf{T}$ is constructed based on the TB which is the subset of the entire input space. Taking the AES128 encryption algorithm as an example, the size of its input space is 2^{128}. It is difficult for the adversary to guess $\mathbf{T}$. Even considering the worst case, the adversary can determine $\mathbf{T}$. In Section 2.3, we have explained that solving P and Q from $\mathbf{T}$ is an exponential complexity $O(2^{(n/2)}\log 2^{(n/2)})$. Therefore, the adversary cannot bypass our detection through the first method. With the second method, the attacker does not need to solve P and Q. However, the Trojan trigger circuit cannot achieve the

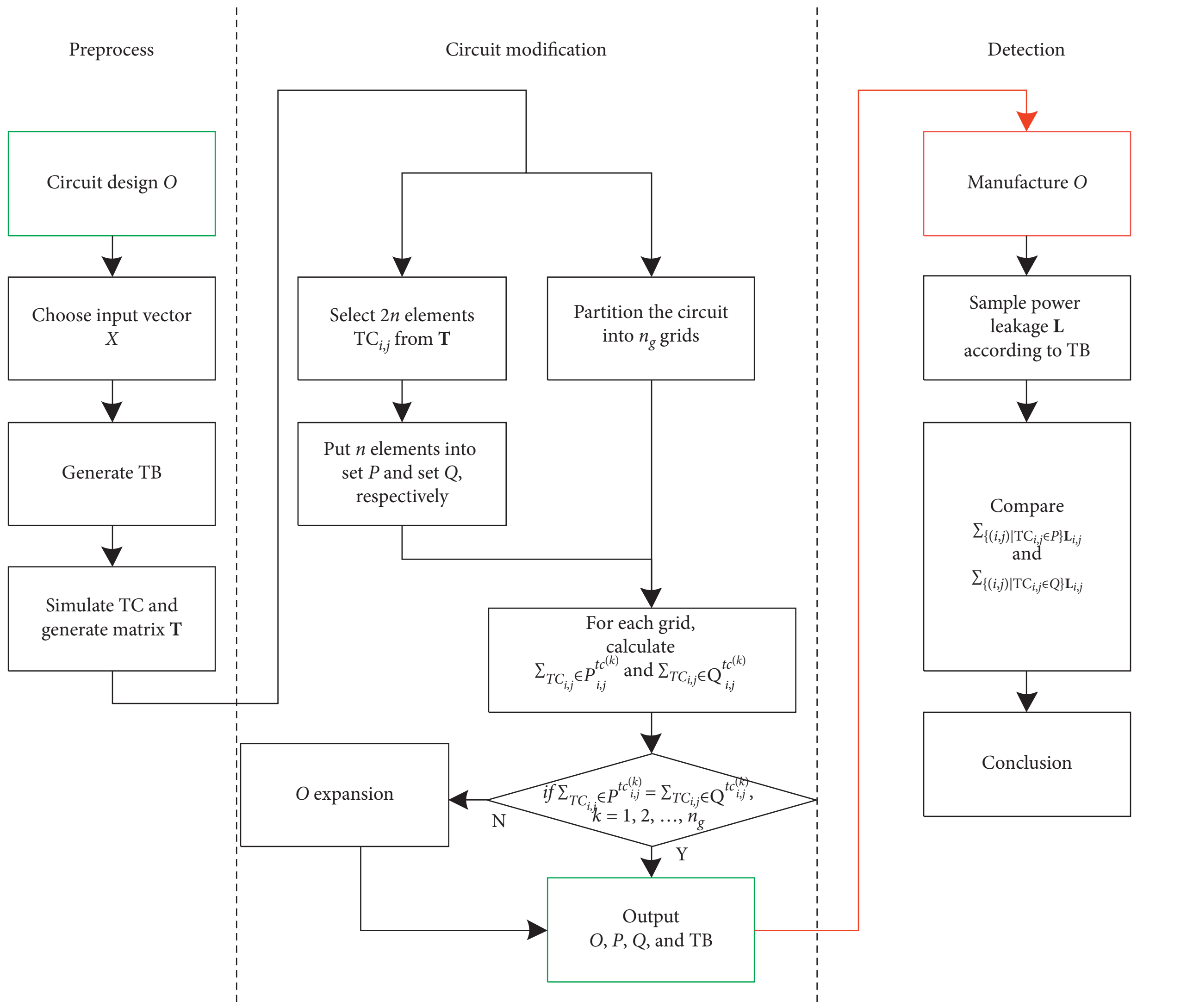
Preprocess
Circuit modification
Detection
Circuit design O
Choose input vector X
Generate TB
Simulate TC and generate matrix $\mathbf{T}$
Select $2n$ elements $TC_{i,j}$ from $\mathbf{T}$
Partition the circuit into n_g grids
Put n elements into set P and set Q, respectively
For each grid, calculate $\sum_{TC_{i,j}\in P} tc_{i,j}^{(k)}$ and $\sum_{TC_{i,j}\in Q} tc_{i,j}^{(k)}$
if $\sum_{TC_{i,j}\in P} tc_{i,j}^{(k)} = \sum_{TC_{i,j}\in Q} tc_{i,j}^{(k)}$, $k = 1, 2, \ldots, n_g$
N
Y
O expansion
Output O, P, Q, and TB
Manufacture O
Sample power leakage $\mathbf{L}$ according to TB
Compare $\sum_{\{(i,j)|TC_{i,j}\in P\}} \mathbf{L}_{i,j}$ and $\sum_{\{(i,j)|TC_{i,j}\in Q\}} \mathbf{L}_{i,j}$
Conclusion

Figure 3: Flow chart of KP-based HT detection scheme.

```
Input: circuit design $O$, input vector space $\mathscr{X}$
Output: matrix $\mathbf{T}$ and TB
(1) $\mathbf{X} = [X_0, X_2, \ldots, X_m] = \textbf{RandomSelect}\ (\mathscr{X})$;
(2) $\text{TB} = [tb_1, tb_2, \ldots, tb_m] = [(X_0, X_1), (X_1, X_2), \ldots, (X_{m-1}, X_m)]$;
(3) $\mathbf{T} = \textbf{Simulate}\ (O, \text{TB})$;
```

Algorithm 2: Preprocess.

same TC state for all test vectors. Because of the small probability of triggering the Trojan, the Trojan trigger circuit must have a large fan-in cone [35]. When the part state meets the trigger condition, the fan-in cone will produce different responses. If the same TC occurs for any input, the trigger circuit loses its function. Therefore, the adversary cannot bypass our detection through the second method.

3.3. Detection Capability of the Scheme. The existence of process variation makes the measured power consumption calculated according to P and Q not completely equal. Let the difference between physical power consumptions without Trojan be $P_{pv\text{-total}}$ and the power consumption of the Trojan trigger circuit be P_{trojan}. If equation (5) is satisfied, the scheme can do detect the injected Trojan:

$$P_{\text{trojan}} > P_{pv\text{-total}}. \tag{5}$$

Because the trigger circuit generates different TC under different inputs, we use average TC to measure the relationship between the size of the Trojan and the original circuit. We

```
Input: original circuit $O$ and $\mathbf{T}$
Output: modified circuit $O'$ and set $P$, $Q$
(1) $(o_1, o_2, \ldots, o_{n_g}) = \mathbf{Partition}\ (O)$;
(2) Select $n$ elements $\mathrm{TC}_{i,j}$ from T and put them into $P$ and $Q$, respectively;
(3) for $k$ from 1 to $n_g$
(4)     if $\sum_{\mathrm{TC}_{i,j} \in P} tc_{i,j}^{(k)} \neq \sum_{\mathrm{TC}_{i,j} \in Q} tc_{i,j}^{(k)}$ then
(5)         Expansion $(o_k)$;
(6)     end
(7)     $k = k + 1$;
(8) end
(9) $O' = (o_1, o_2, \ldots, o_{n_g})$;
```

Algorithm 3: Circuit modification.

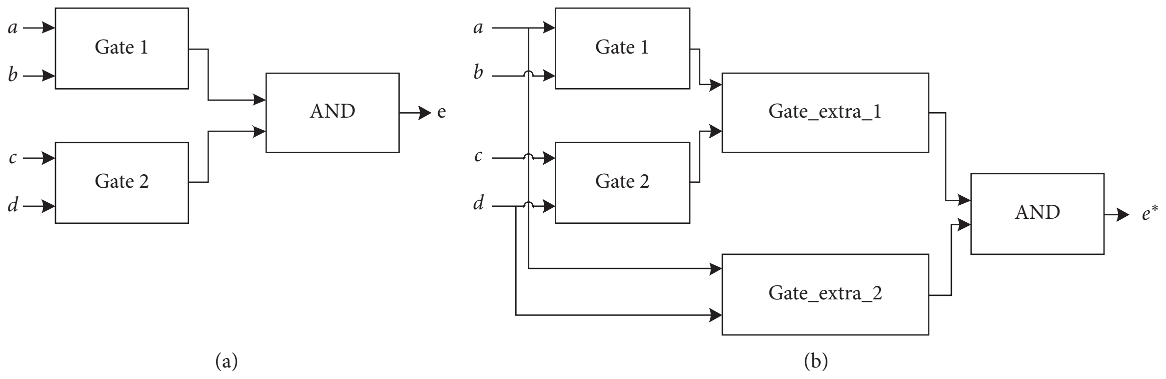

Figure 4: (a) The structure of original AND. (b) The structure of AND after the redundant circuit is inserted.

Table 1: Five methods to expand AND logic.

Number	Origin logic	Extended logic
1	$e = (a \wedge b) \wedge (c \vee d)$	$e^* = ((a \wedge b) \wedge (c \vee d)) \wedge (a \vee d)$
2	$e = (a \wedge b) \wedge (c \wedge d)$	$e^* = ((a \wedge b) \wedge (c \wedge d)) \wedge (a \vee d)$
3	$e = (a \wedge b) \wedge c$	$e^* = ((a \wedge b) \wedge c) \wedge (b \vee c)$
4	$e = (a \vee b) \wedge (c \vee d)$	$e^* = ((a \vee b) \wedge (c \vee d)) \wedge (c \vee d)$
5	$e = (a \vee b) \wedge c$	$e^* = ((a \vee b) \wedge c) \wedge (a \vee b)$

denote the average TC ratio between the Trojan trigger circuit and the original circuit as $\rho = (\mathrm{AVG}(\mathrm{TC}_{\mathrm{trojan}})/\mathrm{AVG}(\mathrm{TC}_{\mathrm{orig}}))$. And, because TC is proportional to the nominal power consumption of the circuit, we get $\rho = (P_{\mathrm{trojan}}/P_{\mathrm{orig}})$. According to the previous equation,

$$\rho > \frac{P_{pv\text{-total}}}{P_{\mathrm{orig}}}. \quad (6)$$

That is, for a hardware Trojan with the toggle scale greater than $(P_{pv\text{-total}}/P_{\mathrm{orig}})$, a 100% successful detection rate can be achieved. The scheme proposed in this paper can effectively deal with process variation. In the physical detection, $P_{pv\text{-total}}$ will be less than the theoretical value, so it can detect a smaller scale Trojan.

4. Experiment

We conducted the experiments on Xilinx Virtex5 XC5VLX30 FPGA. The device granularity on FPGA can only reach the standard FPGA unit, that is, LUT. Therefore, in the experiment, we convert all gate-level signals into LUT output signals.

4.1. Circuit under Test.

For the detection scheme proposed in this paper, we carried out the experiment both with simulated data and physical data. The simulation experiment is based on an FPGA implementation of a 3-share AES S-box masking [36]. We call it TIS16. The physical experiment is based on AES-ECB encryption [37].TIS16 uses a new addition chain to accelerate and lighten the S-box, as shown in Figure 5, where S stands for square operation and M stands for multiplication. The hardware implementation of TIS16 is shown in Figure 6.

The shamul module is used to perform the inversion defined by the addition chain and iterate the square operation according to the affine transformation in GF (2^8). The shamac module performs shared multiplication on constants and then performs shared addition.

This paper implemented TIS16's 3-share design on Xilinx Virtex5 XC5VLX30 FPGA. A round of encryption requires 21 clocks. The entire circuit occupies 530 registers and 883 LUTs.

Based on the physical measurement experiment on the AES-ECB encryption algorithm coming from DPA contest v2 [37], we implemented it on the SASEBO-GII development board. The board is equipped with Xilinx Spartan6 XC6SLX30 FPGA. The S-box of AES is implemented by a look-up table method. Each round of encryption is completed within one

$$x \xrightarrow{3S} x^{8} \xrightarrow{M} x^{9} \xrightarrow{2S} x^{36} \xrightarrow{M} x^{54} \xrightarrow{S} x^{108} \xrightarrow{M} x^{126} \xrightarrow{M} x^{127} \xrightarrow{S} x^{254}$$

Figure 5: The new addition chain of TIS16.

clock, and each encryption contains a total of 11 clocks. This circuit uses a total of 881 registers and 2296 LUTs.

4.2. Reducing Process Variation

4.2.1. Simulation Experiment. There is a proportional relationship between the TC and the physical power consumption of the circuit. Based on the TC, we simulated the effect of process variation on each signal and obtained the simulated power consumption affected by process variation. By measuring the deviation of the physical power consumption under different grid sizes, the ability of our scheme to resist process variation was verified. The simulation of the circuit is based on Xilinx's ISim. The specific process of the experiment is as follows:

(1) *Circuit Simulation.* Postplace and route simulation of the test circuit generates a VCD file, in order to obtain TC of the test circuit under a given input (in this experiment, we focused on the part that actually implements the encryption function. We only simulated the masking part, and the operation of the control circuit was ignored).

(2) *Circuit Partition.* This experiment is based on Xilinx's FPGA, and the circuit can naturally be partitioned by SLICE. The logic elements belonging to the same SLICE are divided into the same grid.

(3) *VCD File Analysis.* Calculate the TC of each SLICE (grid) at each clock, and generate matrix **T** by Algorithm 2.

(4) *Build Pair TC Sets* **P** *and* **Q**. In the experiment, we use the Euclidean distance as an indicator to measure the difference between the TC of two sets in each grid. We build P and Q with the minimized Euclidean distance to ensure that the TC difference of each grid is minimized. It is used to reduce redundant gates.

(5) *Insert Redundancy.* Compared to ASIC circuits, FPGAs specify the total number of hardware resources available to the designer. In the experiment, we choose the speed-first strategy during place and route, leaving a part of resources in each SLICE to add redundancy. On FPGA, for the expansion scheme proposed in 4.1.3, we convert it into the corresponding LUT truth table. For grids with different TC, we construct a LUT that is related to the input. For grids with the same TC, we construct a redundant LUT that is independent of the input.

(6) *Power Simulation.* The simulation power consumption includes the nominal power caused by toggles and the noise power caused by process variation. Research by Chang and Sapatnekar [32] shows that, with the influence of process variation, the leakage power consumption of logic gates approximately follows a logarithmic Gaussian distribution. We simulate the power consumption of each flip as e^{Y_i}, where $Y_i \sim N(\mu_{y_i}, \sigma^2_{y_i})$ is a Gaussian random variable. According to the spatial correlation of process variation parameters, for TC in the same grid, we generate simulation power with the same expectation and variance $(\mu_{g_i}, \sigma^2_{g_i})$. The average power consumption of each grid is denoted as a random variable $M = \{\mu_{g_1}, \mu_{g_2}, \ldots, \mu_{g_n}\}$. We assume $M \sim N(0, \sigma^2_{\text{total}})$. The size of σ^2_{total} affects the proportion of process variation in the total power consumption. In our experiment, we set it to 5%. $\sigma^2_{g_i}$ affects the relevance of TC in the same grid and is related to r. We set $\sigma^2_{g_i}$ of each region to the same value and test the ability of this solution to resist process variation under different $\sigma^2_{g_i}$.

When divided by SLICE size, the circuit is partitioned into 206 grids. After the 4th step, the circuit is activated with input vectors according to P and Q, respectively. The resulting TC are 88291 and 83939. After the redundancy is inserted, their TC are both 88310. And, the final simulated values are 70131.27 and 71335.53. The deviation between the two sets is about 1.72%. Regardless of the partition, we directly choose input to make P and Q toggle the same, which means the entire circuit is in the same grid. The resulting simulation power is 75488.57 and 78639.83, respectively. The deviation between the two is about 4.17%. The values of $\sigma^2_{g_i}$ in the two experiments are set to 0.01 and 0.001. It can be seen from the experiments that the detection scheme proposed in this paper has a certain resistance to process variation, which improves the detection accuracy of Trojan under the same experimental environment.

4.2.2. Physical Experiment. The first five steps of the physical experiment are the same as the simulation experiment. In Step 6, we used the physical power consumption data. In the physical experiment, we tested the experimental effect under different grid sizes. Specifically, in the experiment, the smallest partition unit we use is SLICE ($r = 1$), and each SLICE contains 4 LUTs. Besides, we tested the performance of the larger partitions ($r = 1, 2, 4, 8,$ and 16).

We describe the circuit in the XDL file of Xilinx and instantiate the GII AES design twice. The two instantiations are placed and routed in the same way only with an offset in the phase. The distance of the same function SLICEs is r, as mentioned in Section 2.2.4. r here refers to the ordinate difference of SLICE in the XDLRC file. As shown in Figure 7, r of SLICE1 and SLICE2 equals 1, and r of SLICE1 and SLICE4 equals 3. In this way, when we measure the power of the two different instantiations, it is equivalent to constructing a P set and a Q set under a partition with r.

We choose an AES instance as the reference circuit and collect 100,000 power traces. The traces are statically aligned based on correlation and denoised with Gaussian filtering. The preprocessing power traces of ten rounds' AES encryption are shown in Figure 8. Under different r, we

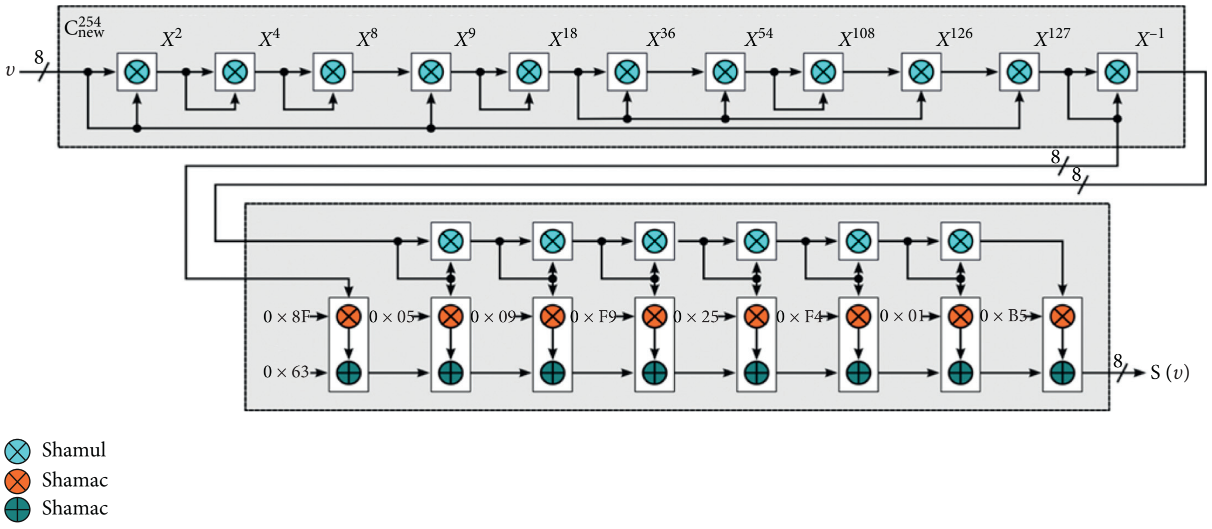

Figure 6: The hardware implementation of TIS16.

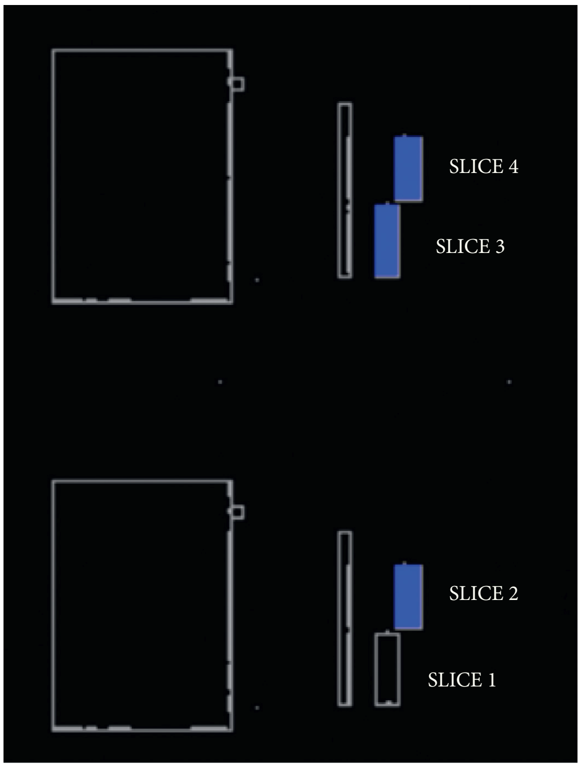

Figure 7: The layout of Xilinx's FPGA.

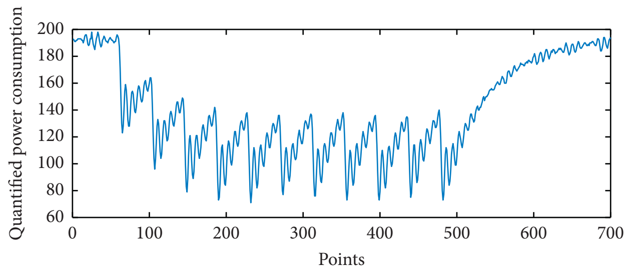

Figure 8: Preprocess power consumption of the reference circuit.

perform the same power collection and preprocessing on other circuits and calculate the power difference between them (as Q set) and the reference circuit (as P set). Figure 9 shows the difference in power consumption between P set and Q set under different r.

From Figure 9(f), it can be seen intuitively that when r is larger, the difference in power consumption between the two sets will fluctuate more. We choose the Euclidean distance as the evaluation index and calculate the distance between the power trace of the reference circuit and the power trace of the circuits under different r. When $r = 1, 2, 4, 8$, and 16, the Euclidean distances are 5.8119, 30.7587, 44.7261, 48.8229, and 53.6777, respectively.

Experimental results show that when the circuit is partitioned into small *grids* and the power consumption is compared by grids, the smaller the grids, the smaller the process variation in the power consumption. And, the comparison result will be more accurate. The scheme proposed in this paper can effectively reduce the influence of process variation on the detection accuracy of Trojans.

4.3. Detecting Hardware Trojans

4.3.1. Simulation Experiment. In order to demonstrate the detection capability of KP-based HT detection, we injected hardware Trojan in TIS16 and conducted the simulation experiment. The HT sample comes from Trust-Hub [38], and the experiment is based on the AES-T800 in Trust-Hub. After place and route, TIS16 uses a total of 305 registers and 745 LUTs, and the Trojan circuit occupies an additional 4 registers and 8 LUTs. Our experiment only considers the Trojan trigger circuit and does not pay attention to the payload (when the payload is not triggered, it will not produce any toggle that affects the detection method). The trigger circuit of the AES-T800 is a finite state machine. When the four consecutive input vectors meet the four given values, the Trojan will be triggered. Based on the circuit used in Section 4.2, we inject the Trojan trigger circuit. In order to

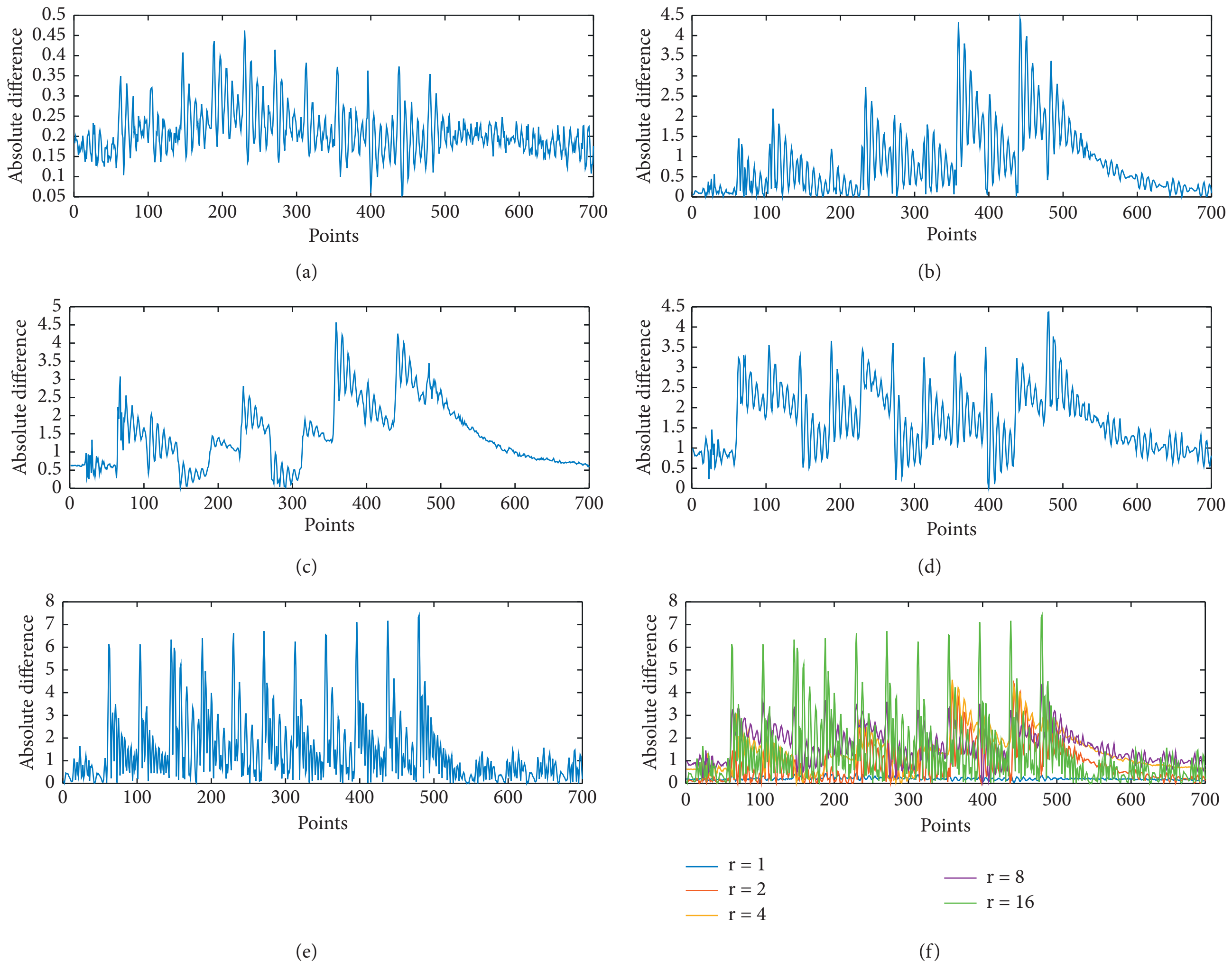

Figure 9: Absolute power difference of different r. (a) $r = 1$. (b) $r = 2$. (c) $r = 4$. (d) $r = 8$. (e) $r = 16$. (f) $r = 1, 2, 4, 8$, and 16.

ensure that the original design will not be modified, we directly inject the Trojan in the XDL file which contains the placement and routing information.

For the circuit with HT, we activate it with the same input in Section 4.2. The simulation power difference between P and Q sets is 4,297.01, which accounts for 6.02% of the original circuit power consumption, and the difference caused by the Trojan trigger circuit accounts for 4.33%. This value exceeds the 1.72% caused by process variation shown in Section 4.2. The result proves that our scheme can successfully detect hardware Trojans.

Note that the number of active gates of the Trojan trigger circuit and the original circuit is completely different under different test benches. Therefore, it may result in different detection effects choosing different test benches. In the experiment, we test different test benches and find that the minimum power consumption difference caused by Trojan is only 565.56 (0.79%). And, the maximum can reach 3297.69 (4.62%). When we select the worst input vectors, the Trojan power consumption may be masked by process variation. This fact reflects the importance of another work in this article, which is to reduce process variation. When the algorithm proposed in this paper is not used, process variation causes 4.17% power deviation in our experiment, which makes it impossible to successfully detect the AES-T800 Trojan under most test benches. When the circuit is partitioned, the process variation is reduced to 1.72% making the detection success rate increase.

4.3.2. Physical Experiment. We have extracted the Trojan trigger circuit in Trust-Hub [38]. In the Trust-Hub Trojan library, there are only three Trojan trigger modes for AES. In addition to the AES-T800 finite-state machine trigger structure used in the simulation experiment, the remaining structures include counter triggers and specific plaintext triggers. However, the solution in this article only pays attention to the toggle power consumption of the trigger circuit, which dilutes the trigger structure of the Trojan. At the same time, we assume that the trigger structure does not change the original function. Therefore, we use redundant circuits with different toggles to represent the Trojan trigger circuits equivalently. In the physical experiment, we insert the "Trojan" circuit, as shown in Figure 10. In the ten rounds of AES encryption, the encryption core calculates a 128 bit

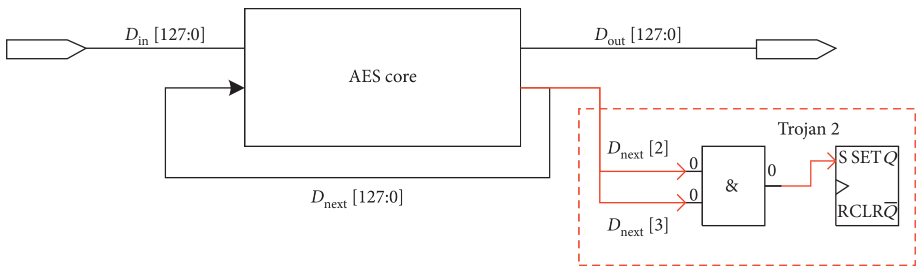

FIGURE 10: Equivalent Trojan trigger circuit.

Dnext signal at the end of each round and applies it to the next encryption. We pull out two bits of D_{next} and perform an AND operation. The result is sent to a redundant register. In this way, we can inject an extra toggle in each round of encryption in the original circuit. The specific implementation is shown in Figure 11.

The total scale of the injected circuit is 20 registers and 40 LUTs, accounting for 2.27% and 1.74% of the original design, respectively. We calculate the absolute difference between the power consumption of the circuit-inserted hardware Trojans and the power consumption of the reference circuit. And, the result is compared with the circuit under different r. As shown in Figures 12(a) and 12(b), when the Trojan trigger circuit contains registers, the extra power consumption is relatively large. Even under the partition length of $r = 16$, the power of the Trojan is still far greater than the power caused by the process variation. The Trojan trigger circuit of this size can be effectively detected. On this basis, we reduced the size of the Trojan trigger circuit, by removing the registers in Figure 10 and turning the Trojan trigger circuit into a pure combinational circuit. As shown in Figures 12(c) and 12(d), under the Trojan scale of 40 LUTs, the power consumption of the Trojan is mixed with process variation when $r = 8$. At this time, due to the influence of process variation, the existence of Trojans cannot be detected. But, when the partition size reaches $r = 1$, the Trojan can still be obviously distinguished. Figure 12(e) shows the difference in power consumption between the Trojan trigger circuit with registers and the Trojan trigger circuit without registers under the same toggles.

Experiments show that the proposed scheme can effectively detect the combinational logic Trojan trigger circuit which accounts for about 1% of the original circuit scale.

4.4. Scheme Efficiency and Overhead. For time and space overhead, we found that the solution time is linear and can be directly calculated based on small data size. At present, under the simulation condition of a clock of 100 ns, the postroute simulation using Isim takes 16 minutes when performing 1000 encryptions (26 clocks each time, including key expansion). And, the generated VCD file reaches 4018405 kB (about 3G). For the above simulation data (data volume at $26 * 108$ moments), the analysis time of the VCD file is about 6 minutes.

In this experiment, we use the knapsack algorithm to solve the multidimensional knapsack. When the knapsack dimension reaches 6 (that is, the total number of partitions ng is 6), for 1000 traces (10 clocks per trace), the size of the state matrix of dynamic programming reaches more than 16 GB. Without state recording, when the dimension reaches 10, the solution time has exceeded 24 hours.

In the previous article on hardware Trojan detection, the work of TeSR [25] and SeMIA [26] are similar to this paper. TeSR leverages on the uncorrelated temporal variations in transient current signature of sequential hardware Trojans to isolate their effect from process and measurement noise. By comparing current signature of a chip for the same input pattern at different time windows, TeSR can only detect sequential hardware Trojans, while the solution proposed in this article can detect both combination Trojans and sequential Trojans. The Trojan example used in TeSR contains 8 registers. Our scheme can detect the Trojans without registers. SeMIA uses the inherent structural self-similarity in the design to detect hardware Trojans. The experiments in SeMIA show that SeMIA can detect hardware Trojans that account for 2.3% of the original circuit. The solution proposed in this paper can still effectively distinguish the Trojans from the process variation when the Trojan accounts for 1.74% of the area.

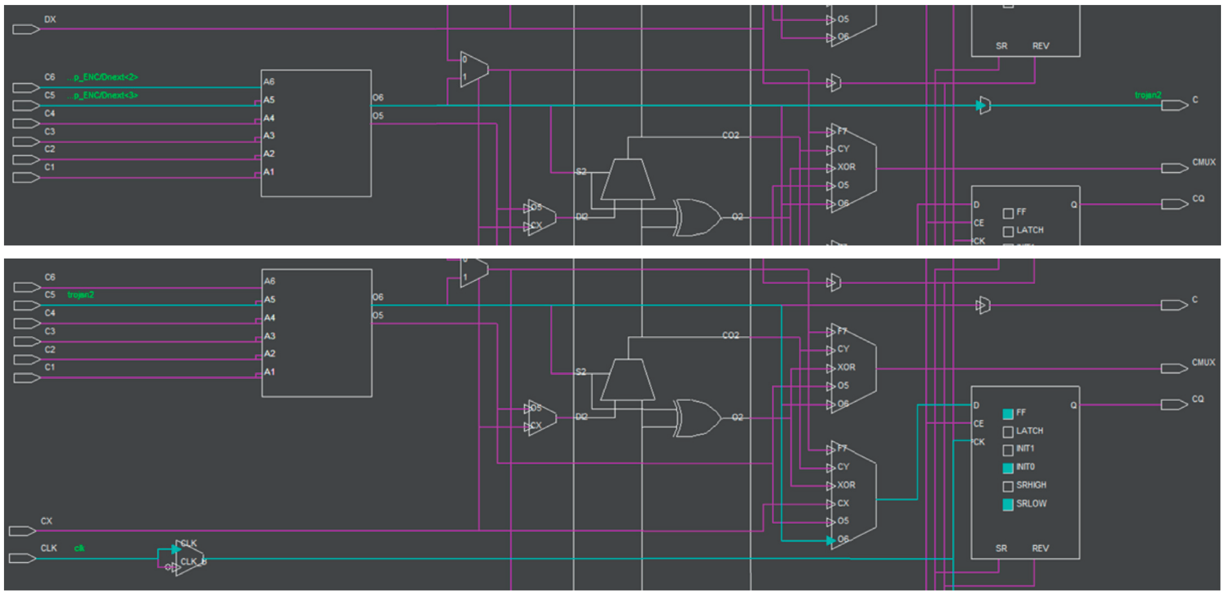

FIGURE 11: The implementation of equivalent Trojan trigger circuit.

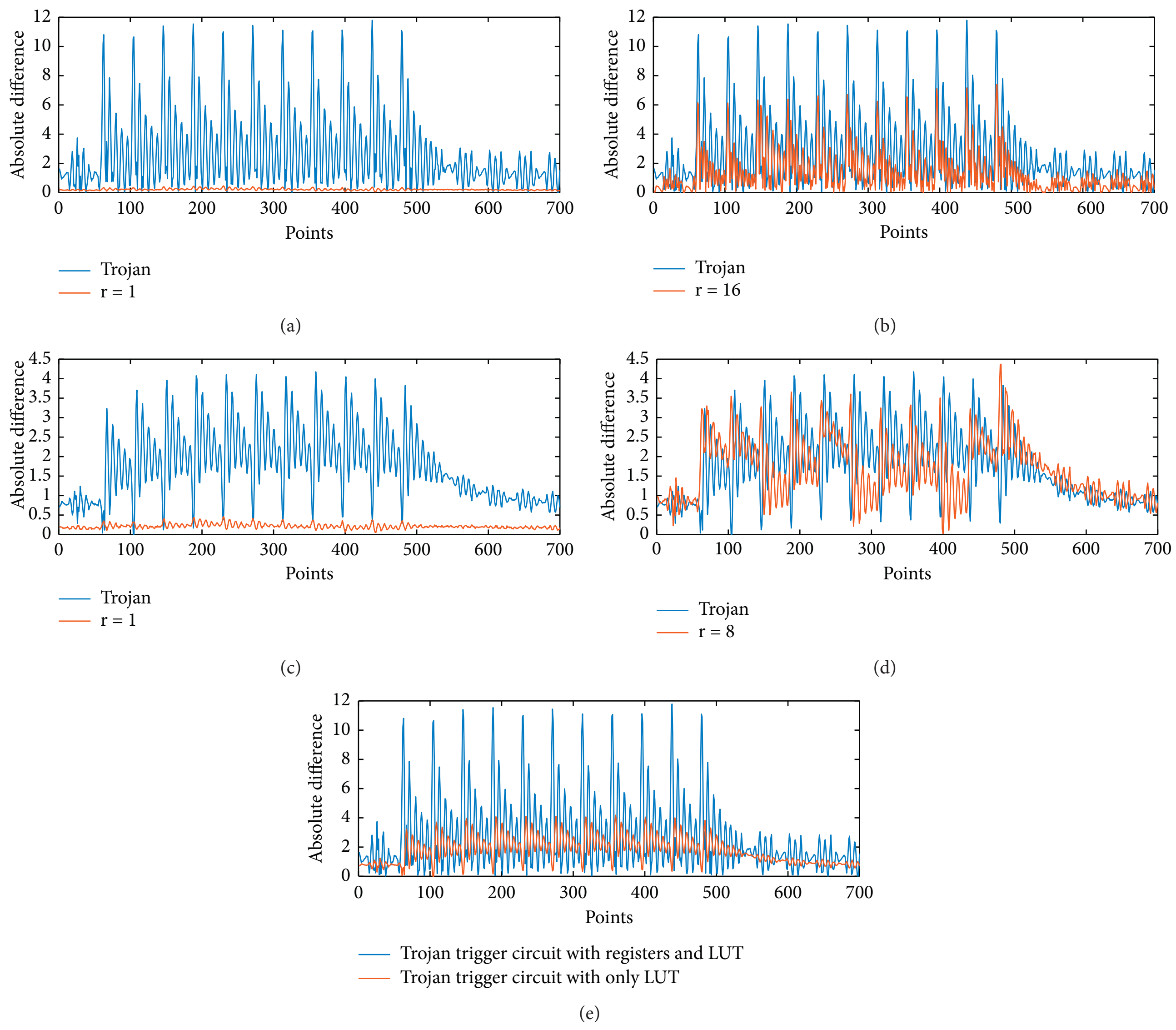

FIGURE 12: The result of Trojan detection. (a) Trojan with reg and LUT vs. $r = 1$. (b) Trojan with reg and LUT vs. $r = 16$. (c) Trojan only with LUT vs. $r = 1$. (d) Trojan only with LUT vs. $r = 8$. (e) Trojan with reg and LUT vs. Trojan with LU.

5. Conclusions

This paper proposes a detection scheme for post-silicon hardware Trojans. Our method combines design and detection. We insert several redundant circuits during the design so that the power consumption selected at a specific time can be superimposed to form two self-referenced power consumption sets. By modifying the design, the problem of requiring a specific structure in the existing self-reference detection scheme is solved. For the modified circuit, we can generate two sets of circuit running moments. The total toggle counts of the two sets are equal. In this way, the physical power consumption corresponding to the two sets are also equal. Their power consumption can be referred to each other, and they can be seen as each other's golden template. The adversary has no knowledge of the redundancy addition process. In order to find the equal toggle counts, they have to solve a knapsack problem. Given that solving the knapsack problem is an NP problem, it proves that even if the adversary obtains the original design, he/she cannot know which power is included in the self-reference set. This guarantees the security of the proposed detection method. Based on the spatial correlation of process variation, this article divides the circuit and extends the knapsack into a multidimensional knapsack. We enable the variation in each grid to reach a balance which minimizes the deviation caused by process variation in the overall power consumption. In this paper, the resistance to process variation is realized by dividing the circuit into small grids, and it is verified in experiments.

5.1. Future Work. In the future, we will study excellent test-bench generation technology to improve the success rate of our detection scheme. When the circuit scale is expanded to a large order of magnitude, our method will be more complicated. We try to study the appropriate algorithm to make the difference between them as small as possible when selecting the elements of the P and Q sets. Besides, we will explore the possibility of the hardware Trojan location through grid division and quantify the relationship between the average TC ratio and process variation.

Data Availability

The simulation power data and physical traces used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by The Laboratory Open Fund of Beijing Smart-Chip Microelectronics Technology Co., Ltd., and was supported in part by the National Natural Science Foundation of China under Grant 61972295, Wuhan Science and Technology Project Application Foundation Frontier Special Project (2019010701011407), and Foundation Project of Wuhan Maritime Communication Research Institute (2018J-11).

References

[1] DARPA, *Trust in Integrated Circuits (TIC)*, DARPA, Virginia Square, VA, USA, 2007, http://www.darpa.mil/MTO/solicitations/baa07-24.

[2] S. Bhasin and F. Regazzoni, "A survey on hardware trojan detection techniques," in *Proceedings of the 2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, Lisbon, Portugal, May 2015.

[3] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: identifying and classifying hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[4] H. Li, Q. Liu, J. Zhang, and Y. Lyu, "A survey of hardware trojan detection, diagnosis and prevention," in *Proceedings of the 2015 14th International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics)*, Xi'an, China, August 2015.

[5] T. Reece, D. B. Limbrick, and W. H. Robinson, "Design comparison to identify malicious hardware in external intellectual property," in *Proceedings of the 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, Changsha, China, November 2011.

[6] W. Adam, M. Suozzo, and S. Sethumadhavan, "FANCI: identification of stealthy malicious logic using boolean functional analysis," in *Proceedings of the ACM SIGSAC Conference on Computer & Communications Security*, Berlin, Germany, November 2013.

[7] M. Hicks, "Overcoming an untrusted computing base: detecting and removing malicious hardware automatically," in *Proceedings of the 31st IEEE Symposium on Security and Privacy, S&P 2010*, Oakland, CA, USA, May 2010.

[8] W. Hu, B. Mao, J. Oberg, and R. Kastner, "Detecting hardware trojans with gate-level information-flow tracking," *Computer*, vol. 49, no. 8, pp. 44–52, 2016.

[9] M. Oya, Y. Shi, M. Yanagisawa, and N. Togawa, "A score-based classification method for identifying hardware-trojans at gate-level netlists," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, March 2015.

[10] K. Hasegawa, "Hardware Trojans classification for gate-level netlists based on machine learning," in *Proceedings of the 2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, Sant Feliu de Guixols, Spain, July 2016.

[11] S. Jha and S. Kumar Jha, "Randomization based probabilistic approach to detect Trojan circuits," in *Proceedings of the 2008 11th IEEE High Assurance Systems Engineering Symposium HASE 2008*, Nanjing, China, December 2008.

[12] S. Wei and M. Potkonjak, "Self-consistency and consistency-based detection and diagnosis of malicious circuitry," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 9, pp. 1845–1853, 2013.

[13] X. Mingfu, A. Hu, and G. Li, "Detecting hardware trojan through heuristic partition and activity driven test pattern generation," in *Proceedings of the IET Conference Publications*, p. 3, Beijing, China, May 2014.

[14] O. Söll, T. Korak, M. Muehlberghuber, and M. Hutter, "EM-based detection of hardware Trojans on FPGAs," in *Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Arlington, VA, USA, May 2014.

[15] N. Yoshimizu, "Hardware Trojan detection by symmetry breaking in path delays," in *Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, Arlington, VA, USA, May 2014.

[16] 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 *Proceedings of the 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Grenoble, France, March 2013.

[17] J. Zhang, G. Su, Y. Liu et al., "On Trojan side channel design and identification," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, November 2014.

[18] R. S. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia, "MERO: a statistical approach for hardware trojan detection," in *Cryptographic Hardware and Embedded Systems*Springer, Berlin, Heidelberg, 2009.

[19] H. Li and Q. Liu, "Hardware Trojan detection acceleration based on word-level statistical properties management," in *Proceedings of the 2014 International Conference on Field-Programmable Technology (FPT)*, Shanghai, China, December 2014.

[20] E. Krotkov and J. Blitch, "The defense advanced research projects agency (DARPA) tactical mobile robotics program," *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 769–776, 1999.

[21] H. Li, Q. Liu, and J. Zhang, "A survey of hardware Trojan threat and defense," *Integration*, vol. 55, pp. 426–437, 2016.

[22] J. He, Y. Zhao, X. Guo, and Y. Jin, "Hardware Trojan detection through chip-free electromagnetic side-channel statistical analysis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2939–2948, 2017.

[23] R. Rad, J. Plusquellic, and M. Tehranipoor, "Sensitivity analysis to hardware Trojans using power supply transient signals," in *Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, Anaheim, CA, USA, June 2008.

[24] D. Du, S. Narasimhan, R. S. Chakraborty, and S. Bhunia, "Self-referencing: a scalable side-channel approach for hardware Trojan detection," in *Cryptographic Hardware and Embedded Systems*, pp. 173–187, Springer, Berlin, Heidelberg, 2010.

[25] T. Hoque, S. Narasimhan, X. Wang, S. Mal-Sarkar, and S. Bhunia, "Golden-free hardware Trojan detection with high sensitivity under process noise," *Journal of Electronic Testing*, vol. 33, no. 1, pp. 107–124, 2017.

[26] Y. Zheng, S. Yang, and S. Bhunia, "SeMIA: self-similarity based IC integrity analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, p. 35, 2015.

[27] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, Springer, Berlin, Germany, 2007.

[28] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked AES hardware implementations," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, Edinburgh, UK, September 2005.

[29] A. Moradi, M. Salmasizadeh, M. T. M. Shalmani, and T. Eisenbarth, "Vulnerability modeling of cryptographic hardware to power analysis attacks," *Integration the VLSI Journal*, vol. 42, no. 4, pp. 468–478.

[30] Y. Li, C.-H. Hwang, T.-Y. Li, and M.-H. Han, "Process-variation effect, metal-gate work-function fluctuation, and random-dopant fluctuation in emerging CMOS technologies," *IEEE Transactions on Electron Devices*, vol. 57, no. 2, pp. 437–447, 2009.

[31] A. Srivastava, R. Bai, A. Blaauw, and D. Sylvester, "Modeling and analysis of leakage power considering within-die process variations," in *Proceedings of the International Symposium on Low Power Electronics and Design*, Monterey, CA, USA, August 2002.

[32] H. Chang and S. S. Sapatnekar, "Full-chip analysis of leakage power under process variations, including spatial correlations," in *Proceedings of the 42nd Annual Design Automation Conference*, Anaheim, CA, USA, June 2005.

[33] M. Tang, Y. Li, Y. Li et al., "A generic TC-based method to find the weakness in different phases of masking schemes," *Tsinghua Science and Technology*, vol. 23, no. 5, pp. 574–585, 2018.

[34] E. Horowitz and S. Sahni, "Computing partitions with applications to the knapsack problem," *Journal of the ACM*, vol. 21, no. 2, pp. 277–292, 1974.

[35] X. Chen, Q. Liu, S. Yao et al., "Hardware Trojan detection in third-party digital intellectual property cores by multilevel feature analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 7, pp. 1370–1383, 2017.

[36] F. De Santis, T. Bauer, and S. Georg, "Hiding higher-order univariate leakages by shuffling polynomial masking schemes: a more efficient, shuffled, and higher-order masked AES S-box," in *Proceedings of the 2016 ACM Workshop on Theory of Implementation Security*, Vienna, Austria, October 2016.

[37] http://www.dpacontest.org/v2/download.php.

[38] https://www.trust-hub.org/.