cryptography

Article

A Novel Two-Level Protection Scheme against Hardware Trojans on a Reconfigurable CNN Accelerator

Zichu Liu, Jia Hou, Jianfei Wang and Chen Yang *

School of Microelectronics, Xi'an Jiaotong University, Xi'an 710049, China;
daolzc199947@stu.xjtu.edu.cn (Z.L.); hjwst0314@stu.xjtu.edu.cn (J.H.); jfwang@stu.xjtu.edu.cn (J.W.)
* Correspondence: chyang00@xjtu.edu.cn; Tel.: +86-13810103039

Abstract: With the boom in artificial intelligence (AI), numerous reconfigurable convolution neural network (CNN) accelerators have emerged within both industry and academia, aiming to enhance AI computing capabilities. However, this rapid landscape has also witnessed a rise in hardware Trojan attacks targeted at CNN accelerators, thereby posing substantial threats to the reliability and security of these reconfigurable systems. Despite this escalating concern, there exists a scarcity of security protection schemes explicitly tailored to counteract hardware Trojans embedded in reconfigurable CNN accelerators, and those that do exist exhibit notable deficiencies. Addressing these gaps, this paper introduces a dedicated security scheme designed to mitigate the vulnerabilities associated with hardware Trojans implanted in reconfigurable CNN accelerators. The proposed security protection scheme operates at two distinct levels: the first level is geared towards preventing the triggering of the hardware Trojan, while the second level focuses on detecting the presence of a hardware Trojan post-triggering and subsequently neutralizing its potential harm. Through experimental evaluation, our results demonstrate that this two-level protection scheme is capable of mitigating at least 99.88% of the harm cause by three different types of hardware Trojan (i.e., Trojan within RI, MAC and ReLU) within reconfigurable CNN accelerators. Furthermore, this scheme can prevent hardware Trojans from triggering whose trigger signal is derived from a processing element (PE). Notably, the proposed scheme is implemented and validated on a Xilinx Zynq XC7Z100 platform.

Keywords: reconfigurable CNN accelerator; hardware Trojans; security protection scheme

check for updates

Citation: Liu, Z.; Hou, J.; Wang, J.; Yang, C. A Novel Two-Level Protection Scheme against Hardware Trojans on a Reconfigurable CNN Accelerator. *Cryptography* **2024**, *8*, 34. https://doi.org/10.3390/cryptography8030034

Academic Editor: Jim Plusquellic

Received: 10 May 2024
Revised: 11 July 2024
Accepted: 29 July 2024
Published: 4 August 2024

1. Introduction

In contemporary times, convolution neural networks (CNNs) have gained widespread application across diverse tasks such as image classification [1,2], face recognition [3,4], and medical diagnosis [5,6]. Due to the substantial parameter sizes often exceeding hundreds of megabytes in CNN models, each inference process entails billions of operations. To enhance inference speed and minimize power consumption, an abundance of CNN accelerators has emerged within both academia and industry. Leveraging a reconfigurable computing architecture, which offers heightened energy efficiency and flexibility [7], numerous CNN accelerators employ reconfigurable processing element (PE) arrays to expedite the CNN inference process [8–13].

While recent years have seen extensive research aimed at improving the performance and energy efficiency of CNN accelerators, investigations into the reliability of these accelerators remain relatively scarce. Consequently, research into hardware security for reconfigurable CNN accelerators assumes paramount importance. Among the array of hardware security concerns, hardware Trojans stand out as a critical threat. Inserted by adversaries at various points in the integrated circuits (IC) supply chain, such as the IP vendor or system-on-chip (SOC) integration [14], hardware Trojans manifest in diverse attack strategies targeting CNN accelerators. Notably, adversaries may modify inputs to alter classification results [15], manipulate processing elements (PEs) by pruning multiply-accumulate operations [16], or introduce logic gates to modify activation functions [17], and even target the reconfigurable interconnect (RI) [18]. In a bid to directly manipulate output, adversaries may obscure targeted errors within triggers, employing multiplexers for output modification [19,20]. Moreover, concealment of hardware Trojans based on the feature map of CNNs has been explored [21,22]. Regrettably, the literature reveals a dearth of studies proposing comprehensive hardware Trojan protection schemes tailored specifically for reconfigurable CNN accelerators. Leibo Liu proposed a secure mapping approach to bolster the resilience of reconfigurable architecture against hardware Trojans [23]. However, this approach lacks specificity in addressing hardware Trojans implanted in CNN accelerators. Shamik Kundu presented two strategies for generating functional test patterns to detect hardware Trojans in CNN accelerators [24]. Yet, these strategies only confirm the presence of hardware Trojans in the neural network accelerator without providing a means to mitigate their impact. Peiyao Sun proposed countermeasures to detect hardware Trojan attacks targeting the pooling layer of CNN implementations [25]. Unfortunately, this approach falls short in eliminating the impact of hardware Trojans and relies on a processing element that is inherently safe by default without providing a methodology for acquiring such safety. Consequently, the current prevention schemes for hardware Trojans in CNN accelerators exhibits several deficiencies.

This paper addresses the identified deficiencies by proposing a comprehensive scheme. In response to the escalating threat of hardware Trojan attacks on reconfigurable CNN accelerators, a two-level protection scheme is introduced. The first protection level, termed the trigger level, is designed to prevent the triggering of hardware Trojans, while the second protection level is focused on detecting and eliminating the harmful effects of triggered hardware Trojans. In contrast to previous protection schemes, our proposed scheme offers three key advantages: Firstly, it can prevent hardware Trojans from triggering. Secondly, it can detect and eliminate the harmful effects of hardware Trojans. Thirdly, it does not need golden elements considered absolutely safe.

The primary contributions of this paper can be summarized as follows:

1. PE Space Randomization (PSR) Solution: This solution is designed to prevent the triggering of hardware Trojans by randomizing the PE space. Simulation results demonstrate that PSR can effectively prevent over 90% of potential hardware Trojans from being triggered.
2. Voting Solution: This solution detects and corrects incorrect configuration words resulting from hardware Trojan attacks. Operating in real-time and without the need for golden elements, the voting solution demonstrates 100% effectiveness in detecting hardware Trojans in the reconfigurable interconnect (RI) and eliminates 100% of the harm caused by these Trojans under attack severities ranging from 1% to 5%.
3. Input–Output Relationship Detection (IORD) Solution: This real-time solution detects the triggering of hardware Trojans inside PE without requiring golden elements. Simulation results indicate that IORD can successfully detect the presence of hardware Trojans in PE with 100% accuracy under attack severities from 1% to 5%.
4. PE Collaboration Correction (PCC) Solution: Developed to eliminate the harm caused by hardware Trojans in PE, this real-time solution demonstrates a probability of at least 95.3% in eliminating harm and ensuring correct accelerator functionality under hardware Trojan attack severities from 1% to 5%.
5. The proposed hardware Trojan prevention scheme is implemented on a reconfigurable CNN accelerator deployed on a Xilinx Zynq XC7Z100 platform. The effectiveness of the protection scheme is validated through experimental evaluation.

The structure of this paper is organized as follows: Section 2 provides background information and discusses related work. Section 3 details the PSR, Voting, and IORD solutions. Section 4 presents an evaluation of the proposed scheme. Finally, Section 5 concludes the paper.

2. Background and Motivation

2.1. Reconfigure PE Array on CNN Accelerators

The conventional structure of reconfigurable processing element arrays (RPA) in convolution neural network (CNN) accelerators is illustrated in Figure 1. A multitude of processing elements (PEs) are organized under specific connections to execute parallel convolution operations. These PEs interconnect based on the configuration of the reconfigurable interconnection (RI), forming distinct computing modules for convolution operations. Chen Yang's proposed Reconfigurable Neural Accelerator (RNA) exemplifies a typical reconfigurable CNN accelerator [10]. The RNA is based on the reconfiguration technology of the dynamic reconfigurable computing architecture. DRA usually has a coarse-grained reconfiguration capability, allowing the configuration of computing resources to be changed dynamically at runtime to adapt to different application requirements. It can be reconfigured frequently during the application running process. This enables it to continuously adjust its architecture when performing different tasks. Its reconfiguration is achieved by changing the configuration word of RI (such as MUX and demultiplexer) through the configuration module, and this process only takes one to several clock cycles [7]. Therefore, the RNA can change the architecture in one to several clock cycles. The RNA consists of a reconfigurable array of 484 PEs organized as a 22 $\times$ 22 rectangle, 22 11-bit $\times$256 filter memories, a ping-pong 2.56 KB image memory, 22 704B multi-bank RAMs as output buffer, 1452 DSP, FSM and a configuration module. As depicted in Figure 1, RNA's RPA encompasses two types of PEs: normal PE (NPE) and special PE (SPE). Each NPE consists of two shift registers, a MAC cluster, an adder and a FIFO. A total of 16 $\times$ 11b shift registers referred to as filter reg and image reg store weights and image data, respectively. A MAC cluster is built to perform the row convolution operation. The FIFO size is 64 $\times$ 11 bit. Conversely, based on the configuration word, the SPE can function either like the NPE or perform convolution operations with a rectified linear unit (ReLU) activation function. The data paths of NPE and SPE are differentiated by green and blue lines, respectively, as indicated in Figure 1.

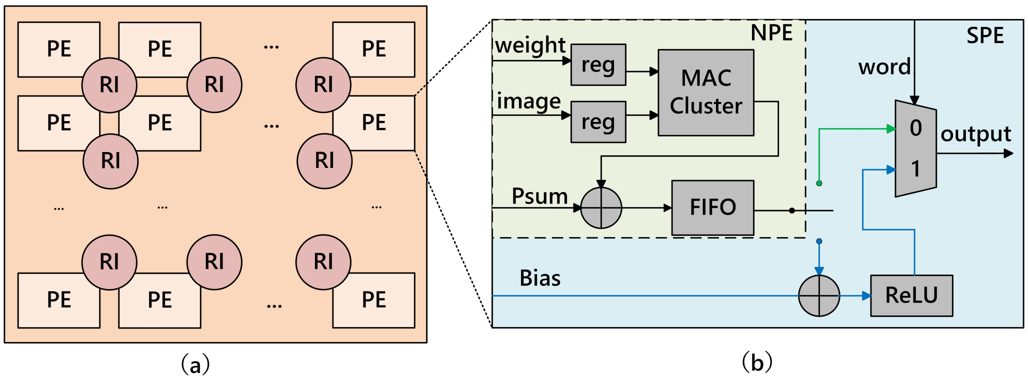


Figure 1. The structure of RPA. (**a**) Overview of RPA; (**b**) the structure of PE.

2.2. Targeting Hardware Trojans Models

Hardware Trojans consist of two main components: the trigger and the payload. The trigger is responsible for receiving a signal and determining whether to activate the hardware Trojan based on this signal. Upon activation, the payload executes a specific task, such as modifying results or causing the chip to malfunction. In normal scenarios, the hardware Trojan system operates seamlessly, resembling a non-Trojan system. However, deviations occur only when the hardware Trojan is triggered, providing effective concealment.

In the context of hardware Trojans targeting CNN accelerators, Odetola et al. [21] utilized a specific output pixel value from the output feature map as a trigger signal, following a normal distribution. The Trojan is triggered when the pixel value falls within a specific interval, intentionally chosen to be improbable under normal circumstances to enhance Trojan concealment. Once triggered, the payload alters the channels of the convolution kernel, resulting in erroneous outcomes. Yang et al. [18] introduced a hardware Trojan with a payload integrated into the reconfigurable interconnection. As depicted in Figure 2, each processing element (PE) is responsible for the convolution operation of a row. Through configuration word adjustments of the multiplexer (MUX), the PE array can perform convolution operations with 3 × 3, 5 × 5, and 6 × 6 kernel sizes. Upon triggering the Trojan, the MUX's configuration word is inverted, leading to a modification in the data path and subsequently producing incorrect results.

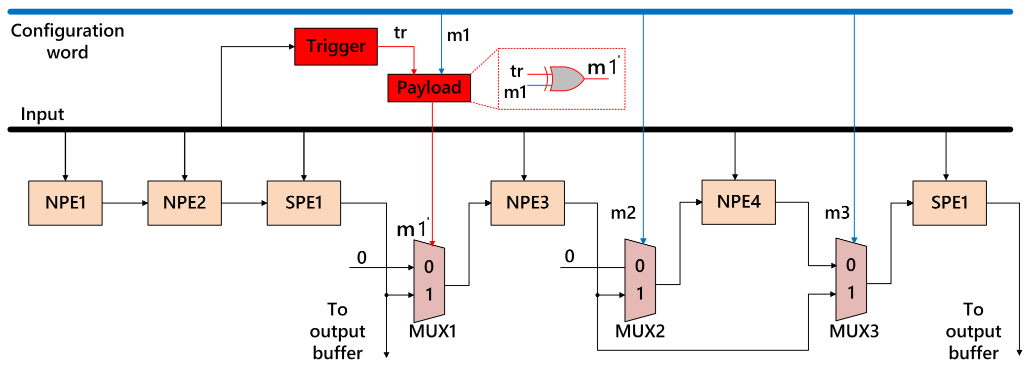


Figure 2. Hardware Trojan in RI.

Li et al. [16] leveraged the sparsity of neural networks to partition the original convolution kernel into two segments: one part is utilized to train harmful models using an incorrect training set, while the other part is employed for training normal models. In the hardware domain, the associated convolution kernel parameters are manipulated by setting the output of the multiplier in the multiply-add tree to 0. This transformation converts the normal model into a harmful one, resulting in erroneous image classification outcomes. The circuit structure of this hardware Trojan is illustrated in Figure 3a. In a separate study, Clements et al. introduced a hardware Trojan attack targeting the rectified linear unit (ReLU) activation function [17]. Figure 3b illustrates the circuit structure of the hardware Trojan, with its payload integrated into ReLU. The trigger signal for the Trojan is derived from the input of ReLU, and upon activation, the payload inverts the corresponding output bit.

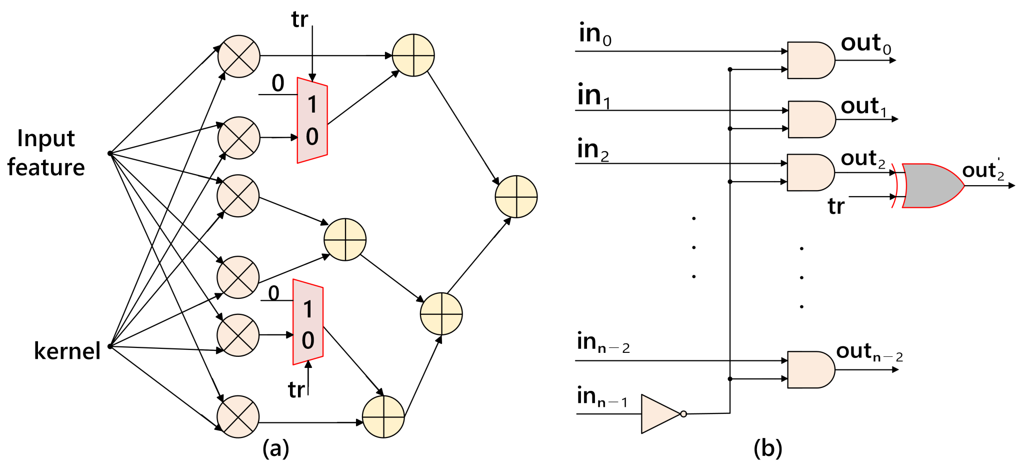


Figure 3. Hardware Trojan of (**a**) MAC and (**b**) ReLU.

This paper focuses on hardware Trojans inserted into the reconfigurable processing element (PE) array, leading to erroneous functional behavior within both the PE and reconfigurable interconnect (RI) to which the Trojan is inserted. The trigger mechanism involves utilizing a signal derived from the computation result of a specific PE to determine whether to activate the hardware Trojan. In terms of payload location, hardware Trojans are categorized into those inserted into the PE and those embedded in the RI. Figure 4 provides an overview of the targeted hardware Trojan model presented in this paper.

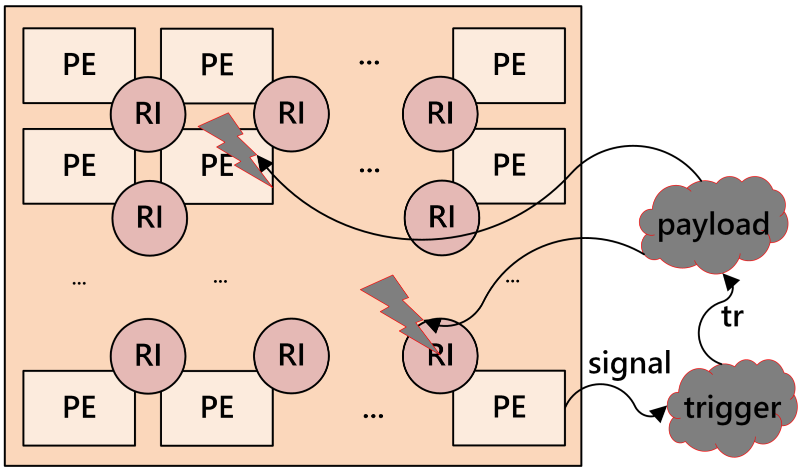


Figure 4. Overview of the targeting hardware Trojan.

2.3. Related Work and Motivation

2.3.1. Related Work

In recent years, convolution neural networks (CNNs) have found widespread application in diverse fields, prompting the development of various reconfigurable CNN accelerators to enhance network inference speed. However, the presence of hardware Trojans in reconfigurable neural network accelerators poses a serious threat to the accuracy of the final inference results [24]. While academia has proposed hardware Trojans for attacking CNN accelerators, the existing countermeasures against hardware Trojans implanted in these accelerators are limited.

Leibo Liu's proposed scheme [23] introduces a mapping approach, termed dynamic resource management, based on security value to enhance the capability of coarse-grained reconfigurable architectures against hardware Trojans. Kundu S's scheme [24] identifies faults in the data path of deep neural network accelerators caused by hardware Trojans and proposes strategies to obtain functional test patterns for detecting functional safety violations. Sun P's proposed scheme [25] introduces a hardware Trojan to modify the result of a pooling operation and suggests two countermeasures. S. Yang proposed a golden-free multidimensional self-referencing technique that analyzes the side-channel signatures in both the time and frequency domains to significantly broaden the Trojan coverage and strengthen the detection confidence [26].

2.3.2. Motivation

Notably, the above schemes exhibit shortcomings. Despite its potential, the scheme proposed by Leibo Liu has limitations, such as not specifically addressing hardware Trojans in CNN accelerators and lacking a defined security value for these accelerators. Also, the scheme proposed by Kundu S is non-eliminative as it only detects the presence of hardware Trojans and lacks real-time monitoring. And the scheme proposed by Sun P faces limitations, such as requiring a hard-to-obtain golden element and only being capable of detecting the presence of Trojans without eliminating their harmful effects. The scheme proposed by S. Yang has limitations as well, such as not specifically addressing hardware Trojans in CNN accelerators and cannot eliminate the harmful effect of hardware Trojans.

The majority of existing schemes for hardware Trojans in CNN accelerators fall short in preventing Trojan triggering. In light of these limitations, this paper proposes a scheme specifically targeting hardware Trojans in reconfigurable CNN accelerators. This scheme not only detects Trojans but also eliminates their harmful effects, prevents triggering, and does not necessitate a golden element.

3. The Novel Two-Level Protection Scheme

3.1. Overview

Two key approaches for mitigating hardware Trojans involve prevention and detection. The prevention strategy aims to either avert the triggering of the hardware Trojan altogether or impede its activation at the designated time intended by the attacker. Conversely, the detection and mitigation approach focus on identifying the triggering of the Trojan and neutralizing its impact after activation. In line with these concepts, this paper introduces a two-level prevention scheme against hardware Trojans tailored for CNN accelerators. An overview of the proposed protection scheme is presented in Figure 5. The first level, the trigger level, aims to thwart the triggering of hardware Trojans at the attacker's designated time. The second level, the payload level, is designed to detect Trojan triggering and subsequently mitigate the impact of the Trojans post-activation. Even if a hardware Trojan breach occurs in the first level of protection, the second level intercepts it, enhancing the overall security of the two-level protection scheme compared to a single-level defense. The trigger level incorporates a solution termed PE Space Randomization (PSR), utilized to prevent Trojan triggering, particularly those triggered by a specific PE's output. On the other hand, the payload level encompasses three solutions: Voting, Input–Output Relationship Detection (IORD), and PE Collaborative Correction (PCC). Voting detects triggered Trojans aimed at the reconfigurable interconnect (RI), correcting the incorrect select signal of RI based on the voting result. IORD identifies triggered Trojans targeting the MAC and ReLU computing modules within the PE. PCC involves PE mutual correction to rectify erroneous PE outputs. This comprehensive two-level scheme not only fortifies prevention measures but also enables effective detection and correction of hardware Trojans in CNN accelerators.

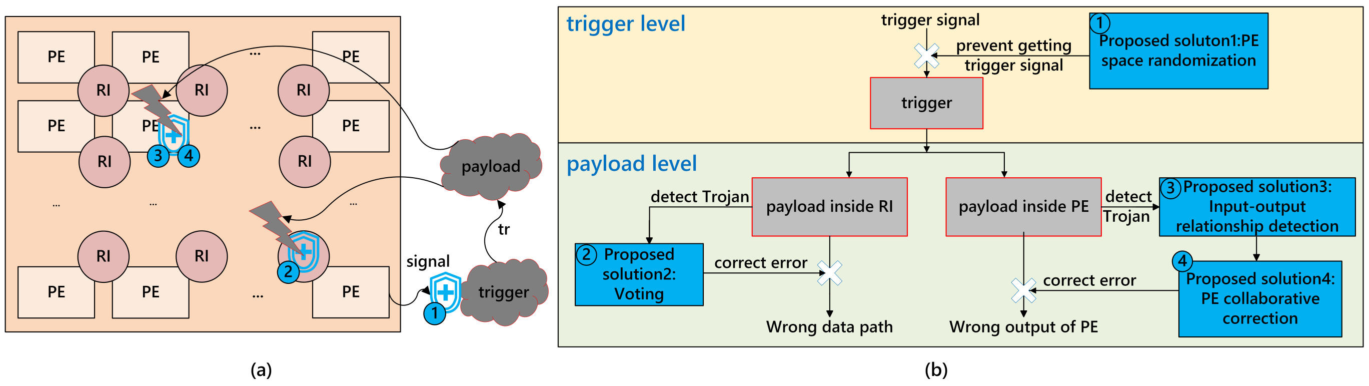

PE
RI
payload
signal
trigger
tr
(a)
trigger level
trigger signal
prevent getting trigger signal
1 Proposed soluton1:PE space randomization
trigger
payload level
detect Trojan
payload inside RI
payload inside PE
detect Trojan
3 Proposed solution3: Input-output relationship detection
2 Proposed solution2: Voting
correct error
correct error
4 Proposed solution4: PE collaborative correction
Wrong data path
Wrong output of PE
(b)

Figure 5. The proposed protection scheme. (**a**) Overview of the scheme; (**b**) detail of the scheme.

3.2. PE Space Randomization

The triggering methods employed by hardware Trojans can be categorized into two groups: random triggering and targeted triggering at specific moments as designed by the attacker. The latter type of attack is particularly pernicious as it enables the attacker to manipulate the classification results of a specific input photo. Numerous Trojans utilize intermediate calculation results of the inference process as trigger signals, sourced from specific processing elements (PEs) within the accelerator [15,17,20]. For instance, Odetola et al. proposed a hardware Trojan trigger method ensuring Trojan concealment by utilizing specific intermediate data as the trigger signal, derived from the output of a fixed PE at a specified time [15].

The dynamic reconfigurable computing architecture, with its abundant interconnection resources and the ability to swiftly remap the algorithm model to the PE array, allows for the calculation of specific intermediate data by different PEs with varying spatial locations on the chip. By changing the configuration word of RI through the configuration module, the source of input of each PE can be changed, so that the PE that calculates a specific intermediate datum becomes unfixed. The configuration module changes the configuration words of RI, and the input and output of PEs can be changed. Based on this capability, this paper introduces a scheme named "PE Space Randomization" to prevent hardware Trojans triggered by specific PEs at designated times. In a reconfigurable CNN accelerator, where multiple PEs simultaneously perform the same type of computation, such as convolution [10], it is assumed, without loss of generality, that N PEs ($PE_0 \sim PE_{(N-1)}$) are engaged in the same convolution operation. An attacker might use a specific output pixel (denoted as O_0) from a certain convolution layer as the trigger signal, assuming it was obtained from PE_0 on a regular CNN accelerator. However, on the reconfigurable CNN accelerator employing the "PE Space Randomization" scheme, O_0 can be obtained from a random PE, making it impossible for the attacker to retrieve the trigger signal from a fixed PE, as in the original scenario. Figure 6 illustrates the "PE Space Randomization" scheme, where O_i is derived from Input[i]. Through RI assignment, the input of PE_i becomes Input[(S+i) mod N] (S is a random number obtained from the configuration module). Given that S is random, the input and, consequently, the output of each PE become unpredictable. This paper assumes that the attacker lacks foreknowledge of our protection scheme. But even if the attacker knows this protection scheme in advance, when this defense mechanism is implemented, because S is a random number, the attacker cannot procure the specific output data needed as a trigger signal, rendering the hardware Trojan trigger scheme ineffective. As the number of PEs involved in the "PE Space Randomization" scheme (denoted as N for PSR) increases, the likelihood of the attacker obtaining the correct trigger signal diminishes.

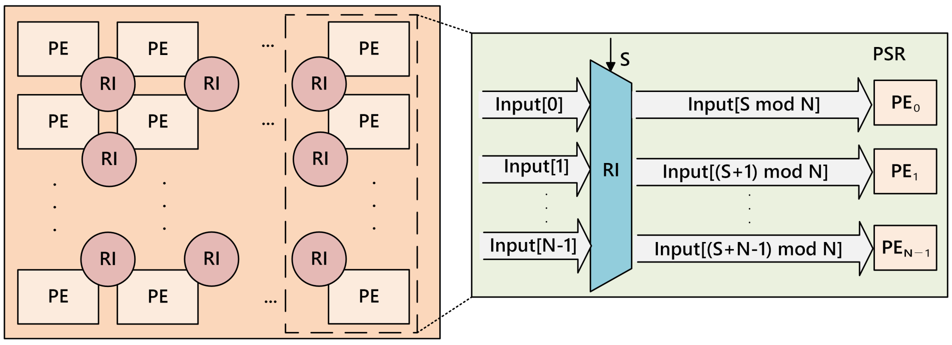


Figure 6. The "PE Space Randomization" scheme.

It is noteworthy that the S selection signal encapsulates the mapping information of the processing element (PE), allowing O_i to be obtained from $PE_{((N+i-S)modN)}$. Consequently, the implementation of the "PE Space Randomization" method has no adverse impact on the normal functionality of the accelerator. In conclusion, the "PE Space Randomization" approach effectively mitigates all hardware Trojan triggering schemes that rely on signals from spatially fixed PEs as trigger signals.

3.3. Voting

Targeting hardware Trojans with payloads inserted in dynamically reconfigurable interconnects [18], this paper introduces a voting-based method designed for the detection and correction of hardware Trojans within the reconfigurable interconnects of a CNN accelerator. In a reconfigurable CNN accelerator, multiple identical circuit structures, comprising processing elements (PEs) and their interconnected configurations, execute parallel operations. For the sake of concealment, attackers can only implant a limited number of hardware Trojans into the reconfigurable interconnects. Excessive Trojan insertions could lead to conspicuous chip area consumption, making the hardware Trojans more easily detectable. Consequently, even if a hardware Trojan alters the configuration word of the reconfigurable interconnects within the CNN accelerator, the majority of configuration words, which are the same under normal circumstances, remain unchanged. This paper presents the "Voting" scheme to counter reconfigurable interconnect hardware Trojans. The scheme exploits the fact that, in the absence of hardware Trojans, the interconnect selection signals should be identical. The Voting process compares these signals and, upon detecting a different configuration word, infers the triggering of an interconnect hardware Trojan. In such cases, Voting conducts a vote, and the resultant configuration word with the highest count becomes the voting result. This result is then employed to modify the configuration word signal. Without loss of generality, assuming that there are N identical circuit structures used to perform convolution operation in parallell, the configuration word of the nth circuit structure is named after m_n and the configuration word has K possible values. Figure 7 depicts an overview of Voting. Voting collects $m_0 \sim m_{(N-1)}$ as votes, and then use counters to count the number of times each value appears in $m_0 \sim m_{(N-1)}$. The MAX module outputs the max cnt. The m_{result} is the most frequent configuration word in m_0 to $m_{(N-1)}$ which wins the most votes. If m_i is different from m_{result}, detect[i] is set to 1 meaning that m_i is wrong and needs to be corrected by being replaced by m_{result}. The flow of the m_{result} generation of the Voting solution is shown in Algorithm 1.

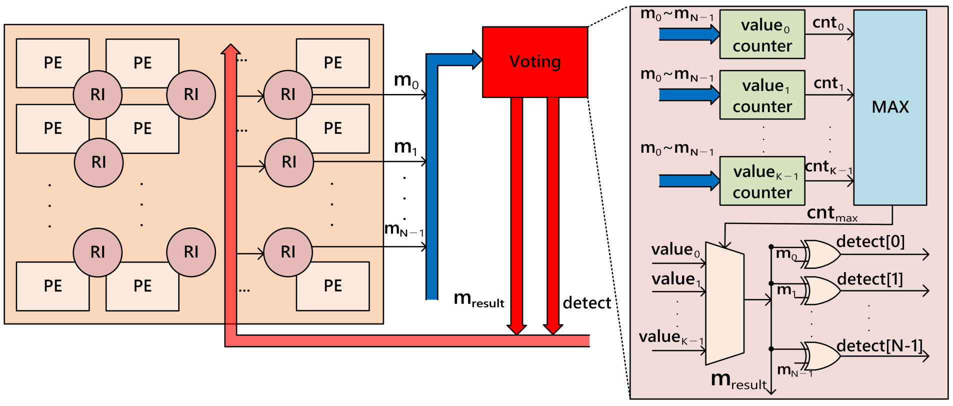


Figure 7. Overview of Voting.

Algorithm 1 Voting solution against hardware Trojan in RI

```
Input: N configuration words: $m_0 \sim m_{N-1}$;
       K possible values of m: $v_0 \sim v_{K-1}$
Output: voting result $m_{result}$; detect[0:N − 1]
1: for(i = 0; i < K; i = i + 1) do
2:     $cnt_0 = 0$
3:     for(j = 0; j < K; j = j + 1) do
4:         if($m_j == v_i$) $cnt_i = cnt_i$+1
5:         else $cnt_i = cnt_i$
6:     end for
7: end for
8: result = max($cnt_0$,$cnt_1$,...,$cnt_i$,...,$cnt_{K-1}$)
9: $m_{result} = v_{result}$
10: for(j = 0; j < N; j = j + 1) do
11:     if($m_j! = m_{result}$) detect[j] = 1
12:     else detect[j] = 0
13: end for
14: return $m_{result}$ and detect[0:N − 1]
```

In summary, the proposed Voting can detect the wrong configuration word of RI caused by a hardware Trojan and correct the wrong configuration word using the voting result.

3.4. Input–Output Relationship Detection

The MAC (multiply-accumulate) and ReLU (rectified linear unit) represent the primary computing modules within the processing element (PE) for convolution and are susceptible to hardware Trojans. Addressing the MAC hardware Trojan introduced in [16] and the ReLU hardware Trojan proposed in [17], this paper introduces the Input–Output Relationship Detection (IORD) method for detecting triggered hardware Trojans with payloads embedded within the PE.

IORD scrutinizes whether the input and output adhere to the correct relationship; any deviation indicates the triggering of a hardware Trojan. In the case of the MAC hardware Trojan outlined in [16], where the results of certain multipliers are set to 0 upon triggering, Figure 8 elucidates the detection details and $in1$ and $in2$ are the two inputs of the multiplier. Specifically, if both $in1$ and $in2$ of the scrutinized multiplier are non-zero, yet the output is 0, it signifies the triggering of the MAC hardware Trojan, and the detect signal is set to 1. For instance, when $in1$ = 2 and $in2$ = 3, with both inputs of AND1 registering as 1, the output of AND1 is 1. If the MAC hardware Trojan proposed in [16] is activated, causing the multiplier's computation results to be set to 0, the inputs of AND1 will remain 1, resulting in a detect signal of 1 for AND2. This detect signal can be expressed using the following equation:

$$detect = (in1! = 0)\&(in2! = 0)\&(out == 0) \tag{1}$$

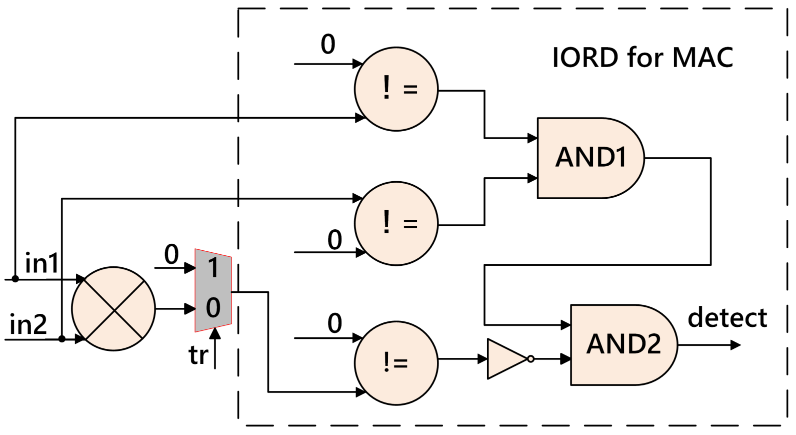


Figure 8. IORD for MAC.

If the ReLU hardware Trojan proposed in [17] is activated, a specific bit in the computation results of the ReLU module undergoes inversion. Concerning the detection of this ReLU hardware Trojan, Figure 9 provides a detailed illustration. To illustrate, without loss of generality, consider the second bit of ReLU's output as the protected bit. If the most significant bit (MSB) of the input ($in_{(n-1)}$) is 1, signifying a negative input for ReLU, the out_2 would be 0. Upon triggering the ReLU hardware Trojan from [17], the out_2 is inverted to 1. Given $in_{(n-1)}$ = 1, detect = out_2, as the inversion of out_2 to 1 signifies the triggering of the ReLU hardware Trojan. Conversely, when $in_{(n-1)}$ is 0 and in_2 does not match out_2, detect = $in_i \oplus out_i$ = 1, indicating the triggering of the ReLU hardware Trojan. Assuming the ith bit of ReLU's output is the protected bit, the corresponding detect can be determined using the following equation:

$$detect = in_{(n-1)}?out_i : in_i \oplus out_i \tag{2}$$

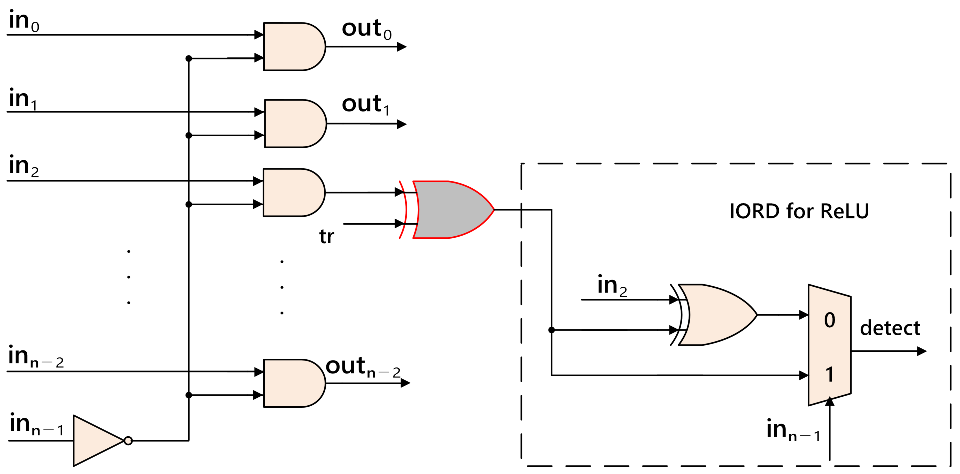


Figure 9. IORD for ReLU.

In summary, the proposed IORD solution checks the relationship between the input and output of the computing block inside PE, which detects the triggered hardware Trojan whose payload is inside the PE.

3.5. PE Collaboration Correction

When a reconfigurable CNN accelerator falls victim to a hardware Trojan attack, merely detecting the trigger of the hardware Trojan is insufficient to ensure the proper functionality of the CNN accelerator. In response, this paper introduces a real-time correction solution termed "PE Collaborative Correction", designed to rectify errors induced by hardware Trojans through collaborative efforts among processing elements (PEs). Following the detection of a triggered hardware Trojan, this method enables the neural network accelerator to operate accurately, mitigating the impact of the hardware Trojan trigger.

The "PE Collaborative Correction" divides PEs performing identical convolution operations into groups of four. Upon detecting a hardware Trojan trigger in a PE within a designated group, all modules in the entire accelerator, except the affected PE group, cease operation for N clock cycles. And N is the number of clock cycles required for PE to complete an operation. During these cycles, other PEs preserve their calculation results, and the computations intended for the PE with the triggered hardware Trojan are rerouted to other PEs within the group without the triggered hardware Trojan. When a hardware Trojan trigger is detected in a PE group, the reconfiguration request is sent to the RI. This signal serves as a control signal for the RI to reconfigure the input and output paths of the four PEs in the PE group. Subsequently, in the subsequent clock cycle, the PE without the triggered hardware Trojan outputs the previously saved calculation results, while the PE with the triggered hardware Trojan outputs the corrected calculation results provided by the other normal PEs. This process ensures correct outputs from all PEs, eliminating the harm caused by hardware Trojans implanted in the PEs. A schematic representation of the PE Collaborative Correction (PCC) solution is illustrated in Figure 10. Within this configuration, four PEs constitute a group, and the input scheduler distributes four different inputs to the PEs in the group for computation. The output scheduler coordinates the output of the four PEs to four distinct output interfaces. The scheduling strategy of the input scheduler and output scheduler is contingent upon the presence of triggered hardware Trojans in the group, yet irrespective of the scheduling, out_i consistently corresponds to in_i.

There are four potential scenarios:

1. When a group of PEs is free from any hardware Trojan triggers, in_i corresponds to the input of PE_i, and out_i represents the computation result of PE_i.
2. In the event of a hardware Trojan implantation in PE_i triggering its operation, the other modules of the accelerator stall for one clock cycle. During this stalling period, in_i serves as the input for PE_x, where x is determined by the equation $x = (i+1)\text{mod}4$, and

out_i denotes the computation result of PE_x. Other output values remain unchanged from their states before the accelerator stall.

3. In scenarios where two PEs in a group triggered hardware Trojans, resulting in their activation, the accelerator's other modules stall for one cycle. The input data designated for these two PEs are processed by the remaining two normal PEs, and the corresponding out will be changed to the output of the other two normal PEs. The out values of the two normal PEs remain unaltered from their states before the accelerator stall.
4. When three or more PEs within a group have been inserted with hardware Trojans and subsequently triggered, the effectiveness of the "PE Co-Correction" scheme diminishes, rendering it unable to rectify the issue. Nonetheless, given the stealthy nature of hardware Trojans, the attacker typically implants only a small number of them. As a result, instances where three or more PEs within a group are both inserted with hardware Trojans and triggered are exceedingly rare.

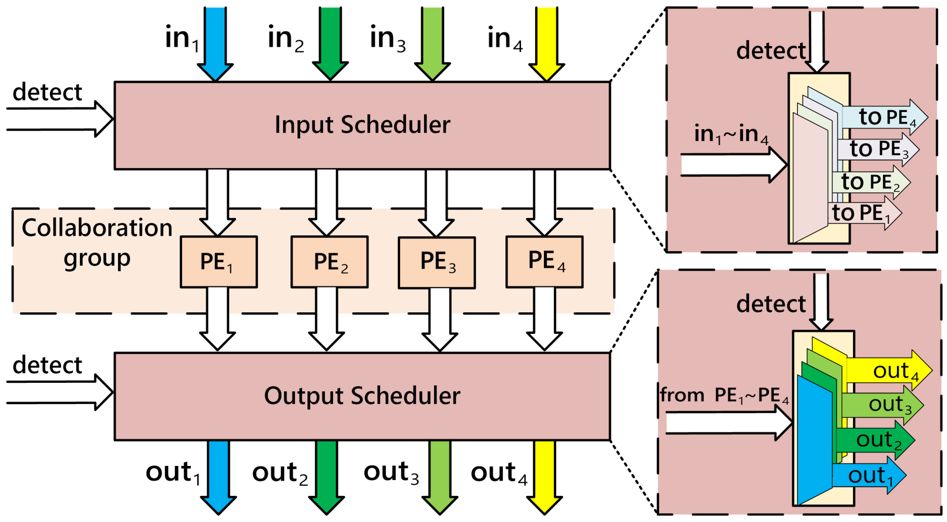


Figure 10. The PE collaboration correction solution.

In summary, the proposed PCC solution effectively mitigates the harm caused by hardware Trojans by collaboratively correcting errors among PEs following the detection of a triggered hardware Trojan.

4. Experimental Results

4.1. Experimental Setup

4.1.1. Benchmark

To conduct the assessment, the Cat VS Dog Dataset was employed to train the AlexNet. This analysis can be extended to different CNN models with various datasets. And a prototype of RNA was utilized to model the CNN for evaluation. The CNN architecture, as outlined by Chen Yang [10], was mapped onto RNA, featuring 22 × 22 PEs.

4.1.2. Experiment Procedure

To assess the effectiveness of the hardware Trojan protection scheme proposed in Section 2, this paper conducted a behavioral simulation experiment, employing the following procedures:

1. Train AlexNet in TensorFlow: Classify the Cat VS Dog Dataset using AlexNet in TensorFlow.
2. Setup Trigger:
   a The trigger setup, proposed by [21], involves assessing the statistical properties of the output feature maps of the first convolution layer of AlexNet to establish

a trigger. Select a specific output feature map pixel of the first convolution layer randomly as the trigger signal. Set the trigger interval based on the data characteristics of the pixel to achieve a triggering probability of 2/10,000 in ordinary scenes.

b Modify the input photographs to ensure that the value of the output pixel selected as the trigger signal falls within the Trojan trigger interval.

3. Randomly Inject Hardware Trojan:

    a Under varying hardware Trojan attack severities, randomly inject RI hardware Trojans proposed by Chen Yang [18] in RIs of RNA.

    b Under different attack severities, randomly inject MAC hardware Trojans proposed by [16] and ReLU hardware Trojans proposed by [17] in PEs of RNA. Attack severity refers to the proportion of PEs or RIs inserted with hardware Trojans among all PEs or RIs, ranging from 1% to 5%.

4. Configure Defense Schemes: Configure all the defense schemes proposed in Section 2.
5. Conduct Hardware Trojan Attacks: Under varying hardware Trojan attack severities, perform 1000 hardware Trojan attacks. In cases where there is no protection, assume that the hardware Trojan in every attack is 100% triggered. Record whether each hardware Trojan in each experiment was triggered, discovered, and whether the harm caused was eliminated.
6. Evaluate Protection Scheme Effectiveness: Based on the experimental results, evaluate the effectiveness of each protection scheme proposed in Section 2 under different hardware Trojan attack severities. In our experiment, hardware Trojans are randomly injected, and the severity of the attack refers to the proportion of PEs or RIs affected by hardware Trojans to the total number of PEs or RIs.

4.2. Evaluation

The proposed hardware Trojan protection scheme is assessed from three perspectives: protection effectiveness, hardware, and power overhead, and comparison with other existing schemes.

4.2.1. Protection Effectiveness

1. PSR Protection Effectiveness:

In the absence of a protection scheme on the reconfigurable CNN accelerator, the trigger probability of inserted hardware Trojans reaches 100%, as the input photographs are deliberately modified to ensure Trojan activation. However, with the deployment of the PE Space Randomization (PSR) scheme in the CNN accelerator, the probability of triggering the hardware Trojan significantly diminishes from 100% to 2.4%. This outcome underscores the robust effectiveness of the PSR scheme in preventing hardware Trojans from being triggered.

In the PSR solution, through the dynamic reconfiguration, by changing the configuration word of RI through the configuration module, the source of input of each PE can be changed, so that the PE that calculates a specific intermediate datum becomes unfixed. When the output of PE is random, the probability of the attacker getting the correct trigger signal is very low. But they still have the possibility of getting the correct trigger signal to trigger the hardware Trojan. For example, if the attacker needs signal A to trigger the hardware Trojan, but signal A comes randomly from N PEs, then the probability of the attacker successfully getting signal A is 1/N; thus, there is still a 2.4% gap for the PSR solution.

2. Voting Protection Effectiveness:

Figure 11 illustrates the protection effectiveness of the Voting scheme under various hardware Trojan attack severities. Protection effectiveness denotes the probability of the Voting scheme successfully mitigating the harm caused by all hardware Trojans in the reconfigurable interconnects (RI) and enabling the normal operation of the accelerator. The figure reveals that when 1% to 5% of RIs are inserted with hardware Trojans, the Voting protection solution can achieve a 100% detection rate for the presence of hardware Trojans and eliminate 100% of the harm caused by these Trojans.

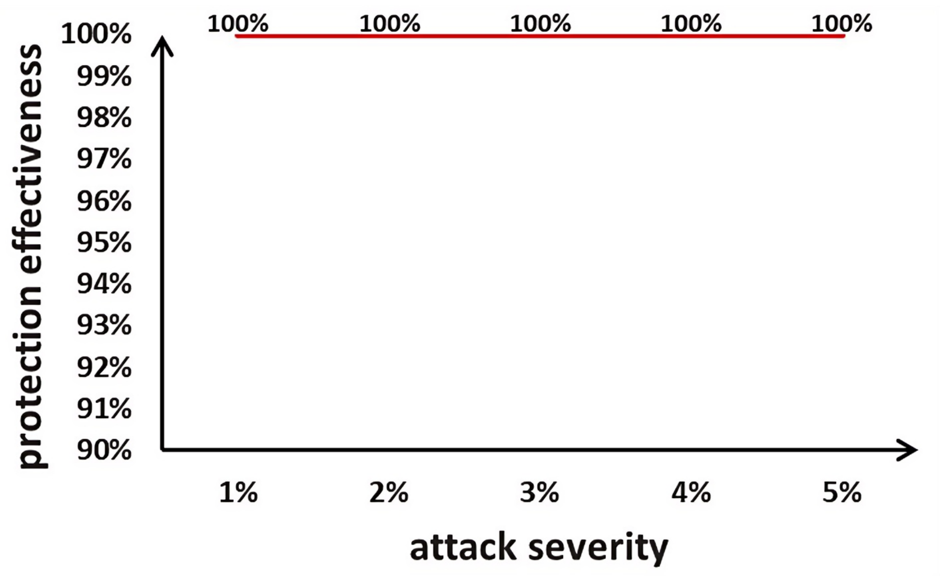


Figure 11. Protection effectiveness of Voting.

3. Voting Protection Effectiveness:

The Input–Output Relationship Detection (IORD) scheme is employed for hardware Trojan detection within the processing elements (PE). Experimental results demonstrate that, irrespective of the severity of the attack, the IORD scheme consistently achieves a 100% detection rate for triggered hardware Trojans.

The Processing Element Collaborative Correction (PCC) scheme is designed to mitigate the impact of hardware Trojans within the PE, offering protection under varying hardware Trojan attack severities. As mentioned in Section 3.5, the PE of the accelerator is divided into groups of four, with a total of 121 groups. Since the hardware Trojans were randomly injected into the PE in the experiment, the four scenarios mentioned in Section 3.5 would appear randomly in 1000 repeated experiments. When scenario 4 occurs, that is, more than two PEs in a group of PEs are injected with hardware Trojans, the PCC protection solution fails. In a thousand repeated experiments, the total number of PE groups is 121,000. The number of PE groups in scenario 1 is 107,121, the number of PE groups in scenario 2 is 13,273, the number of PE groups in scenario 3 is 484, and the number of PE groups in scenario 4 is 122. This experimental result shows that, as mentioned in Section 3.5, the probability of scenario 4 occurring is 0.1%, which is really small.

Figure 12 depicts the protection effectiveness of the PCC scheme. Protection effectiveness, in this context, refers to the likelihood of the PCC scheme successfully neutralizing the harm induced by all hardware Trojans in the PE, thereby enabling normal accelerator functionality. As the attack severity escalates from 1% to 5%, a marginal decrease in protection effectiveness is observed. However, even under a highly severe attack scenario with an attack severity of 5%, the PCC scheme exhibits a noteworthy 95.3% probability of eliminating the harm caused by hardware Trojans in the PE and restoring normal accelerator operation.

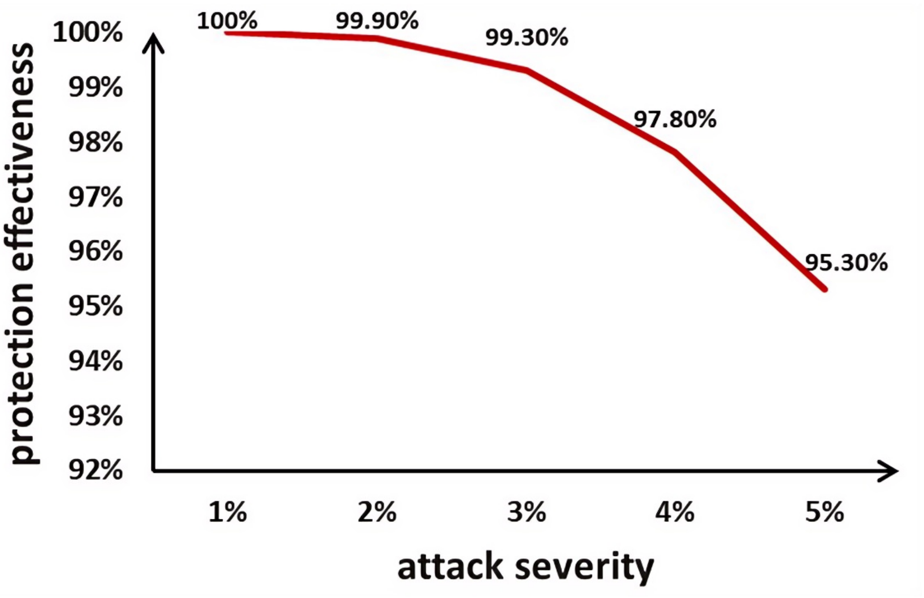


Figure 12. Protection effectiveness of the PCC.

4. Overall Scheme Results:

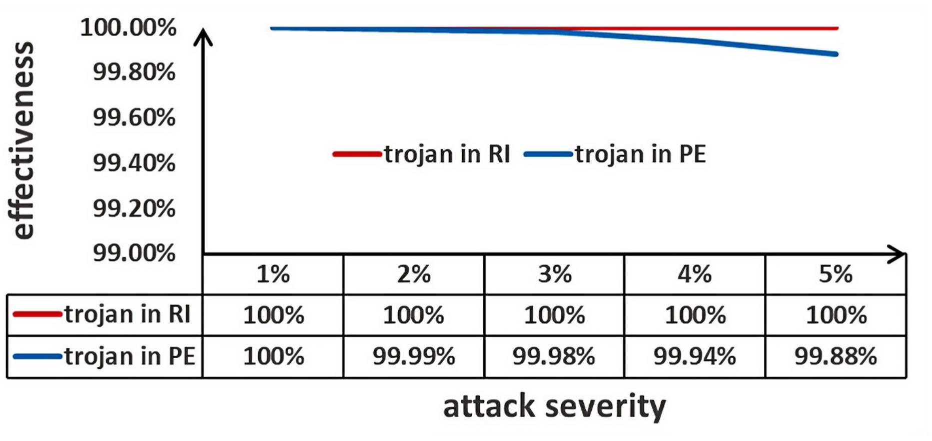
For a hardware Trojan to inflict harm upon a reconfigurable CNN accelerator, it must undergo two distinct processes. Initially, the hardware Trojan's trigger is activated based on a trigger signal. Subsequently, when triggered, the payload executes harmful actions. Disrupting either of these processes is imperative to neutralize the impact of hardware Trojans. To address this, our paper introduces a two-level prevention scheme against hardware Trojans for CNN accelerators, effectively impeding both processes.

Figure 13 illustrates the protection effectiveness of this two-level prevention scheme. The protection effectiveness is the probability of the scheme eliminating all harm induced by hardware Trojans and restoring normal accelerator functionality. The protection effectiveness of this two-level prevention scheme can be computed by the following equation:

$$\begin{aligned} Effectiveness_{two-level} &= Effectiveness_{trigger} \\ &\quad + Effectiveness_{payload} \times (1 - Effectiveness_{trigger}) \end{aligned} \tag{3}$$

In Equation (3), $Effectiveness_{two-level}$ is the protection effectiveness of the two-level scheme; $Effectiveness_{trigger}$ is the protection effectiveness of the trigger level; and $Effectiveness_{payload}$ is the protection effectiveness of the payload level.

The results indicate that when 1% to 5% of reconfigurable interconnects (RIs) are inserted with hardware Trojans, the two-level prevention scheme achieves a 100% success rate in neutralizing all harm caused by hardware Trojans. Regarding hardware Trojans in processing elements (PEs), the effectiveness is 100%, 99.99%, 99.98%, 99.94%, and 99.88% for attack severities of 1% to 5%, respectively. The experimental findings affirm that our proposed two-level prevention scheme can thwart over 99.8% of hardware Trojan attacks targeting reconfigurable CNN accelerators.



	1%	2%	3%	4%	5%
trojan in RI	100%	100%	100%	100%	100%
trojan in PE	100%	99.99%	99.98%	99.94%	99.88%

Figure 13. Protection effectiveness of two-level prevention scheme.

4.2.2. Hardware Overhead

The RNA [10] incorporating the proposed two-level protection scheme is instantiated on a Xilinx Zynq XC7Z100 platform. Table 1 delineates the hardware and power overheads of the pristine accelerator (lacking a protection scheme) juxtaposed with the fortified accelerator (integrated with the protection scheme). The outcomes reveal that the hardware overhead induced by the proposed protection scheme registers an uptick of 32.1%. Although the original RNA consists of 22 11-bit× 256 filter memories, a ping-pong 2.56 KB image memory, 22 704B multi-bank RAMs as output buffer, 1452 DSP and 86 K LUT, there is no overhead of DRAMs and DSPs caused by the application of the protection scheme for the reason that the hardware overhead is mainly composed of the following two parts: reconfigurable interconnects used by the PSR and PCC solutions, and logic resources used by the voting and IORD solutions.

And the power overhead induced by the proposed protection scheme registers an uptick of 12.2%.

Table 1. Hardware and power overheads comparison between the original accelerator and the protected accelerator (containing protection scheme).

	Original Accelerator	Protected Accelerator	Overhead Comparison
LUT	86 K	113.6 K	32.1% ↑
Power	5.50 W	6.17 W	12.2% ↑

4.2.3. Comparison with Other Existing Schemes

Since the types of hardware Trojans we deal with are not dealt with by other works, it is difficult to make a comparison with other works in terms of area/power overheads and the capability to address attacks. Table 2 provides a comparison between the proposed scheme and other existing schemes for a reconfigurable architecture or CNN accelerators against hardware Trojans. The comparison includes whether it is designed for CNN, whether it can prevent hardware Trojans triggering, whether it can eliminate harmful effect of Trojans, whether it can work in real time, and whether it requires a golden element. The above comparison can measure the quality of a protection scheme against hardware Trojans on a reconfigurable CNN accelerator [27].

Table 2. Comparison between the scheme proposed by this paper and other schemes.

Scheme Comparison [27]	Scheme Proposed by Leibo Liu [23]	Scheme Proposed by Kundu S [24]	Scheme Proposed by Sun P [25]	Scheme Proposed by S Yang [26]	Scheme Proposed by This Paper
Design for CNN	✗	✓	✓	✗	✓
Can prevent triggering	✗	✗	✗	✗	✓
Can eliminate harmful effect	✓	✗	✗	✗	✓
Work in real-time	✓	✗	✓	✗	✓
Do not need golden element	✓	✓	✗	✓	✓

5. Discussion

This paper introduces a real-time and golden element-free two-level protection scheme designed for reconfigurable CNN accelerators. The scheme comprises two protective levels: the trigger level and the payload level. At the trigger level, the scheme successfully averts the triggering of 97.6% of hardware Trojans. The payload level is dedicated to the detection and eradication of hardware Trojans, mitigating their detrimental effects. Across hardware Trojan attack severities ranging from 1% to 5%, the payload level consistently eliminates 100% of the harm induced by hardware Trojans in the reconfigurable interconnects (RI) and, impressively, a minimum of 95.3% of the harm in the processing elements (PE). The synergistic application of both protection levels results in a comprehensive harm mitigation of at least 99.88% caused by hardware Trojans.

Although the XC7Z100 platform is a prototype verification platform for the proposed scheme, our scheme is implemented by Verilog code, which is highly portable. Therefore, the result of the protection effectiveness of our scheme is independent of the hardware platform.

The experimental results of the two-level scheme are insensitive to the CNN model and datasets. For example, our IORD and PCC solutions protect each multiplication and addition operation and ReLu operation from the harm of hardware Trojans. These operations are the basic operations of CNN. No matter what the CNN model and datasets are, these operations are the same, just the number is different. Therefore, for different CNN models and datasets, IORD and PCC solutions can effectively protect the normal operation of the accelerator. For other examples, for different CNN models and datasets, the configuration word of RI is different, but the voting solution can protect RI from the harm of hardware Trojans no matter what the configuration word is. Therefore, for different CNN models and datasets, voting solutions can effectively protect the RI from the harm of hardware Trojans. Furthermore, the PCC solution is insensitive to the CNN model and datasets, for the reason that no matter what the CNN model and datasets are, the signals of the PE are random and an attack cannot get the correct signal for a trigger.

In our upcoming research, we intend to scrutinize the multifaceted effects of diverse hardware Trojans on convolution neural network (CNN) accelerators. Our goal is to devise and implement improved countermeasures to strengthen CNN accelerators against potential hardware threats. To ensure our research is anchored in a rigorous and methodologically sound experimental framework, we plan to integrate several widely used open-source and publicly available accelerators into our experiments. This strategy will help validate our findings, thereby enhancing the credibility and reproducibility of our results. Our ongoing research efforts are poised to make significant contributions to the advancement of hardware security in the context of CNN accelerators. However, the hardware and power overhead of our proposed scheme may be deemed high for certain applications. To address this concern, we will suggest potential optimization strategies to alleviate this overhead.

Our solution assumes that the attacker does not know the details of our protection solution. But this is not always true in reality. If the attacker knows our solution in advance, the attacker can bypass our detection of the Trojan, and the payload-level protection measures will fail. But even if the attacker knows our solution in advance, the trigger-level protection measures are still effective. Due to the randomization of the PE space, the attacker cannot obtain a specific signal to trigger the hardware Trojan. The confrontation between hardware Trojan attacks and their prevention methods is an ongoing process. With the continuous advancement of technology, attackers and defenders are keeping abreast of each other's latest strategies and constantly improving their respective technical levels. This "spear and shield" relationship has promoted the continuous development and progress of the hardware security field. Continuous research and innovation are the key to ensuring the security of hardware systems. We believe that our work has great potential in enhancing the hardware security of CNN accelerators.

Author Contributions: Conceptualization, Z.L. and C.Y.; methodology, Z.L.; software, Z.L.; validation, J.H. and J.W.; formal analysis, J.H.; investigation J.W.; resources, Z.L.; data curation, Z.L.; writing original draft preparation, Z.L. and J.H.; writing-review and editing, Z.L. and J.W.; visualization, Z.L.; supervision, C.Y.; project administration, C.Y.; funding acquisition, C.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China, grant number 62176206.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
2. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
3. He, R.; Wu, X.; Sun, Z.; Tan, T. Wasserstein CNN: Learning invariant features for NIR-VIS face recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *41*, 1761–1773. [CrossRef] [PubMed]
4. Miao, G. Application of CNN-based Face Recognition Technology in Smart Logistics System. In Proceedings of the 2021 20th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), Nanning, China, 10–12 December 2021; pp. 100–103.
5. Shen, D.; Wu, G.; Suk, H.I. Deep learning in medical image analysis. *Annu. Rev. Biomed. Eng.* **2017**, *19*, 221–248. [CrossRef] [PubMed]
6. Li, F.; Liu, Z.; Chen, H.; Jiang, M.; Zhang, X.; Wu, Z. Automatic detection of diabetic retinopathy in retinal fundus photographs based on deep learning algorithm. *Transl. Vis. Sci. Technol.* **2019**, *8*, 4. [CrossRef] [PubMed]
7. Liu, L.; Zhu, J.; Li, Z.; Lu, Y.; Deng, Y.; Han, J.; Yin, S.; Wei, S. A survey of coarse-grained reconfigurable architecture and design: Taxonomy, challenges, and applications. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–39. [CrossRef]
8. Chen, Y.H.; Krishna, T.; Emer, J.S.; Sze, V. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J.-Solid-State Circuits* **2016**, *52*, 127–138. [CrossRef]
9. Yin, S.; Ouyang, P.; Tang, S.; Tu, F.; Li, X.; Zheng, S.; Lu, T.; Gu, J.; Liu, L.; Wei, S. A high energy efficient reconfigurable hybrid neural network processor for deep learning applications. *IEEE J. -Solid-State Circuits* **2017**, *53*, 968–982. [CrossRef]
10. Yang, C.; Hou, J.; Wang, Y.; Zhang, H.; Wang, X.; Geng, L. RNA: A Flexible and Efficient Accelerator Based on Dynamically Reconfigurable Computing for Multiple Convolutional Neural Networks. *J. Circuits Syst. Comput.* **2022**, *31*, 2250289. [CrossRef]
11. Yang, C.; Wang, Y.; Wang, X.; Geng, L. WRA: A 2.2-to-6.3 TOPS highly unified dynamically reconfigurable accelerator using a novel Winograd decomposition algorithm for convolutional neural networks. *IEEE Trans. Circuits Syst. Regul. Pap.* **2019**, *66*, 3480–3493. [CrossRef]
12. Fujii, T.; Toi, T.; Tanaka, T.; Togawa, K.; Kitaoka, T.; Nishino, K.; Nakamura, N.; Nakahara, H.; Motomura, M. New generation dynamically reconfigurable processor technology for accelerating embedded AI applications. In Proceedings of the 2018 IEEE Symposium on VLSI Circuits, Honolulu, HI, USA, 18–22 June 2018; pp. 41–42.
13. Liu, L.; Li, Z.; Yang, C.; Deng, C.; Yin, S.; Wei, S. HReA: An energy-efficient embedded dynamically reconfigurable fabric for 13-dwarfs processing. *IEEE Trans. Circuits Syst. Express Briefs* **2017**, *65*, 381–385. [CrossRef]
14. Halak, B. Cist: A threat modelling approach for hardware supply chain security. In *Hardware Supply Chain Security: Threat Modelling, Emerging Attacks and Countermeasures*; Springer: Cham, Switzerland, 2021; pp. 3–65.
15. Odetola, T.A.; Mohammed, H.R.; Hasan, S.R. A stealthy hardware trojan exploiting the architectural vulnerability of deep learning architectures: Input interception attack (iia). *arXiv* **2019**, arXiv:1911.00783.

16. Li, W.; Yu, J.; Ning, X.; Wang, P.; Wei, Q.; Wang, Y.; Yang, H. Hu-fu: Hardware and software collaborative attack framework against neural networks. In Proceedings of the 2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Hong Kong, China, 8–11 July 2018; pp. 482–487.
17. Clements, J.; Lao, Y. Hardware trojan design on neural networks. In Proceedings of the 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 26–29 May 2019; pp. 1–5.
18. Yang, C.; Hou, J.; Wu, M.; Mei, K.; Geng, L. Hardware trojan attacks on the reconfigurable interconnections of convolutional neural networks accelerators. In Proceedings of the 2020 IEEE 15th International Conference on Solid-State & Integrated Circuit Technology (ICSICT), Kunming, China, 3–6 November 2020; pp. 1–3.
19. Ye, J.; Hu, Y.; Li, X. Hardware trojan in fpga cnn accelerator. In Proceedings of the 2018 IEEE 27th Asian Test Symposium (ATS), Hefei, China, 15–18 October 2018; pp. 68–73.
20. Liu, Z.; Ye, J.; Hu, X.; Li, H.; Li, X.; Hu, Y. Sequence triggered hardware trojan in neural network accelerator. In Proceedings of the 2020 IEEE 38th VLSI Test Symposium (VTS), San Diego, CA, USA, 5–8 April 2020; pp. 1–6.
21. Odetola, T.A.; Hasan, S.R. Sowaf: Shuffling of weights and feature maps: A novel hardware intrinsic attack (hia) on convolutional neural network (cnn). In Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS), Daegu, Republic of Korea, 22–28 May 2021; pp. 1–5.
22. Odetola, T.A.; Khalid, F.; Mohammed, H.; Sandefur, T.C.; Hasan, S.R. Feshi: Feature map-based stealthy hardware intrinsic attack. *IEEE Access* **2021**, *9*, 115370–115387. [CrossRef]
23. Liu, L.; Zhou, Z.; Wei, S.; Zhu, M.; Yin, S.; Mao, S. DRMaSV: Enhanced capability against hardware trojans in coarse grained reconfigurable architectures. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2017**, *37*, 782–795. [CrossRef]
24. Kundu, S.; Banerjee, S.; Raha, A.; Natarajan, S.; Basu, K. Toward functional safety of systolic array-based deep learning hardware accelerators. *IEEE Trans. Very Large Scale Integr. (Vlsi) Syst.* **2021**, *29*, 485–498. [CrossRef]
25. Sun, P.; Halak, B.; Kazmierski, T. Towards Hardware Trojan Resilient Design of Convolutional Neural Networks. In Proceedings of the 2022 IEEE 35th International System-on-Chip Conference (SOCC), Belfast, UK, 5–8 September 2022; pp. 1–6.
26. Yang, S.; Hoque, T.; Chakraborty, P.; Bhunia, S. Golden-free hardware trojan detection using self-referencing. *IEEE Trans. Very Large Scale Integr. (Vlsi) Syst.* **2022**, *30*, 325–338. [CrossRef]
27. Xu, Q.; Arafin, M.T.; Qu, G. Security of neural networks from hardware perspective: A survey and beyond. In Proceedings of the 26th Asia and South Pacific Design Automation Conference, Tokyo, Japan, 18–21 January 2021; pp. 449–454.