Hindawi
IET Information Security
Volume 2024, Article ID 2023349, 15 pages
https://doi.org/10.1049/2024/2023349

Research Article

DHRCA: A Design of Security Architecture Based on Dynamic Heterogeneous Redundant for System on Wafer

Bo Mei, Zhengbin Zhu, Peijie Li, and Bo Zhao

PLA Information Engineering University, Zhengzhou 450000, China

Correspondence should be addressed to Zhengbin Zhu; zzb_2385@163.com

Received 21 August 2023; Revised 21 February 2024; Accepted 29 March 2024; Published 12 April 2024

Academic Editor: Taimur Bakhshi

System on Wafer (SoW) based on chiplets may be implanted with hardware Trojans (HTs) by untrustworthy third-party chiplet vendors. However, traditional HTs protection techniques cannot guarantee complete protection against HTs, which poses a great challenge to the hardware security of SoW. In this paper, we propose a computing architecture based on endogenous security theory—dynamic heterogeneous redundant computing architecture (DHRCA) that can tolerate and detect HTs at runtime. The security of our approach is analyzed by building a generalized stochastic coloring petri net (GSCPN) model of DHRCA. The simulation results based on the GSCPN model show that our method can improve the system security probability to 0.8690 and the system availability probability to 0.9750 in the steady state compared with typical triple-mode redundancy and runtime monitoring methods. Furthermore, the impact of different attack and defense strategies on system security of different methods is simulated and analyzed in this paper.

1. Introduction

With the rise of new-generation information technologies epitomized by artificial intelligence and the explosive growth of data in modern society, the demand for computational power within computing systems has reached unprecedented heights. In the realm of integrated circuits, there are several methods aimed at enhancing computational capabilities, including improved process technologies, increased chip area, and adoption of state-of-the-art packaging techniques. However, traditional approaches have encountered bottlenecks due to the physical limits of processes, wafer yield restrictions, and thermal constraints imposed by packaging. As a result, more and more researchers have shifted their focus towards chiplet-based integration systems recently, exemplified by AMD's "zen2" processor [1], Intel's [2] Ponte Vecchio, and Tesla's DOJO [3].

The System on Wafer (SoW) is a chiplet-based integration system characterized by a higher number of integrated chiplets and a larger system scale. By integrating bare die directly onto the wafer substrate without packaging, the SoW can achieve communication bandwidth, energy consumption, and latency that closely resemble those of on-chip systems [4]. However, to achieve lower costs and faster iteration speeds, developers of SoW often integrate commercial chiplets from multiple untrusted third-party sources based on their specific requirements. This practice, unfortunately, gives rise to grave concerns regarding hardware security, particularly the prevalence of hardware Trojans (HTs) issues.

The SoW, a paradigm built upon the foundation of silicon, represents a novel information infrastructure. However, the fundamental challenges about application, system, and network security within the SoW cannot be adequately addressed unless the issue of HTs is duly considered. Unfortunately, current research in the field of SoW predominantly focuses on interconnect network technology, as well as the assembly and integration of chiplets onto the crystalline substrate, neglecting comprehensive investigations into the problem of HTs. Existing defense techniques against HTs do not offer a foolproof solution that ensures complete resilience against their insidious attacks. What's more, traditional studies often assume untrusted entities to be offshore chip manufacturers or third-party IPs integrated within individual chiplets. However, any entity within the commercial chiplet

supply chain of the SoW has the potential to introduce malicious HTs during the design or manufacturing process. The conventional approaches fall short when applied to SoW incorporating commercial chiplets. Therefore, there is an urgent need to develop an HTs defense method for SoW.

This article presents a secure computing architecture for SoW without altering the underlying hardware design. The proposed architecture enables runtime detection of tampered outputs and denial-of-service HTs, while also exhibiting resilience against data leakage HTs. These advancements enhance the security of SoW. The specific contributions of this paper are as follows:

(i) In this paper, we conduct an analysis of the HT threats to SoW in terms of the difficulty of implantation and defense. In a pioneering effort, we propose a secure computing architecture for SoW that embraces dynamic heterogeneous redundancy. This architecture effectively harnesses the unique characteristics of heterogeneous redundant chiplets within SoW, while leveraging its dynamism. Furthermore, it can enhance system security without incurring additional hardware development and integration costs. Furthermore, we give a comprehensive description of the key security mechanisms embedded within this proposed computing architecture.

(ii) We model the behavior of HT attackers in SoW and the key mechanisms of the proposed secure computing architecture using generalized stochastic colored petri net (GSCPN) [5], establishing the dynamic heterogeneous redundant computing architecture (DHRCA) HT attack–defense GSCPN model for SoW.

(iii) We compare our method with the scheme of tri-mode redundancy (TMR) and runtime monitoring TPAD [6] on the GreatSPN simulation platform to verify the security gain of the proposed method and further analyze the security of SoW with different defense approaches in different scenarios and their reasons.

The rest of this paper is organized as follows: Section 2 introduces the relevant background. Section 3 gives the research motivation of this paper, in which we analyze the Trojan threat scenarios of SoW and give the threat model of the research in this paper. Section 4 introduces our approach and proposes the corresponding GSCPN submodel for the relevant attack and defense strategies in DHRCA. Section 5 validates the security of the proposed architecture and analyses the implications of the attack and defense strategy on the security of the system through simulation. Section 6 concludes our work.

2. Background

2.1. HTs and Defense Strategies. HT is a specialized hardware module that can be exploited by attackers. It consists of trigger logic and payload logic. The trigger logic monitors signals within the circuit and activates the payload logic when certain conditions are met, thereby executing specific malicious functions to achieve the attacker's objectives [7]. HTs can be classified according to their malicious payloads, including tampering function, information leakage, and denial-of-service Trojans. Since the emergence of HTs, many researchers have studied how to defend against HT attacks. Currently, HT's defense methods can be categorized into HT detection technologies, trusted design technologies, and runtime protection technologies.

According to the lifecycle of integrated circuits, HT detection techniques can be categorized into pre-silicon and post-silicon detection techniques. Yasaei et al. [8] convert the HDL code of the circuit into a specific data flow graph in the pre-silicon stage, and then they use a graph convolutional network (GCN) to classify the nodes in the graph to determine whether they are infected by HTs, achieving HT detection and localization. Lyu and Mishra [9] map the activation problem of HT trigger logic to the maximum clique cover problem and propose a test vector generation algorithm based on maximal clique sampling to increase the probability of activating hidden HTs, thereby improving the HT detection rate. Yang et al. [10] perform logic testing on post-silicon hard IPs using test vectors, and then they cluster the IPs based on the side-channel information obtained during the testing process. From each cluster, one IP is selected for reverse engineering to determine whether there are HTs in that IP cluster.

Trusted design techniques achieve HT defense from two perspectives: enhancing detection and preventing HT implantation. Guimarães et al. [11] add current sensors to the original circuit to improve the accuracy of signals collected by side-channel analysis methods, thereby increasing the probability of Trojan detection. Li et al. [12] propose a layout padding-based method to prevent HT implantation. By implanting the A2-RO circuit in blank areas of the original circuit, it can prevent attackers from implanting HTs in those areas; once the circuit is removed by attackers, the current information of chip power supply pins will change, enabling the detection of HT implantation. Patnaik et al. [13] combine manufacturing segmentation and layout camouflage to propose a scheme for dividing and manufacturing 3D ICs hierarchically. Different layers are manufactured by different wafer fabrication plants, and the vertical interconnections between layers are obfuscated to prevent HT implantation. Safari et al. [14] proposed the use of vertical obfuscation and horizontal obfuscation with chiplet technology in traditional ICs to prevent the insertion of manufacturer HTs.

Runtime protection techniques are the last line of defense against HT attacks, which can be categorized into runtime monitoring and runtime tolerance techniques. Dong et al. [15] designed a multilevel architecture to protect third-party encrypted IP by secure wrapper and controller. The wrapper checks the input and output signals of the IP, and the controller configures different levels of response measures based on the results of the wrapper to mitigate the security issues of the encrypted IP. Zhu et al. [16] propose an architecture called Jintide, which consists of an IO behavior-tracking chip, multiple memory behavior-tracking chips, and a

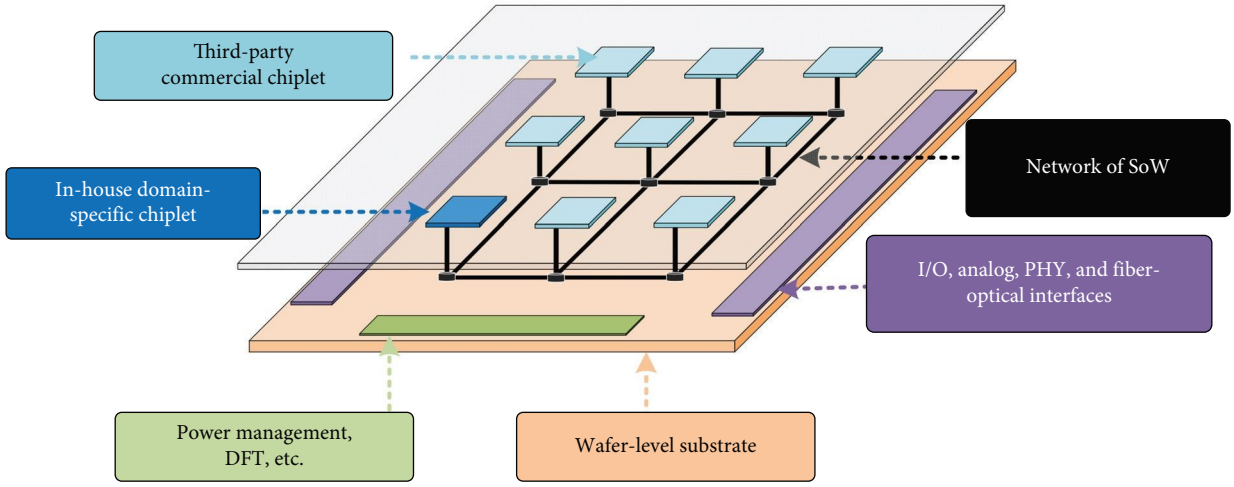

FIGURE 1: SoW schematic.

reconfigurable chip to protect the target CPU. The tracking chip records the CPU's IO and memory transactions and sends the logs to the reconfigurable chip. The reconfigurable chip replays the logs to determine whether there is an HT attack. Gunti and Lingasubramanian [17] use the TMR method to redundantly transform the critical paths in the circuit and determine the presence of an HT based on the majority voting results. Cassano et al. [18] propose a software obfuscation algorithm called DETON to reduce the harm of HT attacks by generating obfuscated versions of protected software. Eslami et al. [19] proposed a security checker utilizing assertions; the security checker synthesizes assertions into circuits to detect rule violations at runtime. Meanwhile, Wu et al. [6] introduced a TPAD approach employing specific CED techniques and selective programmability to safeguard digital systems from HTs attacks, and this method achieves a remarkable 99.998% HTs detection rate with a false positive rate of 0.

2.2. SoW. The SoW, which integrates "super microsystems" with ultra-high density and heterogeneous chiplets assembly, has opened up a new path to enhance the performance of chip systems in the era where Moore's Law is gradually losing its effectiveness. As shown in Figure 1, the SoW is equipped with modules for power management, heat dissipation, I/O, and testability similar to a single chip. Third-party commercial chiplets and self-developed in-house domain-specific chiplets are integrated at an ultra-high density on the wafer-level substrate using advanced packaging techniques. Thermal compression bonding is one of the advanced packaging techniques that involves applying heat and pressure to bond the components together, creating a strong and reliable connection for efficient signal transmission and power distribution. Another advanced packaging technology, through-silicon vias (TSV), provides a method of creating vertical interconnects through silicon wafers; TSV allows for the integration of multilayer circuits and the stacking of multiple chips or chipsets within a single package. Utilizing these advanced packaging techniques, SOW enables high-performance interchiplet communication.

3. Motivation

3.1. Threat Scenario Analysis

3.1.1. Attacker's Perspective.
(1) Increased Probability of HT Implantation. The supply chain of SoW based on chiplet integration relies on various entities spread globally [20]. The supply chain of SoW is depicted in Figure 2. Designers of SoW determine the chiplets to be integrated and their interconnections based on system requirements. They provide the design files of the wafer-level substrate to the manufacturing factory and perform heterogeneous integration of third-party commercial chiplets and domain-specific chipelets designed in-house and fabricated by trusted fabricators on the wafer substrate.

Given the inability to ensure the trustworthiness of third-party commercial chiplets, in our work, we assume that third-party commercial chiplets are untrusted, while the domain-specific chipelets designed in-house and the substrate are trusted. HTs could potentially be implanted during the design or manufacturing stages of commercial chiplets, thereby increasing the possibility of Trojan presence in the system. Assuming the probability of a third-party supplier implants HTs in the chiplets they provide is denoted as P_i. When P_i is 0, none of the chiplets provided by supplier i will have HTs, and when P_i is 1, the supplier is not trustworthy at all, and the supplier implanted HTs in their chiplets. P_{SoW} is the probability that SoW has been implanted with HTs, and the probability that the SoW is not implanted with Trojans is $1 - P_{\mathrm{SoW}}$. The SoW will introduce multiple (set to n in this paper) suppliers, and then the probability that the SoW does not implant an HT is determined by n suppliers, i.e.,

$$1 - P_{\mathrm{SoW}} = P(A_1, A_2, A_3, \ldots, A_n), \tag{1}$$

where A_i $(i = 1, 2, \ldots, n)$ denotes an incident in which supplies i did not implant an HT in the chiplet it supplied. Because we assume that there is no collusion between suppliers in this paper, Whether the vendor implanted an HT is an independent event. Therefore:

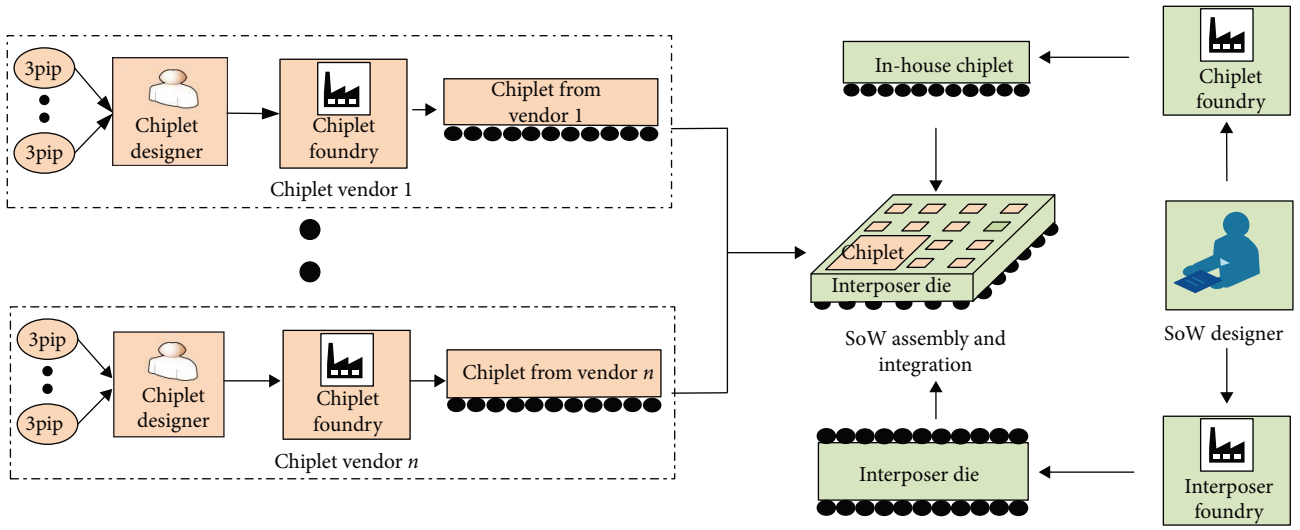

Figure 2: SoW supply chain.

$$1 - P_{\text{SoW}} = P(A_1) \times P(A_2) \cdots\cdots P(A_n), \tag{2}$$

$$P_{\text{SoW}} = 1 - \prod_{i=1}^{n} (1 - P_i). \tag{3}$$

Based on Equations (1) and (2), we can infer that as the number of untrusted entities in the system linearly increases, the system's trustworthiness exhibits an exponential decrease, and the probability of HT implantation in the system also shows an exponential increase.

(2) The Concealment of HTs Increases. Chiplet-based SoW poses new challenges to hardware security, as mentioned in [21–26], where third-party chiplet suppliers can implant a single HT's trigger logic and payload logic into separate chiplets. When the trigger logic in one chiplet is activated, it triggers the malicious payload logic located in another chiplet through the communication channel between chiplets, as shown in Figure 3. Since the trigger and load logic are not in the same chiplet, this can result in lower detection rates for post-silicon side-channel analysis methods. Side-channel analysis is a technique for detecting HTs by utilizing the effect of Trojan implantation on the side-channel information of the parent circuit, and most of them do not rely on the activation of the Trojan for effective detection [27–29], as compared to methods such as logic testing. When performing side-channel analysis on chiplets that only contain the payload logic, the absence of trigger logic that continuously monitors the circuit state reduces the impact of HTs on side-channel information. This, in turn, lowers the probability of Trojan detection and increases the concealment of HTs.

3.1.2. Defender's Perspective. SoW designers can adopt existing Trojan defense methods to resist HT attacks. However, the current methods are not fully applicable to SoW.

(1) Differences in Trusted Entity Assumptions (TEA). Table 1 lists the TEA of HT research in SoC and SoW. Current researches on HTs defense in SoC mainly focus on two untrusted supply chain entities: the untrusted manufacturer

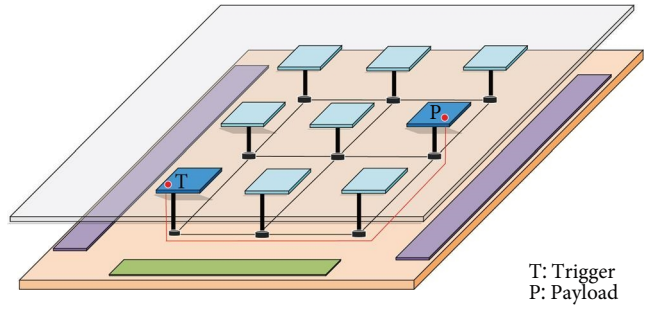

Figure 3: HT hidden enhancement scenario for SoW.

and untrusted third-party IP suppliers. They often assume that the chip designer is trusted. For example, layout fill and split manufacturing methods assume that overseas manufacturing factories are untrusted while the chip design is trusted. Salem and Topham [30] assume that the third-party AXI interconnect IP supplier is an untrusted entity, and the trusted chip designer ensures security by adding wrappers to monitor the interconnect IP. However, the supply chain of SoW involves various entities spread globally, and it cannot be assumed that commercial chiplet designers are completely trusted. Malicious attackers can also implant HTs during chiplet design.

(2) Inadequacy of Traditional HT Methods. To ensure security, the SoW designer/integrator needs to perform security verification of untrustworthy commercial chiplets. Existing methods can effectively defend against HT threats to some extent, but there are still some shortcomings. First, traditional pre-silicon defense techniques (such as pre-silicon inspection) do not apply to post-silicon "hard" chiplets, as commercial chiplet suppliers do not share original design files due to confidentiality. Second, post-silicon methods (such as side-channel analysis) require a golden model without HTs. When the commercial chiplet designer itself is not trustworthy, SoW designers do not have access to the golden model of the commercial chiplet, thus rendering most of the methods relying

Table 1: Comparison of TEA for traditional HT defense and SoW.

TEA in different researches	References	3pip vendor	Chip/third-party chiplet designer	Chiplet foundry	SoW designer
TEA in traditional defense approaches	[15, 30]	✗	√	√	
	[12–14]	√	√	✗	
	[10]	✗	√	✗	
TEAs in SoW	This paper	✗	✗	✗	√

Note: ✗: untrusted entity √: trusted entity.

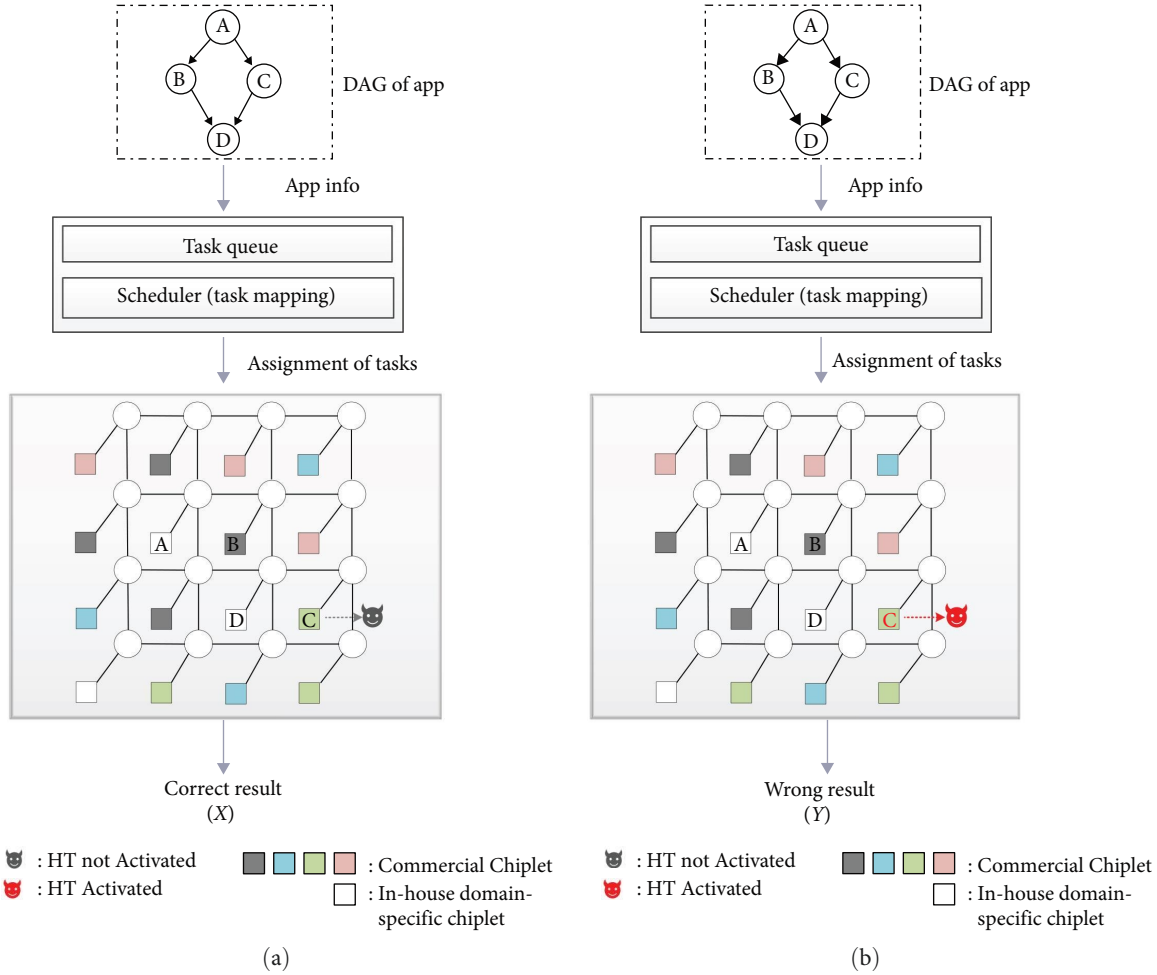

Figure 4: Example of a function—tampering HT attack on SoW: HT not activated (a); HT activated (b).

on the golden model ineffective. Finally, existing runtime protection techniques (such as whitelist-based monitoring techniques) are not effective in defending against unknown HTs with unknown characteristics, and TMR-based runtime tolerance mechanisms cannot tolerate scenarios where two or more redundant units are disabled by an HT attack due to their static nature. In addition, some of the existing studies only target a specific type of HT or a certain class of HT, and it is unable to effectively defend against attack scenarios involving multiple different types of HTs.

Overall, the challenges faced by SoW designers in securing chiplets are significant. Traditional pre-silicon and some existing runtime defense methods may not be directly applicable in such scenarios, emphasizing the need for novel approaches and techniques specifically tailored to the unique requirements of SoW.

3.2. Threat Model

3.2.1. Source of HTs. We assume that any of these entities, from the suppliers of 3pip in commercial chiplets to the manufacturers of commercial chiplets, can implant HTs. SoW designers, domain-specific chipelets designed in-house, wafer-level substrate, and their manufacturers are trusted entities. All of these HTs are chip-level Trojans, which can be at any level of abstraction, such as gate-level or layout-level.

3.2.2. Trigger Mechanism of HTs. We assume that a malicious attacker triggers an HT based on rare signals and states in the

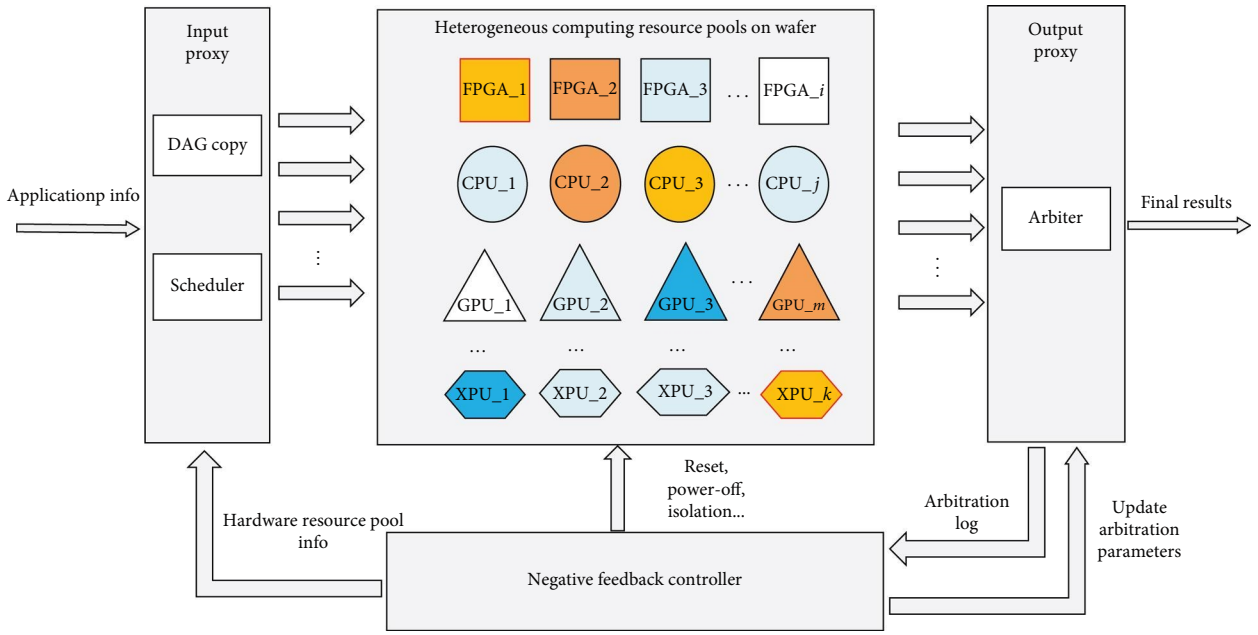

Figure 5: DHRCA.

chiplet, which is probabilistically activated when the chiplet executes a user-submitted computational task. The HT is triggered when the signals or states satisfy the activation conditions.

3.2.3. Payload of HTs. We assume that commercial chiplets can have both function tampering HTs, information leakage HTs, and denial-of-service HTs, and function tampering HTs will alter the normal output of the chiplet, while information leakage HTs do not affect the output. For simplicity of analysis, we assume that individual HTs in this paper have only one malicious function.

3.2.4. Logic Location of HTs. We assume that the trigger and load logic for HTs can be located in the same chiplet or different chiplets from the same vendor.

3.2.5. Others. We assume that there is no collaborative relationship between different commercial chiplet suppliers and that they do not have the same HT logic. This is logical because commercial chiplet suppliers do not share their original design files with commercial competitors, nor do they claim security vulnerabilities in their chiplets and share them with commercial competitors.

Under the above threat model, based on the architecture and computing architecture of SoW, an example of an attacker utilizing an HT that tampers the output of a commercial chiplet to carry out an attack is shown in Figure 4. Figure 4(a) shows that when the HT is not activated, the system outputs the correct result X, and when the HT is activated, the function of the chiplet is modified, the system outputs the incorrect result Y, as shown in Figure 4(b).

4. Architecture Design and Security Modeling

4.1. DHRCA. In this subsection, we combine mimic defense theory with endogenous security properties [31–34] to give a DHRCA for SoW, as shown in Figure 5. The architecture consists of an input proxy module, an output proxy module, a negative feedback control module, and a resource pool of heterogeneous chiplets.

When users submit applications, the input proxy module will abstract the application information and generate a DAG for the application; then, it replicates the DAG. The scheduler maps the application subtasks to the available hardware resources based on the redundant task graph and known resource pool information. When an application is completed by a computing chiplet, the arbiter in the output proxy will use the built-in arbitration strategy to analyze the computation result, output the arbitration result, and submit the arbitration log to the negative feedback controller. The negative feedback controller performs relevant control operations, such as cleaning the HT-attacked chiplets according to the arbitration results and updating the resource pool information and control parameters.

4.2. Key Security Mechanisms. The task mapping and arbitration negative feedback mechanisms in DHRCA determine the security of DHR architecture, and they can effectively increase the uncertainty presented by the DHR architecture to an attacker, making it more difficult for an attacker to trigger an HT and achieve their attack intent. In this subsection, we briefly discuss the key security mechanisms mentioned above.

4.2.1. Task Mapping Mechanism. The task mapping mechanism can be divided into two subquestions: how to map suitable chiplets and when to switch chiplets.

The selection of chiplets can increase the threat awareness of DHRCA and improve system security. In general, chiplets performing the same subtasks online should have a high degree of dissimilarity. The lower the dissimilarity is, the less security the system has. In the extreme case, when the

redundantly executed chiplets are identical, the system is equivalent to a homogeneous redundant system, which can only solve the reliability problem due to random failures and cannot solve the security problem caused by HTs.

The timing of switching chiplets reflects the dynamicity of the architecture, making it more difficult to attack and thus increasing system security. Dynamicity can increase the uncertainty of the system to the outside and disrupt the attack chain of HTs, especially for sequential logic HTs. The dynamicity disrupts the timing conditions, preventing HTs from being triggered. The shorter the switching interval is, the higher the dynamicity is, and the higher attack capabilities of attackers require triggering the HT within a shorter time.

4.2.2. Strategic Arbitration and Negative Feedback Mechanisms. Through the arbitration mechanism, abnormal behavior in the chiplet that executes computing tasks can be detected, and malicious attacks based on HTs can be promptly blocked through negative feedback and scheduling mechanisms. The arbitration mechanism can transform the HT detection problem of a single chiplet into relative judgment between redundant chiplets. Once a differential output phenomenon occurs, it can be determined that an HT has been detected.

Theoretically, the arbitration mechanism can achieve HT awareness once there is an output or state inconsistency across the execution chiplet groups. However, due to the differences between each group of heterogeneous chiplets, their internal states cannot be completely consistent during normal operation. Therefore, the design and implementation of the arbitration mechanism for the DHRCA need to focus on how to normalize the output to improve the accuracy of arbitration reasonably.

The DHRCA does not change the architecture of SoW. It deploys mimic proxy modules in the business perception layer, cognitive decision layer, and resource-aware layer of SoW to form a mimic bracket. The on-wafer computing chiplets that perform the application are the mimetic protection boundary. This architecture can achieve the security goal of avoiding or mitigating uncertain threats caused by the untrustworthiness of third-party supply chains or the inevitability of HTs, ensuring that the SoW can provide high-security and high-reliability services.

4.3. Security Modeling. Since the Petri Net model was proposed in 1962, it has been widely used for modeling parallel, distributed, and non-certainty systems due to its ability to represent complex systems using simple graphical notations. Compared with attack tree models or attack graph models [35], which focus on attack behaviors, the Petri Net model can simultaneously describe the system's state, attack and defense behaviors, and dynamic characteristics of the system's state changes caused by the attack and defense behaviors. To further improve the expressive efficiency of the model, the generalized stochastic petri net (GSPN) and its color extension have been proposed. To better represent the different types of HTs, the different impacts on the system state after triggering different types of HTs, and the temporal relationship between the attack and defense behaviors on the system state, we use GSCPN to model the security of the architecture we proposed in this research.

4.3.1. GSCPN-DHRCA Formal Definition. We also assume that the arrival time of the attack disturbance, the deadline for chiplet task execution, the duration of the HT's activation, and the dynamic scheduling are all memoryless and follow exponential distributions. Finally, to simplify security analysis, we assume that there is at most one HT active in an individual chiplet at any time, while the others are in a dormant state.

The DHRCA for SoW based on GSCPN can be described as follows:

$$\text{GSCPN} - \text{DHRCA} = \{\Sigma, P, T, F, C, G, \lambda, \omega, M_0\}. \quad (4)$$

Among them:

Σ is the set of colors in the net model, consisting of basic and mixed colors, and the color is represented by the <> symbol and its internal parameters in the diagram of this paper.

P is the set of places in the net model, represented by a circle in the diagram.

T is the set of transitions in the net model, $T = T_{\text{time}} \cup T_{\text{immediate}}$, $T_{\text{time}} \cap T_{\text{immediate}} = \emptyset$, T_{time} is the set of time-delayed transitions, and $T_{\text{immediate}}$ is the set of instantaneous transitions, they are represented by hollow and solid rectangles respectively in the diagram.

F is the set of directed arcs in the net model, $F \subseteq (P \times T) \cup (T \times P)$, and the arcs can only be directed from the transition to the place or from the place to the transition, represented in the diagram by the connecting lines with arrows.

C is the color mapping function, $C: P \to 2^{\Sigma}$, 2^{Σ} is a subset of the color set Σ, denoted as the set of possible colors for the token in place.

G is the set of guard functions for the transitions in the net model, $G: T \to$ Bool Expression, when the expression condition is satisfied that the transition may be activated and the Bool Expression = 1 in the absence of special instructions.

λ is the set of average implementation rates of the time-delayed transitions.

ω is the set of normalized weights for the instantaneous transitions, satisfying $\sum_{t \in T_{\text{immediate}}}^{M[t>} w_t = 1$, which means that the sum of the weights of all implementable instantaneous transitions under the marking M is 1. The instantaneous transitions under a certain identity have the same weight in the absence of special instructions.

M_0 is the initial identification of the GSCPN-DHRCA.

We assume that the execution redundancy of the proposed architecture is 3, and the adjudication strategy is a large number of adjudication strategies that adopt a random selection strategy when the output results do not satisfy the adjudication conditions. The corresponding GSCPN submodels of the proposed architecture include the chiplet HT attack behavior submodel, the scheduling mechanism submodel, the task duration submodel, and the arbitration negative feedback behavior submodel. The specific expressions of the guard functions corresponding to the transitions are shown in Table 2.

Table 2: GSCPN-DHRCA guarding functions.

Functions	Bool expression
G(T_HTtri)	$(x == \text{nor}) \wedge (y! = \text{nor})$
G(T_HTsleep)	$(x! = \text{nor}) \wedge (x! = \text{dos}) \wedge (y == \text{nor})$
G(T_HT_out)	$((x == \text{mod}) \wedge (a == \text{right}) \wedge (b == \text{wrong})) \vee ((x == \text{dos}) \wedge (a! = \text{num}) \wedge (b == \text{num}))$
G(T_ANLS_1)	$(a == \text{wrong}) \wedge (b == \text{wrong}) \wedge (c == \text{wrong})$
G(T_ANLS_2)	$(a == \text{num}) \wedge (b == \text{wrong}) \wedge (\text{c} == \text{wrong})$
G(T_ANLS_3)	$(a == \text{wrong}) \wedge (b == \text{num}) \wedge (\text{c} == \text{wrong})$
G(T_ANLS_4)	$(a == \text{wrong}) \wedge (b == \text{wrong}) \wedge (\text{c} == \text{num})$
G(T_ANLS_5)	$((a == \text{num}) \wedge (b == \text{wrong}) \wedge (\text{c} == \text{right})) \vee ((a == \text{num}) \wedge (b == \text{right}) \wedge (\text{c} == \text{wrong}))$
G(T_ANLS_6)	$((a == \text{wrong}) \wedge (b == \text{num}) \wedge (\text{c} == \text{right})) \vee ((a == \text{right}) \wedge (b == \text{num}) \wedge (\text{c} == \text{wrong}))$
G(T_ANLS_7)	$((a == \text{right}) \wedge (b == \text{wrong}) \wedge (\text{c} == \text{num})) \vee ((a == \text{wrong}) \wedge (b == \text{right}) \wedge (\text{c} == \text{num}))$
G(T_ANLS_8)	$(a == \text{right}) \wedge (b == \text{wrong}) \wedge (\text{c} == \text{wrong})$
G(T_ANLS_9)	$(a == \text{wrong}) \wedge (b == \text{right}) \wedge (\text{c} == \text{wrong})$
G(T_ANLS_10)	$(a == \text{wrong}) \wedge (b == \text{wrong}) \wedge (\text{c} == \text{right})$
G(T_NF_23)	$(a == \text{right}) \wedge (b == \text{right}) \wedge (\text{c} == \text{right})$
G(T_NF_24)	$(a! = \text{right}) \wedge (b == \text{right}) \wedge (\text{c} == \text{right})$
G(T_NF_25)	$(a == \text{right}) \wedge (b! = \text{right}) \wedge (\text{c} == \text{right})$
G(T_NF_26)	$(a == \text{right}) \wedge (b == \text{right}) \wedge (\text{c}! = \text{right})$
G(T_NF_27)	$(a! = \text{num}) \wedge (b == \text{num}) \wedge (\text{c} == \text{num})$
G(T_NF_28)	$(a == \text{num}) \wedge (b! = \text{num}) \wedge (\text{c} == \text{num})$
G(T_NF_29)	$(a == \text{num}) \wedge (b == \text{num}) \wedge (\text{c}! = \text{num})$
G(T_NF_30)	$(a == \text{num}) \wedge (b == \text{num}) \wedge (\text{c} == \text{num})$

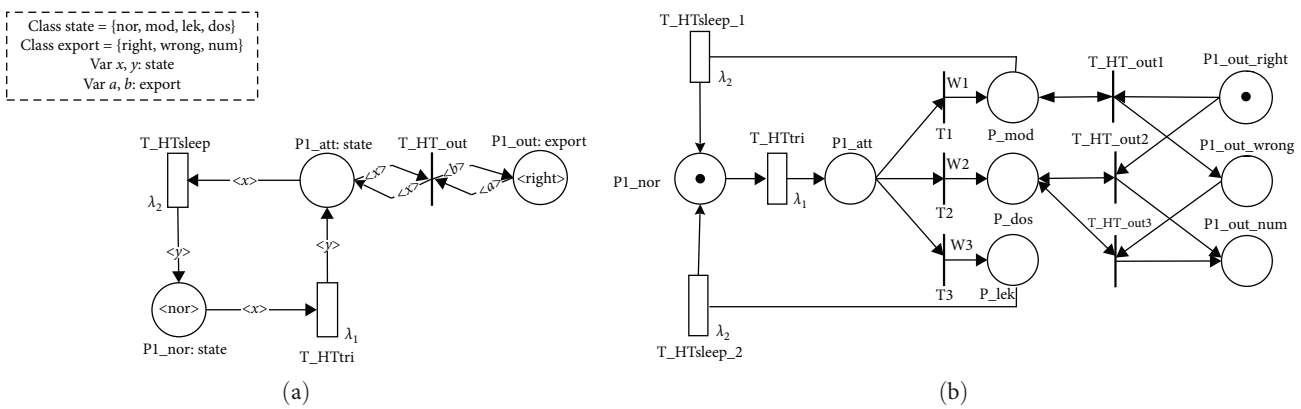

Figure 6: Chiplet HT attack GSCPN submodel (a) and equivalent GSPN model (b).

4.3.2. Chiplet HT Attack Behavior Submodel. As shown in Figure 6(a), it is the chiplet HT attack GSCPN submodel. The token colors for Places P1_nor and P1_att can be <nor>, <mod>, <lek>, and <dos>, representing the states of the chiplet being in HT dormant state, function tampering HT activated state, data leakage HT activated state, and denial-of-service HT activated state. The token colors for Place P1_out can be <right>, <wrong>, and <num>, representing correct output, incorrect output, and no output of the chiplet. Transition T_HWtri represents the activation event of HT, with an average implementation rate of λ_1. Transition T_Hwsleep represents the transition of the HT from activation to a dormant state, with an average implementation rate of λ_2. Immediate transition T_HT_out represents the impact of different types of HTs triggering on the system's output. At the initial moment, there is one <nor> colored token in P1_nor, and the token color in P1_out is <right>. Figure 6(b) is the equivalent GSPN model unfolded from the GSCPN model. Immediate transitions T1, T2, and T3 represent the activated HTs with probabilities ω_1 for function tampering HT, ω_2 for denial-of-service HT, and ω_3 for data leakage HT, respectively.

4.3.3. Scheduling Mechanism Submodel. In the DHRCA, the system scheduling is governed by dynamic random scheduling and arbitration negative feedback scheduling control. The scheduling mechanism of the GSCPN submodel is shown in Figure 7(a). Transition T_dyn_schedule represents the nonarbitration triggered dynamic scheduling switch event,

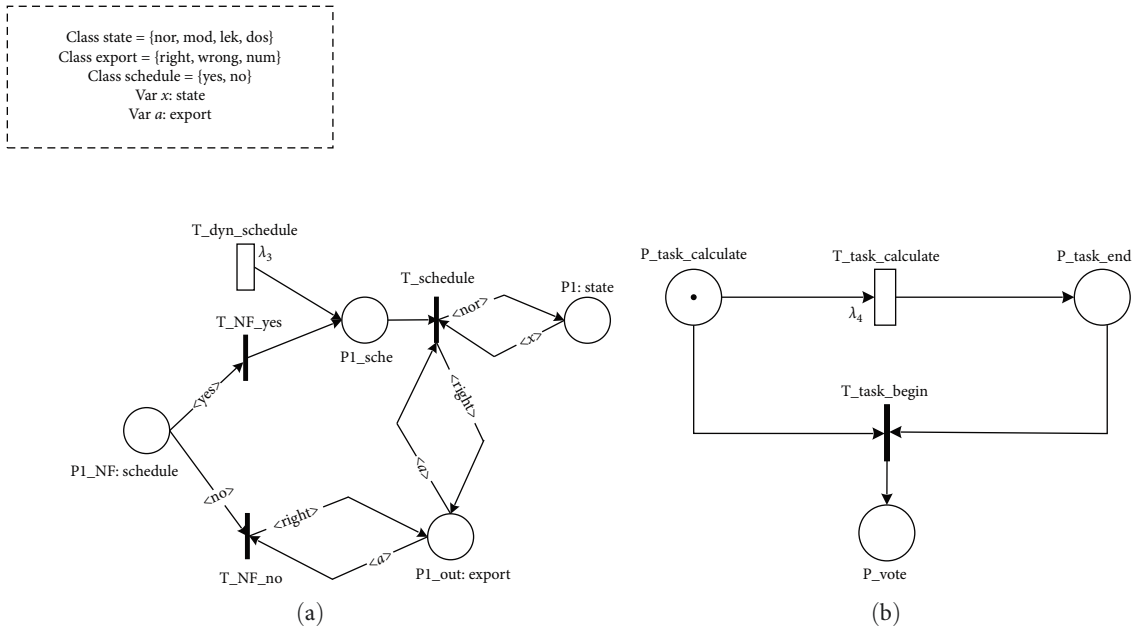

FIGURE 7: Dynamic scheduling submodel (a) and task duration submodel (b).

with an average implementation rate of λ_3. Place P1_NF can contain tokens in <yes> and <no> colors, representing the negative feedback decision triggered by arbitration, i.e., whether to switch the chiplet. Place P1_sche contains tokens indicating the state of being ready for switching chiplet. Transition T_schedule represents the scheduling event's impact on the chiplet's output state and HT state. Transition T_NF_yes and transition T_NF_no are logical judgments and have no actual meaning.

4.3.4. Task Duration Submodel. The task duration submodel is independent of the HT state and output state in the chiplet. So, we use GSPN to model this submodel, as shown in Figure 7(b). At the initial moment, place P_task_caculate holds a token, indicating that the chiplet is in the calculation task state and no arbitration is made. Transition T_task_caculate represents the event of the task calculation time threshold being reached, with an average implementation rate of λ_4. Place P_task_end holds a token indicating that the task execution is completed. Transition T_task_begin represents the start of the next calculation task. Place P_vote holds a token instructing the system to implement the arbitration and negative feedback policy.

4.3.5. Arbitration Negative Feedback Behavior Submodel. The GSCPN model shown in Figure 8 is the arbitration negative feedback behavior submodel. The mixed color classes "Result" and "Action" represent the output state and the negative feedback decision of the system. The transition T_outcol indicates the acceptance of the arbitration instruction and the collection of the chiplets output. Place P_sysstate contains tokens with different colors representing the different output states of the system when arbitrating, which is the input of the arbitration strategy. Transition T_NF indicates making negative feedback decisions based on the system output results. Place P_sysact contains tokens with different colors representing specific negative feedback decisions, i.e., whether to make a scheduling switch for part of the chiplets. Transition T_NFdie disseminates the decision to specific chiplets.

The transition T_NF is expanded, as shown in Figure 9. Due to space limitations, the model is divided into four parts in this paper. The transitions T_ANLS_n ($n = 1,\ldots, 10$) represent the logic for classifying the system output states. Places P4–P13 contain tokens with different colors that represent different system output states after classification. The transitions T_NF_n ($n = 1, 2,\ldots, 38$) determine the corresponding negative feedback decisions. The parameters $\omega_4 - \omega_9$ correspond to the weights of instantaneous transitions, where ω_4 represents the probability that the outputs of two chiplets remain consistent after being tampered with, satisfying $\omega_4 \leq 1$. In this article, ω_4 is set to 0.00001.

5. Model Simulation and Analysis

Our approach belongs to runtime protection, which can be mainly categorized into runtime monitoring as well as runtime tolerance techniques, as described in Section 2. In this section, we compare our approach with the TPAD runtime monitoring technique as well as the redundancy-based tolerance technique, i.e., the TMR technique. The pre-silicon defense techniques and trusted design techniques, such as Split-Fabric, do not belong to the same category as our approach, and we did not include them in our comparison experiments.

It should be noted that in previous studies, using TMR methods in a single chip introduces a large number of redundant logic circuits and voting circuits, which is inappropriate due to the maximum area of a single chip as well as the cost constraints, so HTs defense methods on TMR are few and have not been well appreciated. However, in the research

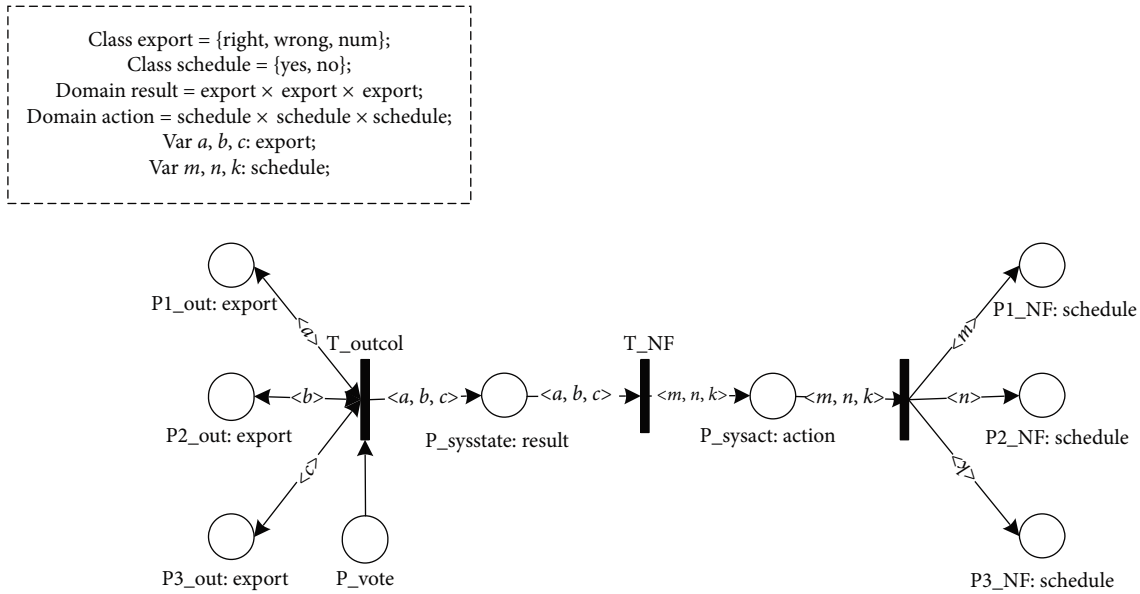

Figure 8: Arbitration negative feedback behavior submodel.

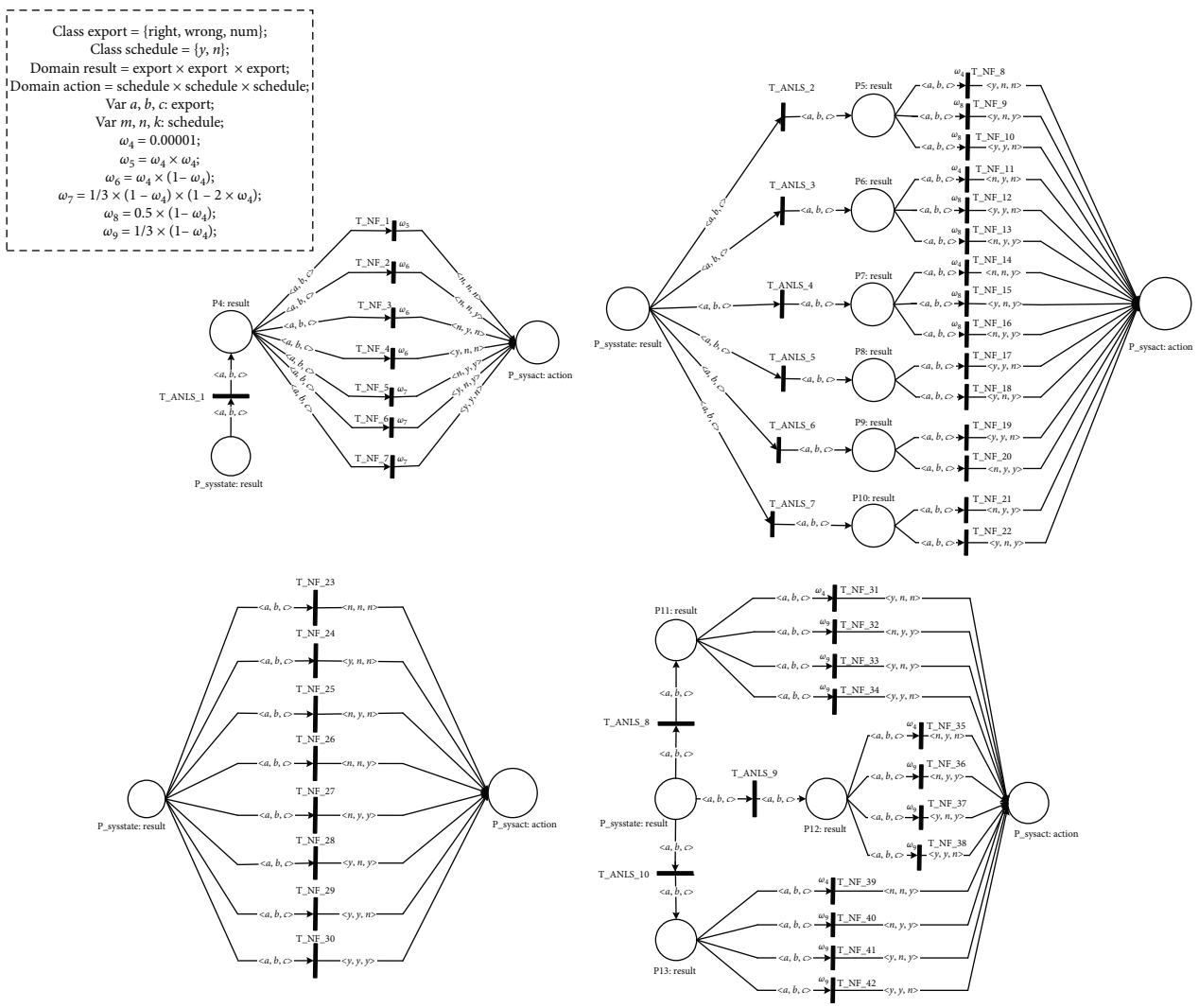

Figure 9: GSPCN submodel after transition T_NF expansion.

Table 3: Experimental parameters and meanings.

Parameter	Value	Meaning
ω_1	0.33	Probability that the triggered HT in the chiplet is a function tampering HT
ω_2	0.22	Probability that the triggered HT in the chiplet is an information leakage HT
ω_3	0.45	Probability that the triggered HT in the chiplet is a denial-of-service HT
ω_4	1.0×10^{-4}	Probability that the outputs of two chiplets remain consistent after being tampered with
λ_1	0.01, 0.1, 10	The average attack time of HTs is 100, 10 hr, and 6 min, simulating weak, medium, and strong attack scenarios
λ_2	0.01, 10	The average time the HTs was active is 100 hr and 6 min, simulating the long and short duration of the HTs
λ_3	0.01	The average time for nonarbitration triggered dynamic switching chiplets is 100 hr, simulating the dynamicity of DHRCA
λ_4	0.01, 1, 10	The average time for the chiplet to perform a task is 100, 1 hr and 6 min, simulating long time, medium time, and short time task
K	0.99998	Probability that TPAD runtime monitoring technology detects HTs when HTs are activated

Note: The parameter settings are based on the HT benchmarks and their distribution provided by the Trust-Hub [36] website. We have analyzed the chip-level HT benchmarks from Trust-Hub (100 in total) and categorized them based on their impact. Out of these benchmarks, there are 33 Trojans that tamper function, 22 HTs that leak information, and 45 HTs that cause denial-of-service.

context of this paper, hundreds or thousands of chiplets are integrated into SoW, and redundancy and heterogeneity are natural attributes of the system. The TMR approach does not bring additional cost overheads, and the drawbacks of traditional TMR techniques are not obvious by utilizing the chiplets of unperformed tasks; therefore, this paper uses it as one of the control groups.

GreatSPN is an open-source tool that uses GSPN and its color extensions to model, validate, and analyze the performance as well as security of the systems. In our research, we use GreatSPN as our simulation platform for system security analysis. GreatSPN is used to simulate models of systems with different runtime protection methods to obtain quantitative results of system security gains. Also, we analyze the impact of HT attack intensity, HT duration, and individual task execution time on the security of SoW with those methods. The security evaluation metrics are as follows:

System security probability (SSP): The probability that all the HTs in chiplets performing computational tasks are dormant, expressed as follows:

$$\mathrm{SSP} = P(\mathrm{all\_ht\_sleep}). \tag{5}$$

System availability probability (SAP): The probability that the SoW can provide normal service to the users, i.e., output the correct result., which can be expressed as follows:

$$\mathrm{SAP} = P\left(\frac{\mathrm{count}\left(\mathrm{export}_{\mathrm{num}} + \mathrm{export}_{\mathrm{wrong}}\right)}{n} < 0.5\right), \tag{6}$$

where n is the implementation redundancy of the different methods, set to 3 for TMR and DHRCA, and 1 for TPAD technique.

The values and meanings of the parameters in the GSCPN model for this experiment are shown in Table 3.

5.1. System Security Comparison. In experiment 1, the method of this paper is compared with the TMR technique and TPAD technique for security; the experimental parameters are set as $(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = (0.1, 10, 0.01, 1), k = 0.99998$, and the experimental results are shown in Figure 10.

As the results show, the TPAD method has the highest SSP and lowest SAP at the initial stage of system operation, because TMR and our methods add redundant runtime units. As the system keeps running, the SSP and SAP of the TPAD and TMR methods gradually decrease and finally converge to 0. In contrast, our method has high security in the steady state of the system, with SSP and SAP of 0.8690 and 0.9750, respectively. It is worth noting that the SSP of the TMR method is always lower than that of the TPAD method, and the SAP also gradually decreases with the system running lower than that of the TPAD method, which is because when more than half of the attacked cores in TMR, the system cannot provide normal service.

5.2. HTs Attack Intensity Impact on Security. In experiment 2, we compare the security of the systems with different approaches under different HT attack intensities. the parameters are set as $k = 0.99998$, $(\lambda_2, \lambda_3, \lambda_4) = (10, 0.01, 1), \lambda_1 = 0.01, 10$, the experimental results are shown in Figure 11.

Based on the simulation results, the security of the SoW for all three methods reaches the steady state in a shorter period of time as λ_1 increases from 0.01 to 10. The SSP and SAP of the TMR and TPAD methods all decrease to 0, while the SSP of our method decreases to 0.0382, and the SAP decreases to 0.1115. From this, we can see that the proposed method in this paper exhibits a certain level of security in strong attack scenarios, but the safety metrics are relatively low; it is necessary to enhance the security by modifying relevant defense strategies, such as reducing decision intervals.

5.3. HTs Duration Impact on Security. In experiment 3, we compare the security of systems with different approaches under different HTs durations. the parameters are set as $k = 0.99998$, $(\lambda_1, \lambda_3, \lambda_4) = (0.1, 0.01, 1), \lambda_2 = 0.01, 10$, and the experimental results are shown in Figure 12.

As the results show, the SSP of all three methods shows improvement while SAP decreases, with the increase of λ_2 before the system reaches the steady state. This is because a decrease in duration reduces the amount of time the Trojan

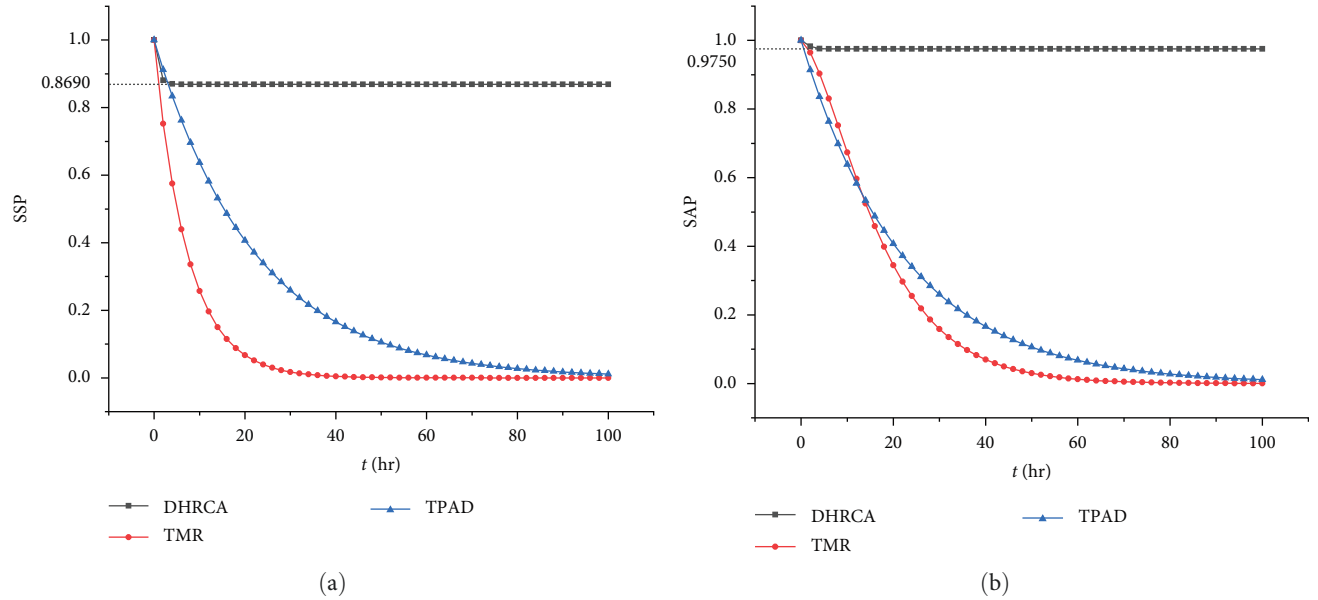

FIGURE 10: Comparison of security of different runtime protection methods. (a) Comparison of SSP and (b) comparison of SAP.

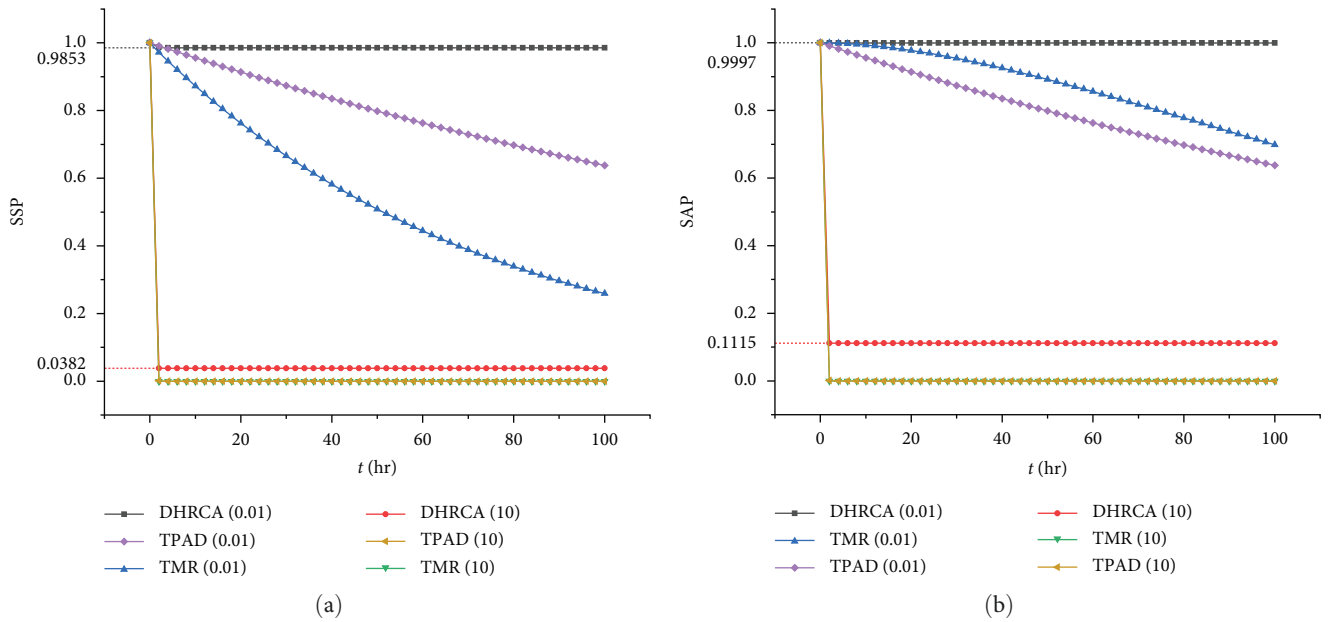

FIGURE 11: Impact of different HTs attack intensity on system security. (a) Impact of different HTs attack intensity on SSP and (b) impact of different HTs attack intensity on SAP.

is active in the chiplet, thus increasing SSP, while a decrease in HTs duration increases the probability of subsequent denial of service and function tampering HTs attacks, thus reducing SAP. Furthermore, the decrease in HTs duration has a significant impact on the SSP of the proposed method but a relatively minor impact on TMR and TPAD methods, with SAP exhibiting the opposite trend. From Figure 13(b), it can be observed that when the system has not yet reached a steady state, the SAP of our method experiences a temporary decrease followed by an increase. This is because the initial operating state of the system may trigger function tampering and denial-of-service HTs, leading to a temporary reduction in availability probability.

5.4. Task Execution Time Impact on Security. In experiment 4, we compare the security of the systems with different approaches under different task execution times; the parameters are set as $k = 0.99998$, $(\lambda_1, \lambda_2, \lambda_3) = (0.1, 10, 0.01)$, $\lambda_4 = 0.01, 10$, and the experimental results are shown in Figure 13.

As the results show, the smaller λ_4 is, i.e., the longer the execution time of the task, the lower the SSP and SAP of our method, the SSP of the method of TMR and TPAD are basically unaffected, and there will be a small decrease in SAP, but the decrease will be lower than that of our method. When the task duration is equal to 100 hr, the long judgment time increases the possibility of the chiplet being attacked by

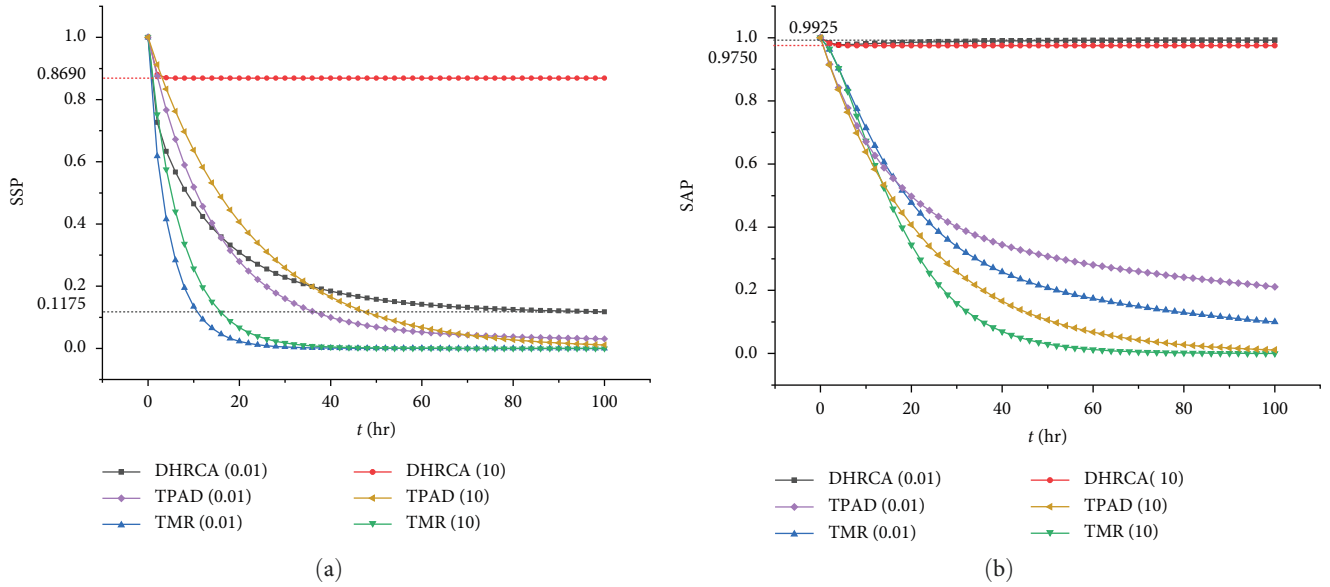

Figure 12: Impact of different HTs duration on system security. (a) Impact of different HTs duration on SSP and (b) impact of different HTs duration on SAP.

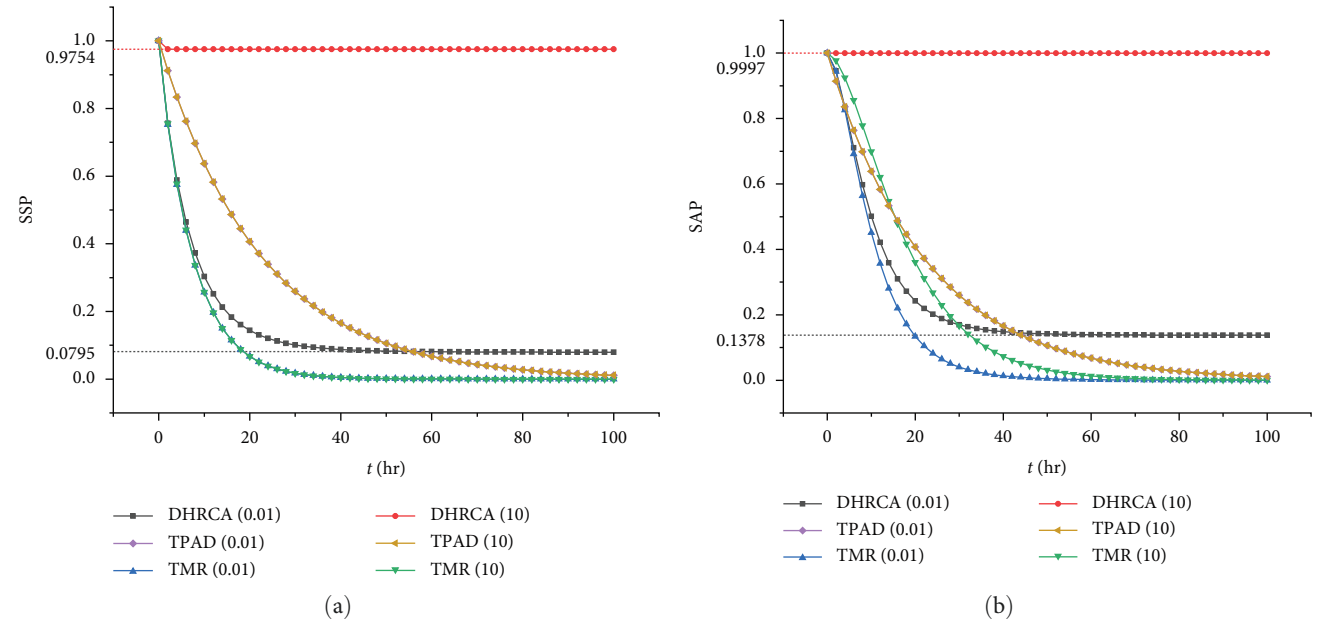

Figure 13: Impact of different task execution times on system security. (a) Impact of different task execution times on SSP and (b) impact of different task execution times on SAP.

HTs during a single task execution. As a result, both the SSP and SAP of our method in this paper are below 0.2 in steady state.

From all the experiments mentioned above, we can conclude that the intensity of the HTs attack, HTs duration, and task execution time all have a significant impact on the SSP of the system implementing our proposed method, which may lead to our method being less secure than the TPAD method in the preoperational period of the system. However, in a steady state, the SSP and SAP of our method are not zero, while both TMR and TPAD methods are zero, i.e., our method has significant security gain in a steady state. The security gain can be adjusted in real-time at runtime, e.g., by assigning the tasks with short execution time to the chiplets with low security or by increasing the checkpoints to adjudicate on the intermediate outputs instead of the final results (equivalent to increasing λ_4).

6. Conclusion

Currently, heterogeneous integration has become the focus in the field of semiconductors. We analyze the serious HT problems faced by the SoW with the basic starting point that the security of the whole supply chain of commercial chiplets cannot be guaranteed. To analyze the security advantages of this paper's approach, we model the architecture using the

GSCPN model and verify our approach has more security advantages than TMR and runtime monitoring approaches through experimental simulations. Further, we analyze the security of the system under different scenarios. In future work, we will analyze the trend of system security with the change of attack and defense strategies to guide the development of low-cost defense schemes to improve security under different attack environments. We will also build a wafer-level system simulator and consider the performance factors in the actual deployment to design a task mapping mechanism and adjudication mechanism applicable to the DHR computing architecture with SoW.

Data Availability

No underlying data were collected or produced in this study.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the National Key R&D Program: Design and Verification of System Level Development Environment for Wafer Level Chip (no. 2022YFB4401401), Song Shan Laboratory (included in the Management of Major Science and Technology Program of Henan Province): Domain-Specific Hardware and Software Co-Computing SoW Pioneering Research (no. 221100211300-02).

References

[1] D. Suggs, M. Subramony, and D. Bouvier, "The AMD "Zen 2" processor," *IEEE Micro*, vol. 40, no. 2, pp. 45–52, 2020.

[2] W. Gomes, A. Koker, P. Stover et al., "Ponte vecchio: a multi-tile 3D stacked processor for exascale computing," in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 42–44, IEEE, San Francisco, CA, USA, February 2022.

[3] E. Talpes, D. Williams, and D. D. Sarma, "DOJO: the microarchitecture of Tesla's exa-scale computer," in *2022 IEEE Hot Chips 34 Symposium (HCS)*, pp. 1–28, IEEE, Cupertino, CA, USA, August 2022.

[4] S. Pal, D. Petrisko, M. Tomei, P. Gupta, S. S. Iyer, and R. Kumar, "Architecting waferscale processors–A GPU case study," in *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 250–263, IEEE, Washington, DC, USA, February 2019.

[5] X.-Y. Li, Y. Liu, Y.-H. Lin, L.-H. Xiao, E. Zio, and R. Kang, "A generalized petri net-based modeling framework for service reliability evaluation and management of cloud data centers," *Reliability Engineering & System Safety*, vol. 207, Article ID 107381, 2021.

[6] T. F. Wu, K. Ganesan, Y. A. Hu, H.-S. P. Wong, S. Wong, and S. Mitra, "TPAD: hardware trojan prevention and detection for trusted integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 521–534, 2016.

[7] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: threats and emerging solutions," in *2009 IEEE International High Level Design Validation and Test Workshop*, pp. 166–171, IEEE, San Francisco, CA, USA, November 2009.

[8] R. Yasaei, S. Faezi, and M. A. Al Faruque, "Golden reference-free hardware trojan localization using graph convolutional network," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 10, pp. 1401–1411, 2022.

[9] Y. Lyu and P. Mishra, "Scalable activation of rare triggers in hardware trojans by repeated maximal clique sampling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 7, pp. 1287–1300, 2021.

[10] S. Yang, P. Chakraborty, P. SLPSK, and S. Bhunia, "Trusted electronic systems with untrusted COTS," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 198–203, IEEE, Santa Clara, CA, USA, April 2021.

[11] L. A. Guimarães, R. P. Bastos, and L. Fesquet, "Detection of layout-level trojans by monitoring substrate with preexisting built-in sensors," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 290–295, IEEE, Bochum, Germany, July 2017.

[12] Z. Z. Li, J. J. He, H. C. Ma, Y. J. Liu, G. X. Qin, and Y. Q. Zhao, "A HT insertion prevention method based on layout filling with A2-RO circuit," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 48, no. 3, pp. 514–521, 2022.

[13] S. Patnaik, M. Ashraf, O. Sinanoglu, and J. Knechtel, "A modern approach to IP protection and Trojan prevention: split manufacturing for 3D ICs and obfuscation of vertical interconnects," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1815–1834, 2021.

[14] Y. Safari, P. Aghanoury, S. S. Iyer, N. Sehatbakhsh, and B. Vaisband, "Hybrid obfuscation of chiplet-based systems," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, San Francisco, CA, USA, July 2023.

[15] Z. Dong, L. Chen, and Y. Li, "A multi-level architecture for hardware Trojan and vulnerability runtime detection and response towards cryptographic IP," *IEICE Electronics Express*, vol. 19, no. 11, Article ID 20220167, 2022.

[16] J. Zhu, A. Luo, G. Li et al., "Jintide: utilizing low-cost reconfigurable external monitors to substantially enhance hardware security of large-scale CPU clusters," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 8, pp. 2585–2601, 2021.

[17] N. B. Gunti and K. Lingasubramanian, "Effective usage of redundancy to aid neutralization of hardware Trojans in Integrated Circuits," *Integration*, vol. 59, pp. 233–242, 2017.

[18] L. Cassano, M. Iamundo, T. A. Lopez, A. Nazzari, and G. Di Natale, "DETON: DEfeating hardware Trojan horses in microprocessors through software ObfuscatioN," *Journal of Systems Architecture*, vol. 129, Article ID 102592, 2022.

[19] M. Eslami, T. Ghasempouri, and S. Pagliarini, "Reusing verification assertions as security checkers for hardware trojan detection," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, pp. 1–6, IEEE, Santa Clara, CA, USA, April 2022.

[20] M. Nabeel, M. Ashraf, S. Patnaik, V. Soteriou, O. Sinanoglu, and J. Knechtel, "2.5D root of trust: secure system-level integration of untrusted chiplets," *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1611–1625, 2020.

[21] V. V. Rao, A. Sasan, and I. Savidis, "Analysis of the security vulnerabilities of 2.5-D and 3-D integrated circuits," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, pp. 1–7, IEEE, Santa Clara, CA, USA, April 2022.

[22] P. SLPSK, S. Ray, and S. Bhunia, "TREEHOUSE: a secure asset management infrastructure for protecting 3DIC designs,"

IEEE Transactions on Computers, vol. 72, no. 8, pp. 2306–2320, 2023.

[23] M. S. M. Khan, C. Xi, A. A. Khan, M. T. Rahman, M. M. Tehranipoor, and N. Asadizanjani, "Secure interposer-based heterogeneous integration," *IEEE Design & Test*, vol. 39, no. 6, pp. 156–164, 2022.

[24] J. Dofe, P. Gu, D. Stow, Q. Yu, E. Kursun, and Y. Xie, "Security threats and countermeasures in three-dimensional integrated circuits," in *GLSVLSI '17: Proceedings of the on Great Lakes Symposium on VLSI 2017*, pp. 321–326, Association for Computing Machinery, Banff, Alberta, Canada, May 2017.

[25] P. Yellu, Z. Zhang, M. M. R. Monjur, R. Abeysinghe, and Q. Yu, "Emerging applications of 3D integration and approximate computing in high-performance computing systems: unique security vulnerabilities," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, IEEE, Waltham, MA, USA, September 2019.

[26] Z. Zhang, J. Dofe, P. Yellu, and Q. Yu, "Comprehensive analysis on hardware trojans in 3D ICs: characterization and experimental impact assessment," *SN Computer Science*, vol. 1, Article ID 233, 2020.

[27] Y. Cao, C.-H. Chang, and S. Chen, "A cluster-based distributed active current sensing circuit for hardware trojan detection," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 12, pp. 2220–2231, 2014.

[28] F. N. Esirci and A. A. Bayrakci, "Delay based hardware Trojan detection exploiting spatial correlations to suppress variations," *Integration*, vol. 91, pp. 107–118, 2023.

[29] 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.

[30] H. Salem and N. Topham, "Trustworthy computing on untrustworthy and Trojan-infected on-chip interconnects," in *2021 IEEE European Test Symposium (ETS)*, pp. 1-2, IEEE, Bruges, Belgium, May 2021.

[31] S. J. Lin, Q. R. Liu, and X. L. Wang, "Competitive arbitration model for mimic defense system," *Computer Engineering*, vol. 44, no. 4, pp. 193–198, 2018.

[32] H. Hu, J. Wu, Z. Wang, and G. Cheng, "Mimic defense: a designed-in cybersecurity defense framework," *IET Information Security*, vol. 12, no. 3, pp. 226–237, 2018.

[33] Q. Ren, Z. Guo, J. Wu et al., "SDN-ESRC: a secure and resilient control plane for software-defined networks," *IEEE Transactions on Network and Service Management*, vol. 19, no. 3, pp. 2366–2381, 2022.

[34] Z. Zhu, Q. Liu, D. Liu, C. Ge, and C. Wang, "MHSDN: a hierarchical software defined network reliability framework design," *IET Information Security*, vol. 17, no. 1, pp. 102–117, 2023.

[35] H. S. Lallie, K. Debattista, and J. Bal, "A review of attack graph and attack tree visual syntax in cyber security," *Computer Science Review*, vol. 35, Article ID 100219, 2020.

[36] "TrustHub.org: Trust-HUB," http://trust-hub.org/benchmarks/trojan.