JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 36, 387-421 (2020)
DOI: 10.6688/JISE.202003_36(2).0016

Enhancing Architecture-level Security of SoC Designs via the Distributed Security IPs Deployment Methodology

ZHAO HUANG AND QUAN WANG
School of Computer Science and Technology
Xidian University
Xi'an, 710071 P.R. China
E-mail: zhhuang@stu.xidian.edu.cn; qwang@xidian.edu.cn

The vulnerability of modern System-on-Chip (SoC) eco-industrial chain model has incurred a variety of rogue entities, such as hardware Trojans, participating in all stages of current SoC design-fabrication processes, resulting in serious security risks. To effectively address the security issues, design-for-security (DfS) technology, *e.g.*, incorporating dedicated on-chip security assurance to facilitate the verification, test, and validation of SoCs, has become the essential strategies in design-time considerations. However, existing DfS measures are targeted at intellectual property (IP) core-level security threats and require specific design modifications to eliminate the dependencies of IP types. In particular, the heterogeneous characteristics of current SoCs and functional diversity of IP types make many IP core-level DfS solutions difficult to adapt or scale to the system level. Moreover, current DfS mechanisms act only at certain stages and thus fail to provide process-wide defense. In this paper, we propose a novel, robust security architecture (MSIPS) to enhance the security of SoCs during the test time and runtime. Unlike existing solutions for the IP core-level problems, our MSIPS also considers the architecture-level security threats and exploits a distributed security IPs deployment strategy to ensure trusted SoC operations with untrusted IPs. In particular, we realize fine-grained IP-protection aware security policies in MSIPS to defend against: (1) hardware Trojan attacks with multi-parameters side-channel analysis primitive; (2) SoC or hardware IP thefts with physically unclonable function (PUF) primitive; and (3) abnormal behavior monitoring and verification with anomaly security auditing primitive. We have implemented this framework on an FPGA platform. Experimental results demonstrate the effectiveness of the proposed approach for providing system protection against diverse attacks. As centralized low-overhead on-chip modules, security IPs reside outside the functional IPs and have the features of flexibility, scalability, and diversity.

Keywords: system-on-chip, design-for-security, security IPs, distributed deployment, hardware trojan detection, IP theft defense, anomaly monitoring

1. INTRODUCTION

The highly-globalized trend of modern System-on-Chip (SoC) design and fabrication supply chain has involved a large number of geographically dispersed participants, such as intellectual property (IP) vendor, system design and integration house, the foundry, and product suppliers, coordinating into the development of electronic devices [1, 2]. However, this distributed nature of current SoC design-fabrication processes has raised a corresponding increased concern about the security and trustworthiness of SoC hardware. This is primarily due to the fact that traditional formal post-manufacturing detection techniques alone are inadequate to detect them completely. Fig. 1 displays di-

Received September 6, 2017; revised August 16 & November 1, 2018; accepted March 12, 2019.
Communicated by Xiaohong Jiang.

verse hardware security issues at different stages of SoC development and deployment cycle [3]. Such hardware-based vulnerabilities might be exploited by any player to launch malicious alterations or surreptitiously compromise the reliability and integrity at some point of this complex ecosystem [2, 4]. In particular, they can potentially incur serious consequences even in the area of mission-critical applications spanning the domains of communications, space, military and nuclear facilities [5, 6].

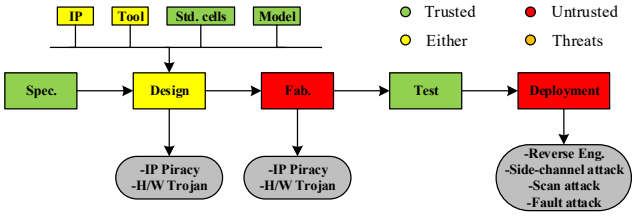


Fig. 1. Security threats at different stages of SoC development and deployment cycle [3].

Typical hardware-based threats that are currently being focused include hardware Trojans, IP thefts, and physical attacks [3, 7-9]. In order to protect SoC hardware against these threats, researchers have directed the design-time considerations as an effective mechanism to prevent or facilitate verification (or recovery) in the case of an attack. For hardware Trojan attacks, security assistance modules, such as multiplexer units [10] and on-chip sensors [11], have been presented as effective design-for-security (DfS) approaches to increase the activity of rare nodes/nets, facilitate Trojan detection or prevent implantation or activation of a Trojan instance. For hardware IP thefts, passive or active hardware protection could be achieved by utilizing the techniques like obfuscation design [12] and physically unclonable function (PUF) [13, 14]. Similarly, diverse mitigation measures have been proposed to prevent the physical attacks such as side-channel attacks [15] and scan-based attacks [16]. These solutions collectively illustrate the applicable of DfS approaches in addressing hardware-based security issues.

However, with the increase of chip integration, SoC designs become more complex, functionally interdependent, and thus incorporating or scaling these DfS mechanisms into SoC designs will encounter the following challenges: (1) these solutions are typically targeted at IP core-level threats and might be insufficient to resist architecture-level issues; (2) the heterogeneous characteristic of modern SoC designs makes it difficult to implement and need to be specifically modified so as to implant or integrate the DfS schemes into an SoC design. This, however, may incur extra overhead and also affect other IPs in the SoC designs; (3) multiple DfS approaches should be incorporated together to address various attacks for a certain type of IPs, which, in turn, may incur design or test conflicts; (4) there are many functional-difference IP cores in the SoC designs, it is difficult to achieve comprehensive protections by solely relying on some certain DfS mechanisms. In particular, the prevalent use of third-party (3P) IPs has also exacerbated the difficulty of incorporating DfS features to provide system-level protection. Considering the above situation, it is critical and challenging to ensure the security of modern SoCs at the architecture level.

In this paper, we propose a Multi-tiered Security IPs architecture, referred to as *MSIPS*, to provide protections for SoCs. Our idea is expanded on the basis of [1, 3]. Ra-

ther than depending on IP core-level features, our solution can address the SoC threats from a perspective of the architecture level. MSIPS is established on a variety of security IPs deployed through distributed strategies and can comprehensively defend against diverse attacks. In particular, security IP acts as a plug-and-play IP and serves as a security defense module, like the firewall in Cyber Defense System. *It can not only effectively protect SoC hardware against IP core-level threats by preventing attacks or facilitating detection/recovery during manufacturing test, but also discover and handle the potential security risks and anomalies from the architecture level of SoC systems during runtime.* We have shown how to design the security IP and associated security policies in this paper. Furthermore, we use three major attack models to validate its performance after integrating different security primitives in MSIPS. With minimal modifications and less overhead, MSIPS can provide comprehensive security protection for SoC designs. In particular, the major contributions of this paper are as follows:

(1) This paper proposes, for the first time to our knowledge, a security architecture based on the distributed deployment of multiple on-chip security IPs. Our method is specifically considered at design time and can provide fine-grained IP-protection for SoCs against various hardware-based threats during test time and run time.
(2) With security IPs integrated, it describes the low-overhead, general wrapper interfaces connecting the functional IPs and MSIPS. Such wrappers can eliminate the heterogeneous impacts of SoCs. In particular, the IP cores in SoC need appropriate design modification for interfacing with MSIPS architecture.
(3) Considering three dominant threat models, *i.e.* hardware Trojan, IP theft, and architecture-level abnormal event or behavior, it studies the design requirements of security policies and the general process for protection against these threats.
(4) Experimental results are presented to demonstrate the functionality of MSIPS, as well as its capability and effectiveness in achieving the security demands of SoC against the threats considered.

The remainder of the paper is organized as follows: In Section 2 we provide the related work on architecture-level protection for SoC designs. Section 3 presents the designs of security policies, followed by the implementation of MSIPS in Section 4. Section 5 provides the implementations of security policies. Simulation steps and experimental results are described in Section 6. Section 7 discusses the functional flexibility, scalability, security, and execution time analysis of the proposed MSIPS. Finally, we conclude this paper in Section 8.

2. RELATED WORKS

In order to achieve the security requirements of the architecture level, some excellent work has been carried out in recent years. Current DfS solutions in this aspect mainly incorporate dedicated on-chip modules into the SoC designs so as to enhance the security and trustworthiness of SoC systems. Such modules are usually programmable or reconfigurable. The demand for dedicated on-chip security modules to facilitate the verification, test, and validation is rising now due to the increasing complexity of modern

SoCs [3]. According to the on-chip modules utilized, existing DfS schemes can be roughly classified into three categories: namely on-chip classifiers, dedicated security IPs, and design-for-debug (DfD) features.

(1) On-chip Classifiers: research advances on this aspect primarily exploit the incorporated on-chip classifiers to implement the hardware Trojan detection during the deployment or application stage of integrated circuits (ICs). For example, Jin *et al.* proposed a general security architecture for wireless cryptographic ICs [17, 18]. It was built on an on-chip neural-network classifier and can discover the inserted hardware Trojans during runtime. However, such approach is only a trust evaluation solution and fails to eliminate the impact created by the Trojan instances embedded. Guha *et al.* presented an intelligent architecture for crypto SoCs [19]. It was established on an on-chip adaptive resonance theory neural-network classifier and can resist the confidential-compromised type hardware Trojan attacks at runtime. However, this solution can only cope with the hardware Trojans which will incur the leakage of sensitive information. On-chip classifiers based security architecture is typically targeted at the IP core-level Trojans. It's Trojan-resistant and aims to complete hardware Trojan detection separately. When it comes to other security threats, these schemes may not be enough.

(2) Dedicated Security IPs: dedicated security IPs based security architecture mainly focuses on providing system-level protection for SoCs with untrusted IPs so as to defend against diverse security threats. It is developed based on the design-for-test (DFT) policy and now becoming the major research hotspot. For example, Wang *et al.* proposed an infrastructure IP for SoC security architecture, referred to as IIPS [3], which can eliminate the heterogeneous impacts of current SoCs. However, IIPS is limited to preventing low-level hardware security vulnerabilities [20]. Basak *et al.* expanded the IIPS and presented a novel SoC security architecture [2, 20]. It implemented a fine-grained IP-trust aware security policy for runtime monitoring. While Kim *et al.* introduced a dynamic function verification (DFV) IP blocks based architecture and can complete the dynamic functions monitoring and hardware Trojan detection at runtime [21]. However, the extra logic for online monitoring will incur the extra area and power overhead of the SoC designs. Dedicated security IPs based security architecture can significantly enhance the security and trustworthiness of SoC designs. However, it requires each functional IP core in SoC designs to be specifically modified at design time, *i.e.* augmented with a customized wrapper. Unfortunately, such situation will raise the complexity, defect rate, and test time or cost of the SoC designs, especially for those contain a large number of IP cores.

(3) DfD Features: In order to reduce the hardware overhead introduced by using the wrapper interfaces, architects attempt to establish the security architectures by reconstructing the infrastructures already available on the chip. Among them, DfD instrumentation is the most widely used one for the reason that it exists in almost all IPs and is often utilized to facilitate the post-silicon validation [22]. There have been some research in this respect. For instance, Basak *et al.* proposed a security architecture which exploits DfD features for system-level trusted validation [23]. It not only contributes to improving test coverage, but also achieving on-field update of security policies. However, such technique is only a trusted evaluation scheme and the security of the SoC designs relies

on the validation functions exploited. In addition, DfD is likewise a double-edged sword. It can in turn be utilized by an adversary (or attacker) to reveal internal information.

In general, current DfS mechanisms are primarily handling the architecture-level or IP core-level threats separately at runtime. Thus, they cannot provide process-wide protections for modern SoCs. Moreover, existing DfS solutions mainly consider incorporating dedicated security IPs as the most frequently used on-chip features into the SoCs because of their flexibility and programmability to withstand diverse security threats. However, such centralized control engines would be restricted by the various security policies they imposed. And they might have not yet considered the type-differences of functional IPs. So it is inappropriate and difficult to integrate all security policies in a single security IP. In particular, most of them may regard the security factors as accessories, rather than the necessaries. To this end, we suggest that dedicated security modules should be built at the very beginning of the SoC designs, be implanted at design time, and provide protection across the entire lifecycle. For type-differences of functional IPs, we also present to build corresponding security blocks and deploy them distributedly, based on which, our security architecture is obtained. *And this is the motivation and objective of our work.*

3. DESIGN OF SOC SECURITY POLICIES

This section describes the threats model and the defense measures involved in the MSIPS. Among them, Section 3.1 gives a brief description of the architecture-level security issues caused by 3PIPs and highlights the specific examples considered in this paper. The corresponding defense policies and security primitives are outlined in Section 3.2.

3.1 Architecture-Level Security Threats Model

Modern SoC designs utilize a large number of 3PIPs to complete the overall system integration, many of which are acquired from untrusted third-party design houses or vendors [20]. Such situation has created opportunities for an adversary to malicious compromise the trustworthiness of the fabricated hardware [24]. In addition to IP core-level security threats, *e.g.* hardware Trojans, IP theft, *etc.*, some existing and emerging threats are also beginning to threaten the security and reliability of SoC designs from the architecture level [3, 5]. Among them, representative examples are currently being considered in an SoC design include IP core-level hardware Trojan attacks, IP theft, architecture-level event or behavior, and so forth [2].

(A) IP Core-Level Hardware Trojan Attacks

Hardware Trojan is the primary security threat in current SoCs or IPs [3, 5]. It can perform an attack independently or in cooperating with software. Furthermore, it can bypass the software defense to incur executions failure, information leakage, performance degradation, and even physical destruction while users may know nothing about that [5]. Due to the stealthy nature of hardware Trojans and practically infinite Trojan space, it is difficult to detect them by traditional formal verification and test [6, 8]. In particular, with the increasing complexity and integration of modern SoCs, hardware

Trojan is now expanding from IP core-level to architecture-level. Architecture-level hardware Trojans are different from the IP core-level ones, whose features are more manifested in the event granularity or behavioral level. For example, rogue elements in a 3PIP core can impact the overall system functions, not just IP core itself [23]. Table 1 highlights the distinction between them in details. Here we specifically consider the IP core-level Trojans and arrange the architecture-level Trojans into architecture-level event or behavior item. Because an architecture-level hardware Trojan can typically influence more about the system level behavior and the activities of other IPs at runtime [2].

Table 1. The distinction between IP core-level and architecture-level trojans.

IP core-level hardware Trojans	Architecture-level hardware Trojans
Feasibility and impact analysis of Trojans in IP modules or processor unit	Impact analysis of IP core-level Trojan on SoC or system
Utilize static IP trust verification for hardware Trojan detection	Potentially suspicious IP behavior detection that affects SoC or system at runtime
No error correction or recovery mechanisms when perform runtime methods	Fine-grained IP-protection security policies and appropriate assert-aware security controls

(B) IP Theft attacks

The theft attack to the functional IPs of an SoC, such as reverse engineering, cracking, counterfeiting, and overbuilding, has become another threat to the SoC hardware [5], [14]. A malicious attacker or foundry with access to IPs may illegally steal and claim the ownership of IPs, or build more than the required number of them without the knowledge and consent of the designers and sell the excess IPs in the gray market [24, 25]. This is called IP thefts. It is noted that the global electronic piracy market is growing rapidly and is now estimated to be $1B/day, of which a significant part is related to hardware IP theft [26]. The counterfeit IPs seriously affect the security of various electronic systems.

(C) Architecture-level Event or Behavior threats

Architecture-level hardware Trojans in untrusted IPs may corrupt the confidentiality and integrity of a SoC design. They could illegally intercept, tamper, and reveal the sensitive information of other IPs or SoCs. For instance, they may maliciously forward messages originally sent to IP *A* to IP *B* [27]. Moreover, they can also disguise themselves as some other blocks and send spurious communications or access to other IPs. In addition, the firmware in some IP cores may also cause them to perform unexpected operations [28]. What's more, the debug interfaces could be utilized to viciously update inauthentic firmware and sniff the sensitive information or information flow inside a SoC [29].

3.2 Mitigation Strategies

To effectively address these security problems, various defense measures have been developed in academia and industries to ensure trusted operations with untrusted com-

ponents [26, 30]. These countermeasures mainly encompass fingerprinting, hardware Trojan detection, and certain analytic methods derived from the structural or activation characteristics of SoCs, which can be utilized to defend against various hardware threats and enhance the security of SoCs or IPs. In addition, recovery from abnormal states (or error correction) is another consideration to ensure the security and reliability of SoC.

(A) For IP core-level hardware trojan attacks.

Reverse anatomy based hardware Trojan detection has been considered to be the most effective method, but it's an invasive mechanism and the test overhead is unaffordable [24, 30]. Non-invasive techniques such as logic testing and side-channel analysis have been proposed as effective solutions to address the hardware Trojan problems. Logic testing uses the specially designed test vectors to activate hardware Trojans and compares the output with the truth table. Any difference will indicate the presence of a hardware Trojan. On the other hand, side-channel analysis takes advantage of the circuit parameters such as power consumption and path delay to identify the Trojan circuits [8, 31-33]. Here we force on the combination of the latter two methods integrated in MSIPS due to the following reasons. First, logic testing based method is conducive to detecting the explicit or small Trojan, but it fails to detect large scale Trojan instances. While side-channel analysis based method is propitious to discover the implicit or large Trojans, but it's susceptible to process variation and system noise [31, 34, 35]. They can complement each other. Second, on-chip monitoring module such as security monitors (SM) makes the collection of power consumption information possible [8, 36].

(B) For IP theft attacks.

Various mitigation strategies, such as watermarking, fingerprint, obfuscation design, IP metering, split manufacturing, have been proposed to address IP theft threats [14, 24, 25]. Among them, device fingerprint has been considered as an effective approach to preventing IP theft attacks. By assigning each legally manufactured SoC or functional IP with a unique identification (ID), called integrated circuit (IC) fingerprints, and registering it in a trusted database, consumers can validate each chip before they use it and thus the illegitimate ones cannot be activated and deployed [3, 37]. However, they could still suffer from the threats of cloning or counterfeiting [38, 39]. An adversary may reveal the stored device ID by invasive or non-invasive methods, and program the cloned devices with the same ID [3, 14]. They will deceive the users and enter into the IC supply chain ultimately. In the content of MSIPS, one measure to solve this problem is to assign each device a unique and unclonable ID and register it in a trusted database, so as to verify the uniqueness of it.

PUF based method has been adapted as a reliable technique to enhance the security. It exploits the uniqueness of inter-device process variations in circuit parameters, such as path delay and frequency, to generate unique and unclonable device signatures [3, 40, 41]. Furthermore, due to the low silicon overhead and configurable nature of the current PUF, as well as the features of easy to integrate and implement, the signature generation and extraction achieve the requirements of standard SoC-level testing process. *According to these reasons*, *configurable ring oscillator based PUF* (*CRO-PUF*) *described in* [50] *is employed here.*

(C) For Architecture-level Event or Behavior Threats

Existing static security verification methods cannot adequately monitor or verify the systematic event or behavior of SoCs during runtime, nor completing a good recovery or fault tolerance mechanism. Therefore, we need to exploit a novel, effective technique to alleviate this situation. Access control model has been considered as an effective method to guarantee the security of information systems [42]. It can be sufficient to prevent unauthorized users or systems from accessing the resources and prevent legal users or systems from accessing unauthorized resources [43-45]. In particular, authorization and audit strategies are the main advantages of access control.

Among the current access control models, the principle of role-based access control (RBAC) technique can be adopted in MSIPS for the following reasons. First, RBAC can assign specific permissions to different IPs [46]. Based on this, a trusted matrix table can be formed inside the security IPs so as to constrain the resource access privilege of the functional IPs. Second, the user, role, permission, and resource in MSIPS are limited and simple. Therefore, it can overcome the drawbacks of RBAC used in information systems such as computer, network, server, *etc.*, [47, 48]. In particular, auditing strategy of the RBAC model is consistent with the on-field trusted evaluation function of the architecture-level event or behavior at runtime. *Thus, we primarily exploit the auditing idea of RBAC to construct the security policies for architecture-level event or behavior verification and monitoring.* As far as our knowledge is concerned, this is the first time that the auditing idea of an access control model is applied to achieve architecture-level security of SoCs at runtime.

4. MSIPS: SECURITY POLICY IMPLEMENTATION ARCHITECTURE

In this section, Section 4.1 provides a description of the proposed MSIPS architecture. And then, a general security threats handling process is introduced in Section 4.2.

4.1 Overview of MSIPS

The security architecture we proposed in this article is built on a series of dedicated, policy-implemented IP blocks called security IPs. These security IPs have been developed and achieved in our preliminary work [1]. Note that previous work focused only on IP core-level Trojan threats, regardless of architecture-level abnormal events or behaviors caused by internal or external attacks. Below we will briefly describe the MSIPS architecture and highlights its advantages in the content of architecture-level event or behavior detection and verification. Following sections will introduce the security policies extension of MSIPS for PUF based signature generation against IP thefts, system level abnormal behavior verification and monitoring.

MSIPS contains many different kinds of IPs, *e.g.* security IPs and functional IPs. These functional different IP blocks could be assigned into different tiers at design time according to the services they provide. The “tier” here is a logical partitioning. The security feature of MSIPS is reflected on the various security policies the security IPs perform. MSIPS architecture is specially established on the basis of these dedicated security IPs which are deployed through the distributed strategies and connected to specific func-

tional IPs through the security interface wrappers (or routing nodes) for individual IPs. Security wrapper interfaces are extended on existing debug/test wrappers and can abstract the internal implementation of individual IPs so as to eliminate the effects of heterogeneity. In particular, the security IPs focus on completing the security requirements of the MSIPS, while the functional IPs are devoted to achieving the functional demands of applications.

Table 2. The different tiers of MSIPS architecture in a generic SoC.

System tiers	IP types	Security policies
System tier	Main security IP	Architecture-level security policies like abnormal event or behavior monitoring and verification
Component tier	Sub security IPs	IP core-level security policies like hardware Trojan detection and signature generation
Functional tier	Functional IPs	Corresponding functional implementations according to the practical requirement

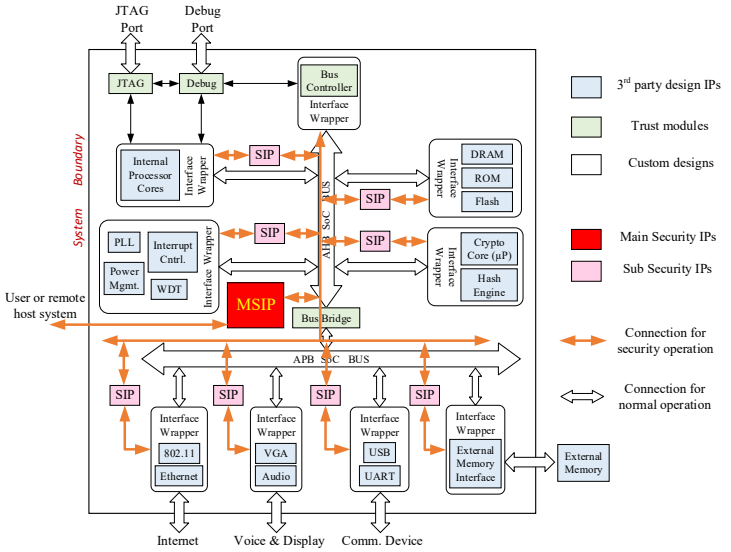


Fig. 2. Block diagram of the proposed MSIPS architecture in a generic SoC.

Fig. 2 illustrates the block diagram of the proposed MSIPS architecture in a generic SoC. It contains three tiers, as shown in Table 2. The bottom tier of MSIPS is the functional tier. It includes a series of functional IPs which are dedicated to achieving the functional requirements of practical applications for user/system or tasks. Functional tier will provide the basic services that SoCs can perform. The middle is the component tier. This tier includes certain numbers of sub security IPs (abbreviated as SIPs). SIPs serve as user-customized security modules and can present IP core-level security verification, testing, and monitoring, so as to protect functional IPs against hardware Trojan attacks, hardware IPs theft, *etc.* Such security policies are designed and programmed according to the target IPs they serve. The top is the system tier which contains a centralized secu-

rity policy control IP called main security IP (MSIP for short). Such security IP can verify and handle the architecture-level abnormal event or behavior by the continuously track of system security states and enforce necessary restrictions imposed by security policies so as to prohibit the illegal access of sensitive assets.

The biggest difference between our solution and the previous works is that previous works only utilize a centralized on-chip security module to address all the security transactions while our solution explores the distributed security IPs to do that, *i.e.* MSIP handles global SoC system level security threats and SIPs local IP core-level security issues. Such scheme makes the SoC systems highly trustworthy and fault-tolerant. *The proposed MSIPS architecture shares the same motivations and design principles.*

4.2 Security Policies Execution Processes

The entire procedure of security verification, test, and monitoring at test time or run time can be represented by a finite state machine (FSM). When enabled, MSIP and SIP are initialized based on the configuration of the users or the remote host system. Then, the testing control procedure of IP core-level security verification and test is performed. Upon the completion of the signature generation and ID verification of MSIP or SIPs during the authentication stage, security IP will then send standard test vectors generated by the security policy control unit to the input ports of functional IPs. As the outputs have been acquired from the output ports, security IP then makes a comparison of the results with those gotten from the reliable trusted library to confirm whether the functional IP verified is security during PUF based authentication and hardware Trojan detection. After that, SIP will monitor each functional IP and send current states to MSIP. Finally, MSIP will analyze the system security states and ensure whether the behavior is trusted or not according to the architecture-level security policies and restrictions preset. In short, MSIPS serves as a programmable control & management system. Security IP block generates the required test patterns, processes the output results with corresponding security primitives, compares with the truth tables and then draws a conclusion. In particular, the continuous track of IP states performed by SIPs constitutes the foundation of architecture-level security policies. Following presents the specific process of IP core-level security protection.

State transition diagram of each SIP achieving the IP core-level verification, test and monitoring is illustrated in Fig. 3. After the SIP is turned on, it starts from the *IDLE* state. A *request* from the requestor can bring the SIP to SIP_INIT state to complete the initial configuration. After the requestor receives the *response* from SIP, it then sends the *ID authentication request* to verify the legality of SIP itself. If failed or time out, this process will be executed again. After accepted, SIP can be set to *SIP_EN* state to perform testing, verification and monitoring of the functional IPs. Either or both of testing mode and monitoring mode can be performed. During the testing mode, PUF based authentication and hardware Trojan detection of functional IPs will be executed, respectively. Testing model is designed to precede the PUF state and hardware Trojan state because both PUF based authentication and hardware Trojan detection require at least one active test vector. During the monitoring mode, specific control signals are required so as to obtain current state information of functional IPs. In order to switch between different models, specific test vector sequence has to be applied to SIP, and when the

task is completed, SIP can be reset to *IDLE* state with the reset signal *SIP_RST* and send the results to MSIP for trust evaluation. In the overall processes of SIP working, all of the operations and state information should be recorded into the logging units.

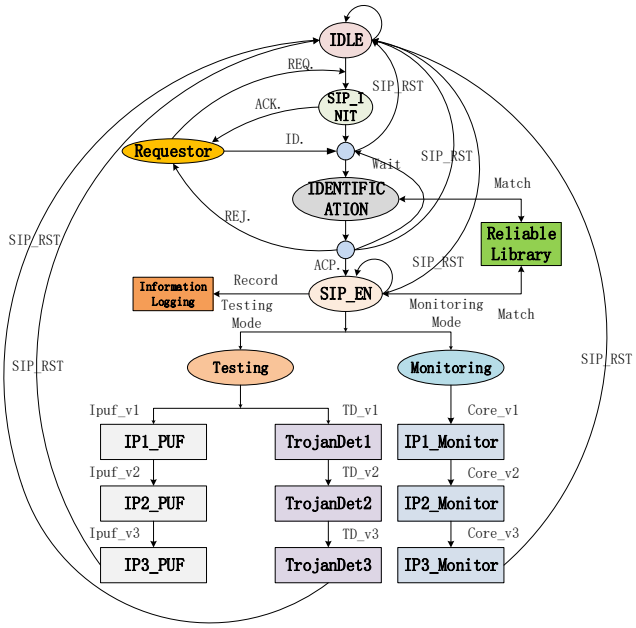


Fig. 3. State transition diagram of the sub security IP module.

5. SECURITY POLICY IMPLEMENTATION

This section describes the specific implementation of security policies. To achieve the target security requirements, Section 5.1 first designs and completes the security IPs required. Then, security wrappers for connecting security IPs and functional IPs are implemented in Section 5.2. Specific security primitives corresponding to security policies are explained in Section 5.3. Section 5.4 provides a use case analysis by applying our proposed architecture.

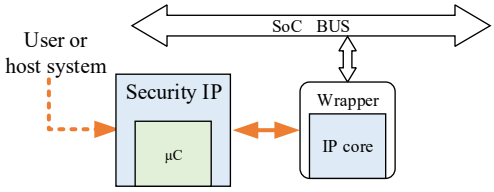


Fig. 4. Block diagram of Security IPs connected to other individual IPs.

Fig. 4 illustrates a block diagram of security IP modules connected to the functional IPs. It contains three main components: (1) a centralized on-chip dedicated IP; (2) the

custom-designed security interface wrappers; and (3) the security policies. Under ideal conditions, security IPs can implement various architecture-level or IP core-level security policies. These security policies are incorporated into the security IPs in the form of microcode or firmware so as to provide fine-grained IP protection for SoC designs. Below we provide a concrete implementation of each component.

5.1 Security IPs Implementation

Security IPs refer to a range of IP cores that are specifically designed by the security team and implanted into SoCs as a DfS mechanism at design time. These dedicated security modules are devoted to facilitating the trust verification, test, or detection of functional IPs so as to defend against thefts, abnormal event or behavior, and hardware Trojan attacks. For example, Synopsys Inc. provides a set of security-IP products to enhance the security capability of hardware-level SoCs [55, 56]. Furthermore, these IPs are instantiated into SoCs after manufacturing test and acts as on-chip modules to achieve the target security standards such as anti-theft, abnormal behavior monitoring and Trojan detection.

The block diagram of a generic security IP module is shown in Fig. 5. It consists of a security management component that enforces the security policies, and a security support component that provides the necessary hardware resources to assist the implementation of security policies.

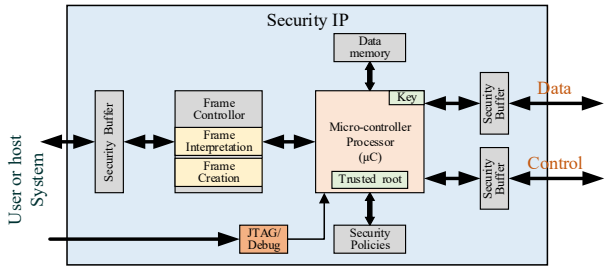


Fig. 5. Block diagram of security IP module.

(A) Security Management Component

Security management component is mainly composed of a *micro-control processor IP*. Such microprocessor is a programmable central security policy controller established on the *trusted root*. It plays the role of CPU, together with security wrappers to enforce the predefined restrictions for each functional IP. Functionally, microprocessor serves as a finite state machine (FSM) whose primary responsibility is to manage the executions of various security policies according to the custom-designed security requirements. In particular, *trusted root* is equivalent to a programmable and updated firmware module. It can assist in the initialization of security IPs according to the contents of *configurable register* at the beginning of procedures.

(B) Security Support Component

As an indispensable part of security IPs, security support component provides the

security management component with the hardware resources necessary to accomplish various security transactions. It typically consists of five major components, *viz*., (1) instruction memory; (2) data memory; (3) frame controller; (4) JTAG port; and (5) security buffer registers. The details of each unit are described in the following.

Instruction memory and data memory are all internal storage. In particular, *instruction memory* is utilized to store the security policies enforced by micro-control processor cores, while *data memory* contains the data required to execute the security policies such as event log, test vector, truth table, or test results, *etc.* All of these data are stored in the static segment manners and have fixed offset addresses. Due to the features of security IPs distributed deployment within SoCs, the communication between the MSIP and SIPs is achieved in the form of frame. Thus, the *frame controller* can interpret and package the event frames received or sent. Fig. 6 shows the format of a typical frame. Moreover, *JTAG port* permits users or system to update the security policies and data to fulfill the specific security applications or use cases. In addition, *security buffer registers* are allocated for the temporary storage of intermediate data.

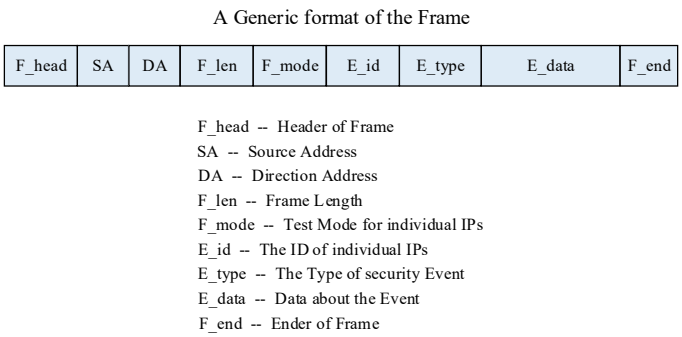


Fig. 6. Fields of a generic event frame.

5.2 Security Interface Wrappers Implementation

To eliminate the effects of heterogeneity and implement the DFT mechanism, each IP core in the SoC designs should have a security wrapper [3]. Such security interface wrapper can obtain the security-critical information from the operating states of the underlying IPs and provides a standard way for security IPs to communicate with each individual IP [20]. Furthermore, the input-output ports of an individual IP can be selectively connected to the security IPs for on-field trust verification and abnormal detection; at other time they can be connected to the SoC bus or chip input-output pins as required to ensure normal system operations [21].

Typical block diagram of the proposed security interface wrapper is shown in Fig. 7. The *control logic* component is similar to a configuration register which could be configured by security IPs at boot time to achieve particular security applications or user cases. The wrapper permits each input-output port of individual core contains a boundary register cell, referred as *wrapper boundary registers* (*WBR*), for data buffer. The security policies to defend against three major threats considered in this paper are completed by *PUF control logic*, *activity detect logic*, and *on-chip current sensor*, respectively. They

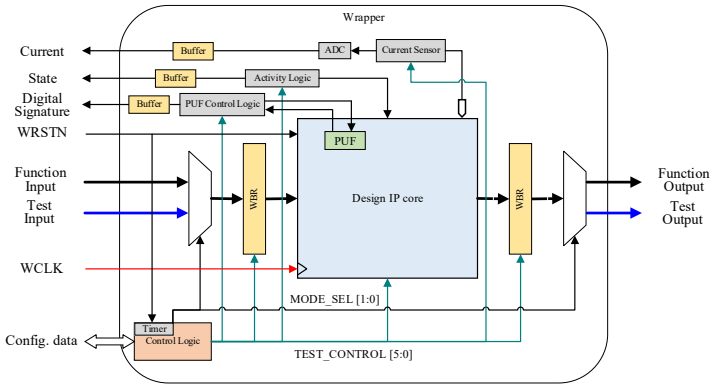


Fig. 7. Block diagram of security IP module.

are controlled by the *control logic*. The wrapper also provides *local clock domains* to satisfy the demand of some events. In general, the security wrapper acts as a control & switch component which can change the communication paths IP cores connect to other modules and enable the acquisition of security-critical information.

5.3 Security Primitives Implementation

In order to effectively address the security threats considered in this paper, we have specified the considered mitigation strategies as hardware security primitives and implemented them here.

(A) Hardware Trojan Detection Primitive

Hardware Trojans will inevitably incur some deviations in circuit parameters, which could be exploited to discover the Trojan instances. Much progress has been made in this area. However, current single-parameter side-channel analysis based methods cannot detect the hardware Trojans effectively due to the effects of environmental noise, process variations, and system noise [8, 32]. *Therefore, multi-parameter side-channel analysis based approaches for hardware Trojan detection* ***proposed in [6]*** *could be utilized as the hardware Trojan detection primitive to incorporate into the security IPs.* Such approaches exploit the intrinsic relationship between transient supply current (I_{ddt}) and the maximum operating frequency (F_{max}) of a circuit to identify the Trojan-infected circuits. In particular, the total transient current of an IC can be approximately expressed as the following formula:

$$I_{ddt} = \sum_{g \in IC} I_g \approx k_{av} \cdot n_{tot} \cdot (V_{DD} - V_T - \Delta V_T)^{\alpha}. \quad (1)$$

While the delays for the gates on the critical path of the IC can be approximated by:

$$T_{crit} = \sum_{g \in P_{crit}} t_{dg} \approx \beta_{av} \cdot n_{crit} \cdot (V_{DD} - V_T - \Delta V_T)^{-\alpha}. \quad (2)$$

Hence, the maximum operating frequency of the IC is given by:

$$F_{\max} = \frac{1}{T_{crit}} \approx \frac{1}{\beta_{av} n_{crit}} \cdot (V_{DD} - V_T - \Delta V_T)^{\alpha}. \tag{3}$$

Combining Eqs. (2) and (3), the relationship between I_{ddt} and F_{max} is given by:

$$\frac{I_{ddt}}{F_{\max}} \approx k_{av} \cdot \beta_{av} \cdot n_{tot} \cdot n_{crit}. \tag{4}$$

Where k_{av} and β_{av} are the gate-dependant constant, n_{tot} is the total number of switching gates in the IC, and n_{crit} is the number of gates on the critical path P_{crit} of the IC. If a Trojan circuit, with n_{trojan} switching gates, is implanted in an IC, the value of the transient current will change to:

$$I_{ddt,trojan} = \sum_{g \in IC} I_g \approx k_{av} \cdot (n_{tot} + n_{trojan}) \cdot (V_{DD} - V_T - \Delta V_T)^{\alpha}. \tag{5}$$

The relationship between I_{ddt} and F_{max} will become:

$$\frac{I_{ddt,trojan}}{F_{\max}} \approx k_{av} \cdot \beta_{av} \cdot (n_{tot} + n_{trojan}) \cdot n_{crit}. \tag{6}$$

Hence, the slope value of I_{ddt} and F_{max} can be used to identify the hardware Trojan circuit. In addition, logic testing solutions, as another hardware Trojan detection primitive, could also be used here. Both logic testing based method and side-channel analysis based method together constitute the hardware Trojan detection primitive of MSIPS.

(B) CRO-PUF Primitive

An RO is a simple circuit which is composed of a set of inverters connected in a loop. The basic idea of CRO-PUF is to generate random sequences by using the delay difference among ROs [40, 49, 50]. The simplest form to generate the output *logic*-0 and *logic*-1 is to compare the frequency difference between a pair of ROs. Multiple pairs of ROs can be utilized to generate more bits in the same way. The frequency depends on the path delay of each inverter and the wires, and it varies randomly with the intrinsic process variations of ICs due to the impact of manufacturing process and other uncertain factors.

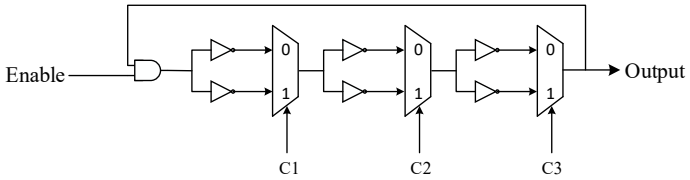


Fig. 8. The basic structure of a CRO circuit.

Fig. 8 illustrates the basic structure of a CRO circuit integrated in each functional IP. We can obtain eight different ROs via the control inputs *C*1, *C*2, and *C*3 of the three 2:1 MUXs rather than using eight separate RO circuits [50]. The configurable logic significantly reduces the silicon area overhead.

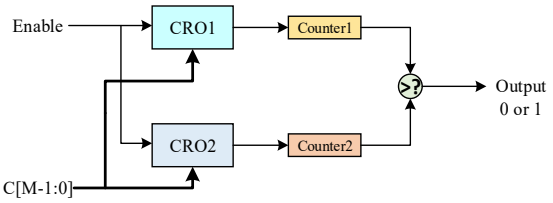


Fig. 9. A general structure of CRO-PUF module.

Fig. 9 presents a typical example of the CRO-PUF circuit. It contains two CRO circuits with which the number of control inputs is M. In theory, the length of each bit string generated by the CRO-PUF is 2^M. By applying the same control input value, these two CRO circuits can be configured into the identical structure. It is desirable that these 2^M pairs of RO circuits will have varying frequency difference due to the process variations in circuit parameters.

The output value of CRO-PUF can be expressed by the following formulation:

$$output = \begin{cases} 1, & f_{CR01} > f_{CR02} \\ 0, & \text{otherwise} \end{cases}. \tag{7}$$

Therefore, because of the negligible hardware overhead, as well as the high uniqueness and robustness, *it is appropriate to integrate the low-overhead CRO-PUF* ***proposed in [50]*** *as a security primitive into each functional IP in order to generate the unique ID signature for device authentication*.

(C) Anomalies Verification Primitive

Architecture-level abnormal events or behaviors include communication interruption, message interception, data modification, information sniffing, malicious access or control, denial of service (DoS), fault injection, spoofing attacks, *etc.* Unlike IP core-level threats, these anomalies are more reflected in the impacts to the SoC system. To effectively identify these security issues, we need to construct a trusted matrix table based on the RBAC and predefine the systematic abnormal events or behaviors first. When a particular exception occurs at runtime, MSIPS will then handle it depending on the corresponding security policies. Unfortunately, security policies in SoC designs are extremely complicated and current IP core-level protection strategies rely on the scheme of static verification [20].

To overcome these problems, the runtime architecture-level behaviors must be validated by either dynamic or formal analysis with the help of trusted matrix table. Table 3 provides the trusted matrix table required. It contains the representative set of architecture-level exceptions and the priority of IPs in each tier. For example, MSIP in Role 3 has the highest operational authority and it can interrupt the operations of other IPs. In particular, to satisfy the requirements of SoC dynamic verification, we translate the behavior characteristics of security events into formal metadatas and simplify the complex security execution strategies into simple operation primitives like *permit*, *reset*, *avoid*, *disable*, *etc.* Finally, the trusted matrix table must be integrated into the *data memory* of security IPs so that the runtime security verification could be executed accordingly to prevent the malicious manipulations introduced by untrusted IPs.

Table 3. Trusted matrix table of architecture-level abnormal events or behaviors.

Role	Type of IPs	Abnormal Events	Associated Metadata	Security Executions
Role 1	Processor IPs	tamper threads, DoS, interrupt or firmware update request, failure, Trojan, theft	status flag, circuit parameters, outputs	permit, reset, avoid, disable
	Memory IPs	read/write specific address, execution mode, failure, theft	status flag, address, data, timestamp	permit, disable, segment
	Communication IPs	invalid, hijack, DoS, Trojan, theft, tamper address or data	packet size, baud rate, address, data, transfer power	permit, disable, avoid
	System-required IPs	power or frequency change, Trojan, theft	frequency, flag, voltage	permit, reset, avoid
	User-defined Hard logic IPs	key exposure, invalid, DoS, failure, Trojan, theft, malicious request	local clock domain, timestamp	permit, reset, avoid, disable
Role 2	Sub security IPs	trust	frame	–
Role 3	Main security	trust	frame	–

5.4 Use Case Scenario Analysis

Here we consider a practical application of MSIPS in the SoC execution scenario. In this use case, we assume that the adversary has already implanted a rogue Trojan instance, which will incur a DoS attack, into the 3PIPs and design the rare conditions based trigger logic for it. The attack model is as follows:

(A) DoS Attacks via Hardware Trojans

Consider such a situation where a hardware Trojan has been implemented into *IP B*. Once triggered, this Trojan circuitry will modify the values of the configuration register, causing the firmware of *IP B* to perform malicious operations. For example, *IP B* keeps asking *IP A* to perform some specific computations. When *IP A* receives the request, it executes the corresponding operation and responds the replies to *IP B*. However, when *IP C* also asks *IP A* to perform other particular operations, the request of *IP C* cannot be executed since *IP B* continuously sends requests to *IP A*, thereby resulting in DoS attack.

(B) Assumptions

We assume that the attacker (*e.g. IP B*) can utilize the rare input act as a trigger condition for the inserted Trojan instance. The firmware in *IP B* contains a piece of malicious code and cooperates with the Trojan circuitry to complete the attack. We also assume that *IP B* is malicious independent and does not collude with other IPs. All access requests to *IP A* will be prohibited by the associated security IPs until the dynamic analysis or verification of the architecture-level abnormal events or behaviors are intended correlation. MSIP will then check the status flag and the event log of each IP, so as to eliminate the effects caused by *IP B*. Below are the as illustrated in Fig. 10.

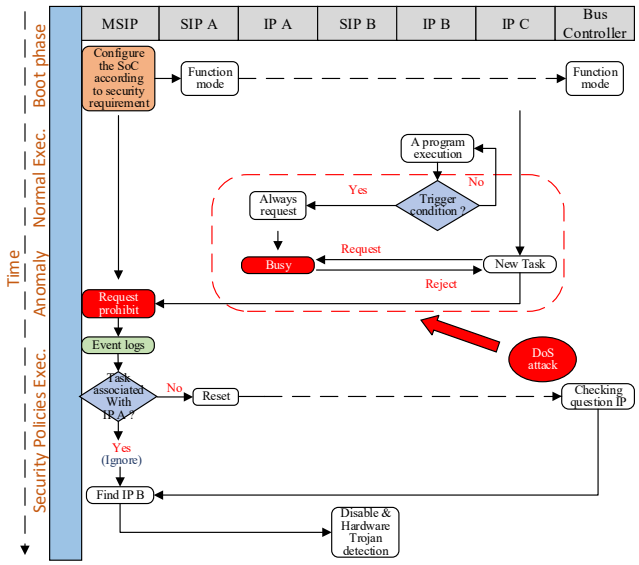


Fig. 10. Operation flow of the proposed solution for preventing DoS attacks.

(C) Flow of the Procedure

(1) At the boot phase, SoC completes the initialization of each security IP as the target requirements of the user or system. For example, all the security IPs are assigned to configure the functional IPs to function mode, so as to perform normal system operations;
(2) During the normal execution, for a program executing on the *IP B*, a rare condition is achieved and then the hardware Trojan in *IP B* is activated. It modifies the configuration register to incur the execution of the malicious code in firmware. At this time, the relevant SIP may not enable the dynamic verification security primitive;
(3) *IP B* starts up secretly and sends the requests to *IP A* so as to execute some specific calculations. Such process continues, keeping the request buffers queue of the *IP A* always in a filled status;
(4) When another IP (*IP C*) asks *IP A* to perform a new operation, the request message is always blocked and never gets a response. And after a period of time, *IP C* reports a feedback to MSIP that the requested operations cannot be executed by *IP A*. Such situation triggers the dynamic monitors and verification policies;
(5) MSIP sends a request frame to the associated security IP to execute the dynamic verification security policies and feedback the current running status of *IP A*.
(6) After MSIP receives the response frame, it then discovers that *IP A* is always working now, thus the execution request is never responded. MSIP then sends the event logs for checking that whether a task is associated with *IP A* (assuming the event logs to be legitimate);
(7) MSIP discovers that *IP A* is communicating with *IP B* via the bus controller, then repeats the step (6) again to identify the illegal operations of *IP B*. A reset command

is sent to *IPA*. Simultaneously, the *disable* and *verify* commands are sent to IPA and IPB;

(8) The DoS attack is now been thwarted by the proposed solution, thus the adversary fails to block the *IP A*.

6. EXPERIMENTAL RESULTS

In order to evaluate the security and effectiveness of MSIPS, a representative SoC benchmark system has been performed on an FPGA platform where the Altera FPGA development boards were used to emulate the ASIC scenario. The MSIPS was implemented on TMSC 90nm Cyclone II FPGA devices used in our self-developed boards. It includes three reference circuits (s1423, s5378, and s9234 from ISCAS-89 benchmark suite) and a security IP (established as a micro-control finite state machine) [51, 52]. Fig. 11 presents the SoC benchmark system created on FPGA platform.

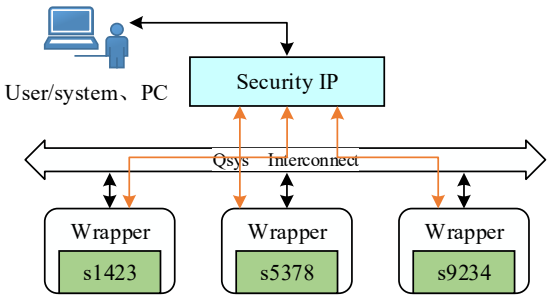


Fig. 11. Block diagram of our SoC benchmark system.

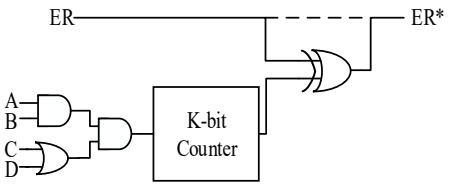


Fig. 12. A generic hardware Trojan circuits implanted in SoC benchmark system.

6.1 Hardware Trojan Detection

A golden chip and its variant with a hardware Trojan instance (called Comb. HTI) are mapped on the FPGA platform. Generally speaking, most hardware Trojan circuits are composed of two basic components: *Trigger logic* and *effective Payload* [31, 33]. Fig. 12 shows the hardware Trojan circuits inserted in each functional IPs. Trigger logic monitors a set of trigger inputs for the sake of activating the Trojan under the prove conditions. Once triggered under rare events or conditions, the hardware Trojan becomes activated and the payload circuit injects an error signal which will alter the values of data bus or control bus [34]. Comb. HTI has 4-trigger inputs (*i.e. A*, *B*, *C*, and *D*) and one

k-bit counter. It will be activated as long as the k-bit counter reaches a predetermined time. The payload circuits are a XOR gate. Assume that the switching probability of each trigger input is 0.5, the activation probability is only $(3/16)^k$. Hence, such hardware Trojan is more difficult to be triggered under the normal conditions.

(A) Test Steps

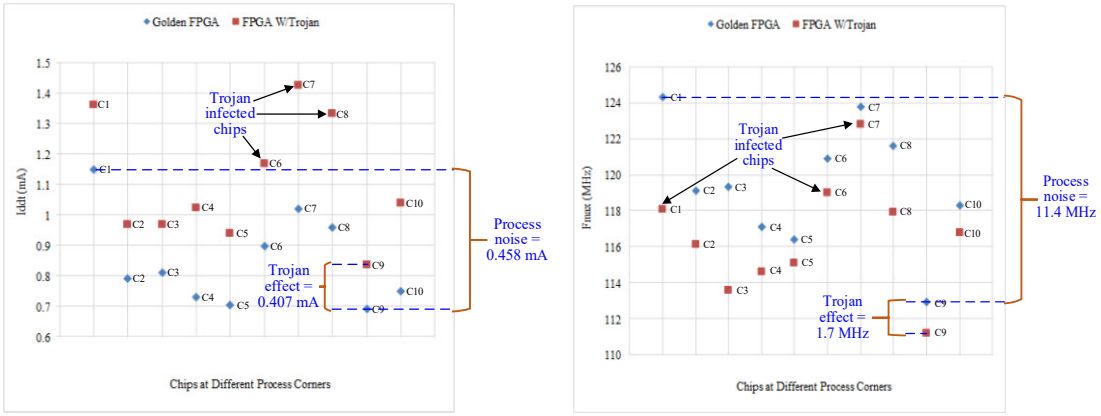
Hardware validation of the hardware Trojan detection is performed using the FPGA platform to emulate the practical scenario. In our experiment, we take the s1423 circuit as an example to validate the hardware Trojan detection technique. The detailed procedure involves three steps, as described follows:

(1) Acquisition of I_{ddt} values. The identical test vectors are applied to the golden chips and its variants separately, and then obtaining the I_{ddt} values.
(2) Extracting F_{max} values. At the same time, we extract the corresponding frequency values (F_{max}) of each chips.
(3) Calculating the slope. According to the I_{ddt} and F_{max} values acquired, we have calculated the slope values of each FPGA chip and infer whether the chip under detection is infected with a Trojan circuit.

In order to measure the I_{ddt} of each sample circuit, we first insert eight 30mR precision current sense resistors at each V_{icco} port of an FPGA device. The shunt I_{ddt} values are acquired and recorded through a dedicated current/voltage collecting module. Then, the frequency, F_{max}, are measured through a 15-inverter RO circuit with an on-chip counter. Here we choose a 1GHz clock signal as reference to complete the counting. Furthermore, we perform the detection procedure on ten different FPGAs, which are from the same lot and placed in the same platform. The same designs are also mapped into each FPGA chip. In particular, we randomly select two of the FPGA chips, chip 3 and chip 4, to implement the Comb. HTI Trojan infected s1423 circuits and the remaining eight FPGA devices are selected as golden chips to map the gate-level netlists of s1423 original circuits.

(B) Results

The experimental results for the side-channel analysis based hardware Trojan detection approach are provided in Figs. 13 and 14. Among them, Fig. 13 shows the single-parameter based hardware Trojan detection results on FPGA while Fig. 14 illustrates the results for multi-parameter based testing method. The results in Fig. 13 demonstrates that since the impact of a Trojan circuit on the circuit parameters may be drowned in the process noise, the measurement values of I_{ddt} (Fig. 13 (a)) or F_{max} (Fig. 13 (b)) only may not be able to capture the effects of a Trojan under parameter variations and isolate the FPGA chips infected with Trojan instances from the uninfected ones. Taking the chip C9 as an example, the Trojan effect on the I_{ddt} and F_{max} is respectively 0.407mA and 1.7 MHz, which are less than the process noise (*i.e.* 0.458mA and 11.4MHz). Thus, the deviations in I_{ddt} or F_{max} due to variation can easily mask the effect of the inserted Comb. HTI Trojan, making it infeasible to isolate from the process noise. In addition, since the boundary between them is not clear, a simple comparison can result in a large number of false detection (as shown in Fig. 13).



(a) (b)

Fig. 13. The single-parameter based Trojan detection results on FPGA; (a) Measurement of average I_{ddt} values only at 10 random process corners; (b) Corresponding F_{max} values.

On the other hand, multi-parameter based testing approach can overcome this issue and effectively differentiate between the original and tampered versions (as shown in Fig. 14). For a set of golden FPGA devices, the relationship values, *i.e.* I_{ddt} vs. F_{max}, follow an expected variation curve under process noise, while the one which deviates from the variation curve implies that it may have a Trojan circuit inside. The black breakpoint line in Fig. 14 shows this variation curve. Since the C3 and C4 already have been implanted a Trojan instance, multi-parameter based testing will find that they stand out from the variation curve. Moreover, the different trigger conditions of a Trojan circuit will result to the difference experimental results between the infected chips and the golden ones.

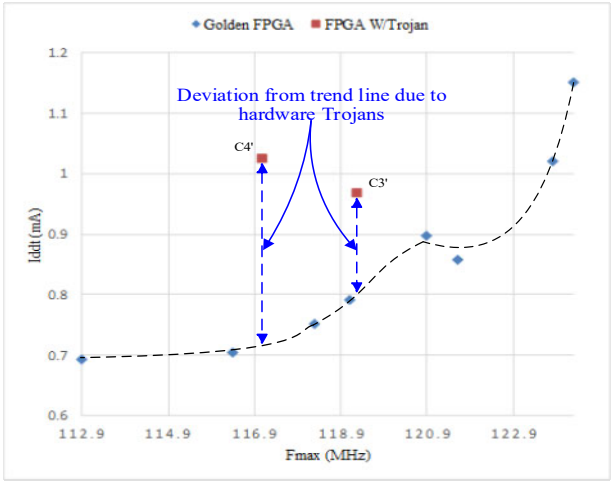


Fig. 14. Variation curve of the slope values (I_{ddt} vs. F_{max}) acquired on 10 FPGA chips using multi-parameter based Trojan detection technique.

6.2 IP Authentication

For IP authentication, a set of test vectors is transmitted by the security IPs to generate the unique digital signature for the purpose of resisting IP thefts. Here the functional IPs are configured and assigned so as to initialize the PUF controller module and then to proceed with the procedure of digital signature generation, extraction, and verification. In the experiment, each of the CRO circuits utilized in the CRO-PUF primitive can be configured to 128 different RO circuits. It includes fourteen inverters and seven 2:1 MUX circuits, and seven inverters are cascaded on each RO circuit as a buffer to analyze the delay distribution of RO circuits under process variations [50]. Thus, the CRO-PUF circuits can generate N 128-bit signatures using the control inputs under the ideal conditions. However, due to the impact of structural differences, only the pairs of RO circuits with the same structure are compared.

(A) Test Steps

The signature generation and authentication are performed as follows:

(1) Security IP initialization: In the testing state, security IPs complete the initialization of the CRO-PUF design, generate the required test vectors, and reset each port of the CRO-PUF module simultaneously.
(2) Signature generation: Launch the control input values to C[6:0] ports after the initialization of security IPs. Then Enable port is set and the counter starts working. After a period of time t_{period}, compare the value of the counting values and output the result.
(3) Signature extraction and verification: Change the value of control inputs, security IP first reads the results of output sequentially, then compares the signature obtained with the results stored in data memory to draw a conclusion whether the functional IP under validated is security.

(B) Results

Hardware validation of the CRO-PUF scheme is performed on the Altera Cyclone II FPGA platform. In order to eliminate the influence of systemic variations, the entire configurable RO-PUF circuits are created into the continuous logic array blocks (LABs) [42]. The "Logic lock" function of Altera FPGA makes the region partition on the floorplan of a FPGA chip come true. Fig. 15 shows the generic LABs with a CRO circuit inside.

We have also implemented a CRO-PUF design with identical routing and layout in ten different FPGA devices, and validate them from the point of uniqueness and reliability through the Hamming distances (HDs). In particular, the HDs can be expressed by the following formulation which represents the number of positions in two binary strings corresponding to different bits.

$$HD(R_i, R_j) = \sum_{i=1}^{n} R_i[1] \oplus R_i[1] \oplus R_j[1] \tag{8}$$

(1) Uniqueness

Uniqueness represents how unique a PUF response can be, *i.e.*, all the digital signatures generated by each chip are different even the identical silicon PUF silicon with the same routing and layout. Furthermore, the different routing and layout of the same

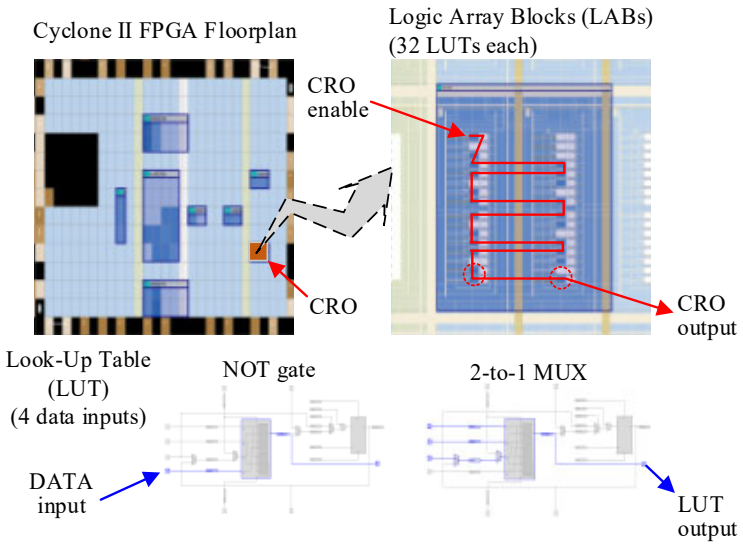
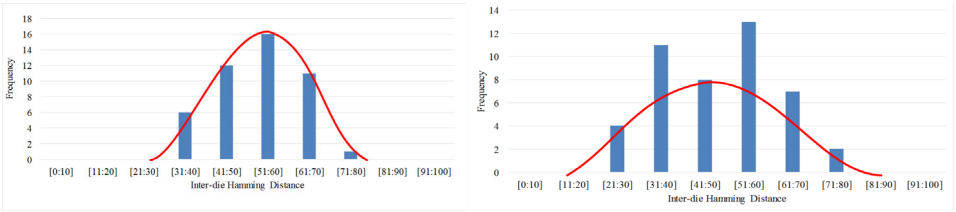


Fig. 15. Generic LABs with a CRO circuit implemented on FPGA.

PUF structure in the identical chip will also introduce to different responses. We estimate the uniqueness of the CRO-PUF by the average inter-die HDs over a group of chips. With a pair of signatures, R_i and R_j ($i \neq j$), both having n-bit response, the average inter-die HDs among a group of k chips can be expressed as:

$$uniqueness = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i, R_j)}{n} \times 100\%. \tag{9}$$

Here we use 10 FPGA chips to evaluate the uniqueness of the CRO-PUF, and all the experimental results are obtained under the normal operating condition (20 degrees operating temperature and 1.2V V_{icco} voltages). The ten digital signatures are compared and calculated with each other by the average inter-die HDs. The length of the digital signature generated is 128 bits. Fig. 16 exhibits the uniqueness distribution among 10 FPGA chips. From Fig. 16, the differences of pairs of path delay for *logic*-0 to *logic*-1 transition approximately follow the Gaussian distribution and the CRO-PUFs are more concentrated on the ideal expected 64 than the common RO-PUFs. In addition, we have also calculated the average inter-die HDs of the CRO-PUF and the common RO-PUF.



(a) Inter-die HDs of the CRO-PUFs. (b) Inter-die HDs of the common RO-PUFs.

Fig. 16. The uniqueness distribution among 10 FPGA chips.

Table 4. Uniqueness comparison between common RO-PUF and configurable RO-PUF.

Uniqueness	Com. RO-PUF	Config. RO-PUF
Average Inter-die HDs	38.1%	42.7%

Table 4 shows the comparison of uniqueness between them. From Table 4, we can see that the average inter-die HDs of the CRO-PUFs is 42.7% of PUF signature bits and are a little larger than the RO-PUFs (*i.e.* 38.1% of PUF signature bits), which means that the CRO-PUFs are a slight higher unclonable and random than the common RO-PUFs. In particular, the inter-die HDs will be 50% under the ideal conditions.

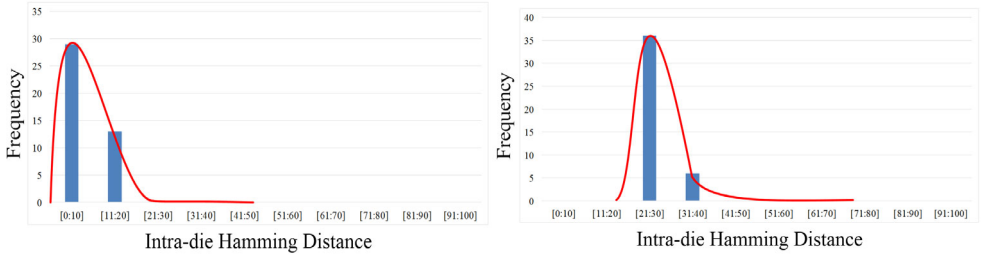
(2) Reliability

The PUF responses are expected to be the same under the same challenges. However, a variety of environmental variations such as temperature and voltage may introduce the changes in them, and thus affect the stability of the PUF responses. Reliability means that the difference between any two signatures generated by the same device should be slight under different conditions. In particular, the average intra-die HDs can be exploited to evaluate the reliability of PUFs, *i.e.* $HD(R_i, R_{i,y})$ over k samples. The following formulation can be defined as:

$$reliability = \frac{1}{k}\sum_{y=1}^{k}\frac{HD(R_i, R_{i,y})}{n}\times 100\%. \tag{10}$$

To estimate the reliability of the CRO-PUF, we randomly select a FPGA chip i from the 10 FPGA chips and extract a 128-bit signature from the chip i at the normal operating condition. After that, we extract the same 128-bit response under different operating conditions. Fig. 17 shows the reliability distribution among different operating temperatures. It should be noticed that the distribution of intra-die HDs is more concentrated on the ideal expected 0 than the common RO-PUFs. In addition, we then calculate the average intra-HDs of the CRO-PUFs and the common RO-PUFs.

Table 5 presents the comparison of reliability between them. From Table 5, we can see that the CRO-PUFs have the higher average intra-die HDs which means that the CRO-PUFs are much more reliability and reproducibility than the common RO-PUFs with the changes of operating temperatures. In particular, the intra-die HDs will be 0% under the ideal conditions.



(a) (b)

Fig. 17. The reliability distribution among a variety of operating temperature; (a) Intra-die HDs of the CRO-PUFs; (b) Intra-die HDs of the common RO-PUFs.

Table 5. Reliability comparison between common RO-PUF and configurable RO-PUF.

Reliability	Operating Conditions (Temperature)	Com. RO-PUF	Config. RO-PUF
Average Intra-die HDs	20 degree	16.4%	7.0%
	30 degree	16.3%	7.0%
	40 degree	17.1%	5.8%
	50 degree	17.9%	6.7%
	60 degree	17.0%	6.3%
	70 degree	19.3%	5.9%
	80 degree	18.2%	6.0%

6.3 Abnormal Event or Behavior Verification

To validate the practical application of MSIPS described in Section 5.4, we have designed an instance of the proposed MSIPS scheme for this SoC benchmark system model using the FPGA platform (as shown in Fig. 18). We configure the *IP B* and *IP C* with the similar functionality, *e.g.* they are simple NIOS II embedded processors with serial communication ports, while *IP A* could be configured as a memory IP, such as RAM memory IP or FIFO IP, *etc.* All of these IPs (including the security IP) are interconnected through an Avalon-Memory Mapped (Avalon-MM) pipeline bridge. Avalon-MM can provide a mechanism for simultaneous connection a slave interface to both a processor master local to a subsystem and an external processor master elsewhere in the hierarchy. Moreover, we also provide protection of mutually exclusive accessing with respect to shared *IP A*. In this case, only one processor has ownership of the hardware mutex at any time. In particular, we have created and implanted a hardware Trojan into *IP B*, which runs at 50Mhz in the tested FPGA platform. After triggered, such a Trojan circuit could incur DoS attacks to *IP A*, that is, will not release the hardware mutex. Thus, *IP C* can never access *IP A*.

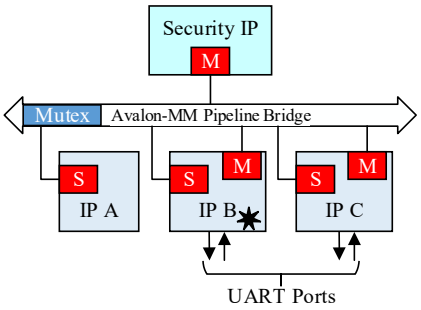


Fig. 18. Block diagram of a SoC benchmark system instance.

(A) Test Steps

For security IPs performing anomalies verification, we first select *IP B* to receive the data through UART ports and write them into *IP A*. Then, *IP C* reads the data from *IP A* and transmits them via UART ports. In particular, the Trojan instance designed is a

FSM circuit which is triggered after monitoring a continuous input string of "*HTD*". After that, it changes the values of a private register so as to incur the execution of a piece of malicious code, thus *IP B* no longer releases the hardware mutex. The detailed procedure can be presented as follows:

(1) Initialization. After power up, SoC system first completes the initialization according to the predefined configuration and waits for receiving data;
(2) Normal execution verification. Then, we continuously send a series of strings to the target SoC system through the serial ports of host PC. After a while, check whether the host PC has received the sent strings. This step is intended to verify the normal operation of the target SoC system;
(3) Trigger the hardware Trojan. A specific string which contains the piece of "*HTD*" is sent to the target SoC system to trigger the hardware Trojan inside *IP B*;
(4) Abnormal behavior occurs. The host PC will receive the abnormal message due to the reason that *IP C* fails to access and read *IP A*;
(5) Security policies launch. The security IP will launch the corresponding security policies and feature values to verify the abnormal behaviors and eliminate it;
(6) Anomalies recovery. The host PC will receive the handling results and begin to receive the strings normally.

(B) Results

Collecting the relevant data based on the hardware behavior analysis is the first and important step in the abnormal behavior verification. Each feature should be better correlation between the result and feature, rather than themselves. Therefore, we do select relevant features based on feature correlation analysis. Here we consider the following features for anomalies verification:

– System Mode: Current Mode of SoC System
– Problem Core: Suspicious Core Number
– Current Status 1: Current Status of Source Core
– Event Log Record 1: Task Number of Problem Core
– Relevant Core: Relevant Core Number
– Current Status 2: Current Status of Distance Core
– Event Log Record 2: Task Number of Relevant Core
– Specific Features (if Needed): Specific Features relevant to each functional IP

Table 6 gives some examples of test records from the security IP in case of hardware Trojan attacks and without Trojans. Observations 1, 3 and 4 show the DoS attacks whereas observations 2 and 5 show the test and normal operation respectively.

We examine observation 3 as example. Here, we set the baud rate of UART to 9600 pbs. When a transfer and storage operation is completed, the time consumption is approximately 6.57 msec. During the verification, security IP find that *IP B* is now locking *IP A* through the *OWNER* filed (generated by CRO PUF) of the hardware mutex register and find a latest task (*11: IP B writes to IP A*) existed in the system logs. Till now, suspicious IP core cannot be inferred because this *alarm* may be caused by the normal operation which takes a bit more time than the threshold value of the interval time ($T_{interval}$ = 10 *msec*). So, we further verify the configure registers of each IP. For this, security IP

compare the current value of configure registers to the user- or pre- defined ones. Hence, the mismatch reveals the existence of a Trojan horse that is triggered by external conditions. In particular, the execution results are shown in Fig. 19.

Table 6. Observation results from security IP.

Feature / Example	System Mode	Problem Core	Current Status	Config. Register	Task ID	Relevant Core	Current Status	Task ID	Config. Register	Results
1	0	IP A	1	1	05	IP B	1	04	–	0101
2	1	IP A	1	1	08	SIP	–	–	–	1000
3	0	IP A	1	1	11	IP B	1	11	0	0110
4	0	IP A	1	1	13	IP B	0	11	–	0111
5	0	IP A	1	1	17	IP B	1	17	1	0100
…										

* "System Mode" –0 normal mode, –1 test mode.
"Current Status" –0 standby, –1 busy.
"Config. Register" –0 mismatch, –1 match.
"Results" – result values. For example, "0110" means relevant core is executed under normal mode and injected by a DoS Trojan.

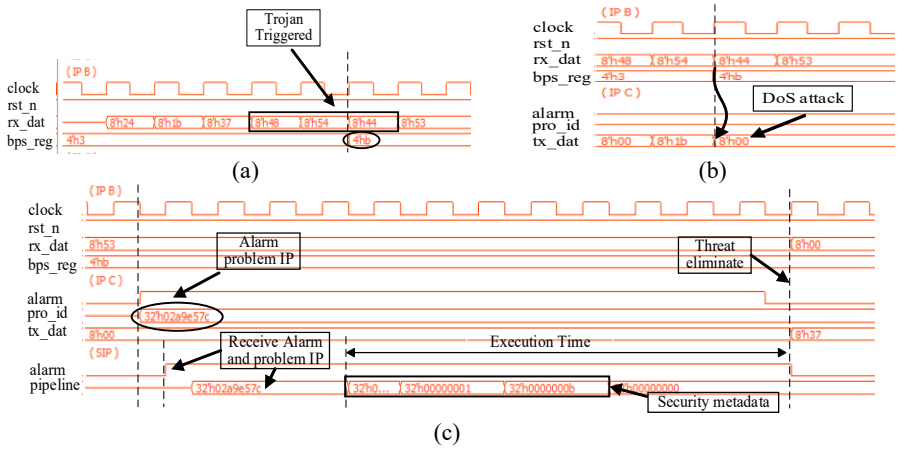


Fig. 19. Simulation results of security IP against DoS attacks; (a) Trigger the Trojan instance; (b) *IP C* fails to read *IP A*; (c) Alarm threats and eliminate them.

6.4 Overhead Analysis

(A) SoC Benchmark System Overhead

The estimated hardware overhead of the SoC benchmark system incurred by IIP and MSIPS is presented in Table 7. It contains three functional IPs: an s1423, an s5378 and an s9234 circuits. Moreover, we not only consider the micro-controller implementations due to the fundamentally invariant characteristics with respect to the structures and security policies in various security IPs, but also include implementing the security interface wrappers for functional IPs to assist protection for SoC system against various attacks. Most notable is the instruction and data memory, a 4K-Bytes RAM memory is added to the base memory controllers to calculate the security IP overhead.

Table 7. Hardware overhead of security IP with respect to SoC benchmark system (*using TMSC 90nm CMOS Press Model*).

Cores/SoC (# of LUTs)	SoC benchmark system								
	Functional IP cores						SoC	IIPS	MSIPS
	S1423	Wrapper	S5378	Wrapper	S9234	Wrapper			
Wang *et al.* [3]	728	25(+3.43%)	2192	43(+1.96%)	3376	30(+0.89%)	6394	1559 (+24.38%)	–
Our scheme	728	67(+9.20%)	2192	86(+3.92%)	3376	72(+2.13%)	6521	–	1561 (+23.94%)

From Table 7, we conclude that the implantation of a security IP, and wrappers, *etc.*, in the SoC benchmark system makes the total area overhead increased by about 23.94%, compared with the IIPS architecture incurring 24.38% increasing area overhead. However, from the sample values in Table 7, the overhead incurred by wrappers of MSIPS is a little larger than that of IIPS, which reduces the proportion of security IP in SoC system.

(B) Hardware Trojan Overhead

Area overhead of the hardware Trojan employed in the SoC benchmark system is given in Table 8. The combinatorial hardware Trojan has an advantage in area overhead over the sequential hardware Trojan. It is worth taking note that the combination of these two hardware Trojan circuits may incur a slight increase in area overhead, but it will produce a more concealed hardware Trojan which is more difficult to be detected by current hardware Trojan detection methods. In particular, the Comb. HTH instance only accounts for 3.16% of the total s1423 circuit.

Table 8. The hardware overhead of hardware trojans.

Overhead \ Circuit		HTH benchmark	IP Core
		Comb. HTI	S1423
Average hardware overhead	Area(# of LUTs)	23	728
	Gates/Registers	14/17	490/74
	Input Lines	4	17
	Output Lines	1	22

(C) CRO-PUF Overhead

Furthermore, we make a comparison on the area overhead of common RO-PUF and CRO-PUF in generating one 128-bit signature. For the common RO-PUF, it actually includes 128 RO rings and two 128:1 MUXs. For the CRO-PUF, two CROs are contained in it. In addition, it also requires two 32-bit counters and a 32-bit comparator for each RO-PUF. The areas of each RO-PUF are listed in Table 9, which is acquired from the FPGA compiler and synthesis. Since the 2:1 MUX is inserted into the RO rings to form the configurable feature, the total logic elements of Configurable RO-PUF only take up 15.29% of the common RO-PUF in the case of generating the same length of signatures.

Table 9. Area comparison between common RO-PUF and configurable RO-PUF.

	Com. RO-PUF	Config. RO-PUF
# of LUTs	1524	233
# of Registers	74	74

7. DISCUSSION

This section mainly analyzes the characteristics of our proposed MSIPS approach. The flexibility and scalability features of Security IPs are discussed in Sections 7.1 and 7.2, respectively. Integrity and security effectiveness are demonstrated in Section 7.3. Section 7.4 anlyzes the exection time of the proposed MSIPS.

7.1 Flexibility

The MSIPS architecture presented in this article is established on a variety of security IPs deployed through a distributed methodology. Flexibility is the primary characteristic of security IPs, which is mainly reflected in the following aspects; (1) the diversity of security primitives. Since the types of functional IPs are various, the corresponding security primitives are also different. These defense measures can be integrated into the security IPs, which could achieve system-level target security protection for SoC designs; (2) the arbitrariness of deployment. As low-overhead on-chip modules, security IPs are implanted into the SoC designs and reside outside the functional IPs. In particular, they resemble normal functional IPs and act only when a security event occurs, making it a challenge for adversaries to seek them; (3) the selectivity of protecting objects. Users can optionally choose to protect for part or all function IPs in a SoC design.

Furthermore, MSIPS reflects good portability and compatibility. Since the security IPs are connected to functional IPs through the test wrapper interfaces, the design and/or integration of them can be adjusted accordingly. In case of SoCs with another test wrappers, they can be extended to interface with them easily.

7.2 Scalability

The proposed MSIPS architecture exhibits good scalability in respects of integrating more security primitives or defense policies at minimal extra overhead. It can be adjusted to provide protection against other threats, *e.g.* scan-based attack or side-channel attack on crypto cores, reverse engineering. For instance, a small noise injector can be embedded so as to mask the critical information disclosed through electromagnetic, power consumption, transient current [15]. Moreover, MSIPS can simultaneously employ multiple countermeasures for certain attacks to improve the overall security and trustworthiness of SoCs [53]. For example, logic testing and side-channel analysis based method can be incorporated to accomplish the hardware Trojan detection (see Section 3.1). In addition, MSIPS can utilize the same information to achieve different verifications. An on-chip current monitoring module can be integrated to detect hardware Trojans at the test time [54], or to assist in supervising the abnormal behavior of functional IPs at the run time. Finally, security functions can also be implemented in hardware (*e.g.* FPGA) or software (*e.g.* microcontroller) programmable fabric so that it can be tailored dynamically according to the capabilities.

In addition, MSIPS shows also good scalability when applied in large complex SoCs due to the reason that defenders could choose more security IPs to construct the MSIPS scheme according their security demands. Though this policy increases the area overhead, it can comprehensively protect for large complex SoCs from the system-wide perspective.

7.3 Security of MSIPS

Security of MSIPS itself against various threats is also a critical issue which must be considered. The trustworthiness of security IPs is important to ensure the effectiveness of its security policies. Security IPs which are designed or integrated into a SoC can be acquired from untrusted 3P vendors, and the MSIPS itself can be specifically created by the security teams or SoC architects in untrusted environments. In particular, malicious modification of MSIPS or security IPs can possibly be made in an untrusted foundry. Hence, to enhance the security and trustworthiness of MSIPS, low-overhead hardware obfuscation or camouflaging techniques could be employed here to resist the security issues mentioned above [24, 25]. These solutions can also combat the reverse engineering attacks.

On the other hand, the trusted matrix table stored in the *data memory* of security IPs may also be compromised. Since the on-field upgrades of security policies are permitted, any player who has the right to access it may illegally update or forge the trusted matrix table, or even counterfeit the security IPs themselves. This can seriously affect the robustness of MSIPS, especially the security IPs. To effectively handle these threats, we implement an authentication mechanism based on challenge-response keys (as shown in Fig. 5). We avoid on-chip key storage for the reason that they could suffer reverse engineering and be cloned. Instead, keys are generated at boot time using a PUF-based standard technique. Then, the keys are exploited to complete security authentication. Only authorized players are permitted to access or update the trust matrix table.

7.4 Execution Time Analysis

When protecting against various security issues, time is of the essence since they can cause hardware Trojan attacks, thefts, or abnormal behaviors. In our MSIPS, the execution of security policies for a certain threat is processed in parallel due to the distributed deployment of security IPs. Thus, the detection time is relied on the worst case of the IPs. While in IIPS, due to the scan chain features, the total detection time is the superposition of the time consumption of each IP. Although the advantages of our approach may not be reflected well in the single core test, it could greatly reduce the time overhead when executed in the multiple cores detection. In particular, our scheme is a typical space-for-time defense strategy.

8. CONCLUSION

In this paper, we proposed a centralized on-chip security architecture, referred as Multi-tiered Security IPs (MSIPS), to provide adequate protection against diverse threats.

It can effectively achieve the verification, test, and authentication of SoC during test time and run time, and recover from abnormal states. Besides, it can also monitor the running states of functional IPs in order to discover their abnormal behaviors. It features ease of integration, low-overhead, flexibility, diversity, and scalability. We have validated the functionality and security of MSIPS through experimental measurements on a FPGA platform. FPGA implementations have shown the feasibility of integrating a PUF primitive, a hardware Trojan detection primitive, and abnormal behavior or event validation and monitoring primitive into security IP to provide security defense. The FPGA implementation shows that MSIPS incurs low hardware overhead. Based on this preliminary study we conclude that much interesting work remains to be done in terms of the achievement of MSIPS and its security validation. Future work will include: to integrate the security IP with other bus topology; to supervise and identify the abnormal behavior by machine learning; as well as to extend its capability so as to provide protection against other attacks.

ACKNOWLEDGEMENT

This work was supported by a grant from the National Natural Science Foundation of China Program (Program ID 61572385), and completed under the guidance of Professor Quan Wang. Opinions, findings, conclusions and recommendations expressed in this material are those of the authors and may not reflect the views of the funding entities. Quan Wang is the corresponding author.

REFERENCES

1. Z. Huang and Q. Wang, "MSIPS: Multi-tiered security IPs architecture for secure SoC design," in *Proceedings of International Conference on Networking and Network Applications*, 2017, pp. 203-208.
2. A. Basak, S. A. Basak, S. Bhunia, T. Tkacik, and S. Ray, "Security assurance for system-on-chip designs with untrusted IPs," *IEEE Transactions on Information Forensics and Security*, Vol. 12, 2017, pp. 1515-1528.
3. X. M. Wang, Y. Zheng, A. Basak, and S. Bhunia, "IIPS: Infrastructure IP for secure SoC design," *IEEE Transactions on Computers*, Vol. 64, 2015, pp. 2226-2238.
4. A. Antonopoulos, C. Kapatsori, and Y. Makris, "Trusted analog/mixed-signal/RF ICs: A survey and a perspective," *IEEE Design and Test*, Vol. 34, 2017, pp. 63-76.
5. Y. Q. Lv, Q. Zhou, Y. C. Cai, and G. Qu, "Trusted integrated circuits: Problem and challenges," *Journal of Computer Science and Technology*, Vol. 5, 2014, pp. 918-928.
6. S. Narasimhan, D. D. Du, R. S. Chakraborty, *et al.*, "Hardware trojan detection by multiple-parameter side-channel analysis," *IEEE Transactions on Computers*, Vol. 62, 2013, pp. 2183-2195.
7. Y. X. Lu, M. P. O'Neill, and J. V. McCanny, "FPGA implementation and analysis of random delay insertion countermeasure against DPA," in *Proceedings of International Conference on ICECE Technology*, 2008, pp. 201-208.

8. Y. Q. Zhao, J. L. He, S. Yang, and S. F. Liu, "Research on defense technology against hardware trojans in integrated circuits," *Computer Engineering*, Vol. 42, 2016, pp. 128-137.
9. P. Bernardi, M. Rebaudengo, and M. S. Reorda, "Exploiting an I-IP for in-field SoC test," in *Proceedings of IEEE Internatioinal Symposim on Defect and Fault Tolerance in VLSI Systems*, 2004, pp. 404-412.
10. B. Zhou, W. Zhang, S. Thambipillai, *et al.*, "Cost-efficient acceleration of hardware trojan detection through fan-out cone analysis and weighted random pattern technique," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 35, 2016, pp. 792-805.
11. L. A. Guimaraes, R. P. Bastos, and L. Fesquet, "Detection of layout-level trojans by monitoring substrate with preexisting built-in sensors," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, 2017, pp. 290-295.
12. B. Liu and B. Wang, "Reconfiguration-based VLSI design for security," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Vol. 5, 2015, pp. 98-108.
13. A. Johnson, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-enabled secure architecture for FPGA-based IoT applications," *IEEE Transactions on Multi-Scale Computing Systems*, Vol. 1, 2015, pp. 110-122.
14. U. Juin, K. Huang, D. DiMase, *et al.*, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," in *Proceedings of the IEEE*, Vol. 102, 2014, pp. 1207-1228.
15. X. Wang, W. Yueh, D. B. Roy, S. Narasimhan, Y. Zheng, S. Mukhopadhyay, D. Mukhopadhyay, and S. Bhunia, "Role of power grid in side channel attack and power-grid-aware secure design," in *Proceedings of the 50th ACM/EDAC/IEEE Design Automation Conference*, 2013, pp. 1-9.
16. S. Karmakar and D. R. Chowdhury, "Scan-based side channel attack on stream ciphers and its prevention," *Journal of Cryptographic Engineering*, Vol. 8, 2017, pp. 327-340.
17. Y. Jin, D. Maliuk, and Y. Makris, "A post-deployment IC trust evaluation architecture," in *Proceedings of IEEE 19th International On-Line Testing Symposium*, 2013, pp. 224-225.
18. Y. Jin, D. Maliuk, and Y. Makris, "Post-deployment trust evaluation in wireless cryptographic ICs," in *Proceedings of Conference on Design*, *Automation and Test in Europe and Exhibition*, 2012, pp. 965-970.
19. K. Guha, D. Saha, and A. Chakrabarti, "RTNA: Securing SoC architectures from confidentiality attacks at runtime using ART1 neural networks," in *Proceedings of the 19th International Symposium on VLSI Design and Test*, 2015, pp. 1-6.
20. A. Basak, S. Bhunia, and S. Ray, "A flexible architecture for systematic mplementation of SoC security policies," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, 2015, pp. 536-543.
21. L. W. Kim and J. D. Villasenor, "Dynamic function verification for system on chip security against hardware-based attacks," *IEEE Transactions on Reliability*, Vol. 64, 2015, pp. 1229-1242.
22. B. Vermueulen, "Design-for-debug to address next-generation SoC debug concerns," in *Proceedings of IEEE International Test Conference*, 2007, p. 1.

23. A. Basak, S. Bhunia, and S. Ray, "Exploiting design-for-debug for flexible SoC security architecture," in *Proceedings of the 53rd ACM/EDAC/IEEE Design Automation Conference*, 2016, pp. 1-6.
24. M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," in *Proceedings of the IEEE*, Vol. 102, 2014, pp. 1283-1295.
25. M. Rostami, F. Koushanfar, J. Rajendran, and R. Karri, "Hardware security: Threat models and metrics," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design* , 2013, pp. 819-823.
26. P. Mishra, S. Bhunia, and S. Ravi, "Tutorial T2: Validation and debug of security and trust issues in embedded systems," in *Proceedings of the 28th International Conference on VLSI Design and the 14th International Conference on Embedded System*, 2015, pp. 3-5.
27. Y. Y. Zhang, Y. L. Shen, H. Wang, *et al.*, "On secure wireless communications for IoT under eavesdropper collusion," *IEEE Transactions on Automation Science and Engineering*, Vol. 13, 2016, pp. 1281-1293.
28. S. Krstic, J. Yang, D. W. Palmer, *et al.*, "Security of SoC firmware load protocol," in *Proceedings of IEEE International Symposium on Hardware-Oriented Security and Trust*, 2014, pp. 70-75.
29. S. Ray, J. Yang, A. Basak, and S. Bhunia, "Correctness and security at odds: Post-silicon validation of modern SoC designs," in *Proceedings of the 52nd ACM/EDAC/IEEE Design Automation Conference*, 2015, pp. 1-6.
30. M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, Vol. 27, 2010, pp. 10-25.
31. Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *Proceedings of IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 51-57.
32. K. Xiao and M. Tehranipoor, "BISA: Built-in self-authentication for preventing hardware trojan insertion," in *Proceedings of IEEE International Symposium on Hardware-Oriented Security and Trust*, 2013, pp. 45-50.
33. D. Agrawal, S. Baktir, D. Karakoyunlu, *et al.*, "Trojan detection using IC fingerprinting," in *Proceedings of IEEE Symposium on Security and Privacy*, 2007, pp. 296-310.
34. S. Yao, X. M. Chen, J. Zhang, *et al.*, "DeTrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans," in *Proceedings of IEEE International Test Conference*, 2015, pp. 1-10.
35. D. Mukhopadhyay and R. S. Chakraborty, "Testability of cryptographic hardware and detection of hardware trojans," in *Proceedings of Asian Test Symposium*, 2011, pp. 517-524.
36. M. Abramovici and P. Bradley, "Integrated circuit security – New threats and solutions," *China Gems & Jades*, 2009, pp. 1-3.
37. V. V. D. Leest and P. Tuyls, "Anti-counterfeiting with hardware intrinsic security," in *Proceedings of Design*, *Automation and Test in Europe Coference and Exhibition*, 2013, pp. 1137-1142.
38. X. X. Sun, H. Wang, J. Y. Li, and Y. Zhang, "Injecting purpose and trust into data anonymisation," *Computer and Security*, Vol. 30, 2011, pp. 332-345.

39. J. Zhang, X. Tao, and H. Wang, "Outlier detection from large distributed databases," *Journal of World Wide Web*, Vol. 17, 2014, pp. 539-568.
40. J. L. Zhang, G. Qu, Y. Q. Lv, and Q. Zhou, "A survey on silicon PUFs and recent advances in ring oscillator PUFs," *Journal of Computer Science and Technology*, Vol. 4, 2014, pp. 664-678.
41. M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Transactions on Reconfigurable Technology and Systems*, Vol. 2, 2009, pp. 1-33.
42. W. Stallings and L. Brown, *Computer Security: Principles and Practice*, 3rd ed., 2016, pp. 81-95.
43. A. Ouaddah, H. Mousannif, A. A. Elkalam, and A. A. Ouahman, "Access control in IoT: Survey & state of the art," in *Proceedings of the 5th International Conference on Multimedia Computing and Systems*, 2016, pp. 272-277.
44. E. Sahafizadeh and S. Parsa, "Survey on access control models," in *Proceedings of the 2nd International Conference on Future Computer and Communication*, Vol. 1, 2010, pp. v1-1-v1-3.
45. F. N. Shang, W. M. Wu, and Y. H. Gu, "A unified model of RBAC and DAC," in *Proceedings of the 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce*, 2011, pp. 4547-4550.
46. M. A. Aftab, M. A. Habib, N. Mehmood, M. Alsam, and M. Irfan, "Attributed role based access control model," in *Proceedings of Information Assurance and Cyber Security*, 2015, pp. 83-89.
47. M. E. Kabir, H. Wang, and E. Bertino, "A role-involved purpose-based access control model," *Information Systems Frontiers*, Vol. 14, 2012, pp. 809-822.
48. X. Sun, M. Li, H. Wang, and A. Plank, "An efficient hash-based algorithm for minimal K-anonymity," in *Proceedings of the 31st Australasian Conference on Computer Science*, 2008, pp. 101-107.
49. M. Mustapa, M. Niamat, M. Alam, and T. Killian, "Frequency uniqueness in ring oscillator physical unclonable functions on FPGAs," in *Proceedings of IEEE 56th International Midwest Symposium on Circuits and Systems*, 2013, pp. 465-468.
50. A. Maiti and P. Schaumont, "Improved ring oscillator PUF: An FPGA-friendly secure primitive," *Journal of Cryptology*, Vol. 24, 2011, pp. 375-397.
51. D. Bryan, "The ISCAS'85 benchmark circuits and netlist format," in *Proceedings of International Symposium on Circuits And Systems*, 1985, pp. 1-4.
52. F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran," in *Proceedings of International Symposium on Circuits And Systems*, 1985, pp. 695-698.
53. F. Koushanfar and A. Mirhoseini, "A unified framework for multimodal submodular integrated circuits trojan detection," *IEEE Transactions on Information Forensics and Security*, Vol. 6, 2011, pp. 162-174.
54. S. Narasimhan, W. Yueh, X. Wang, S. Mukhopadhyay, and S. Bhunia, "Improving IC security against trojan attacks through integration of security monitors," *IEEE Design & Test of Computers Special Issue on Smart Silicon*, Vol. 29, 2012, pp. 37-46.
55. Synopsys security IP: https://www.synopsys.com/designware-ip/security-ip.html.

56. Synopsys Verification IP: http://www.synopsys.com/Tools/Verification/Functional Verification/VerificationIP/Pages/default.aspx.

Zhao Huang (黄釗) received the M.S. degree in Computer System Architecture from Xidian University, Xi'an, China. He is currently pursuing the Ph.D. degree in the School of Computer Science and Technology, Xidian University, Xi'an, China. His research interests include embedded system security, hardware Trojan detection and design for security of integrated circuits.

Quan Wang (王泉) received the B.S., M.S. and Ph.D. degrees in Computer Science from Xidian University in 1992, 1997 and 2008 respectively. He is now a Professor at the School of Computer Science and Technology and the Director of the Institute of Computer Peripheral Equipment. His research interests include wireless embedded system, 3-D printing, and wireless networks.