Received 15 September 2023, accepted 21 November 2023, date of publication 23 November 2023, date of current version 29 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3336565

RESEARCH ARTICLE

A Comprehensive Framework for Systemic Security Management in NoC-Based Many-Cores

RAFAEL FOLLMANN FACCENDA[1], GUSTAVO COMARÚ[1], (Graduate Student Member, IEEE), LUCIANO LORES CAIMI[2], AND FERNANDO GEHM MORAES[1], (Senior Member, IEEE)
[1]School of Technology, Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre 90619-900, Brazil
[2]Department of Computer Science, Federal University of Fronteira Sul (UFFS), Chapecó 89815-899, Brazil

Corresponding author: Fernando Gehm Moraes (fernando.moraes@pucrs.br)

This work was supported in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—CAPES under Grant 001, in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico—CNPq under Grant 309605/2020-2 and Grant 407829/2022-9, and in part by Fundação de Amparo à Pesquisa do Estado do RS—FAPERGS under Grant 21/2551-0002047-4.

ABSTRACT Many-core Systems-on-Chip (MCSoC) are increasingly used in various applications domains such as high-performance computing, embedded systems, and Internet of Things devices. As MCSoCs permeate various industries and applications, the potential consequences of security issues are becoming increasingly severe. Therefore, security is a fundamental design constraint, addressing vulnerabilities and protecting valuable data from threats. This requires the development of robust security mechanisms and countermeasures against potential threats. The reviewed works on security for MCSoCs addressed frameworks and mechanisms to treat different security threats. Despite these proposals, the integration of security mechanisms still needs to be improved, enabling a security manager to make decisions using monitoring data for mitigating threats more effectively. This integration is the primary goal of our work, aiming to create a comprehensive framework for security management. The framework adopts a Monitoring-Detection-Countermeasure loop. A distributed monitoring infrastructure detects suspicious behaviors, generating warnings to different system actors. These actors decide the warning severity, firing security countermeasures. Countermeasures may be local (e.g., discarding a packet) or taken at the system level (e.g., aborting a malicious application). The results use an MCSoC modeled at the RTL level, providing accuracy at the clock cycle (cc) level. Five different attack scenarios are evaluated, showing that the gap between attack detection and countermeasure takes less than one millisecond (15,000 cc at 100 MHz). The area overhead in the communication infrastructure corresponds to 48.8%. These results show that the framework can effectively manage the system's security while maintaining the performance of the applications.

INDEX TERMS Countermeasures, monitoring, NoC-based many-cores, security framework.

I. INTRODUCTION

High-performance computing architectures increasingly employ Many-Core Systems-on-Chip (MCSoC) that use Networks-on-Chip (NoC) as their communication infrastructure to meet the ever-growing demand for computational power. These systems are widely used in diverse domains, such as embedded systems, data centers, and high-performance computing. Prominent examples of MCSoCs include Esperanto [1], Epiphany-V [2], Kalray MPPA [3], Celerity [4], Kilo-Core [5]. As MCSoCs become more prevalent, ensuring their security has become a crucial design consideration.

As MCSoCs permeate various industries and applications, the potential consequences of security issues are becoming increasingly severe. Therefore, security is fundamental to system design, addressing vulnerabilities, and protecting valuable data from threats. This requires the development of robust security mechanisms and countermeasures against potential threats to prevent unauthorized access,

The associate editor coordinating the review of this manuscript and approving it for publication was Engang Tian.

tampering, or data leakage. Such proposals must also consider non-functional requirements such as power, performance, and area.

MCSoCs are susceptible to various attacks, such as hardware Trojans (HT), side channel attacks (SCA), denial of service (DoS), and spoofing [6]. Current security methods in MCSoCs include the implementation of firewalls [7], secure routing algorithms [8], encryption [9], authentication [10], anomaly detection [11], among others. These proposals focus on mitigating a single type of attack rather than seeking to create an integrated security framework.

This paper introduces a systemic security management approach designed for NoC-based MCSoCs. Our primary *objective* is to create a comprehensive framework that enables the integration of diverse security mechanisms, monitoring, analysis, and decision-making processes. We present the framework, alongside a set of attacks and recovery costs, to assist system designers and researchers in effectively improving the security of MCSoCs. By adopting our proposed framework, designers can achieve more resilient and secure architectures, ensuring safe and reliable operation across a variety of applications and industries.

A recurrent observation in the literature is the absence of systemic and integrated security mechanisms that simultaneously monitor, detect, and mitigate a broad spectrum of threats in real-time. The *novelty* of our work is to fulfill this gap by proposing a comprehensive framework for security management. The proposed framework monitors the system at several locations, allowing the detection of threats. Threat detection fires countermeasures and generates security warnings to a security manager. None of the reviewed works presents this monitoring-detection-countermeasure security loop.

This paper is organized as follows. Section II reviews security proposals for NoC-based systems. Section III overviews the architecture and threat models. Section IV details the security mechanisms added to the reference architecture. Section V presents the main contribution of this work, the security framework, which adopts a Monitoring-Detection-Countermeasure loop. Section VI evaluates the framework in different attack scenarios and evaluates the area overhead due to added hardware mechanisms. Section VII concludes this paper and points out directions for future work.

II. RELATED WORK

In a previous study, Fiorin et al. [12] emphasized the need of a security framework designed to gather data from monitors integrated into network interfaces (NI) or routers strategically positioned within critical areas of the NoC. The authors propose to monitor: (*i*) buffer occupancy, (*ii*) anomalous behavior of power manager; (*iii*) unauthorized access to secure memory locations; (*iv*) violation of execution of critical routines. These authors propose in [7] the adoption of firewalls integrated into the NI to manage memory accesses using a lookup table containing the access rights. The authors only evaluate the firewall area. In more recent work, Fiorin et al. [13] propose the insertion of a configurable Probe device inside the NI that can detect events and collect values about throughput, latency, resource utilization, and message characteristics. After detecting events, a message to the Probe Management Unit (PMU) reports the detected set of events that can trigger runtime management functions. The probe module was evaluated for area, energy, and traffic overhead.

Azad et al. [9] propose a mechanism that allows the configuration of security zones in MPSoCs. The authors assume that the NoC is untrustworthy and that an attacker can tamper with the on-chip communication. Secure zones are created at runtime by configuring firewalls through authenticated encryption. Differently from our security zone approach described in Section IV-A, the proposal is, in effect, a Network Interface design.

Thejaswini et al. [10] present an HT detection and mitigation approach using obfuscation and key-based authentication. The authors aim to prevent illegal transactions between routers, protecting the NoC against packet misrouting and information leakage. The evaluation uses the Gem5 simulator in a 4×4 mesh NoC-based SoC with 16 cores. Despite the reported high detection rate and effective mitigation against HT attacks, there are no identified concerns related to security management.

Meng et al. [14] propose a framework for systematically detecting security violations in SoC designs resulting from vulnerabilities in NoC communication. The threat model includes message misdirection, message mutation, delivery prevention, and network congestion. The proposed framework, SEVNOC, extracts a control-flow graph of the design that enables analysis of security properties through state exploration. The framework does not detect attacks at runtime. The authors' goal is to detect vulnerabilities in the RTL design using a symbolic approach.

Sharma et al. [15] analyze the security aspects of MPSoCs, discussing several defense mechanisms known in the literature, such as secure zones, firewalls, and key agreement, and then expand the discussion to the Cloud of Chips scope. Furthermore, the authors propose a software-defined network-on-chip (SDNoC) as an alternative that can reserve resources, avoiding congestion and harmful paths. The paper provides a broad view of MPSoC security and how effective are the defense mechanisms against DoS, Hardware Trojan, and Side-channel attacks. However, the proposal does not include runtime monitoring of threats, which results in a gap in detection and countermeasures once a threat is detected.

Ruaro et al. [16] also adopt SDN to establish a programmable path based on different policies, such as power, QoS, and security. In addition, the authors propose a secure path configuration based on key authentication that avoids DoS and flooding attacks since packets that fail the authentication are discarded. However, the authors point out that their approach is still vulnerable to HT attacks.

Kumar et al. [17] propose a methodology to protect NoCs against HTs. The authors propose a 3-tier approach that includes a Trojan cognizant routing algorithm (TCRA),

TABLE 1. Related work summary.

Author	Security Location	Defense Mechanism	Monitoring	Detection	Countermeasure
Fiorin et al. [7] (2008)	NI, to protect external shared memories	Data Protection Unit (DPU), for memory access control	Memory access rights	Correctness of access rights	Packet discarding; negative acknowledgment to the initiator
Fiorin et al. [13] (2013)	NI	–	Throughput, Latency, Resource Utilization	–	–
Azad et al. [9] (2019)	NI (CEASAR Core) with security mechanisms	Firewalls, security zones, authentication, lightweight cryptography	Firewall rights	Correctness of firewall rights	Packet discarding
Thejaswini et al. [10] (2022)	NoC	Bit-obfuscation of critical packet fields	Key-based path verification (XOR authentication mechanism)	Per hop key authentication	Packet discarding and rerouting computation
Meng et al. [14] (2023)	SoC	Security analysis at design-time of CFGs (control-flow graph)	–	–	–
Sharma et al. [15] (2021)	NoC (SND) for MPSoC and Cloud of Chips	SDNoC-based security, security-aware routing	–	–	–
Ruaro et al. [16] (2020)	NoC (SDN)	Secure SDN configuration; sub-network authentication	Key authentication for path configuration packets	Wrong authentication key	Packet discarding
Kumar et al. [17] (2021)	Router and NI	Trojan cognizant routing algorithm (TCRA), Trojan-resilient network interface.	–	Error detection code (ECD)	Routing algorithm and ECD
This Work	PEs, NoC and NI with IO devices	Opaque Secure Zones (**OSZ**), authentication, adaptative routing, secure NI for IO devices	Session protocol in OSZs, master-slave comm. protocol, packet authentication	Missing packets, unexpected data, fail key auth., intense access attempts	Reroute, key renewal; packet discarding, **warnining for a security manager**

a Trojan detection and diagnosis module, and a Trojan-resilient network interface. The detection and diagnosis module is responsible for identifying and locating HTs in the system, while the network interface provides a secure communication channel between the NoC and the external world. The Authors used a NoC simulator (NoCTweak) to test the TCRA under different scenarios, including single and multiple HTs. The results of the experiments show that the proposed approach effectively mitigates the impact of HTs on NoCs. The TCRA outperforms other methods regarding average throughput, packet delivery, and free link availability.

Sudusinghe et al. [18] propose a machine learning-based approach for detecting eavesdropping attacks in NoCs, which can be launched by inserting HTs. The approach involves training machine learning models on data sets that emulate different application executions and malicious router scenarios. The performance of the models is evaluated based on accuracy, F1 score, precision, and recall. The results show that the accuracy of the models increases with the percentage of information snooping due to an eavesdropping attack. Experiments use a 4×4 mesh NoC modeled using Garnet NoC with the Gem5 system simulator, testing the model using the Garnet synthetic traffic injector. The results show that the models can detect eavesdropping attacks with high accuracy, even when the percentage of snooping is as low as 25%. The authors conclude that the proposed approach can detect eavesdropping attacks in NoC architectures with high accuracy and low false positive rates. This work highlights the potential of machine learning to detect security threats in complex system-on-chip designs.

Charles et al. [6] survey NoC security attacks and countermeasures. The authors evaluate five types of security attacks and the corresponding countermeasures: eavesdropping, spoofing, denial-of-service (DoS), buffer overflow, and side channel (SCA). For each attack, the survey presents proposals to mitigate or avoid them, presenting the overhead and effectiveness of the proposal. One of the future directions pointed out by the authors is the integration of security mechanisms. They argue that a proposal that mitigates one particular type of threat does not effectively secure the SoC.

Table 1 provides a qualitative comparison of the works discussed in this section. The column **Security Location** denotes the system components endowed with security mechanisms. Most proposals focus primarily on protecting the communication infrastructure, encompassing NoC and NI components. In addition to employing these components, our work uses PEs to monitor and configure NoC and NI peripherals at runtime.

The column **Defense Mechanisms** presents the security mechanisms adopted by the authors. Our approach is distinct due to adopting opaque secure zones, which reserve communication and computation resources for a specific application, together with secure mechanisms for communicating with peripherals.

The **Monitoring** column lists the policies used for system monitoring. Here appears a gap in current literature, as fundamental mechanisms are firewalls or authentication, lacking systemic mechanisms. The **Detection** column is a consequence of the monitoring methods. As discussed later, our work monitors multiple events simultaneously,

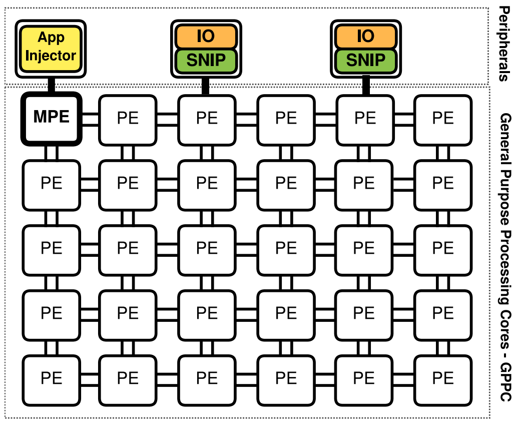


FIGURE 1. **MCSoC model. PE: Processing Element, MPE: Manager PE; Peripheral: contains a SNIP (Secure Network Interface with Peripheral) and an IO device. *App. injector*: transmits applications to execute into the system.**

enabling the detection of a broader range of security events.

Lastly, the **Countermeasure** column shows that the most common countermeasure is packet discarding, followed by rerouting. Besides incorporating these countermeasures, our work notifies a security agent about suspicious events. This enables this agent to know the system's status and implement suitable measures to ensure its secure operation.

The reviewed proposals on security for NoC-based systems addressed frameworks and mechanisms to treat different security threats. Despite these advances, the integration of security mechanisms still needs to be improved, enabling a security manager to make decisions using monitoring data to mitigate threats more effectively. This integration is the primary goal of our work, aiming to create a comprehensive framework for security management.

III. ARCHITECTURE AND THREAT MODELS

A. ARCHITECTURE MODEL

Figure 1 overviews the architecture model, based on [19]. It contains two regions. The first is the "General Purpose Processing Cores" (GPPC), with identical general-purpose processing elements (PEs). One of those PEs is the **MPE**, which controls the system, also being the **security manager** of our proposal. The second region contains peripherals and the application injector (App_{inj}), responsible for deploying applications into the system.

All PEs (including the MPE) have the same hardware architecture, with two routers, private memory, a processor, and a DMNI (Direct Memory Network Interface) [20] module. The system contains two 2D-mesh NoCs, a packet-switching data NoC and a control NoC. The data NoC uses duplicated physical channels, enabling the adoption of XY and adaptive source routing. The control NoC uses broadcast as the default transmission mode [21].

Peripherals are specialized nodes that provide IO interface and hardware acceleration for tasks running on the GPPC. Examples of peripherals include hardware accelerators, communication modules (e.g., Ethernet interface), and shared memories. The App_{inj} is responsible for transmitting applications to be executed in the GPPC through the data NoC. Peripherals and App_{inj} are connected to the boundaries of the GPPC to obtain regular floorplanning.

B. THREAT MODEL

The reference architecture has two trustworthy entities: (*i*) control NoC and (*ii*) MPE. Message authentication code (MAC) protects the operating system and tasks load against boot attacks. Only the operating system running on PEs may access the control NoC, preventing attacks on it from malicious tasks.

This work considers in the threat model attacks that involve denial-of-service, spoofing, and eavesdropping executed from different sources (Hardware Trojans, malicious applications, malicious IO devices), arising in the following vulnerabilities:

1) Resource sharing between applications: Execution of secure and non-secure applications in the same region, using the same structures for computation (PEs) and communication (NoC routers). This characteristic presents a major system vulnerability since malicious applications could access sensitive application data or generate an intensive communication flow that disrupts the surrounding communication structures.
2) Applications deployment: Applications must be deployed securely to ensure they are not tampered or compromised during the object code transmission through data NoC.
3) Hardware Trojans (HTs): Malicious hardware can be inserted into the NoC or PEs [22] to perform unauthorized operations, steal sensitive data, or disturb traffic by blocking ports.
4) Access to IO devices: Applications that communicate with IO devices demand additional security measures to prevent unauthorized access or exploration of protocol violations.
5) Malicious IO devices: IO devices that communicate with the external environment may try to attack applications by injecting malicious code or data. In addition, IOs can be a gateway for information leakage.

It is crucial to consider these vulnerabilities and implement security measures to mitigate the risks to ensure the safety and reliability of the system.

IV. SECURITY MECHANISMS AND APPLICATION EXECUTION

This section initially details the security mechanisms (hardware and software) added to the reference architecture (IV-A). In the sequel (IV-B), we present the actions executed at system startup, responsible for carrying out security-related configurations. Next (IV-C), we detail the deployment of applications with security requirements into the system, and

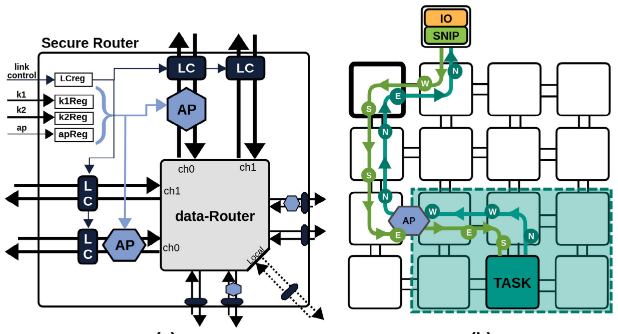


FIGURE 2. (a) Secure Router. LCs: activated by *LCreg*. APs: activated by *apReg*. {*k*1*Reg*,*k*2*Reg*} authentication key registers. Each AP has an interface with the control NoC to notify suspicious packets. (b) Example of SR paths Task↔SNIP through the AP.

we conclude this section (IV-D) with a summary of the adopted security mechanisms. Once an application runs, the framework (Section V) monitors for suspicious events and takes countermeasures.

A. SECURITY MECHANISMS

The primary security mechanism is the Opaque Secure Zone (**OSZ**) [23]. OSZs are system regions with PEs reserved for executing an application with security requirements – App_{sec}. The purpose of OSZs is to protect communication and computation. The boundaries of the OSZ are closed by link control modules (LCs) to ensure that the communication within an OSZ remains secure. The LC modules block any communication trying to cross the OSZ boundaries, and the App_{sec} traffic remains inside the OSZ. Any traffic attempting to cross an OSZ is automatically deviated, thus preserving the App_{sec} security. OSZs protect the App_{sec} by ensuring that it does not share processors and routers with other applications, preventing interference from other applications that could disturb the processing and communication within the OSZ, such as unauthorized access or data tampering.

The mechanism that enables the OSZ method is the support for adaptive source routing (**SR**). The SR is used in: (*i*) paths circumventing OSZs to avoid losing packets; (*ii*) paths between OSZs and peripherals; (*iii*) paths inside the OSZ when detecting a threat (discussed in more detail later). The SR is a security mechanism once it obfuscates the source and target addresses. It means that the packet header contains the turns the packet takes at each hop, rather than including the target addresses in the header.

To enable the communication of an App_{sec} with a peripheral, it is necessary to open a link in the OSZ without compromising the OSZ security mechanism. To enable this secure communication, two hardware modules are used: Access Point (**AP**) and the Secure Network Interface with Peripherals (**SNIP**).

The AP is a hardware module inserted on all router ports, except the local one, with its operation mode defined by memory-mapped registers. Figure 2(a) shows the “secure

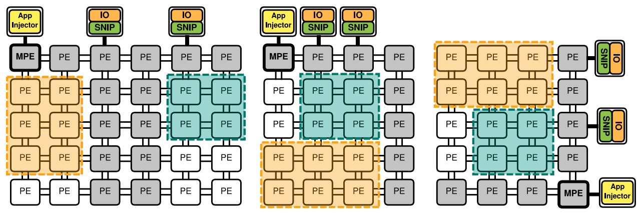


FIGURE 3. Three examples of gray (gray PEs) and safe areas (white PEs), with OSZs created in the safe areas. The MPE is part of the gray area despite not executing user tasks.

router”, with nine LCs (link control modules), four APs, and the data router. All links have an LC module, including the local port, that blocks the traffic when activated. The AP verifies the authenticity of the packets using a lightweight authentication protocol [24], detects suspicious traffic, and notifies such events to the MPE through the control NoC.

The SNIP (Secure Network Interface with Peripherals), placed between the NoC and a peripheral (Figure 1), executes the functions related to a network interface and is responsible for **authenticating** the communication between App_{secs} and IO devices. The paths between APs and SNIPs are defined by SR. Figure 2(b) presents an example of an App_{sec} task communicating with a peripheral.

B. SYSTEM INITIALIZATION ACTIONS

During the boot process, the MPE carries out a series of security-related actions. Initially, the MPE blocks communication with all peripherals by activating the LC modules positioned between the NoC and the SNIPs. This measure is crucial for preventing peripherals from attempting to inject malicious code into the system.

Subsequently, the MPE creates the **safe** and **gray** areas, with examples presented in Figure 3. Gray areas serve two purposes: (*i*) execute applications without security requirements; (*ii*) ensure that a path always exists between a border of the safe regions and peripherals located at the NoC borders. The safe area is reserved for allocating OSZs. The shape of these areas is a function of the peripheral’s location and the amount of PEs required to run App_{secs}.

Moreover, the MPE generates random initialization keys for each PE and peripheral. Since no other applications or traffic exist in the system at this step, these keys can be transmitted without encryption. This approach eliminates the need for complex key distribution mechanisms and guarantees the confidentiality and integrity of these keys, as they are not accessible to applications or IO devices.

Lastly, the MPE grants access to the *AppInj*, a trusted entity responsible for deploying applications into the system.

C. DEPLOYMENT OF APPLICATIONS WITH SECURITY REQUIREMENTS

The MPE receives requests from the App_{inj} to execute applications into the system. If an application has security requirements, the MPE runs the following algorithms:

1) Mapping: The MPE defines the shape and location of the OSZ, ensuring that at least one side of the OSZ is adjacent to a gray area.
2) Secure deployment: The MPE controls the transmission of the object code to the selected PEs. Each PE verifies the object code integrity using a MAC.
3) Closing the OSZ: The MPE activates the LC modules to close the OSZ, and elects a PE to manage the AP, the PE_{AP}.
4) Path Computation: The computation of the paths between each peripheral that the App_{sec} communicates with, called IO_{path}.
5) Transmission of security data: The MPE sends $\{IO_{path}, App_{sec}\ id, key\ generation\ parameters\}$ to peripherals through the data NoC using the initialization keys for authentication. The PEs receive these parameters through the control NoC.
6) Start the App_{sec} execution: The MPE sends a message to the App_{sec} to start its execution.

At the end of the execution of a given App_{sec}, the MPE and PEs clear the memory and management structures allocated for the application's tasks to prevent information leakage, and after release LCs and AP.

D. SUMMARY OF THE DEFENSE MECHANISMS

Figure 4 presents in its left part the list of the defense mechanisms adopted in the current work:

D1 Opaque Secure Zones (Section IV-A);
D2 Authentication keys, used in the AP ↔ SNIP communication;
D3 MAC, protects the binary codes;
D4 Source routing, obfuscate the source and target address;
D5 Key renewal, ensures the periodic change in the authentication keys;
D6 Safe and gray areas (Figure 3);
D7 SNIP, a NI that authenticates the communication.

V. FRAMEWORK FOR SYSTEMIC SECURITY MANAGEMENT

The proposed framework adopts an actuation loop based on Monitoring-Detection-Countermeasure, as illustrated in Figure 4, which protects the system when executing applications with security requirements. This framework integrates into the many-core defense mechanisms, summarized in Section IV-D, with distributed monitoring methods (Section V-A) that enable the detection of threats and activation of countermeasures (detailed in Section V-B).

Monitoring mechanisms (`M1-5`) observe system resources and generate warnings (`W1-6`) in case of suspicious behavior. Based on the severity of the alerts, the system triggers countermeasures, which can be local (`C1-5`) or system-level actions. System-level countermeasures are triggered upon detecting a more complex attack (`A1-3`).

A. MONITORING AND DETECTION OF SUSPICIOUS BEHAVIOR

To protect internal communication within the OSZ, we adopt a protocol called "session protocol" (`M1`) [25]. This protocol includes sending control messages via the control NoC alongside the data messages to monitor the arrival of packets inside the OSZ. Packets are only accepted upon receiving both control and data packets that confirm the source and target of this packet. The "session protocol" can raise two warnings: `W1` (Missing Packet), when only the control message arrives or the data packet is delayed beyond a certain time threshold; `W2` (Unexpected Data) when a data packet arrives without the control message, making it impossible to confirm the source of the message.

The communication API (Application Programming Interface) with peripherals adopts a master-slave protocol, which is monitored (`M2`). Any IO transaction must always start from the App_{sec} (master), and the IO device must answer this request (slave). PEs monitor the packets' arrivals. Whenever a packet arrives without being requested, the PE raises `W2` (Unexpected Data), or if an answer packet from an IO takes too long to arrive, the PE sends an `W1` (Missing Packet).

The Access Point (AP) (`M3`) monitors all packets trying to enter the OSZ. The AP can raise four warnings:

- `W2` – Unexpected Data. When a packet tries to enter the OSZ without being requested. The AP has two counters, CT_{in} and CT_{out}, which count the number of packets entering and leaving the OSZ. Due to the master-slave communication protocol, the number of received packets cannot exceed the transmitted packets, i.e., $CT_{in} < CT_{out}$. Whenever this condition becomes false, the AP generates the warning.
- `W3` – Wrong Packet Type. The SNIPs add an identifier in the packets signalizing that it is generated by a peripheral. This warning is raised if the packet does not have this identifier. This monitoring avoids packets generated by applications running on PEs to try to enter the OSZs.
- `W4` – Wrong Authentication Key. The AP executes a lightweight authentication protocol that verifies the packet's authenticity. The warning is raised when the authentication fails, signalizing a forged packet.
- `W6` – Intense Access Attempts. This warning signalize a potential DoS attack. A counter in the AP monitors the number of received packets within an interval defined according to the application profile.

Thus, a packet only enters the OSZ, passing through the AP, satisfying three conditions: (*i*) $CT_{in} < CT_{out}$; (*ii*) successful authentication; (*iii*) packet from a peripheral.

The SNIP also monitors packets (`M4`). The SNIP authenticates packets, sending warning `W4` if the authentication fails. In addition, the SNIP can raise `W6` if the number of packets received within a time window is larger than a given threshold.

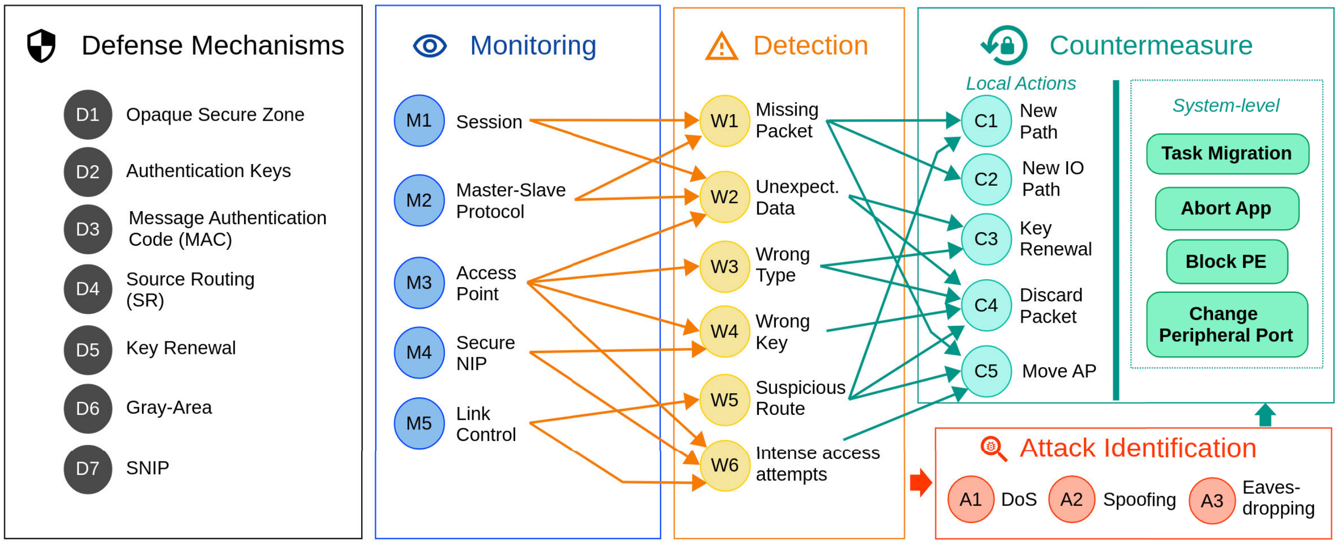


FIGURE 4. **Security management framework overview.**

The last monitoring element is the Link Control (LC) (M5). Packets should not arrive at enabled LCs due to the routing method, which circumvents the OSZs. Thus, LCs generate a W5 (Suspicious Route) warning for any packets arriving at an activated LC, especially if the packet is trying to exit the OSZ (this only may occur if an HT infected the secure application). LCs also have a counter to generate a warning signalizing a possible DoS attack (W6).

B. COUNTERMEASURE

Countermeasures are actions that reinforce system security upon the detection of suspicious behavior. Such actions are divided into two groups: local and system-level, which include immediate actions executed upon receiving warnings and actions taken based on broader systemic information, respectively.

The five local countermeasures are triggered by receiving warnings, as depicted on Figure 4. Depending on the warning severity, a single warning activates the countermeasure, or it is necessary to receive a set of notifications to trigger it.

The C1 (New Path) action triggers the computation of a new routing path for a message that did not reach the final target detected by the Session protocol (W1 from M1), or detected by the master-slave Protocol (M2) in the case of an IO communication.

New IO Path (C2) is a countermeasure triggered by W1 (Missing Packet) emitted by any PE that initiates an IO communication but does not receive the answer within a given time window (M2 master-slave protocol). This process is also a tool that can be requested by other countermeasures that affect the IO paths, such as C5 (Move AP).

Key renewal can be periodic or reactive. The periodic key renewal is part of the authentication method, aiming to enhance its security. Although the keys are not transmitted in plaintext, unauthorized access to the flits with the keys could enable a brute-force attack. The reactive key renewal (C3) is a countermeasure to refresh the keys, triggered by W2 or W3. For example, a packet with correct keys arrives at an AP but without a request (W2), or wrong type (W3). Even though the AP blocked the packet, a key renewal must occur since the authentication keys were correctly forged.

Packet Discarding (C4) is the fourth local countermeasure. This is the most frequent countermeasure action due to the OSZ method. The AP of the OSZ discards packets that fail authentication at any layer: whether due to the absence of request (W2), wrong type (W3), or incorrect key (W4). Additionally, activated LCs discard packets that attempt to cross it (W5). Moreover, the SNIP discards packets arriving with incorrect keys, and PEs may discard packets considered suspicious.

Move AP (C5) countermeasure is triggered when the MPE receives a W6 warning from an AP, meaning that heavy traffic on the AP is affecting the IO communication of an App_{sec}. W1 also may trigger the Move AP to change the route of a packet that could not reach the peripheral during an IO communication. The MPE then elects new AP location and triggers C3 to refresh the keys and C2 to recalculate the IO path since the AP coordinate changed. The Suspicious Route warning (W5) collected throughout the application execution time can be used as information to avoid mapping the AP at ports that could have been under attack.

In addition to local countermeasures, the MPE can identify threats and suspicious behaviors because it receives warnings and has a global view of the platform. Thus, it is possible to execute decision-making heuristics and perform system-level countermeasures, such as:

- **Task Migration**: Secure applications under attack, can be migrated to less susceptible regions of the MCSoC.
- **Abort Application**: if the MPE identifies that a given PE is the source of an attack, it identifies the tasks

running in the PE, and thus the malicious application(s), sending a message to abort the potential malicious application(s).

- **Block PE**: if the previous countermeasure fails, the identified PE may contain, e.g., an HT, generating malicious traffic. Thus, the MPE sends a control message to activate the LC of the local port, isolating it from the rest of the system.
- **Change peripheral Port**: SNIPs may have a secondary channel connected to another Router. When an attacker targets the SNIP, the primary channel can be swapped to the secondary one.

VI. SECURITY ANALYSIS AND COSTS

This section explores attacks identified in the Threat Model, including Denial of Service (DoS), Spoofing, and Eavesdropping. It outlines the system's responses in such events (refer to Figure 4) and discusses the associated costs of implementing countermeasures.

The many-core system is modeled at the RTL (Register-Transfer Level) level using VHDL and SystemC hardware description languages, meaning that the many-core description is synthesizable for FPGAs or ASICs. We use digital circuit simulators, such as Modelsim [26] or Incisive [27], to evaluate the experiments. Thus, the accuracy achieved in the experiments is at the clock cycle level, reflecting the actual system behavior. Applications run in the digital simulator for a few dozen milliseconds due to the complexity of the low-level simulation. On average, the simulation time for a 5×5 system (i9@3.10GHz with 32 GB RAM) is 131s for each ms. Thus, a 1-second simulation would require 36 hours. We do not use network simulators such as OPNET [28] for the experiments, as we are not considering generic networks but traffic within an integrated circuit. In the experiments, no external tool is used other than the digital circuit simulator.

The simulation includes the execution of the software on each processor modeled at the RTL level. Each processor runs an in-house microkernel with multitasking and message-passing support. Communicating task graphs (CTGs) model applications as a set of communicating tasks. We use the C language to describe applications.

We adopt two mechanisms to execute attacks. A peripheral named "packet injector" is directly connected to the system without using the SNIP. The goal of using this peripheral is to inject controlled traffic into the system to emulate attacks. The second mechanism is an HT circuit (based on [29]) connected to routers. The HT trigger may be a malicious application or a given condition (e.g., time-triggered HT) that can duplicate, misroute, or block flows.

Simulations focus on five specific attack scenarios:

A. *DoS Flooding*: In this attack scenario, the "packet injector" transmits packets at a high throughput rate of 0.85 flits per clock cycle to an OSZ or SNIP. The objective is to saturate the NoC links and buffers to render the NoC unavailable for legitimate operations.

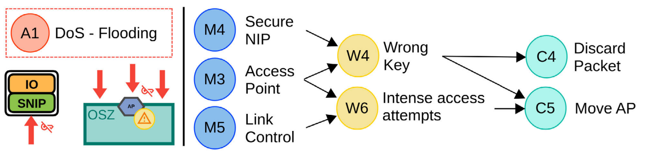


FIGURE 5. DoS flooding attack scenario.

B. *Spoofing*: This attack is analogous to the DoS flooding attack but operates at a lower injection rate of 0.05 flits per clock cycle. Furthermore, the injected packets contain the correct keys to bypass the AP or SNIP, simulating a situation where the keys have been compromised.

C. *Eavesdropping*: a time-triggered HT infects a router inside the OSZ. The attack initiates after 5 milliseconds of simulation time has elapsed, duplicating packets traversing the infected router.

D. *Internal OSZ DoS blocking*: a time-triggered HT infects a router inside the OSZ. The HT blocks all router links for 1 millisecond at regular intervals of every 5 milliseconds throughout the simulation time.

E. *External OSZ DoS blocking*: similar to the previous attack, but in a router belonging to the GA.

A. DoS - FLOODING

Figure 5 depicts the first attack scenario, in which malicious flows with forged packets target the SNIP or the AP of a given OSZ.

Packets that arrive in the SNIP without the correct keys are automatically discarded. Malicious packets arriving at the LCs and the AP of a given OSZ are also discarded. However, the W6 warning signal is triggered if access attempts become too frequent. This alert informs the AP selection heuristic running in the MPE to avoid mapping an AP to this port due to the attack attempt and also initiates a Move AP countermeasure (C5).

If the malicious flow bypasses the hardware barrier (AP) and reaches a PE, the master-slave protocol may identify unexpected data and subsequently discard the packet, as depicted by countermeasure C4.

The countermeasure C4 in hardware instantly discards the packet. Conversely, in software (discard action performed by the PE), the process takes 378 clock cycles (*cc*), measured from the point of packet arrival interruption to the complete clearing of the DMNI slots where the packet was initially stored.

B. SPOOFING

Figure 6 illustrates a Spoofing attack where the malicious flow has the correct authentication keys, enabling it to pass through the AP or even gain access to the SNIP.

When this malicious flow reaches the SNIP, the master-slave protocol generates an answer packet to the address stored in the SNIP Application Table. Note that

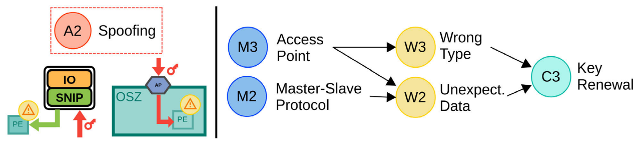


FIGURE 6. Spoofing attack scenario.

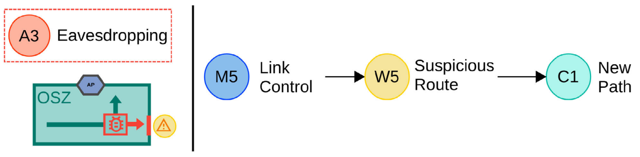
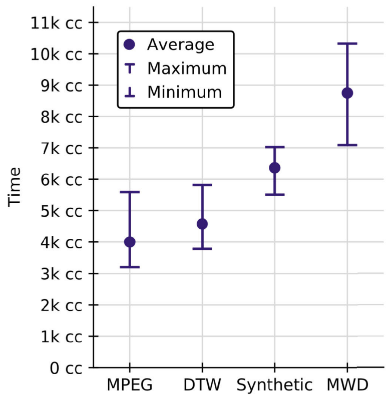


FIGURE 8. Eavesdropping attack scenario.



(a)

App	# Tasks	Comm. Vol.
MPEG	5	Low
DTW	6	Low
Synthetic	6	High
MWD	12	High

(b)

FIGURE 7. Key renewal overhead for different applications.

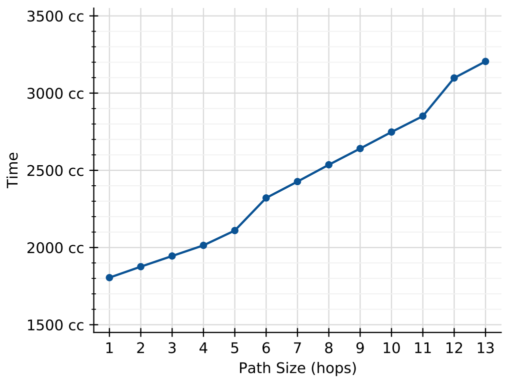


FIGURE 9. Searchpath overhead for different path sizes.

the answer does not go to the attacker but to the registered application. This answer packet reaches a PE in the OSZ, and the PE triggers an unexpected packet warning (W2), given that the application is not expecting answers from IO operations.

Another scenario occurs when the malicious packet reaches the AP with the correct keys but either at the incorrect moment or with the wrong type. As a countermeasure in both cases, a Key Renewal C3 is executed to refresh the keys and prevent further unauthorized access.

Figure 7 presents the Key Renewal cost, in clock cycles (*cc*), for four applications. The key renewal costs have fixed and variable components. The fixed part refers to the time to process the key renewal request and decide on the new key renewal parameters. The variable component of the cost is related to synchronization costs, which includes the time taken to receive the key renewal acknowledgment from all tasks of the application plus the time needed for the tasks to finish their pending IO transactions before changing the keys.

For the two applications with low IO communication volume (MPEG and DTW), the average time for key renewal is 4,000 and 4,500 clock cycles, respectively. Increasing the IO communication volume (Synthetic) and the number of tasks (MWD) directly impact the Key Renewal execution, reaching average values of 6,370 and 8,753 clock cycles, respectively. The fixed component of the cost corresponds to 600 cc (average values).

C. EAVESDROPPING

Figure 8 presents an example of an Eavesdropping attack, where a malicious hardware in the NoC duplicates a packet to send it outside the OSZ. However, this packet hits an LC or the AP at the OSZ border, triggering an alert W5 to indicate that the packet has taken a suspicious route. To mitigate this threat, a new path is calculated within the OSZ that avoids passing through the identified suspicious router.

The New Path countermeasure uses the control NoC to build a new path avoiding the suspicious router. To achieve this, a *searchpath* message is broadcasted to all PEs in the OSZ, and a *backtrack* message is subsequently received with the correct sequence of hops. Figure 9 presents the time to build new paths ranging from 1 to 13 hops.

The overhead curve starts at a path size equal to 1, consuming 1805 clock cycles. It then increments by 70 clock cycles per hop until the hop size reaches 6. From 7 hops, the overhead corresponds to 100 clock cycles per hop until the hop size equals 12. These periodic increases in the curve at every multiple of six are due to the Source Routing (SR). SR uses flits within the packet to store the direction for forwarding the packet. Each flit may store the directions for six hops. Consequently, the cost increases when a new word is required in the SR path.

D. DoS-BLOCKING (OSZ)

Figure 10 presents an example of DoS blocking inside an OSZ. In this case, a malicious entity (e.g., HT infecting an NoC router) blocks any communication trying to pass through it. The Session protocol detects this behavior due to the control message emitted alongside the data message. Due to the broadcast transmission, the control message arrives at the PE and not the data packet. This behavior raises the Missing Packet warning (W1).

To avoid this malicious entity, the system triggers a New Path C1 countermeasure to build a new route circumventing

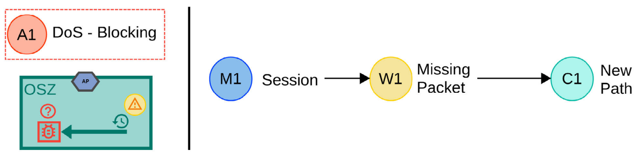


FIGURE 10. DoS blocking scenario inside an OSZ.

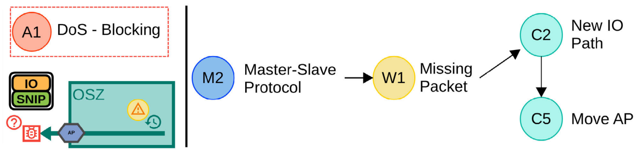


FIGURE 11. DoS blocking scenario in the gray area.

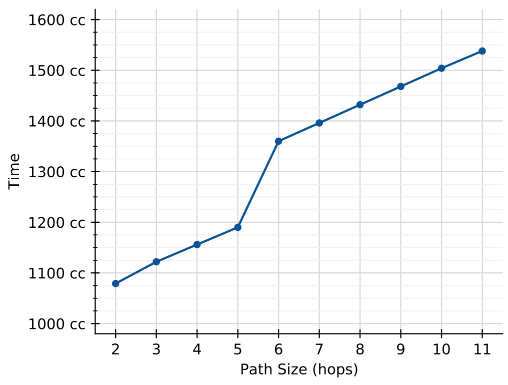


FIGURE 12. Overhead for calculating a new path for peripherals.

the suspicious router. The costs related to this countermeasure are the same presented on Section VI-C.

E. DoS-BLOCKING IN THE GRAY AREA

Figure 11 shows a DoS blocking attack occurring during an IO transaction in the gray area. In this case, the master-slave protocol detects the attack since the secure application never receives the answer from the packet sent to the IO device.

The system initiates a countermeasure if an IO packet fails to receive an answer within a predetermined time threshold. The first countermeasure is to create a New IO Path (C2), corresponding to a new path to the IO device, traversing the gray area. Figure 12 shows the time to build a new IO path for different hop counts. This path construction is faster than the New Path (C1) because the kernel entirely calculates it, while C1 also uses the control-NoC to find a new path inside the SZ.

The system can also trigger a Move AP (C5) countermeasure if the IO communication is still blocked even with the execution of C2. For the Move AP process, the MPE decides a new position for the AP and notifies the PE that it must now configure this new AP. This process also triggers a Key Renewal and a New IO Path for every SNIP that this application communicates with because the route using the old AP is now closed.

TABLE 2. Average cost of Move AP countermeasure for different applications.

App	Overhead (cc)		
	Change AP	Path Config.	Total
MPEG	5,754	3,132	8,887
DTW	5,634	3,118	8,753
Synthetic	7,467	3,182	10,047
MWD	11,457	3,433	14,890

Table 2 shows the costs of the Move AP countermeasure triggered on four applications: MPEG, DTW, Synthetic and MWD. The costs of Move AP can be analyzed as the cost to change the AP location and synchronize the new keys, plus the cost of the New IO Paths.

The MPEG and DTW applications show similar overheads when changing the AP location as they have a small number of tasks (5 and 6, respectively). Conversely, the Synthetic application requires additional 1,200 clock cycles due to its higher communication volume, while the MWD application requires around 5,800 additional clock cycles due to its larger task count (12). Both characteristics, IO communication volume and task number, affect the synchronization of the new AP location and key renewal. As all applications interact with two SNIPs, the costs of calculating a new IO path (column "Path Configuration") remain similar, around 3,100 cc. This value refers to the cost to search for the affected SNIPs, construct the new IO paths (one for each SNIP - Figure 12), build the path and send the new path packets.

This countermeasure, which involves relocating the AP, is one of the most complex in terms of protocols and processing time. The worst-case scenario observed corresponds to 14,890 clock cycles, equivalent to 0.1489 ms@100 MHz. Consequently, the proposed countermeasures add a negligible impact on applications' performance due to their low computational overhead.

F. AREA EVALUATION

Table 3 presents the results of the logic synthesis to assess the hardware impact of the security elements added to the system. The synthesis includes the Secure Router, Control Router, and SNIP. Each processing element (PE) has two routers (data and control routers), a processor, local memory, and a network interface. The SNIP is configured with an application table with four rows, and input/output buffers for 16 slots for 16-bit flits. The Data Router has two disjoint channels, with 16-bit flits and 8-flit input buffers. The Control Router has a CAM size with 8 slots.

Table 3 shows that the SNIP has a low area overhead, representing 55.6% of the data router area. Comparing the baseline router area (17,973 μm^2) to the new communication infrastructure (control and secure router – 26,766 μm^2), the area overhead corresponds to 48.8% in the communication infrastructure.

TABLE 3. Synthesis results - 28nm FDSOI - CADENCE GENUS 21.12-s068.

Synthesis Results	SNIP	Control Router	Secure Router – Figure 2(a)			
			Data Router	1 AP (Avg)	All LCs + Regs	Total
Cell Count	3,666	3,203	7,049	169	50	7,776
Cell Area (μm^2)	8,665	4,656	13,768	291	86	15,019
Net Area (μm^2)	2,470	2,090	4,204	81	472	5,000
Total Area (μm^2)	**11,135**	**6,747**	**17,97**	**372**	**558**	**20,019**
Timing Slack @500 MHz (ps)	373	29	-	-	-	136
Total Est. Power (mW)	**3.95**	**2.69**	**-**	**-**	**-**	**7.46**

TABLE 4. Summary of the countermeasure costs, in clock cycles.

Countermeasure	Worst-case (cc)
New OSZ path	3,205
New IO path	1,538
Key Renewal	10,912
Packet discarding	378
Move AP location	14,890

The communication infrastructure represents no more than 20% of the PE area (in [4], the NoC represents 7.7% of the PE area). Thus, it is expected that an increase of 50% in the communication infrastructure corresponds to an increase in the PE area between 5% an 10%.

VII. CONCLUSION AND FUTURE WORK

This paper presented a framework that combines several defense mechanisms to detect and protect against security threats such as Denial of Service, Spoofing, and Eavesdropping. The framework can help designers and developers to identify and mitigate security risks and threats, and to ensure the security and reliability of the system. The framework can also contribute to developing secure and trustworthy many-core systems for various applications.

Five attack scenarios were used to measure the costs of the countermeasures that the system manager can apply to reinforce the system security. Table 4 illustrates the costs of each countermeasure, considering the observed worst-case. The cost to create the OSZ (20.5K clock cycles in [30]) does not impact the application performance since it is executed before its execution starts. The table shows that the highest costs are related to renewing the keys and moving the AP location. The wost-case is 0.15 ms at 100 MHz, demonstrating that the proposed countermeasures are lightweight, adding security to the applications.

The hardware to support the framework added an overhead of 48,8% in the communication infrastructure (NoC). This is an acceptable cost considering that the communication infrastructure is a part of the processing element (PE) and may lead to an actual area overhead at the PE level between 5% and 10%.

This work paves the way for developing system-level heuristics to increase security in many-cores. A fusion of monitoring data by the security manager will allow the detection of more complex attacks and the execution of a more extensive set of countermeasures to those proposed in this paper.

REFERENCES

[1] O. Peckham. (2020). *Esperanto Unveils ML Chip with Nearly 1,100 RISC-V Cores*. [Online]. Available: https://www.hpcwire.com/2020/12/08/esperanto-unveils-ml-chip-with-nearly-1100-risc-v-cores

[2] A. Olofsson, "Epiphany-V: A 1024 processor 64-bit RISC system-on-chip," 2016, *arXiv:1610.01832*.

[3] B. D. de Dinechin, D. van Amstel, M. Poulhiès, and G. Lager, "Time-critical computing on a single-chip massively parallel processor," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2014, pp. 1–6, doi: 10.7873/DATE.2014.110.

[4] A. Rovinski et al., "Evaluating celerity: A 16-nm 695 Giga-RISC-V instructions/s manycore processor with synthesizable PLL," *IEEE Solid-State Circuits Lett.*, vol. 2, no. 12, pp. 289–292, Dec. 2019, doi: 10.1109/LSSC.2019.2953847.

[5] B. Bohnenstiehl, A. Stillmaker, J. J. Pimentel, T. Andreas, B. Liu, A. T. Tran, E. Adeagbo, and B. M. Baas, "KiloCore: A 32-nm 1000-processor computational array," *IEEE J. Solid-State Circuits*, vol. 52, no. 4, pp. 891–902, Apr. 2017, doi: 10.1109/JSSC.2016.2638459.

[6] S. Charles and P. Mishra, "A survey of network-on-chip security attacks and countermeasures," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–36, Jun. 2022, doi: 10.1145/3450964.

[7] L. Fiorin, S. Lukovic, G. Palermo, and P. di Milano, "Implementation of a reconfigurable data protection module for NoC-based MPSoCs," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, Apr. 2008, pp. 1–8, doi: 10.1109/ipdps.2008.4536514.

[8] A. Sarihi, A. Patooghy, M. Hasanzadeh, M. Abdelrehim, and A. A. Badawy, "Securing network-on-chips via novel anonymous routing," in *Proc. 15th IEEE/ACM Int. Symp. Networks-on-Chip (NOCS)*, Oct. 2021, pp. 29–34.

[9] S. Payandeh Azad, M. Tempelmeier, G. Jervan, and J. Sepúlveda, "CAESAR-MPSoC: Dynamic and efficient MPSoC security zones," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2019, pp. 477–482, doi: 10.1109/ISVLSI.2019.00092.

[10] P. Thejaswini, G. Vivekananda, H. Anu, R. Priya, P. Krishna, and M. Nischay, "Hardware Trojan detection and mitigation in NoC using key authentication and obfuscation techniques," *EMITTER Int. J. Eng. Technol.*, vol. 10, no. 2, pp. 370–388, Dec. 2022, doi: 10.24003/emitter.v10i2.716.

[11] K. Wang, H. Zheng, Y. Li, J. Li, and A. Louri, "AGAPE: Anomaly detection with generative adversarial network for improved performance, energy, and security in manycore systems," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2022, pp. 849–854, doi: 10.23919/DATE54114.2022.9774693.

[12] L. Fiorin, C. Silvano, and M. Sami, "Security aspects in networks-on-chips: Overview and proposals for secure implementations," in *Proc. 10th Euromicro Conf. Digit. Syst. Design Archit., Methods Tools (DSD)*, Aug. 2007, pp. 539–542, doi: 10.1109/dsd.2007.4341520.

[13] L. Fiorin, G. Palermo, and C. Silvano, "A configurable monitoring infrastructure for NoC-based architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 11, pp. 2438–2442, Nov. 2014, doi: 10.1109/TVLSI.2013.2290102.

[14] X. Meng, K. Raj, S. Ray, and K. Basu, "SeVNoC: Security validation of system-on-chip designs with NoC fabrics," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 2, pp. 672–682, Feb. 2023, doi: 10.1109/TCAD.2022.3179307.

[15] G. Sharma, G. Bousdras, S. Ellinidou, O. Markowitch, J.-M. Dricot, and D. Milojevic, "Exploring the security landscape: NoC-based MPSoC to cloud-of-chips," *Microprocess. Microsyst.*, vol. 84, Jul. 2021, Art. no. 103963, doi: 10.1016/j.micpro.2021.103963.

[16] M. Ruaro, L. L. Caimi, and F. G. Moraes, "A systemic and secure SDN framework for NoC-based many-cores," *IEEE Access*, vol. 8, pp. 105997–106008, 2020, doi: 10.1109/ACCESS.2020.3000457.

[17] J. Y. V. M. Kumar, A. K. Swain, K. Mahapatra, and S. P. Mohanty, "Fortified-NoC: A robust approach for Trojan-resilient network-on-chips to fortify multicore-based consumer electronics," *IEEE Trans. Consum. Electron.*, vol. 68, no. 1, pp. 57–68, Feb. 2022, doi: 10.1109/TCE.2021.3129155.

[18] C. Sudusinghe, S. Charles, S. Ahangama, and P. Mishra, "Eavesdropping attack detection using machine learning in network-on-chip architectures," *IEEE Des. Test. Comput.*, vol. 39, no. 6, pp. 28–38, Dec. 2022, doi: 10.1109/MDAT.2022.3202995.

[19] M. Ruaro, L. L. Caimi, V. Fochi, and F. G. Moraes, "Memphis: A framework for heterogeneous many-core SoCs generation and validation," *Design Autom. Embedded Syst.*, vol. 23, nos. 3–4, pp. 103–122, Dec. 2019, doi: 10.1007/s10617-019-09223-4.

[20] M. Ruaro, F. B. Lazzarotto, C. A. Marcon, and F. G. Moraes, "DMNI: A specialized network interface for NoC-based MPSoCs," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 1202–1205, doi: 10.1109/ISCAS.2016.7527462.

[21] E. Wachter, L. L. Caimi, V. Fochi, D. Munhoz, and F. G. Moraes, "BrNoC: A broadcast NoC for control messages in many-core systems," *Microelectron. J.*, vol. 68, pp. 69–77, Oct. 2017, doi: 10.1016/j.mejo.2017.08.010.

[22] H. Wang and B. Halak, "Hardware Trojan detection and high-precision localization in NoC-based MPSoC using machine learning," in *Proc. 28th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2023, pp. 516–521. [Online]. Available: https://ieeexplore.ieee.org/document/10044814

[23] L. L. Caimi and F. G. Moraes, "Security in many-core SoCs leveraged by opaque secure zones," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2019, pp. 471–476, doi: 10.1109/ISVLSI.2019.00091.

[24] R. F. Faccenda, G. Comarú, L. L. Caimi, and F. G. Moraes, "Lightweight authentication for secure IO communication in NoC-based many-cores," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2023, pp. 1–5, doi: 10.1109/iscas46773.2023.10181962.

[25] R. F. Faccenda, L. L. Caimi, and F. G. Moraes, "Detection and countermeasures of security attacks and faults on NoC-based many-cores," *IEEE Access*, vol. 9, pp. 153142–153152, 2021, doi: 10.1109/ACCESS.2021.3127468.

[26] Siemens. (2023). *ModelSim Simulator*. [Online]. Available: https://eda.sw.siemens.com/en-U.S./ic/modelsim

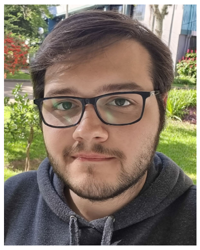
[27] Cadence. (2023). *Incisive SystemC, VHDL, and Verilog Simulation Training*. [Online]. Available: https://www.cadence.com/en_US/home/training/all-courses/82115.html

[28] OPNET. (2023). *OPNET Network simulator*. [Online]. Available: https://opnetprojects.com/opnet-network-simulator

[29] I. Weber, G. Marchezan, L. Caimi, C. Marcon, and F. G. Moraes, "Open-source NoC-based many-core for evaluating hardware Trojan detection methods," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5, doi: 10.1109/ISCAS45731.2020.9180578.

[30] L. L. Caimi, V. Fochi, E. Wachter, and F. G. Moraes, "Runtime creation of continuous secure zones in many-core systems for secure applications," in *Proc. IEEE 9th Latin Amer. Symp. Circuits Syst. (LASCAS)*, Feb. 2018, pp. 1–4, doi: 10.1109/LASCAS.2018.8399904.

RAFAEL FOLLMANN FACCENDA received the degree in computer engineering and the M.Sc. degree from Universidade Federal de Santa Maria (UFSM), Santa Maria, Brazil, in 2018 and 2020, respectively. He is currently pursuing the Ph.D. degree in computer science with PUCRS, Brazil. His research interests include many-cores, NoCs, embedded systems, and security for NoC-base many-cores.



GUSTAVO COMARÚ (Graduate Student Member, IEEE) received the degree in computer engineering from the Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil, in 2022, where he is currently pursuing the M.Sc. degree in computer science. His research interests include security and reliability of hardware and embedded systems.

LUCIANO LORES CAIMI received the M.Sc. degree in electrical engineer from the Federal University of Santa Catarina (UFSC), Florianopolis, Brazil, in 1998, and the Ph.D. degree in computer science from PUCRS, Porto Alegre, Brazil, in 2019. He is currently an Adjunct Professor with the Federal University of Fronteira Sul (UFFS). His main research interests include multiprocessor systems on chip (MPSoCs) and security for embedded systems.

FERNANDO GEHM MORAES (Senior Member, IEEE) received the degree in electrical engineering and the M.Sc. degree from Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil, in 1987 and 1990, respectively, and the Ph.D. degree from Laboratoire d'Informatique, Robotique et Microélectronique de Montpellier, France. He has been a Full Professor with PUCRS, since 2002. He has authored and coauthored 53 peer-refereed journal articles in the field of VLSI design. His primary research interests include microelectronics, FPGAs, reconfigurable architectures, NoCs, and MPSoCs.