Open-source NoC-based Many-Core for Evaluating Hardware Trojan Detection Methods

Iaçanã Weber*, Geaninne Marchezan*, Luciano Caimi†, César Marcon*, Fernando G. Moraes*
*School of Tecnhology - PUCRS – Av. Ipiranga 6681, 90619-900, Porto Alegre, Brazil
{iacana.weber, geaninne.lopes}@edu.pucrs.br, {cesar.marcon, fernando.moraes}@pucrs.br
†UFFS – Av. Fernando Machado 108E, 89802-112, Chapecó, Brazil
lcaimi@uffs.edu.br

***Abstract*—In many-cores based on Network-on-Chip (NoC), several applications execute simultaneously, sharing computation, communication and memory resources. This resource sharing leads to security and trust problems. Hardware Trojans (HTs) may steal sensitive information, degrade system performance, and in extreme cases, induce physical damages. Methods available in the literature to prevent attacks include firewalls, denial-of-service detection, dedicated routing algorithms, cryptography, task migration, and secure zones. The goal of this paper is to add an HT in an NoC, able to execute three types of attacks: packet duplication, block applications, and misrouting. The paper qualitatively evaluates the attacks' effect against methods available in the literature, and its effects showed in an NoC-based many-core. The resulting system is an open-source NoC-based many-core for researchers to evaluate new methods against HT attacks.**

***Index Terms*—Many-core, NoC, Security, Hardware Trojans.**

I. Introduction

NoC-based many-core systems, beyond their inherent scalability, provide massive parallelism and high performance to the users. In such systems, several applications execute simultaneously, sharing computation (processors and memories) and communication (routers and links) resources. This resource sharing leads to security and trust problems, requiring the design of solutions to avoid malicious entities exploring vulnerabilities and breaking security principles, like confidentiality, availability, integrity, and authentication [1].

Most of the attacks in these systems come from malicious software [2]. Nonetheless, the literature reports the insertion of Hardware Trojans (HTs) in several stages of the production process [3], compromising the original design. The objective of HTs includes stealing sensitive information, performance degradation, Denial-of-Service (DoS), and, in extreme cases, physical damage to the device.

This article explores vulnerabilities related to resource sharing at the communication level. One way to exploit this vulnerability is through Side-Channel Attacks (SCA) [4], [5]. Another way to exploit this vulnerability is to add an HT in the communication infrastructure [6]–[8], which enables to access the data of sensitive applications without directly attacking the application and changing its performance.

In general, HTs are classified into two categories according to their activation mechanism: always-on Trojan and trigger Trojan [3]. Always-on HTs are easily detected using, e.g., side-channel analyses. Trigger Trojans are not easily detected since they are usually in sleep mode, being wake-up by some event.

This paper assumes as case-study an infected router, distributed as a third-party intellectual property (3PIP), available as a netlist mapped to a given technology. This router comes with documentation, performance data (area, power, frequency), and test benches. The motivation to add an HT in a router is to have a backdoor, triggered by software. The malicious software may be deployed in the many-core and access sensitive information, induce DoS attacks and interrupt the execution of sensitive applications.

This work has as main *goals*: (*i*) to detail the insertion of an HT in an open-source NoC; (*ii*) to show that the security methods presented today in the literature present limited effectiveness against attacks carried-out by HT; and (*iii*) to demonstrate the efficiency of the attacks.

The *contribution* of the work is the availability of an open-source NoC-based many-core for evaluating HT detection methods. The many-core is described at the RTL level (VHDL and SystemC), with a multitask operation system, and benchmarks described in C language.

II. Attack Model

Our proposal assumes the insertion of an HT at design time by the Intellectual Property (IP) provider. At the beginning of many-core execution, the router behavior does not reveal the presence of HT, as HT is only enabled with a configuration packet sent by a malicious task. According to this packet, HT can execute three types of attack: (*i*) *misrouting*, deflecting packets to an invalid address; (*ii*) *block* the local port, disabling the communication of the IP with other IPs; and (*iii*) *packet duplication*, transmitting the packet to its destination and the malicious task. Once the HT awoke, it usually remains active until receiving another configuration packet to stop the attack.

Figure 1 illustrates the attacks explored in this work. The system contains two hardware layers, the trusted Processing Elements (PEs) and the untrusted NoC. Distinct PEs execute tasks A and B (T_A and T_B), being an application with security requirements, and T_Z references a malicious task. T_A sends data to T_B, represented by the green arrow.

Figure 1(*i*) illustrates the misrouting attack. T_Z configures the HT to misroute the T_A outgoing packets (red dotted flow). Consequently, T_A transmits packets to a PE, which is not waiting for data, or to an invalid address. These packets are not consumed, resulting in a DoS attack that obstructs other flows and may block the entire NoC operation. Figure 1(*ii*) illustrates the local-port blocking attack. In this case, the application is interrupted, and the result may be catastrophic for critical applications, as in autonomous driving. Figure 1(*iii*) shows the packet duplication attack. During the attack,

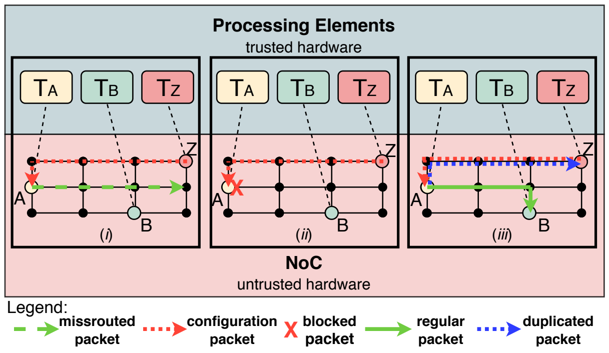

Fig. 1. Representation of attacks that may be executed using the proposed HT. (*i*) misrouting, (*ii*) local port blocking and (*iii*) packet duplication.

packets generated by T_A goes to T_B and also to T_Z, leaking sensitive data for a malicious task. The HT triggering process is not carried out by the local PE, but from other PE of the many-core; therefore, the protection measures implemented at the local Network Interface (NI) [8] become inefficient. The entire attack process is carried out in the router, over the local port. Thus, the application does not identify the information leakage, as previously mentioned. This attack method also easiness the interruption of the application execution, as well as DoS attacks.

III. NoC with Hardware Trojan Implementation

We modified an open-source NoC [9], including an HT for performing the attack model described in Section II.

Figure 2 illustrates the configuration packet that activates or deactivates the HT, which is a 3-flit packet with a 32-bit flit. The first two flits correspond to the packet header and the 3rd flit to the payload. The target address is stored in the 16 least significant bits of the 1st flit, and the 16th bit indicates the routing algorithm: XY (default routing) or YX (used by the HT). The 2nd flit is the payload size, always equal to one. The 3rd flit contains the attack signature (“10101010101010”), the attack mode (“00”: duplicate, “01”: misrouting, “11”: local port blocking), and the PE address holding the malicious task.

Figure 3 presents the top-level router architecture, encompassing three main blocks: input buffer, crossbar, and switch control. Additionally, we designed the HT block as well as the necessary modifications (in red) for this block to operate with the other blocks to implement an infected router.

Input Buffers connected to the West, East, North, and South ports received a 4-state Finite State Machine (FSM) that verifies if the flit addresses the local port, the payload size

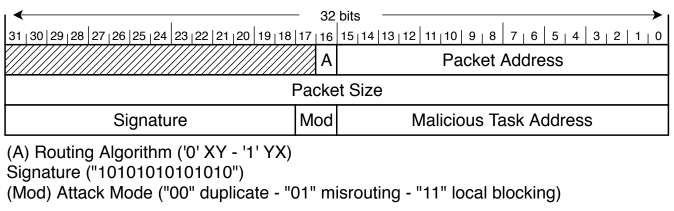

Fig. 2. HT configuration packet.

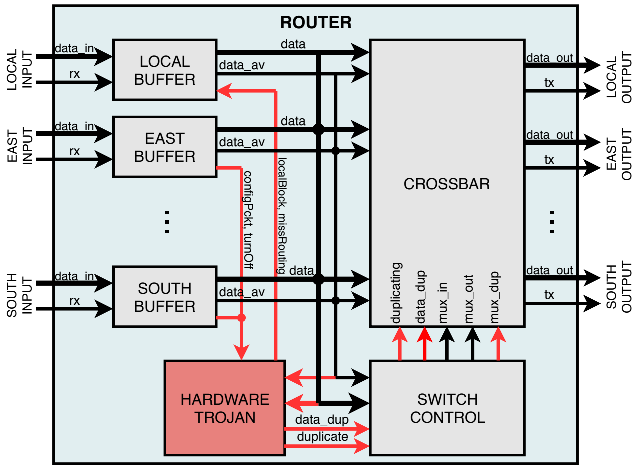

Fig. 3. Top-level architecture of the infected router.

is equal to one, and the 16 most significant bits of the 3rd flit matches with one of the predefined signatures. Validating these conditions, this FSM sends to the HT the malicious task address, the attack type, and through signals *configPckt* and *turnOff* if the HT should be activated or deactivated, respectively. The FSM does not interfere with the buffer operation, thus not affecting its performance. The local port buffer implements the misrouting attack by adding a constant to the target address (1st flit), and the blocking attack by masking the control flow signals, preventing any communication between the PE and router.

Switch Control is responsible for: (*i*) answering routing requests from input buffers; (*ii*) executing the routing algorithm; (*iii*) configuring the crossbar. We modified this module to implement the packet duplication attack, considering that standard and duplicate packets follow the XY and YX routing algorithms, respectively (deadlock in this case is not a concern). Initially, to answer a routing request, it is necessary to verify if both output ports are available. Next, the FSM responsible for executing the routing algorithm must set the connection between the input buffer with both output ports. We added a second entry in the routing table to enable the duplicated connection. The intrusiveness on this module is higher than the input buffers because it was necessary to modify the internal modules of the Switch Control. Using two output ports to duplicate packets may result in a latency overhead because both must be available. From one side, this latency overhead may reveal the presence of the HT, but from the other side, this is a usual overhead in NoCs with congested scenarios. Thus, machine learning-based methods could be adopted to detect the presence of HT [10].

Crossbar is a combinational circuit that uses the routing table to make the connection between buffers and output ports. The *duplicate* signal notifies the crossbar to use the added entry in the routing table to transmit packets to two output ports.

Hardware Trojan is a hardware of low complexity, with a 4-state FSM starting in a waiting state. When an input buffer receives an activation packet, the FSM reads the attack type, notifying the router modules to execute the actions related to the attack. For misrouting and blocking attacks, only the local input buffer is affected. For the duplication packet attack, the HT notifies the switch control to make the routing using two output ports. The reception of a packet equal to the one used to start the attack stops the HT, returning the FSM to the waiting state.

Discussion — the area consumption and power dissipation overheads, targeting a 32-bit flit width and 16-bit buffer depth configuration, were 8.13% and 10%, respectively (Cadence Genus 18.1, CMOS 65nm technology). It is noteworthy that the adopted router has a simple architecture. The literature reports similar values [3], [7], but for routers with several virtual channels and more complex architectures. This case-study showed that the addition of an HT corresponds not only to add a new hardware module, but also to define how it interacts with the hardware to be attacked.

IV. Many-core Protection Methods and Vulnerabilities

This Section presents, in a non-exhaustive way, methods found in the literature aiming for the protection of NoCs and many-cores against security attacks. Six methods are discussed, presenting at the end of each one their vulnerabilities against the proposed HT.

A. Firewalls. Hu et al. [11] propose a three-level firewall to provide access control, authentication, and availability of the communication system, preventing information leakage and DoS attack. The proposed method makes a design-time analysis of the traffic and the NoC architecture to select the levels and placement of the firewalls: (*i*) between a PE and a router or; (*ii*) between routers. Grammatikakis et al. [12] adopt NoC firewalls to check the physical address and reject untrusted CPU requests to on-chip memory; thus, protecting legitimate processes running in a multicore System-on-Chip (SoC) from the injection of malicious instructions or data to shared memory. The major problem in the adoption of firewalls is its operation mode, allowing or releasing packets according to a set of security policies. Then, even with strict security policies, if an infected router duplicates a legitimate packet and sends it to a malicious task, firewalls will let it pass.

B. DoS Detection and Prevention Mechanisms. DoS is an attack caused by a malicious core flooding the network with unnecessary packets that degrade the system performance through NoC congestion. Charles et al. [13] propose a distributed DoS detection scheme to measure the performance degradation of sensitive flows and to detect the router connected to the core that is the DoS source. Chaves et al. [14] propose a lightweight and real-time DoS attack detection mechanism. Once a potential attack has been flagged, the method finds the malicious core using the latency data gathered by NoC components. The location of the core responsible for the DoS attack solves one of the actions that the HT described above can perform, misrouting. However, an HT that only duplicates packets during specific periods does not produce a behavior perceived as a DoS attack. Thus, works that seek to detect anomalies in the NoC latency, possible do not detect this type of attack.

C. Specialized Routing Algorithms. Boraten and Kodi [15] propose a Non-Interference Based adaptive Routing (NIBR) to protect NoCs from side-channel and DoS attacks. The Authors employ adaptive routing algorithms to dynamically and spatially disperse network traffic among under-utilized routers by classifying traffic types into a set of sub-security domains. Sepúlveda et al. [16] present a runtime method to prevent timing side-channel attacks and information leakage. The work proposes two mechanisms: adaptive routing and random arbitration. The proposed method assumes that a malicious task in the path of a memory access may extract sensitive data from the temporal behavior of the communication flow. These examples detect DoS attacks, mitigating them by modifying the routing algorithm or arbitration policies. These mechanisms are inefficient against the proposed attack. The routing algorithm may change at runtime, but this action does not interfere with the HT actions, as packet replication.

D. Cryptography. Sharma et al. [17] propose an encryption mechanism for zone-to-zone secure communication. The authors present a runtime protocol that generates private and public keys for each IP core of the SoC. The secure zones are created dynamically, having a node responsible for the secure communication with other zones (anchor node). The protocol enables dynamically creation of session keys used to encrypt the message flows between the anchor nodes, protecting the communication. Sepúlveda et al. [18] protect computation and communication resources using spatial isolation of PEs with encryption mechanisms. Encrypting data is a mechanism that adds security to the flows, but with a significant latency penalty [19]. Although the data is encrypted, the proposed attack is still able to duplicate the packets. Thus, cryptography mitigates the attacks, but using methods to recover the cryptography key may steal sensitive data.

E. Task migration. Ancajas et al. [7] propose Fort-NoC, a three-layer security mechanism placed in each NI of the NoC. The 1st layer is the Data Scrambling (DS) that uses XOR cipher encryption to encrypt the data before sending it to the NoC. The 2nd layer is the Packet Certification (PC) that attaches an encrypted tag at the end of the packet before injecting it into the NoC. The 3rd layer is the Node Obfuscation (NObf), obtained with task migration, decoupling the source and destination nodes of a communication to increase side-channel resilience. The first two layers correspond to the adoption of ciphering methods. Task migration can be useful in ensuring application security, as data generation ceases in the PE, whose HT is scheduled to perform the attack. However, task migration is a costly process, and cannot be done frequently, because it would compromise the application performance. Even with task migration, the malicious task can schedule the attack on other routers, until finding the address of the migrated task.

F. Secure zones. The use of secure zones is a technique adopted to reduce resource sharing at the computation and/or communication level. Sepúlveda et al. [20] propose a Non-minimal Odd-Even Region-based routing NoC (NOE-RNoC), an architecture that combines the Region-Based Routing (RBR) mechanism at design time and non-minimal adaptive routing at runtime to encapsulate sensitive traffic efficiently. With this approach, the sensitive traffic is not shared with applications belonging to other applications. Caimi and Moraes proposed the Opaque Secure Zones (OSZ) [21], which is a method that reserves a rectilinear region at runtime to execute a sensitive application. The application traffic remains restricted to the OSZ, mitigation most attacks related to computation and communication sharing. A rerouting mechanism deviates the traffic that should traverse the OSZ. The creation of secure zones, using dedicated routing algorithms [20], or reserving PEs to execute a single application [22] is prone to the attacks proposed by the HT previously described. The adoption of OSZ mitigates the attack induced by packet replication because the duplicated packet that hits the OSZ boundary is discarded. However, the OSZ approach is vulnerable to the other attacks.

G. Discussion. Considering the three attacks that the HT may execute, none of the approaches presented in the literature can deal with them, as detailed in Table I. The effect of packet duplication may be avoided by using OSZ and mitigated by encryption methods at the cost of significant performance penalty. Packet misrouting, if considered as a DoS attack, may be treated by the methods presented in [13] [14]. Blocked packets interrupt the application execution, and there is no available method to deal with this attack. A possible solution may come from fault-tolerant approaches. An unresponsive PE is considered faulty, with the system manager remapping the task executing in the PE connected to the infected router in another PE, restarting the application after the remapping. Note that task migration can mitigate all attacks, since the position of the task in the system changes. However, as mentioned earlier, the cost of migrating a given task is high and has a significant impact on application performance.

TABLE I
PROTECTION METHOD VERSUS ATTACK TYPE (**PROTECT** / **MITIGATE** / **INEFFICIENT**).

Attack	Duplication			Misrouting			Blocked Pkt		
Method	P	M	I	P	M	I	P	M	I
Firewalls			●			●			●
DoS detection			●		●				●
Rout. algorithms			●			●			●
Cryptography		●				●			●
Task migration		●			●			●	
Secure zones			●			●			●
OSZ	●					●			●

In most of the studies founded in literature, the Authors propose security techniques for their HT. As a result, the proposed solutions are specific and not feasible in real NoC/SoC designs. The availability of an open-source NoC-based many-core that includes the described HT may encourage generic security techniques proposals.

V. Results

This section presents the HT operation in a 4x4 MPSoC, modeled at the RTL level [23], running four real applications: MPEG – 5 tasks; Dynamic Time Warping (DTW) – 6 tasks; producer-consumer – 2 tasks; malicious application – 1 task.

The malicious software performs the attacks targeting the DTW application. The Y-axis and X-axis of Figure 4 correspond to the NoC throughput (in $flits/ms$) and time (in ms), respectively. The black and orange curves are the baseline execution without the HT action and the execution with the HT activated by the malicious task, respectively. The first 1.5 ms corresponds to the warmup period, and the packet duplication attack occurs between 1.7 and 3.5 ms. Both scenarios start having the same number of flits received by the PEs, and the attack is characterized by the increase in the number of flits.

According to Table I, cryptography and OSZ, may protect the flow being duplicated. After the attack, both curves have the same behavior again. In 6 ms, the DTW application is blocked, and it is observed the reduction of the NoC throughput. If the application monitors its latency, task migration can mitigate this attack. Finally, from 8 ms, the DoS attack starts. The network quickly blocks, and at 11 ms, there is no more data being transmitted into the NoC. DoS detection mechanisms may find where the NoC is blocked and discard packets to release the network.

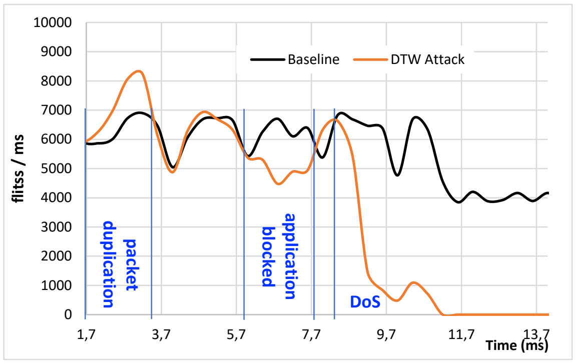

Fig. 4. Scenario with the DTW application under attack.

This result demonstrates the effectiveness of all three types of attacks. Without effective methods of detecting them, applications with security requirements can be attacked without them realizing that they are under attack.

VI. Conclusion

This article inserted an HT on an open-source NoC to evaluate qualitatively counter-measures presented in the literature, concluding that they are not enough to block the proposed attacks. Finally, results showed the efficiency of the attacks in an NoC-based many-core. In the Author's knowledge, this is the first work providing an open-source many-core equipped with HTs in the NoC to evaluate security counter-measure. As future works, investigation of formal methods to detect the HT using assertions is a direction to follow.

Acknowledgement

The Author Fernando Gehm Moraes is supported by FAPERGS (17/2551-0001196-1) and CNPq (302531/2016-5), Brazilian funding agencies. César Marcon is supported by CNPq (303460/2017-2), brazilian funding agency.

References

[1] J. Ramachandran, *Designing Security Architecture Solutions* . John Wiley & Sons, Inc., 483p, 2002.

[2] J. Sepúlveda, G. Gogniat, D. Flórez, J. P. Diguet, C. Zeferino, and M. Strum, "Elastic security zones for NoC-based 3D-MPSoCs," in *ICECS*, 2014, pp. 506–509.

[3] J. Wang, S. Guo, Z. Chen, and T. Zhang, "A Benchmark Suite of Hardware Trojans for On-Chip Networks," *IEEE Access*, vol. 7, pp. 102 002–102 009, 2019.

[4] A. K. Biswas, "Side channel attack on NoC-based MPSoCs are practical: NoC Prime+Probe attack," in *SBCCI*, 2016, pp. 1–6.

[5] J. Wang, S. Guo, Z. Chen, and T. Zhang, "Efficient Timing Channel Protection for Hybrid Packet/Circuit-Switched Network-on-Chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 5, pp. 1044–1057, 2018.

[6] T. Boraten and A. K. Kodi, "Mitigation of Denial of Service Attack with Hardware Trojans in NoC Architectures," in *IPDPS*, 2016, pp. 1091–1100.

[7] D. M. Ancajas, K. Chakraborty, and S. Roy, "Fort-NoCs: Mitigating the Threat of a Compromised NoC," in *DAC*, 2014, pp. 1–6.

[8] V. Y. Raparti and S. Pasricha, "Lightweight Mitigation of Hardware Trojan Attacks in NoC-based Manycore Computing," in *DAC*, 2019, pp. 1–6.

[9] F. G. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *Integration*, vol. 38, no. 1, pp. 69–93, 2004.

[10] A. M. Kulkarni, Y. Pino, M. French, and T. Mohsenin, "Real-time anomaly detection framework for many-core router through machine-learning techniques," *JETC*, vol. 13, no. 1, pp. 10:1–10:22, 2016.

[11] Y. Hu, D. Müller-Gritschneder, M. J. Sepúlveda, G. Gogniat, and U. Schlichtmann, "Automatic ILP-based Firewall Insertion for Secure Application-Specific Networks-on-Chip," in *INA-OCMC@HiPEAC*, 2015, pp. 9–12.

[12] M. D. Grammatikakis, P. Petrakis, A. Papagrigoriou, G. Kornaros, and M. Coppola, "High-level security services based on a hardware NoC Firewall module," in *WISES*, 2019, pp. 73–78.

[13] S. Charles, Y. Lyu, and P. Mishra, "Real-time Detection and Localization of DoS Attacks in NoC based SoCs," in *DATE*, 2019, pp. 1160–1165.

[14] C. G. Chaves, S. P. Azad, T. Hollstein, and J. Sepúlveda, "A Distributed DoS Detection Scheme for NoC-based MPSoCs," in *NORCHIP*, 2018, pp. 1–6.

[15] T. H. Boraten and A. K. Kodi, "Securing NoCs Against Timing Attacks with Non-Interference Based Adaptive Routing," in *NOCS*, 2018, pp. 1–8.

[16] M. J. Sepúlveda, J. Diguet, M. Strum, and G. Gogniat, "NoC-Based Protection for SoC Time-Driven Attacks," *Embedded Systems Letters*, vol. 7, no. 1, pp. 7–10, 2015.

[17] G. Sharma, S. Ellinidou, V. Kuchta, R. A. Sahu, O. Markowitch, and J. Dricot, "Secure Communication on NoC Based MPSoC," in *SecureComm*, 2018, pp. 417–428.

[18] M. J. Sepúlveda, D. Flórez, V. Immler, G. Gogniat, and G. Sigl, "Efficient security zones implementation through hierarchical group key management at NoC-based MPSoCs," *Microprocessors and Microsystems - Embedded Hardware Design*, vol. 50, pp. 164–174, 2017.

[19] B. S. Oliveira, H. M. Medina, A. C. Sant'Ana, and F. G. Moraes, "Secure Environment Architecture for MPSoCs," in *SBCCI*, 2018, pp. 1–6.

[20] J. Sepúlveda, R. Fernandes, C. A. M. Marcon, D. Florez, and G. Sigl, "A security-aware routing implementation for dynamic data protection in zone-based MPSoC," in *SBCCI*, 2017, pp. 59–64.

[21] L. L. Caimi and F. G. Moraes, "Security in Many-Core SoCs Leveraged by Opaque Secure Zones," in *ISVLSI*, 2019, pp. 471–476.

[22] M. M. Real, P. Wehner, V. Migliore, V. Lapotre, D. Göhringert, and G. Gogniat, "Dynamic spatially isolated secure zones for NoC-based many-core accelerators," in *Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2016, pp. 1–6.

[23] M. Ruaro, L. Caimi, V. Fochi, and F. G. Moraes, "Memphis: a Framework for Heterogeneous Many-core SoCs Generation and Validation," *Design Automation for Embedded Systems*, 2019.