Hardware Trojan Localization for Untrusted Network-on-chips

Gustavo Comarú*, Rafael Follmann Faccenda*, Luciano Lores Caimi†, Fernando Gehm Moraes*
*School of Technology, Pontifical Catholic University of Rio Grande do Sul – PUCRS – Porto Alegre, Brazil
{gustavo.comaru, rafael.faccenda}@edu.pucrs.br, fernando.moraes@pucrs.br
†Federal University of Fronteira Sul – UFFS – Chapecó, Brazil – lcaimi@uffs.edu.br

***Abstract*—Manycore platforms are designed to provide high performance through parallelism, addressing the current demand for embedded devices with power consumption and communication constraints. A manycore system contains processing elements (PEs) interconnected by complex communication infrastructures, such as Networks-on-Chip (NoCs). As the adoption and complexity of manycores increase, data protection emerges as a critical design requirement. Furthermore, the widespread use of third-party intellectual property cores (3PIPs) to meet time-to-market constraints and reduce design costs increases the risk of malicious hardware insertion through Hardware Trojans (HTs), thereby making manycores more vulnerable. The literature presents security techniques, such as cryptography, authentication codes, error correction codes, and the creation of secure zones. However, these countermeasures are only effective if the system can identify the location of the attack source. The goal of this work is to present a non-invasive method for localizing HTs in an NoC. The proposed method employs a probing protocol with test packets to detect the presence of HTs. The results demonstrate the effectiveness of this method in accurately identifying HTs within the NoC.**

***Index Terms*—NoC-based manycores, security, hardware Trojan (HT), HT localization**

I. Introduction

As the adoption and complexity of manycores increase, data protection emerges as a critical design requirement [1]. Manycores may be deployed in scenarios where high availability is essential, and downtimes must be minimized. Additionally, these systems may process sensitive information, requiring data protection from unauthorized access.

The increasing number of features and functionalities integrated into a single chip has led to the adoption of third-party intellectual property cores (3PIPs) to meet time-to-market constraints and reduce design costs. These IPs are sourced from various vendors, which increases the risk of Hardware Trojan (HT) insertion [2]. HTs that infect the NoC can execute various attacks that compromise fundamental security principles [3], such as *confidentiality* by redirecting messages to malicious agents, *availability* by dropping messages or blocking communication paths, and *integrity* by corrupting the content of packets traversing the NoC.

The literature presents countermeasures to protect manycores, including cryptography [4], authentication codes [5], error correction codes [6], the creation of communication flow profiles to detect anomalous behavior [7], and the use of Opaque Secure Zones [8]. However, most countermeasures are limited in effectiveness without knowledge of the HT's location. Although the literature offers solutions to localize attacks in NoCs (Section II), these methods typically involve adding hardware to the untrusted router, which can compromise the security of the localization mechanism itself.

The goal of this work is the creation of a non-invasive method to find the location of HT-infected routers in an untrusted NoC by probing the network with test packets.

II. Related Work

The literature proposes similar security mechanisms for either locating an HT within the NoC [9]–[12], identifying attacks originating from an IP connected to the NoC [13, 14], or protecting applications from an insecure NoC [15]–[17]. Table I categorizes these related works.

The first columns of the table pertain to the threat model. ***Attacker location*** indicates where the malicious component is deployed, such as within the routers or links of the NoC, or in the software of a malicious application. Some works [11, 13, 14] do not specifically identify the attacker as an HT, but their proposed solutions are still applicable for locating HT-related attacks. ***Attack type*** defines the specific attacks that can be executed, such as DoS (Denial of Service), spoofing, information leakage, or packet tampering.

The next columns address the security mechanism proposed to tackle the threat model. The column ***Security mechanism*** overviews the features implemented to protect the system. ***Detection*** relates to the type of monitoring used to detect the occurrence of an attack: it can be either Hop-to-Hop (H2H) with monitors deployed in every router; or End-to-End (E2E) with monitors in the communication endpoints. Hussain et al. [10] use both types of detection, enabling the usage of more precise H2H modules only after the attack is detected by the E2E monitors. ***Mechanism location*** defines where the security features are implemented in the system (e.g., integrated into the NoC routers, or as a network interface).

Our work distinguishes itself from others for its *non-invasive* localization mechanism, meaning it is not implemented within the NoC. Most proposals suggest adding security mechanisms to NoC routers or links, which has drawbacks. Authors [11]–[14, 16, 17] add security hardware to NoC routers, which is effective against malicious cores but unsuitable if HTs infect the NoC itself, as the security hardware inside an untrusted router cannot be trusted. Hussain et al. [10] suggest deploying detection units on every NoC link, leading to significant area overhead. Instead, our approach uses OS-sent probe packets to detect infected NoC routers without adding security features to the NoC.

This work was financed in part by CNPq (grants 309605/2020-2 and 407829/2022-9), and FAPERGS (grants 21/2551-0002047-4 and 23/2551-0002200-1).

TABLE I
CLASSIFICATION OF WORKS RELATED TO HT LOCALIZATION IN NOC-BASED MANYCORES.

Work	Year	Attacker location	Attack type	Security mechanism	Detection	Mechanism location
Wang et al. [9]	2023	NoC router Network Interface	Packet tempering	Tempering detection (Machine Learning) – Credit table	E2E	(Not disclosed)
Hussain et al. [10]	2018	NoC router	Information leakage Flooding (DoS) Performance degradation Packet tempering	Selective activation of H2H units Worm-based algorithm	E2E H2H	Network interface NoC links
Sinha et al. [11]	2021	IP block	Flooding (DoS) Performance degradation	Flooding detection (ML) Probe packets	H2H	NoC router
Daoud et al. [12]	2019	NoC router	Packet dropping (DoS)	Ack signal	H2H	NoC router
Sepúlveda et al. [15]	2017	NoC router NoC links	Information leakage Packet tempering	Packet encryption Authentication codes	–	Network interface
Charles et al [13]	2020	IP block	Flooding (DoS)	Diagnosis message	H2H	NoC router
Chaves et al. [14]	2019	Application software	Flooding (DoS)	Path collision	E2E	NoC router
Bhamidipati et al. [16]	2024	NoC router	Information leakage	Hash tables of communicating pairs Packet counters	H2H	NoC router
Hossain et al. [17]	2024	NoC router	Flooding (DoS) Performance Degradation	NoC performance counters AI model for attack detection	H2H	NoC router Software
This work		NoC router NoC links	**DoS** **Information leakage** **Spoofing**	**Probe packets**	**E2E**	**OS (software)**

III. MANYCORE AND ARCHITECTURAL ASSUMPTIONS

This work adopts a modified version of the HeMPS [18] manycore. The main features of the HeMPS platform are: (i) an NoC with a 2D-mesh topology, input buffering, credit-based flow control, round-robin arbitration, and XY-routing algorithm; (ii) homogeneous PEs with private memory, a processor, and a DMNI (Direct Memory Network Interface) module.

This work keeps the original PE architecture but uses two NoCs:

Data-NoC (D_{NoC}) [19]: The original NoC is modified to use two 16-bit disjoint physical channels, enabling full adaptive routing through *source routing* (with default routing being XY).

Control-NoC (C_{NoC}) [20]: A lightweight NoC where all packets consist of a single flit. In default broadcast mode, packets reach all PEs, enabling the communication between a source and a target PE, even with faults or HTs in the D_{NoC}. For security reasons, only the OS can access the C_{NoC}, preventing its misuse by malicious applications.

The proposed method for detecting HTs is agnostic to the D_{NoC} model; however, it is assumed that the D_{NoC} is equipped with the following features:

- Signals indicating the start and end of the packet - **BOP** (beginning of packet) and **EOP** (end of packet);
- **Router port reset**, which involves flushing the flits of a specific input buffer. This feature is required because the presence of HTs can leave flits indefinitely stored in buffers with the router remaining switched, i.e., with an active connection between an input and an output port.

The manycore is modeled at the RTL level, part in SystemC (memory, processor, DMNI), and part in VHDL (data and control routers).

IV. THREAT MODEL AND HT TYPES

The premise of this work is that all manycore components are secure, with only the D_{NoC} being vulnerable to HT attacks. This work assumes the following threats and HT types:

DoS:

- **Black Hole HT**: An HT that drops any packet attempting to traverse a given link.
- **Blocking HT**: Induces performance degradation by temporarily blocking packets, leading to QoS and real-time constraints violations.
- **Flooding HT**: Injects a flow of flits into the D_{NoC}, causing congestion.

Information leakage:

- **Misrouting HT**: redirects a packet to a malicious task or IO device.

Spoofing (identity falsification):

- **Packet Injector HT**: Sends a packet through the D_{NoC}. The injected packets proceed to be routed as regular packets, arriving at a PE, thereby enabling the extraction of sensitive information.

HT attacks can be implemented using different activation mechanisms. The simplest is the ***always-on*** activation, where the HT is always active, making locating it easier. In contrast, ***static time-triggered*** activation uses an internal counter to activate the HT within a predefined time window, starting and stopping the attack based on specific clock cycles. A more complex approach is ***intermittent time-triggered*** activation, where the HT is activated and deactivated at random intervals, making it harder to detect due to its unpredictable behavior.

V. HT LOCALIZATION PROTOCOL FOR NOCS

The existing literature on HT localization in NoCs proposes solutions that either add extra hardware to the unreliable router or add security circuitry in every NoC link. Our proposal does not add extra hardware to the insecure D_{NoC} but instead leverages the C_{NoC} to search for the Trojan-infected routers.

The method we propose is named "*path probing*". Path probing consists of exploring (i.e., probing) different routes in the D_{NoC} and recording whether they are successful or broken. If several broken paths intersect at a single link, then it is probable that this link is infected with an HT. By using source routing, it is possible to choose the path to probe, and the C_{NoC} can be used to detect the success or failure of the probed path.

A. Path Probing API

The task of localizing HTs is attributed to a Manager PE (MPE). The MPE receives security warnings from other components and analyzes the suspicious segments of the network to find the location of HTs. The MPE sends probe packets to test whether a specific network segment is working as expected or not, using the Path Probing API:

Send Probe Request ($f1$ in Fig. 1): when the MPE decides to initiate a new probe, it calls the *send_probe_request* function, specifying the source and target PEs, and the path to be taken by the probe packet. A unique identifier is created, and the information related to the probe is stored in a local table at the MPE.

Handle Probe Result ($f2$ in Fig. 1): when the probing is finished, the MPE receives a PROBE_RESULT packet and the function *handle_probe_result* is called. This function receives the value of Probe_ID and the result specifying if the probe was a success or a failure.

Figure 1 presents the Probing Protocol. The MPE initiates the protocol by sending two packets to the Probe Source PE: (i) PROBE_REQUEST contains Probe Source Address (*source*), Probe Target Address (*target*), and Probe_ID; (ii) PROBE_PATH contains the source-routing path to be followed by the probe.

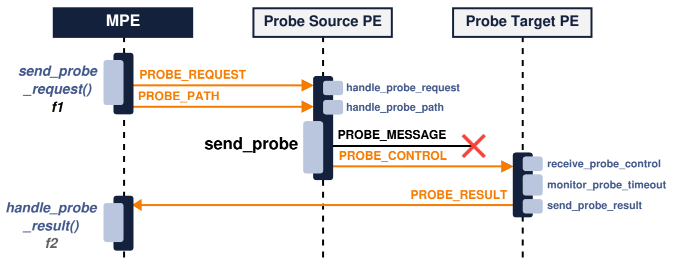

Fig. 1. Probing Protocol. Black and orange arrows represent packets sent via the D_{NoC} and C_{NoC}, respectively. The functions executed at each step are represented in blue.

After receiving both packets from the MPE, the *source* sends two probe packets to the *target*: (i) PROBE_CONTROL is sent via the C_{NoC} , and it is guaranteed to arrive due to the broadcast nature of the network. This packet contains the Probe_ID, which triggers a time counter. (ii) PROBE_MESSAGE, is the probe packet itself, transmitted through the D_{NoC}. The PROBE_MESSAGE may or may not arrive at the *target*, as the path the packet takes in the D_{NoC} is susceptible to HT attacks. In the scenario presented in Figure 1, the probe is dropped by an HT and does not arrive at its destination. If the data NoC is healthy, both packets arrive within a short interval of each other, and the probe is considered to be a success. But if the PROBE_MESSAGE packet does not arrive or is delayed, a timeout occurs due to the time counter violation started by the PROBE_CONTROL, and the probe is considered to be a failure. In both cases, a PROBE_RESULT packet with the probe's result is sent to the MPE.

B. Binary Search Localization

This section presents the "Binary Search Localization" (**BSL**) algorithm for locating an HT. When a PROBE_MESSAGE packet fails to reach its destination, the *target* sends a warning message to the MPE, which initiates the **BSL** algorithm. The algorithm divides the suspicious path into two segments and tests each segment individually with different probes. Segments that function correctly are considered HT-free. Segments that fail are further divided, and the process is repeated recursively. The HT is considered localized when the suspicious path is reduced to a single hop. At this point, since the path cannot be divided further, the HT is located in the single link that was probed.

Figure 2 illustrates how a suspicious path can be divided to search for HT-infected hops. When a path is divided it produces two new paths with half the size of the original: the *left-side path* and the *right-side path*. The method considers each probe to be a different search. This allows the **BSL** algorithm to take maximum advantage of the support for parallel probes: when the path is broken, the MPE sends probes to the left side and right side simultaneously; and when a PROBE_RESULT packet arrives, it is handled independently.

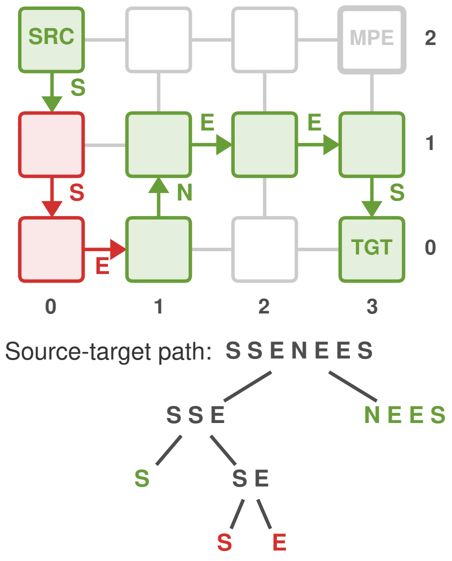

Fig. 2. Example of BSL used to localize HTs on a suspicious path. The upper part of the figure shows the path between SRC and TGT. Routers with infected links are in red, HT-free routers are in green, and routers outside the path are in light gray. Infected links (red arrows) block any packet attempting to traverse them. The lower part of the figure presents a tree with the probes performed to localize the HT. Black probes fail and are divided in half. Green probes succeed and are not divided. The two red probes fail but cannot be divided, identifying HTs at 0x1-South and 0x0-East.

Considering that each probe can be treated as a separate search makes implementing the algorithm straightforward. The MPE keeps a table called the Binary Search Table which stores the information for each probe used in the search: the Probe_ID and the Source Routing path. When a suspicious path is broken into two halves, the MPE allocates two more lines in the table, and when the result for a probe comes back the MPE retrieves the parameters from the table and then releases the "parent" line.

TABLE II

EFFECTS OF FAULTY PACKETS ON THE SYSTEM AND THE PROPOSED SOLUTIONS.

Faulty-packet issue	Cause	Effected location	Effect	Proposed solution
Packet stuck in router	Blocking HT	NoC router	Residual switching and congestion	Port Reset
Packet without tail	Black Hole HT Blocking HT Packet Injector HT	NoC router Network interface	Residual switching and congestion Target NI waits for non-existing packet tail	Port Reset Reception Timeout
Packet without header	Black Hole HT Blocking HT+Reception Timeout Packet Injector HT	Network interface	Packet is routed to the incorrect target	BOP Signal
Packet without header and tail	Intermittent Black Hole HT Intermittent Blocking HT+Reception Timeout Packet Injector HT	NoC router	Packet is routed to incorrect target Packet leaves behind residual switching	BOP Signal

Note that the **BSL** algorithm may not detect the HT on the evaluated path if the HT is deactivated after the attack. In such cases, the path is marked as suspicious, prompting the MPE to conduct further searches that include the routers on the suspicious path.

C. Faulty Packets

HTs induce *faulty packets*, which can cause significant issues for the overall system, including the complete blockage of the D_{NoC}. While the literature acknowledges the occurrence of attacks and faults, it does not offer solutions for their side effects on the D_{NoC}.

Four types of faulty packets that negatively impact the system were identified: (i) a packet stuck in a router, such as in the case of a Blocking HT; (ii) a packet missing a tail, caused by either a cropped packet or an attack suppressing the EOP signal; (iii) a packet without a header; and (iv) a packet lacking both header and tail. These faulty packets may induce *residual switching*, which occurs when the links in the path affected by the HT remain switched (i.e., connecting an input port to an output port) even after the packet was transmitted, thereby preventing other flows from using these ports.

Table II presents the proposed solutions to protect the manycore from faulty packets. The architectural mechanisms to prevent faulty packets, as described in Section III, include BOP and router port reset. At the NI we added a reception timeout. In wormhole switching mode, flits arrive continuously at the NI (we set this timeout arbitrarily to 30 clock cycles).

VI. RESULTS

A framework was implemented to evaluate the HT localization protocol, enabling the simulation of the HT attacks described in Section IV. HTs were modeled as hardware blocks that can be placed at any NoC link. Figure 3 presents two scenarios used to validate the protocol. It is important to mention that a comprehensive attack campaign was conducted to validate the proposal.

Figure 3(a) shows a producer-consumer application exchanging messages through a path infected with 3 Black Hole HTs. During the execution of the application, the HTs are activated simultaneously, triggering the BSL algorithm, which, by the end of the simulation, successfully finds the coordinates of all HTs. Figure 3(b) shows the execution of a DTW application. Tasks *P1-P4* communicate with *Rec* using a shared path, which is infected with a Black Hole HT. When the HT is activated, each communicating pair affected by the attack sends a warning to the MPE, each initiating a different

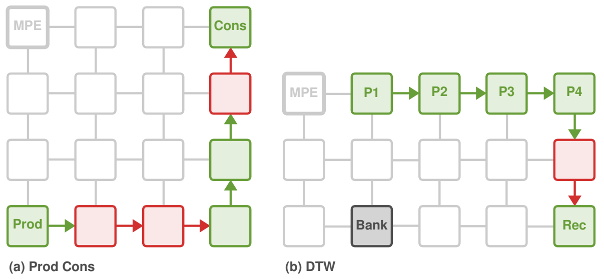

Fig. 3. Examples of testcases used to validate the HT localization protocol. Labels show which task is executed in each PE. The paths compromised by HTs are colored: green routers are not infected; red routers contain HTs, and drop packets trying to cross the infected link (shown as a red arrow). (a) Producer-consumer application trying to communicate through a path with multiple HTs. (b) DTW application with four workers (P1-P4) sending messages to the Recognizer task (Rec) using the same path.

binary search. The MPE queues the incoming requests and handles one at a time. Once the first search finishes and the HT is localized, the MPE starts the second search. Since the same HT is responsible for all the attacks, all the binary searches performed in this scenario find the same HT coordinate.

In all scenarios, applications executed correctly, fulfilling the following actions: (1) attack notification; (2) BSL execution, accurately locating the HT(s); (3) execution of countermeasures to flush residual flits in the NoC; (4) execution of the path-search mechanism to define a new source-target path; and (5) packet retransmission. The *originality* of this work lies in the localization proposal (BSL algorithm) and the treatment of residual switching and residual flits.

VII. CONCLUSIONS AND FUTURE WORK

This work presented a non-invasive approach to locating HTs in NoC, employing a probe mechanism and a binary search-based algorithm. The method is non-invasive as it is entirely implemented in software. However, it requires a control NoC and mechanisms in the data NoC, such as BOP and router port reset, to manage residual switching and residual flits.

The attack campaign revealed that certain HTs were not located due to their intermittent behavior. Future work includes: (1) proactive searches for HTs on paths marked as suspicious; (2) move suspicious paths/routers to healthy ones when the absence of HTs is confirmed; (3) adoption of statistical methods to locate HTs based on scores returned by probes; (4) use the HT locations for systemic measures, such as avoiding the mapping of applications that may use links with HTs.

References

[1] S. Baron, M. S. Wangham, and C. A. Zeferino, "Security mechanisms to improve the availability of a Network-on-Chip," in *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2013, pp. 609–612.

[2] H. Li, Q. Liu, and J. Zhang, "A survey of hardware Trojan threat and defense," *Integration, the VLSI Journal*, vol. 55, pp. 426–437, 2016.

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

[4] S. Charles and P. Mishra, "Securing Network-on-Chip Using Incremental Cryptography," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2020, pp. 168–175.

[5] G. Sharma, V. Kuchta, R. A. Sahu, S. Ellinidou, S. Bala, O. Markowitch, and J. Dricot, "A twofold group key agreement protocol for NoC-based MPSoCs," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 6, pp. 1–18, 2019.

[6] H. Gondal, S. Fayyaz, A. Aftab, S. Nokhaiz, M. Arshad, and W. Saleem, "A method to detect and avoid hardware trojan for network-on-chip architecture based on error correction code and junction router (ECCJR)," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 4, pp. 581–586, 2020.

[7] S. Charles, Y. Lyu, and P. Mishra, "Real-Time Detection and Localization of Distributed DoS Attacks in NoC-Based SoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4510–4523, 2020.

[8] L. L. Caimi and F. Moraes, "Security in Many-Core SoCs Leveraged by Opaque Secure Zones," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2019, pp. 471–476.

[9] H. Wang and B. Halak, "Hardware Trojan detection and high-precision localization in NoC-based MPSoC using machine learning," in *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, 2023, pp. 516–521.

[10] M. Hussain, A. Malekpour, H. Guo, and S. Parameswaran, "EETD: An energy efficient design for runtime hardware trojan detection in untrusted network-on-chip," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2018, pp. 345–350.

[11] M. Sinha, S. Gupta, S. S. Rout, and S. Deb, "Sniffer: A machine learning approach for DoS attack localization in NoC-based SoCs," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 2, pp. 278–291, 2021.

[12] L. Daoud and N. Rafla, "Detection and prevention protocol for black hole attack in network-on-chip," in *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, 2019, pp. 1–2.

[13] S. Charles, Y. Lyu, and P. Mishra, "Real-time detection and localization of distributed DoS attacks in NoC-based SoCs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4510–4523, 2020. [Online]. Available: https://doi.org/10.1109/TCAD.2020.2972524

[14] C. G. Chaves, S. P. Azad, T. Hollstein, and J. Sepúlveda, "DoS attack detection and path collision localization in NoC-based MPSoC architectures," *Journal of Low Power Electronics and Applications*, vol. 9, no. 1, p. 7, 2019.

[15] J. Sepúlveda, A. Zankl, D. Flórez, and G. Sigl, "Towards protected MPSoC communication for information protection against a malicious NoC," *Procedia computer science*, vol. 108, pp. 1103–1112, 2017.

[16] P. Bhamidipati and R. Vemuri, "QA-NoCs: Quantitative Analysis for Trojan Detection in Network-on-Chips," in *2024 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2024, pp. 757–761.

[17] N. Hossain, A. Buyuktosunoglu, J. D. Wellman, P. Bose, and M. Martonosi, "Noc-level threat monitoring in domain-specific heterogeneous socs with socurity," in *ACM/IEEE Annual International Symposium on Computer Architecture*, 2024.

[18] E. A. Carara, R. P. de Oliveira, N. L. Calazans, and F. G. Moraes, "HeMPS - a framework for NoC-based MPSoC generation," in *2009 IEEE International Symposium on Circuits and Systems*. IEEE, 2009, pp. 1345–1348.

[19] 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, the VLSI Journal*, vol. 38, no. 1, pp. 69 – 93, 2004.

[20] E. Wachter, L. L. Caimi, V. Fochi, D. Munhoz, and F. G. Moraes, "BrNoC: A broadcast NoC for control messages in many-core systems," *Microelectronics Journal*, vol. 68, pp. 69 – 77, 2017.