# Security Aspects of Real-Time MPSoCs: The Flaws and Opportunities of Preemptive NoCs

Bruno Forlin, Cezar Reinbrecht, Johanna Sepúlveda

# Security Aspects of Real-time MPSoCs: The Flaws and Opportunities of Preemptive NoCs

Bruno Forlin[1], Cezar Reinbrecht, and Johanna Sepúlveda[2]

[1] Federal University of Rio Grande do Sul
Av. Paulo Gama, 110 - Farroupilha, Porto Alegre - RS, Brazil
bruno.eforlin@inf.ufrgs.br
[2] Airbus Defence and Space GmbH
Willy-Messerschmitt-Straße 1, 82024 Taufkirchen, Germany
johanna.sepulveda@airbus.com

**Abstract.** Multi-Processor System-on-Chips (MPSoCs) is a standard platform used in time-critical applications. These platforms usually employ Priority-Preemptive NoCs (PP-NoCs), a widely used real-time on-chip interconnection structure that offers communication predictability. A deep analysis of the PP-NoC parameters and their impact on system security is required. Moreover, countermeasures that can protect the system while guaranteeing the real-time capabilities should be proposed and evaluated. To this end, this paper explores and evaluates the impact of the PP-NoCs parameters on system security; exploits PP-NoCs vulnerabilities and demonstrates for the first time two very powerful attacks; and proposes and integrates three new security countermeasures: RT-blinding, RT-masking, and RT-shielding. Results show that PP-NoCs are vulnerable to attacks and that is possible to uncover victim's information with high accuracy (up to 96.19%). On the other hand, protection techniques were able to harden the system, effectively and efficiently mitigating the vulnerabilities while maintaining deterministic behavior.

**Keywords:** Network-on-Chip, Secure MPSoC, Timing Side-channel Attacks

## 1 Introduction

Real-time applications (RTA) are becoming very popular as more embedded systems enter in daily life. Examples include health care equipment, automotive safety mechanisms, smart greenhouses, agriculture, avionics, and aerospace technology. Most of these systems require a powerful and efficient hardware platform, where Multi-processor Systems-on-Chips (MPSoCs) are the *status quo*. Current MPSoCs already support RTAs through ad-hoc solutions, such as real-time processors [1], operating systems for critical applications [2] and Networks-on-Chip (NoCs) with Quality-of-Service (QoS) [3–5]. Although all these components perform RTAs successfully, the techniques, architectures, and implementations used

to ensure their predictability create security vulnerabilities. Consequently, any security issue in these critical systems will also be critical. Security backdoor can be easily exploited to affect the safety-related characteristics of the systems. Therefore, to ensure the development of safety-critical systems, it is mandatory to understand and to guarantee the security of these systems. In this work, we contribute by identifying and discussing the flaws and opportunities in designing secure real-time MPSoCs. In particular, we focus on this work in the on-chip communication structure based on NoCs.

Recent works have shown the effectiveness of Priority-Preemptive NoCs (PP-NoCs) in guaranteeing real-time requirements and support mixed-critically traffic [4–7]. The works of [6] and [7] demonstrated that low priority traffic at PP-NoCs can affect the high priority traffic behavior. Furthermore, these works elaborate on a mathematical model able to estimate such effects in terms of packet latency. Despite these accurate models were developed to support the design of predictable communication structures for critical applications, they also provide a means for attackers to successfully retrieve sensitive data from the MPSoCs. An adversary, for example, may use this information to perform the so-called NoC timing attacks [8–10]. These attacks exploit the microarchitecture of the NoC, the shared communication nature and the main role of the communication in the system operation to passively gather information during the normal operation of the system [11]. A processor inside the MPSoC can be infected (i.e., an external malicious application with hidden functionalities or backdoors) and then, it may start to inject dummy packets in the network. In this scenario, the attacker aims to collide its traffic with a victim's traffic. In the absence of collisions, the attacker throughput is maximal. However, the degradation of throughput sensed by the attacker reveals the presence of collisions. Sensitive information, such as communication affinity (to whom the victim communicates most), communication rates and size of packets, may be extracted using this technique. In addition, this information may be used to further enhance powerful attack (e.g., cache attacks), as described in [12] [13]. The identification of vulnerabilities and further exploitation of cover channels on the real-time NoCs, like PP-NoCs, are still unexplored.

Previous attacks were demonstrated in a wide variety of NoC architectures. However, NoC attacks on real-time MPSoC have been not widely explored. For the best of our knowledge, our previous work in [14] is the only study in this direction. This work introduced two attacks named *Direct Interference* and *Back-Pressure*. These attacks exploit two MPSoC vulnerabilities. First the traffic predictability and shaping of sensitive applications. Second, the shared memories. As a protection mechanism to circumvent potential attacks, this former work proposed two NoC countermeasures. In this paper, we further extend the work in [14] by providing a refined and deep analysis of the PP-NoCs vulnerabilities. We explore the different design parameters that define the configuration of a PP-NoC and evaluate their impact on the overall MPSoC security. Also, we propose a new countermeasure, which tunes the PP-NoCs parameters at design

time to avoid the main known vulnerabilities. In summary, the contributions of the paper are:

- Vulnerability exploration of preemptive NoCs through the evaluation of the impact of the PP-NoCs configuration parameters on the MPSoC security;
- Optimization of the previous two PP-NoCs attacks: *Direct-Interference* and *Back-Pressure*; and
- Design, implementation and evaluation of three new protection techniques for PP-NoCs: i) RT-Blinding; ii) RT-Masking and; iii) RT-Shielding.

This paper is organized as follows. Chapter summarizes existing NoC attacks in literature. Then, chapter 3 describes the Priority-Preemptive Network-on-Chip concept, as well as, our hardware implementation of it. In chapter 4, we show how to exploit the analytical models of PP-NoCs to elucidate NoC vulnerabilities. Thereafter, two attacks are described in chapter 5. Then, security countermeasures and their impact are presented in chapter 6. Section 7 presents the experiments and evaluation results. From the results a discussion is made in chapter 8. Finally, conclusions are drawn in Section 9.

## 2 Related Works in NoC Attacks and Protections

In this chapter, we describe the previous works that deal with the NoC vulnerability exploration and security integration. The NoC-timing attack was the first attack against NoCs mentioned in scientific articles [8, 10]. These works described for the first time the exploitation of NoC covert channels. A malicious agent (attacker) inside the MPSoC can elicit channel leakage by the evaluation of the throughput of the own injected packets (attacker packets). As routers are shared, different packets must compete for the resources when they are being communicated simultaneously. The communication collisions between the attacker packets and sensitive traffic cause latency perturbations. Thus, affecting the attacker communication throughput. This effect is shown in Figure 1, where the attacker traffic is represented as $\lambda_O$ while victim traffic as $\lambda_V$. As a result, this congestion reveals the sensitive traffic information of the victim. Examples of characteristics that can be extracted by this attack are mapping, topology, routing, communication pattern and volume of communication. The collisions are sensed by the attacker due to the reduction of throughput to inject new packets in the network.

The first demonstration of NoC timing attacks was presented in [12]. In this work, the authors show the effectiveness of the classical NoC timing attack and of a powerful variation: Distributed NoC Timing Attack (DNTA). It uses two or more attackers inside an MPSoC to better tune the MPSoC congestion and thus to maximize the attacker observation capabilities. It was demonstrated that DNTA was immune to the NoC countermeasures proposed in [10] and [15]. In order to avoid NoC timing attacks and DNTA, the authors proposed Gossip NoC, a security enhanced NoC able to identify traffic anomalies and avoiding attacks through the on-chip traffic distribution. Each NoC router included a

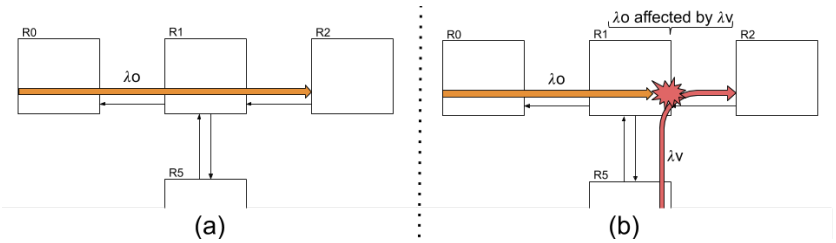



Fig. 1: Attack scenarios: (a) Attacker traffic $\lambda_O$ only; (b) Attacker traffic $\lambda_O$ blocked by victim traffic $\lambda_V$.

monitor to detect possible points of attack. This information was used to adapt the routing of packets. The attack was executed in an FPGA MPSoC prototype. Similar approach was used in [16]. Despite the effectiveness of the detection and protection mechanisms, these secure NoCs cannot meet hard communication deadlines and only offer best-effort communication services.

Recently, the NoC timing attack has been used to enhance the capabilities of dangerous cache attacks, creating extremely powerful attacks such as the NoC Prime+Probe [17] and Earthquake [11]. In [17], the authors propose the NoC Prime+Probe attack which uses the NoC timing attack to extract information regarding the communication behavior between a cryptographic IP core and a shared cache, a usual configuration of secure MPSoCs. By using the NoC timing attack, it is possible for the attacker to detect, in a non-intrusive way, the optimal point in time to probe the shared cache [3]. For a system that implements the AES (Advanced Encryption Standard) symmetric cryptography based on a transformation table (T-Table) implementation, the best attack point (increases the efficiency of the attack) to probe the shared cache is at the end of the first AES round. By using the NoC Prime+Probe, the authors retrieved 12 of the 16 key bytes after 80 AES encryptions.

In the Earthquake attack of [11], the authors use the NoC timing attack to collect the time where the third round of AES starts. Earthquake manipulates the input of the cryptographic IP core to force several cache collisions (i.e., cache hits) until the third round of AES, thus causing faster encryptions/decryptions. The faster results can be used to break the key. Since the important timing information resides within the first three rounds, the NoC timing attack allows the attacker to sample effectively (less noise) the time. This work presented the first practical implementation of timing attacks.

Although NoC timing attack has been studied in different NoC configurations and scenarios, attacks to real-time NoCs, specifically Priority-Preemptive NoCs, have been not widely explored yet. For the best of our knowledge, the only works that address attacks to real-time NoCs are [19] and [14]. The work of Indrusiak et al. [19] showed the impact of the NoC routing in the security of the

[3] Following the classical cache attack, Prime+Probe from Osvik et al. [18], the best moment to probe a cache is when all the accesses to the cache depends only on the value of the secret key used for encryption.

system. As a protection mechanism, the authors proposed a packet route randomization mechanism to increase NoC resilience against side-channel attacks. The route randomization was based on an evolutionary optimization approach for effectively controlling the impact on hard real-time performance guarantees. In our previous work in [14], we have demonstrated for the first time an attack that exploits preemptive NoC-based MPSoCs. Also, two countermeasures were proposed.

The protection against NoC timing attacks has been addressed in the works of [20] [21] [22] [10] and [15]. The works of Yao and Suh [20] and Wassel et al. [21] proposed the integration of hard Quality-of-Service (QoS) mechanisms to isolate the sensitive information. They included temporal network partitioning based on different priorities arbitration: high priority [20] and bounded priority [21]. The work of Sepúlveda et al. [22] presented a secure enhanced router architecture that dynamically configures the router memory space according to the communication and security properties of the traffic. Furthermore, the work of Sepúlveda et al. [10] proposed random arbitration and adaptive routing as protection techniques. The work of Stefan and Goossens [15] introduced the usage of multiple path communication for sensitive flows. The work of Reinbrecht et al. [12] showed the Gossip NoC, an NoC architecture with a distributed protection mechanism that changes the routing algorithm in the presence of abnormal traffic.

## 3 Priority Preemptive NoCs

Priority Preemptive NoCs (PP-NoCs) allow that high priority communication flows preempt low priority packets on the NoC router. Thus, higher priority packets are preferentially communicated while lower priority packets remain stored inside the router buffers. High priority traffic is assumed to be either periodic or sporadic, as to avoid starvation of low priority packets due to continuous high priority interference. This chapter presents the target MPSoC platform (architectural scenario) used to demonstrate the attacks and the security of the countermeasures. The on-chip communication structure is a parameterizable PP-NoC. The architecture overview and functionality are further detailed.

### 3.1 Target MPSoC Platform

The MPSoC platform allows performing the practical study of the PP-NoCs vulnerabilities. In such environment, the proposed PP-NoCs attacks and countermeasures (Sections 5 and 6, respectively) can be evaluated. In this work, we use the Glass MPSoC, a parameterizable hardware platform presented in [11] and which has been already used to evaluate logical side-channel attacks. To evaluate the PP-NoC vulnerabilities, the Glass NoC was modified to support the priority-preemptive flow control.

Glass MPSoC is presented in Figure 2. It integrates 16 tiles (from $IP_0$ to $IP_15$) through $4 \times 4$ mesh-based PP-NoC. It supports several layers of memory hierarchy. The MPSoC tiles include an inclusive shared cache (64KB, 16-way

set-associative cache L2) at $IP_0$, serial UART interface at $IP_3$ and 14 RISC-V processing elements. We used a verified and functional RISC-V distribution called RI5CY core, from the Pulpino Platform [23]. Besides the processor, each of the 14 processing tiles integrates local instruction and data memories (8KB, direct-mapped cache L1), a cycle-accurate timer and a network interface to communicate with the NoC. The NoC routers implement credit-based flow control with four virtual channels (each one at a different priority level) and Priority-Preemptive routers.

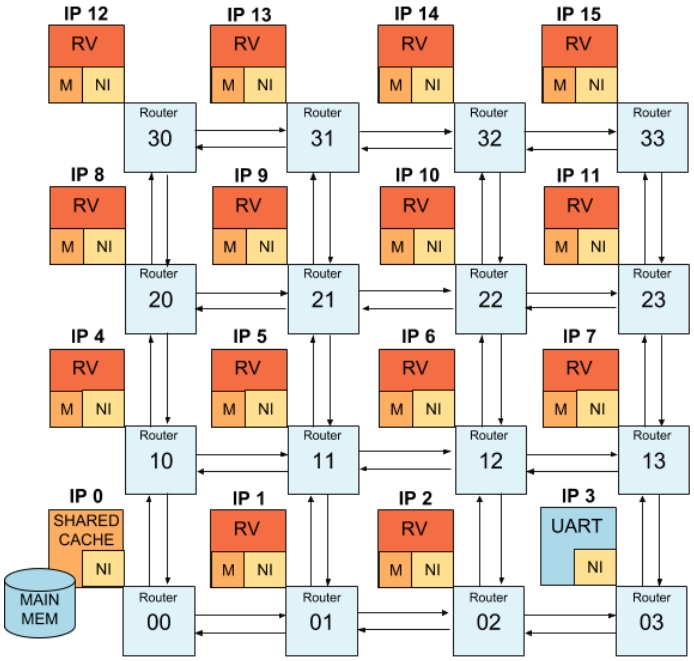



Fig. 2: Glass-V MPSoC Platform

### 3.2 Priority-Preemptive (PP) NoC Architecture and Functionality

To guarantee different real-time and mixed-critically requirements, communication on-chip is prioritized. That is, communication flows that are characterized by tight delay requirements are granted the highest communication priority. A preemptive policy allows a higher priority packet to anticipate (preempt) an already progressing lower-priority packet. To support the preemptive communication technique, routers should be enhanced through the integration of virtual channels. These additional structures are capable of storing a packet blocked during its communication. When packets with the same priority-level dispute a communication resource (collision), any arbitrary decision algorithm can be applied, such as Round-Robin [3] or aging [5].

The structure of the packets of the PP-NoC follows the typical packet organization. It includes a header flit, $N$ payload flits and a trailer flit, where $N$ can

reach a maximum of 1024 flits per packet. Besides these flits, nine control bits are present in the links: header/trailer identification (1 bit), packet priority (2 bits), handshake (2 bits) and credit information (4 bits). A detailed description of the router is given in the following paragraphs.

**Priority-Preemptive (PP) Router:** The proposed PP-router is defined to support four different priority levels. It uses credit-based virtual channels to handle different priority messages simultaneously (each virtual channel is a different priority lane). Since the preemption of low priority packets by higher priorities is allowed, this router supports packet interleaving. The PP-router integrates six components as shown in Figure 3:

1. *Priority multiplexer:* It is responsible for monitoring the packet priority and selecting the proper input buffer.
2. *Routing computing unit:* It defines the output port of each packet.
3. *Virtual channel (PP) allocator:* It selects the active priority level at the output port.
4. *Switch allocator:* It is a unit included for each priority level. It is used to attend each request (of the same priority) using a Round-Robin arbiter to cope with resource conflicts.
5. *Crossbar:* Also included for each priority level. It is used to connect the defined input and output of the router.
6. *Virtual channel demultiplexer:* It links the active priority crossbar with the output of the router.

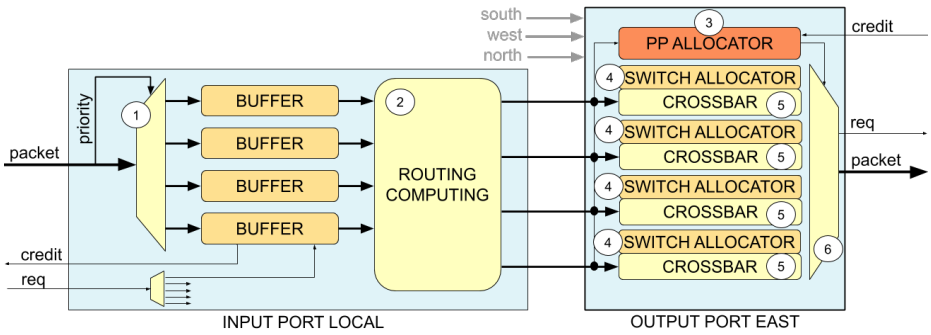



Fig. 3: Priority-Preemptive Router Architecture

When a packet arrives at the input port, the priority information in the control bits defines which input buffer will be used. This information is used for referring the packet to the proper virtual channel. The buffers request the proper output port for each packet being handled, where the virtual channel allocator (VA Allocator) at the output port decides which virtual channel will be granted by the crossbar. Hence, VA Allocator is responsible for providing the preemption feature. The output port always chooses the higher priority packet to perform the data commutation. When different input ports dispute the same

output port and the level of priority is the same, the Round-Robin arbitration takes place. It is implemented in the Switch Allocator of Figure 3.

## 4 Exploiting Priority-Preemptive Models for Security

In this chapter, we show how the priority-preemptive models can be used to evaluate vulnerabilities of the Preemptive Network-on-Chip (PP-NoC). First, we present the existing models in the literature that predicts in detail the traffic behavior. Thereafter, we use the *IBN* model to explore the PP-NoC vulnerabilities to NoC timing attacks. Also, we explore the impact that the PP-NoC parameters have on the overall system security.

### 4.1 Priority-Preemptive Models

Scheduling of real-time systems requires the calculation of upper-bounds for packet transmission delay. To evaluate if the system can meet the application deadlines, analytic models can be used. Although these models were elaborated to support designers to validate the real-time constraints, in this work we use them as an important tool to evaluate vulnerabilities and elaborate NoC timing attacks.

The Priority-Preemptive models allow estimating the worst-case latency for different flows of packets in the NoC. We define flow $i$ as $\lambda_i$, where the flow represents the first flit leaving the origin node until the last flit arrives at the destination node. In the sequence, three models developed in previous works are here described: i) SB model; ii) XLWX model; and iii) IBN model.

***SB Model*** The work of [6] presents the *SB model* to predict packet network latency. It is based on direct and indirect interference from other traffic flows and calculates the upper-bound interference suffered by a communication flow. When no flow interferes with $\lambda_i$, the worst-case latency is given by the flow's zero-load latency ($C_i$), given by Equation (1):

$$C_i = RouteL * (route_i - 1) + LinkL * (route_i) + LinkL * (L_i - 1) \quad (1)$$

Where $RouteL$ is the router latency in cycles, $route_i$ is the contention domain of $\lambda_i$ in hops, $LinkL$ is the link latency in cycles and $L_i$ is the number of flits in each packet of $\lambda_i$. The direct interference presented in this model can be seen in Equation (2). The worst-case response $R_i$ is quantified by the summation of two components: the flow's zero-load latency ($C_i$) and the worst possible delays resulting from blocking and preemption caused by higher priority packets.

$$R_i = C_i + \sum_{\lambda_j \in S_i^D} \left\lceil \frac{R_i + J_j}{T_j} \right\rceil C_j \quad (2)$$

The worst-case delay is the summation of the effects of all interfering flows on the contention domain ($cd_{ij}$), defined as the interference Set $S_i^D$, which is the path $\lambda_i$ intersects with the interfering higher priority $\lambda_j$. This equation is a recurrent ceiling function that depends on the relation between release jitter ($J$) in cycles, the period of the flow ($T$) in cycles, and the flow latency ($R$), also in cycles. The equation is calculated until the result converges.

***XLWX Model:*** Xiong et al. [24] presented the *XLWX model*. The work extended the *SB model* to support Multi-point Progressive Blocking (MPB) for downstream indirect interferences. MPB takes place when a flow $\lambda_i$ is preempted by a flow $\lambda_j$ by more than its base latency $C_j$. This scenario usually occurs when a third flow $\lambda_k$ interferes with $\lambda_j$ downstream from the link that $\lambda_j$ interferes with $\lambda_i$. The model can be represented by Equation 3.

$$R_i = C_i + \sum_{\lambda_j \in S_i^D} \left\lceil \frac{R_i + J_j + J_j^I}{T_j} \right\rceil (C_j + I_{ji}^{down}) \tag{3}$$

Where $J_j^I = R_j - C_j$ is used to calculate the effects of upstream interference of $\lambda_j$, $I_{ji}^{down}$ is the number of hits suffered by $\lambda_j$ from every $\lambda_k$ in the downstream indirect interference Set of $\lambda_i$, given by Equation (4):

$$I_{ji}^{down} = \sum_{\lambda_k \in S_{I_i}^{down_j}} \left\lceil \frac{R_j + J_k}{T_k} \right\rceil (bi_{ij}) \tag{4}$$

Where $bi_{ij} = buf \cdot LinkL \cdot |cd_{ij}|$ is the maximum buffered interference over the contention domain $cd_{ij}$, $R_j$ is the worst-case latency experienced by $\lambda_j$, $buf$ is the routers FIFO buffer size, and $T_k$ is the release period of packets for $\lambda_k$.

***IBN Model:*** The work of Indrusiak et al. [7] proves that the analysis proposed by Xiong et al. [24] for downstream indirect interference is overly pessimistic since it treats all interferences as direct interference. The authors improve the *XLWX model* by presenting an upper-bound analysis. In order to find $R_i$, two cases are considered when calculating the upper-bound for downstream interference $I_{ij}^{down}$. First, when interference is caused by flows that do not suffer from upstream and downstream interference. Second, when interference is caused by flows that do suffer from upstream interference. The first case is described by Equation 5. It includes the effects of the maximum buffered interference ($bi_{ij}$) and the high priority flows downstream. The latter case is id$cd_{ij}$entical to the analysis proposed by Xiong et al. [24].

$$I_{ji}^{down} = \sum_{\lambda_k \in S_{I_i}^{down_j}} \left\lceil \frac{R_j + J_k}{T_k} \right\rceil min(bi_{ij}, C_k + I_{kj}^{down}) \tag{5}$$

Even though these models give guarantees regarding the system's ability to meet the deadlines, the predictability of the system allows an attacker to take

advantage of the additional information about the system behavior, especially packet delay times. This data could be used to create a refined NoC timing attack (exploiting thresholds), allowing an attacker to skip the costly attack setup and tune phases as showed in [12]. These phases usually are dedicated to monitor the throughput in order to find a value that allows an attacker to have a communication sensibility such that sensitive packets are efficiently and effectively detected (attack threshold). The details regarding the estimation of the latency experienced by preempted packets will be explored in the next section 4.2. We show that it is possible for an attacker to further extract information of the sensitive flow (victim's flow), such as packet sizes. We refer to the improved version of the *XLWX model* as the *IBN* model, and it will be used for the rest of this paper.

### 4.2 PP-NoC Vulnerabilities

The analytical model used to predict the behavior of the PP-NoCs can reveal information regarding the system operation that can be exploited by an attacker. As described in [14], two vulnerabilities can be found at PP-NoCs: *Direct-interference* and *Back-pressure*.

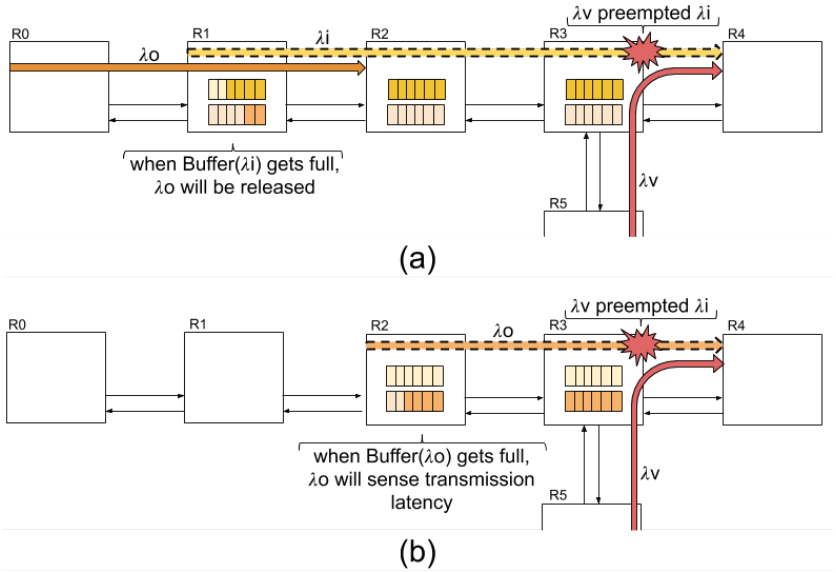



Fig. 4: Flow representation of the discussed vulnerabilities: (a) Back-Pressure (b) Interference

***Direct-Interference Vulnerability*** It takes place when two flows with different priorities dispute the same output port, such as in Figure 4.b. The flow $\lambda_o$ is preempted by a higher priority flow $\lambda_v$ and a contention occurs at router $R_3$. Direct measurements of transmission latency times can be used to retrieve

predicted access times of secure flows. In this case, only direct interference was considered. We call the preemption caused by high priority messages as interference, and since high priority messages can occur all over the system, we consider this feature as a vulnerability only when the attacker and the victim can be placed close enough to avoid external interference (caused by a third IP core communication flow). In the case of an attacker be placed distantly from victim traffic, the interference behavior from higher priority packets can be used as a protection technique, since it becomes a source of noise in the attacker measurements. As a result, a high amount of false-positive sensitive packet detection is caused to the attacker.

***Back-Pressure Vulnerability*** This vulnerability is based on the MPB scenario. It considers three communication flows $\lambda_v$, $\lambda_i$ and $\lambda_o$, originating from routers $R_5$, $R_1$ and $R_0$, ordered from highest priority to lowest, respectively. In this scenario, represented in Figure 4.a, the flow $\lambda_i$ preempts all packets from $\lambda_o$. However, when the high priority flow $\lambda_v$ is injected, a contention occurs at $R_3$ and $\lambda_i$ is preempted. Then, packets start to accumulate on the upstream routers from $R_3$ until all buffer space in the upstream path is used. As a result, all of the buffer credits from $\lambda_i$ are expended, allowing $\lambda_o$ to take over the transmission in the path. Based on Equation (3), it is noticed that when $\lambda_v$ stops transmitting, $\lambda_o$ gets preempted by more than $\lambda_i$ baseline latency, due to the accumulation in the$cd_{ij}$ buffers.

### 4.3 Exploiting PP-NoC through IBN Model

To evaluate the vulnerabilities of the PP-NoC, the *IBN model* can be used. In this case, the impact of each PP-NoC parameter of the IBN model on the security of the system is explored. The multiple parameters described by the equations in the previous sections can be classified into three different categories: i) Network Interface (jitter); ii) Router (Buffer size); and iii) Application parameters (transmission period and packet size). As described in subsection 4.1, each of the analyzed flows ($\lambda_v$, $\lambda_o$, and $\lambda_i$) has a set of proprieties as: zero-load latency ($C$) in cycles, worst-case latency ($R$) in cycles, and packet period ($T$) in cycles. Also, the network itself has parameters that influence the latency of packets, such as router buffer-size in ($buf$) flits, packet release jitter ($J$) in cycles, link latency ($LinkL$) in cycles, and the contention domain $cd_{ij}$ (e.g., the routers where two flows intersect). The range of values explored for each one of the parameters of the PP-NoC is shown in Table (1).

***Direct Interference Evaluation*** Each PP-NoC configuration is obtained after defining the values of the different PP-NoC parameters. The previously discussed IBN model can be used to further understand the behavior of the system. The Equation (2) can be used with the corresponding PP-NoC values. Initially, an oscillation of the resulting value $R_i$ is observed and by using recursively this mathematical expression, until a stable value is reached. However, for some PP-NoC configurations the stable value is never reached (the equation diverging to

Table 1: Selected Parameters for Network Interface, Router and Application.

| Parameters | Values | | | | | |
|---|---|---|---|---|---|---|
| Router Buffer Size (Flits) ($LinkL$) | 4 | 8 | 16 | 32 | 64 | |
| Packet Transmission Period (% of 1000 cycles) | 10% | 20% | 30% | 40% | 50% | |
| Release jitter (Cycles) | 4 | 8 | 16 | 32 | 64 | 128 |
| Flow Packet Size (Flits) | 4 | 8 | 16 | 32 | 64 | 128 |

infinity). These scenarios lead to stalls and they are not considered nor evaluated in this paper. Table (2) presents 10 different PP-NoC configurations considered as representative candidates of all possibilities evaluated.

Table 2: Different configurations of parameters for Direct Interference.

| Parameter | Config 1 | Config 2 | Config 3 | Config 4 | Config 5 | Config 6 | Config 7 | Config 8 | Config 9 | Config 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Jitter (Cycles) | 4 | 4 | 4 | 4 | 8 | 16 | 32 | 128 | 64 | 64 |
| $\lambda_v$ Period (Cycles) | 200 | 200 | 500 | 300 | 200 | 200 | 200 | 500 | 500 | 200 |
| $\lambda_o$ Size (Flits) | 4 | 128 | 128 | 128 | 128 | 16 | 64 | 128 | 64 | 64 |

The Direct-interference vulnerability may be used to exploit the PP-NoC as a cover channel. By manipulating specific communication flows, the information of a victim flow can leak. In the Direct-Interference case, the victim $v$ is a high priority flow that will directly affect the low-priority attacker $o$. Figure (5) presents the impact of the victim's packet size on the attacker's flow delay. Results show that some PP-NoC configurations exhibit a greater latency sensibility when the packet size increase. This evaluation has not been presented before and the results allow the designer to identify potential leakages in PP-NoCs. Consequently, there are configurations that mitigate the observation of a potential attack. The strategy uses this benefit to protect the MPSoC will be further discussed and presented in Section 6.

Figures 6(a) and 6(b) present the difference in latencies for all considered configurations. Each area represents the delta for subsequent packet sizes, meaning that there is an expressive difference in terms of the latency when the packet size is increased. Some configurations (e.g., Dif 16-32) raise this difference even further (an increase of almost 400%). When the difference is large enough, the attacker is able to easily differentiate the victim‘s packet size.

***Back-Pressure Vulnerability Evaluation*** In order to evaluate the Back-Pressure vulnerability (or indirect interference), Equations (3) and (5) must be used. These calculations involve a wide variety of variables. As a consequence, the exploration space is wider, that is, there are a wide variety of PP-NoC configurations. In this paper, a representative set of this design space was selected. Table 3 presents the 10 different PP-NoC configurations selected for the study.

Figure 7 shows that the victim's packet size does not produce a linear influence for all configurations of the PP-NoCs. For some PP-NoC configurations, such as *Config 2* and *Config 5*, the packet size does not influence at all on the latency experienced by the observer. In contrast, for some PP-NoC configurations,

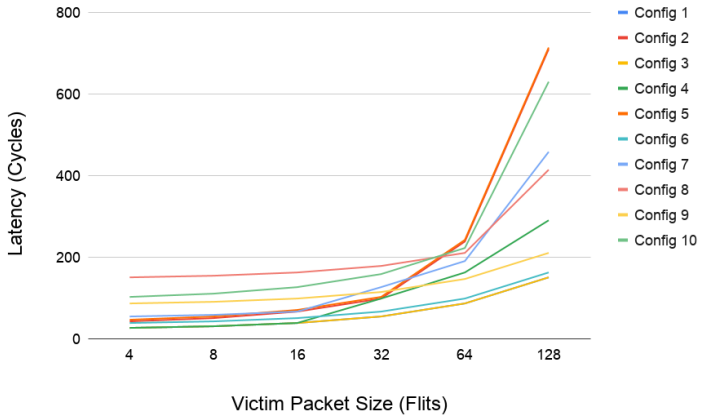
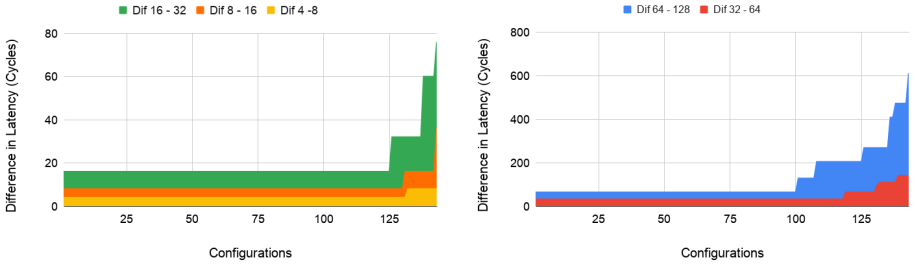



Fig. 5: Interfering Packet Size effect in Direct Interference latency.




(a) 4, 8, 16 and 32 flits (b) 32, 64 and 128 flits

Fig. 6: Value of the difference in latency between packet sizes of interfering flow

Table 3: Different configurations of parameters for Back-Pressure.

| Parameter | Config 1 | Config 2 | Config 3 | Config 4 | Config 5 |
|---|---|---|---|---|---|
| Injector Packet Size (Flits) | 4 | 4 | 8 | 8 | 8 |
| Observer Packet Size (Flits) | 4 | 4 | 4 | 8 | 32 |
| Injector Period (Cycles) | 100 | 300 | 100 | 300 | 500 |
| Victim Period (Cycles) | 100 | 300 | 200 | 500 | 300 |
| Jitter (Cycles) | 4 | 8 | 8 | 64 | 8 |
| Buffer Size (Flits) | 4 | 16 | 4 | 64 | 4 |
| | | | | | |
| Parameter | Config 6 | Config 7 | Config 8 | Config 9 | Config 10 |
| Injector Packet Size (Flits) | 16 | 16 | 32 | 32 | 128 |
| Observer Packet Size (Flits) | 8 | 32 | 64 | 128 | 32 |
| Injector Period (Cycles) | 500 | 300 | 200 | 500 | 300 |
| Victim Period (Cycles) | 200 | 300 | 100 | 100 | 500 |
| Jitter (Cycles) | 32 | 64 | 16 | 4 | 4 |
| Buffer Size (Flits) | 32 | 32 | 8 | 32 | 8 |

such as *Config 8* and *Config 9*, the packet latency is not easily predictable. As

a conclusion, for some PP-NoC configurations, the Back-Pressure attack could not yield information about packet sizes, even if it could detect them.

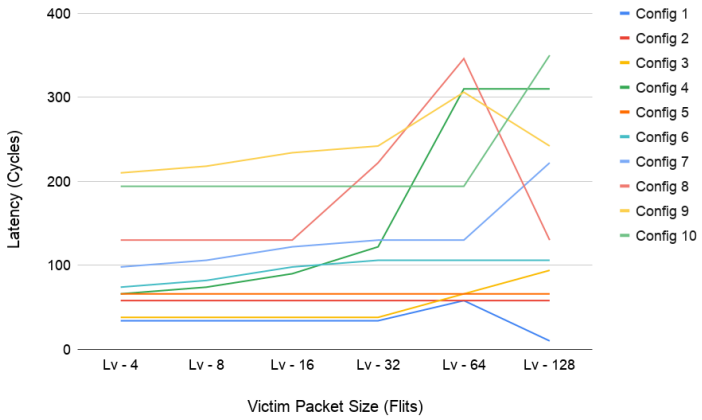



Fig. 7: Interfering Packet Size effect in Indirect Interference latency.

## 5 Proposed Attacks

This section presents two attacks based on the vulnerabilities already presented.

### 5.1 Direct-Interference Attack

This attack explores the interference vulnerability and it is performed in three phases. In the first phase, the latency upper and lower-bounds are calculated based on the different parameters (PP-NoC configuration and attacker's flow). This generates a range of values of the expected latency for the observer's packets being preempted. This information can be used to increase the precision of the NoC timing attacks. In the second phase, an IP core is infected (through a malicious software or the trigger of a Trojan), in order to create the $\lambda_o$ flow from the attacker through the secure flow's path (see Figure 4(b)). The Direct-Interference attack requires a close engagement of the attacker on the secure traffic observation. The attack applies measurement of the interference from the secure flow close to the target path while trying to avoid at maximum interference from non-victim flows (also called indirect interference). The third and final phase employs a mathematical algorithm to correlate the timing results collected by the attacker's monitor to infer an unknown key or private information. From this point, the attacks presented in [25] or [26] can be performed to retrieve secret information.

**Attack Conditions:** In order to execute this attack on a PP-NoC, the following conditions are required:

- The attacker can infect at least one IP in the MPSoC with malicious software or any other infection technique.
- This IP has to be, at most, one-hop away from the secure flow.
- The attacker knows where the target elements are located in the NoC (logical addresses).
- The attacker knows the topology and routing algorithm of the communication infrastructure.

**Attack Optimization:** This attack can be optimized in a way that can retrieve not only the victim's traffic pattern but also the size of the messages exchanged. The size of the packets provide important information that can be correlated with the sensitive application running. For example, it can reveal if the AES cryptography uses T-table (32-bit information) or S-Box (8-bit information) implementation. In addition, it may reveal the bit width of the keys distributed in the system. Finally, by knowing the granularity of the information and the moments of data transfers, more sophisticated and efficient attacks can be elaborated.

In order to achieve this, modifications are performed in the phases of the Direct-Interference attack. In the first phase, a range of values for $R_o$ and $C_o$ is calculated, based on the jitter, attacker packet size, and possible victim period and packet size. The attacker will then monitor the latency of it's packets and filter configurations that lead to an $R_o$ value smaller than the one measured. For example, assuming a NoC with a Jitter of 4 cycles, if the attacker chooses a large packet size (e.g., 128 flits), there will be a discernible difference in $R_o$ for victim packet sizes (e.g., 710 cycles for 128 flits, 98 cycles for 32 flits, and 42 cycles for 4 flits), as it can be seen in *Config 2* in Figure (5). Therefore, an attacker can distinguish with less effort the size of victim's packet.

### 5.2 Back-Pressure Attack

This attack explores back-pressure vulnerability. It uses the same first and third phases of the Direct-Interference attack. In contrast with the previous attack, the second phase of the Back-Pressure attack requires the infection of two IPs: an injector IP and an observer IP. The injector IP is responsible for creating traffic interfering with the observer IP, $\lambda_i$ flow as shown in Figure 4(a). The $\lambda_i$ flow intends to accumulate back pressure until the buffers are filled. When the priority flow (victim flow), $\lambda_v$, preempt the injector, the observer flow, $\lambda_o$, will be released to proceed. Therefore, the observer understands that the secure flow has been communicated through the increase of its transmission throughput. Besides, the observer can use Equation (3) to calculate specific features of the secure flow, such as message size.

The predicted advantages of this type of attack are its ability to infer sensitive information of high priority packets indirectly, while not being necessarily close to the target path of the secure flow. This allows more flexibility for the attacker and expands the range of MPSoC configurations that could be targeted. In the same manner as the previous attacks, from this point, different methodologies can be applied to successfully perform a complete logical Side-Channel attack.

**Attack Conditions:** In order to execute this attack on a PP-NoC, the following conditions are required:

- The attacker can infect at least two IPs in the MPSoC with malicious software.
- The attacker knows where the target elements are located in the NoC (logical addresses).
- The attacker knows the topology and routing algorithm of the communication infrastructure.
- The attacker is able to create low and medium priority messages.

**Attack Optimization:** In the same manner as the previous attack, this attack can be optimized to retrieve other characteristics of the victim's traffic rather than pattern. To retrieve the granularity of the packets, modifications are preformed to the Back-Pressure attack.

Before describing the optimized methodology, note that Back-Pressure vulnerability is affected by much more parameters of the NoC than direct-interference. Hence, it is more difficult to guarantee the attacker knows or can infer all required information. However, once the attacker knows the parameters, the optimized attack is possible. Therefore, the attacker has first to conduct an exploration of the NoC parameters, and then evaluate the best configuration of attacker packet sizes and period to match with the victim's behavior. For example, in an MPSoC with jitter of 64 cycles and buffer size 64 flits, if the attacker applies an injector period of 300 cycles, and packet sizes of 8 flits, the latency experienced by the observer would be 310 cycles for 128 flit packets, 122 cycles for 32 flits, and 66 cycles for 4 flits. As observed, the differences allow the attacker to distinguish the size of the messages.

## 6 Proposed Countermeasures

Our proposed attacks depend heavily on the preemption caused by secure flows. Hence, we propose three main strategies to mitigate the risk of a successful attack on PP-NoCs: a) RT-Blinding, b) RT-Masking and c) RT-Shielding.

### 6.1 RT-Blinding

The blinding strategy relies on the timely delivery of payloads by the secure IP. The Back-Pressure attack identifies high priority flows and assumes they are sensitive flows being exchanged through the PP-NoC. One possible way to avoid detection is to use dummy high priority payloads intentionally. Delivered at predefined intervals, these payloads could be replaced by an actual secure packet when needed. In this scenario, the victim has fixed time slots to send it's high priority secure packages. Since the attacker has no way to differentiate between a secure flow and a simple high priority flow, the attacker would not be able to determine whether it is an actual payload.

This interval would be determined by calculating the maximum acceptable delay that the secure application could endure. This application-dependant value is then used as a baseline for all high priority transmissions. The secure application must calculate it's zero-load latency $C$, with Equation 1. The drawback of this approach is the increase in traffic in the NoC, which, in turn, may compromise the overall system performance. However, since secure flows are intended to be sporadic, it is expected that the average time defined in the time slots will be high enough to avoid performance penalties.

### 6.2 RT-Masking

The masking strategy proposes the saturation of the channels when a *high priority (secure) flow* is passing. Our proposed protection technique, nicknamed as Distraction, employs several high priority dummy packages sent prior and after the actual secure package to pad the channel. In this scenario, the secure flow is extended to have a random number of dummy packets sent with the secure packet. This would effectively mask the actual timestamp of the packet and message size, as each secure package would be sent in a random position of this enlarged secure flow. When the high priority flows are replaced by *normal priority flows*, our proposed protection technique, nicknamed as Avoidance, can be triggered. It masks the secure flow with the saturation of the normal priority channels.

An IP, defined at design time, would be responsible for generating a random series of large normal priority packets directed to the sensitive path of the target IP. The goal of this technique is to generate interference with the normal priority attacker packets which will be defined by the round-robin arbiter. Thus, the secure packets could be sent to the defined router, where they would preempt both the attacker's traffic and the companion normal priority buffered traffic.

### 6.3 RT-Shielding

Both of the previous countermeasures could be employed at the application level or with little modification to the design. The RT-Shielding strategy uses instead the models, described in section 4.2, to build a NoC that makes the distinction of victim traffic by the attacker more arduous. It accomplishes this by setting the routers or the network interface with a specific set of parameters (i.e., buffer size, jitter). In this manner, the attacker would not be able to attack since by definition of the models certain configurations do not present interference in low priority packets. Furthermore, if the most security-critical paths in the system can be established at design time, only few routers and network interfaces require adjust, reducing the overall costs of choosing this strategy.

The RT-Shielding strategy relies on the redesign of NoC parameters, such as jitter and buffer sizes to prevent any possible interference to be used by an attacker. The direct interference equations only use jitter parameter, so an exhaustive analysis for different configurations was elaborated, where 17 of these different configurations can be seen in Figure (8). It is clear on Figure 8 that jitter

has an effect on Latency at higher values, while maintaining a more constant behavior at lower settings. Hence, to avoid direct interference in the system, the IPs considered as victim in the system should have network interfaces with fast injection times (reduced jitter). In practice, this means to design a network interface with a small size of buffers, as they increase the delay to inject a packet in the network. Based on our results, we consider any value below 32 cycles as secure.

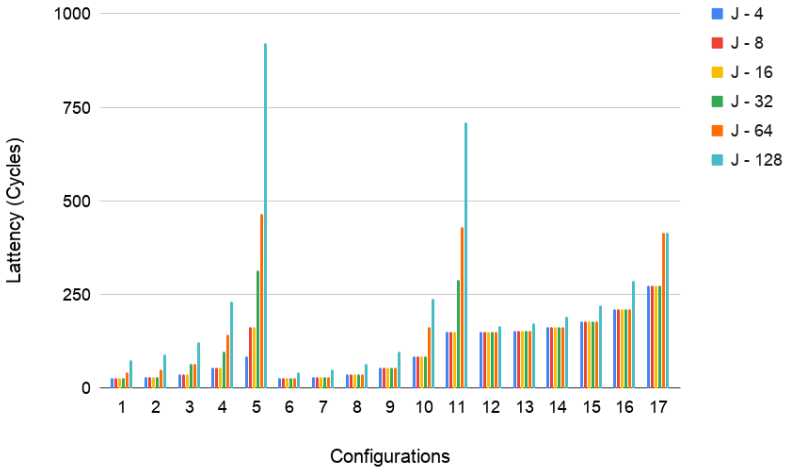



Fig. 8: Effect of jitter in different configurations.

For indirect interference, buffer sizes of the routers and the jitter have to be taken into account. The impact of these parameters was evaluated by calculating all possible configurations of Router, Network Interface, and applications. As with direct interference, some configurations resulted in infinite latencies, so these were filtered, only leaving the viable sets. The plot of all these configurations can be seen in Figure 9. On the other hand, in Figure 9, it is possible to see the impact of the buffer size of the routers by itself, and how this size emphasizes other variables, creating new spikes of latency. This effect is probably caused by buffered interference, as the accumulated flow $\lambda_i$ will take longer to dissipate. In the case of the indirect interference vulnerability, large buffer sizes between the observer and the injector are detrimental to the attack, as there is plenty of time for victim behavior to be obfuscated. Therefore, for a secure NoC, Buffer Sizes outside critical paths should aim for higher values.

## 7 Experiments and Results

This section presents the setup of all experiments, the results of the efficiency of the attacks, and the decrease of attack efficiency under countermeasures.

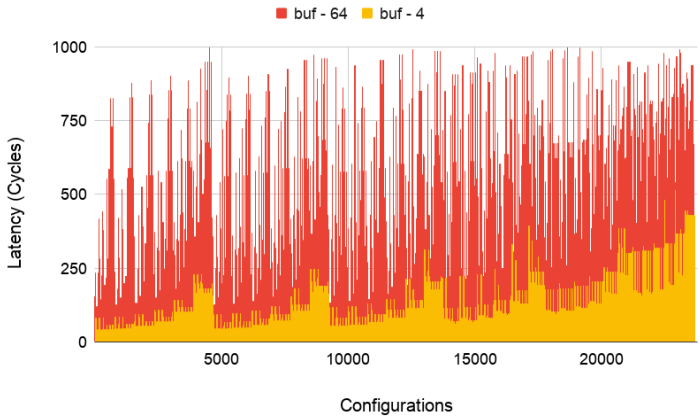



Fig. 9: Latencies for all viable configurations, when comparing 4 Flit buffer and 64 Flit buffer.

### 7.1 Setup of Experiments

In this subsection, the scenarios considered and metrics used to evaluate the attack's efficiency are presented. All experiments were executed through RTL MPSoC Glass simulations. The Routers were configured to have a buffer size of 32 flits and a jitter of 64 cycles.

***Scenarios*** Two scenarios were elaborated and mapped into the target platform. The first scenario (*Scenario 1*) uses $IP_{13}$ as a trusted RISC-V that provides encryption services. This processor runs the AES-128 encryption as a T-Table implementation [13], whose algorithm uses huge tables that must be accessed in the shared cache located at $IP_0$. In the second scenario (*Scenario 2*), $IP_1$ and $IP_{13}$ exchange messages to perform the Diffie-Hellmann (DH) key establishment protocol. DH protocol requires intense communication, therefore, three message sizes (packet granularity) were evaluated in our experiments: 32 flits (KEY32), 64 flits (KEY64), and 128 flits (KEY128). The following paragraphs describe the preparation and execution of the attacks.

***Evaluation Metrics*** To evaluate the effectiveness of the attacks, three metrics are defined:

- *False Positives* (FP), which measures the percentage of wrong guesses among all guesses of the attacker. Note that these guesses represent the victim's traffic occurrence time
- *Observation Efficiency* (OBS), which refers to the percentage of correctly guessed (observed) by the attacker of the total victim's traffic occurrence times
- *Attack Efficiency*, that relates the FP and OBS. It is described by Equation (6). Note that the FP shows the quality of the information collected and OBS shows the sensitivity of the attack observation

$$Efficiency = Obs * (100\% - FP) \tag{6}$$

***Direct-Interference Attack Execution*** To perform the Direct-Interference attack in the *Scenario 1* the three phases are executed. During the first phase, $IP_5$ is infected and the desired flow configuration (packet size and injection rate) is calculated so to maximize the attacker‘s observation. We desire to calculate the attacker latency $R_a$ for direct interference, therefore we use Equations (1) and (2). The system parameters needed by the attacker are all queryable by the attacker in the infected IP.

During the second phase, $IP_5$ starts to inject a high volume of small packets (4 flits) to the $IP_9$ (at low priority). Meanwhile, the encryption runs at $IP_{13}$. When a cache miss occurs at the local L1 memory of $IP_{13}$, a request to the L2 shared cache is performed through the NoC. The shared cache L2 responds to $IP_{13}$ request with high priority flow. Such flow preempts the attacker's flow at $IP_5$ at the $Router11$.

For *Scenario 2*, the key exchange application, the same behavior of preemption takes place. $IP_1$ and $IP_{13}$ exchange a high volume of high priority messages, which are intersected by the attacker's flow at $Router11$. The attacker records all latencies of the transmission. Any increase in latency above the calculated threshold $C_a$ is marked as a sensitive traffic point.

***Back-Pressure Attack Execution*** The execution of the Back-pressure attack at *Scenario 1* is performed in three phases. First, $IP_6$ and $IP_7$ are infected, becoming the observer (low-priority) and the injector (medium-priority), respectively. Then, the attacker calculates a range of expected latency values for the defined attacker flow.

In contrast with the Direct-Interference attack, Equations (3) and (5) are used to calculate the indirect interference from the injector. In the second phase, the planned flow is executed. Injector IP generates large packets (128 flits) addressed to the $IP_9$. Meanwhile, the observer sends data to $IP_5$ (4 flits). The encryption is performed at $IP_{13}$, provoking cache misses. As a result of the cache hierarchy handling, the shared cache responds to the data requests with high priority flows. This traffic preempts the flow from $IP_6$ at the $Router11$. Thus, $IP_6$ flow gets buffered on the route. Hence, the flow of $IP_7$ is now free to transmit its packets downstream.

For *Scenario 2*, the cores exchanging key information are $IP_1$ and $IP_{13}$. Since this operation also uses high priority messages, the injected packets of the attacker at $IP_6$ are also preempted at $Router11$. This condition releases observer traffic. Any delay of the observer latency at the execution stage, based on the threshold found in phase 1, is considered as the identification of a sensitive packet. For our experiments, the injector packet's were sized so to guarantee that any interference always generates maximum back-pressure.

### 7.2 Evaluation of Attacks

Table 4 shows the results for the Direct-Interference attack, in which the attacker acts as the injector and observer. Small packets are not able to fill the buffer space in the route fast enough for contention to occur. On the other hand, bigger packets take longer time to be generated, and therefore, sensitive information can be lost in the meantime. Especially, when dealing with smaller packets as in an AES execution, where the victim received packages of 16 flits from the shared memory. The results for the Back-Pressure attack show an overall lower detection capacity in comparison with the Direct-Interference attack. However, these strategies were able to correctly detect smaller packets, such as the packets generated by the AES encryption. The main reasoning behind this is that since the observer is constantly starved, the liberation of its flow is almost instantaneous, while the smaller packets guarantee a faster observation in the increase of the throughput. This occurs even in the case where the injector flow is preempted for a very small period. We can also observe that in general, bigger packets are easier to detect in the NoC. As in the cases of key exchanges with larger granularity.

Table 4: Evaluation results of the attacks under unprotected MPSoC in two scenarios.

| | No Protection | | | | | |
|---|---|---|---|---|---|---|
| | DI Attack | | | BP Attack | | |
| | FP | Obs. | Efficiency | FP | Obs. | Efficiency |
| Scenario 1 - AES | 0% | 71.29% | 71.29% | 0% | 72.07% | 72.07% |
| Scenario 2 - KEY32 | 0% | 93.53% | 93.53% | 0% | 80.43% | 80.43% |
| Scenario 2 - KEY64 | 0% | 93.95% | 93.95% | 0% | 81.25% | 81.25% |
| Scenario 2 - KEY128 | 0% | 96.19% | 96.19% | 0% | 87.18% | 87.18% |

### 7.3 Evaluation of Countermeasures

The objective of the countermeasures is either distracting the attacker or avoiding the attacker through false traffic. The countermeasures added 2% of overhead in performance, which is related to the setup time to activate the defense mechanisms. Note that the elaboration of countermeasures also took into account the IBN model. As a result, all real-time constraints were met. Table 5 shows the results of both countermeasures under both scenarios.

***RT-Blinding*** The RT-Blinding strategy pad the sensitive information with high priority packets. These padding packets are sent from the victim's IP in a time table-fashion. In addition, they are identical in terms of size and destination. Therefore, without a frame of reference, the attacker could not identify which of these packets (the sensitive packet or the padding packets) is the actual secure message. In this test scenario, one of each four packages is a real secure packet. In our test system, these values were defined as maintaining a latency below an arbitrary threshold. In a real system, this would be defined at design time, based

on the critical application and the hard time constraints. In both attacks, the result of the countermeasure is a plummet of efficiency values through all of the scenarios. This comes at the cost of having four times as many secure packages on the NoC, possibly preempting other flows beside the attackers.

***RT-Masking*** The RT-Masking strategy saturates the attacker with low-priority packages. To accomplish this, without undermining performance for the secure process, we employ another IP (Defender IP) which sends medium-sized (32 flits) and low priority packages at random intervals (pseudo-randomness achieved by a Linear Feedback Shift Register function). For the *Scenario 1*, this dedicated IP is defined as $IP_1$ at design time, and ideally, it would be placed as close as possible to the cache memory, as the attacks target the returning message of the cache. For the *Scenario 2*, the Defender IP is placed at $IP_2$, and in other scenarios should be placed as close as possible to the secure processor. As observed in table 5, the effects of this strategy in the Direct-Interference Attack are closely related to the package size of the secure flow. When the size of the Defender's IP package is equal or close to the secure package, there is a heavy loss of efficiency. This is a consequence of the inability of the attacker to differentiate the latencies caused by the congestion of the same priority packets (Defender) and the high priority packets (Victim). However, larger secure packages produce a higher effect on preemption, thus turning easier to distinguish the latency thresholds. The Back-Pressure attack remains unaltered by the countermeasure. This is the result of the utilization of the sensitive path by the Defender IP, whose packets are exchanged with low priority to avoid important performance penalties. Since the Back-Pressure attack uses medium priority in the sensitive traffic path, the defender IP will not affect this attacker.

***RT-Shielding:*** The RT-Shielding strategy avoids direct interference and back-pressure interference by designing the NoC properly. It has large enough buffers to sustain the throughput of potentially malicious traffic while providing a structure to quickly transmit sensitive packets. This approach can be theoretically guaranteed by applying the IBN model equations discussed in section 4.2. In our experiments, the four routers in the sensitive path (i.e., routers 11, 12, 21 and 22) had its buffers increased. Two configurations were evaluated, using 64 flits and 128 flits. To decrease the time to propagate sensitive packets in the system, the sensitive IPs (i.e., IP 13 and IP 1) were configured with low jitter - 4 cycles - Network Interfaces. To guarantee a minimal difference between the jitter of trusted and non-trusted IPs, the other network interfaces were configured with a jitter of 32 cycles, especially for IP 5, IP 6, IP 9, and IP 10. In addition, to comprise a scenario where a jitter of 4 cycles would affect the performance of the sensitive IPs, we also tested the system with 32 cycles as low jitter (for sensitive IPs), and 128 cycles as high jitter (for other IPs). In total, four different PP-NoCs were evaluated under both Direct-Interference and Back-Pressure attacks. As expected, the attacks did not experienced latency degradation, obtaining zero observability in the system.

Note that RT-Shielding provides a very efficient countermeasure that avoids any performance issue. However, it imposes several limitations as a protection mechanism. First, it increases the hardware considerably. Second, it is tailored to specific attacks; thus it is not guaranteed that variations of these attacks can be avoided as well. And finally, since this is defined at design time, there is no flexibility concerning the applications running.

Table 5: Evaluation results of the attacks under RT-Blinding and RT-Masking countermeasures (S1 - *Scenario 1*; S2 - *Scenario 2*)

| | Masking Countermeasure | | | | | | Blinding Countermeasure | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DI Attack | | | BP Attack | | | DI Attack | | | BP Attack | | |
| | FP | Obs. | Efficiency | FP | Obs. | Efficiency | FP | Obs. | Efficiency | FP | Obs. | Efficiency |
| S1.AES | 72.38% | 71.29% | 19.69% | 0% | 72.07% | 72.07% | 75% | 71.29% | 17.82% | 75% | 72.07% | 18.02% |
| S2.KEY32 | 91.85% | 93.53% | 7.41% | 0% | 80.43% | 80.43% | 75% | 93.53% | 23.38% | 75% | 80.43% | 20.11% |
| S2.KEY64 | 88.08% | 93.95% | 11.20% | 0% | 81.25% | 81.25% | 75% | 93.95% | 23.49% | 75% | 81.25% | 20.31% |
| S2.KEY128 | 20.75% | 96.19% | 76.23% | 0% | 87.18% | 87.18% | 75% | 96.19% | 24.05% | 75% | 87.18% | 21.79% |

## 8 Discussion

In this section, we clarify why the assumptions used for the attacks are practical. Regarding having a malicious IP inside the MPSoC, mobile and embedded systems allow external software to run on the devices (under low privileges). These external applications may hide malicious code (also known as Trojans), which configures a system or IP infection. Another way to have an attacker in the system is when a hidden functionality is embedded in a third-party hardware IP (also known as Hardware Trojan). Regarding the knowledge of the logical location (i.e., mapping in the NoC), typically the Operating System provides an API that points system services to logical addresses. Even if the IP is in another processor, there will be some logical identification by the system manager. Sometimes, the documentation clarifies the logical (or even the physical) addresses of the system components. About topology and routing knowledge, if the technical documentation of the device does not disclose this information, it is possible to infer it by injecting traffic into the NoC and observing the physical behavior (power, timing). Regarding the application privilege levels, note that the attacks do not require a high priority level. It is expected, that any application will have a minimum of privilege in the real-time service (at least two levels of privileges). For the one-hop location requirement, the attacker does not need to know the distance. The drop in attack efficiency will reveal to the attacker that this condition is not met. This allows location tuning by the attacker.

## 9 Conclusion

In this paper, we have shown that Priority-Preemptive NoCs are vulnerable to logical side-channel attacks. The accurate analytical model developed for these

systems to prove their demanding time constraints may be used to develop powerful attacks. The predictability of such systems provides the attacker with accurate information before the chip infection. We create two three-phase attacks: Direct-Interference and Back-Pressure. These attacks exploit two MPSoC vulnerabilities: i) the traffic predictability and shaping of sensitive applications. These attacks may detect key updates in the MPSoC. Key refreshing in sensitive applications is a common practice and it is usually performed through a key exchange protocol that presents a very specific traffic pattern. This can be exploited to determine the attack's momentum; and ii) the shared memories, which is a very common practice in MPSoCs. The time or access of the cache can be integrated with NoC timing attacks to reveal the secret key. We show that critical time systems must consider security already during the design stage. We demonstrated that providing lightweight security to critical systems while guaranteeing the time constraints is feasible.

## References

1. Sapra, D., Altmeyer, S.: Work-in-progress: Design-space exploration of multi-core processors for safety-critical real-time systems. In: 2017 IEEE RTSS. (Dec 2017) 360–362
2. Rockwood, M., Joshi, V., Sullivan, K., Goubran, R.: Using a real-time operating system for multitasking in remote patient monitoring. In: 2014 IEEE MeMeA. (June 2014) 1–5
3. Bolotin, E., Cidon, I., Ginosar, R., Kolodny, A.: Qnoc: Qos architecture and design process for network on chip. Journal of Systems Architecture **50** (2004) 105–128
4. Lo, S., Lan, Y., Yeh, H., Tsai, W., Hu, Y., Chen, S.: Qos aware binoc architecture. In: 2010 IEEE IPDPS. (April 2010) 1–10
5. Nikolić, B., Petters, S.M.: Edf as an arbitration policy for wormhole-switched priority-preemptive nocs — myth or fact? In: 2014 EMSOFT. (Oct 2014) 1–10
6. Shi, Z., Burns, A.: Real-time communication analysis for on-chip networks with wormhole switching. In: Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008). (April 2008) 161–170
7. Indrusiak, L.S., Burns, A., Nikolić, B.: Buffer-aware bounds to multi-point progressive blocking in priority-preemptive nocs. In: 2018 DATE. (March 2018) 219–224
8. Yao, W., Suh, E.: Efficient timing channel protection for on-chip networks. In: NOCS '12 Proceedings of the 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip, Lyngby, Denmark, IEEE (May 2012) 142–151
9. Daoud, L., Rafla, N.: Analysis of black hole router attack in network-on-chip. In: 2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS). (Aug 2019) 69–72
10. Sepúlveda, J., Diguet, J.P., Strum, M., Gogniat, G.: Noc-based protection for soc time-driven attacks. IEEE Embedded Systems Letters **7**(1) (March 2015) 7–10
11. Reinbrecht, C., Forlin, B., Zankl, A., Sepúlveda, J.: Earthquake - a noc-based optimized differential cache-collision attack for mpsocs. In: 2018 DATE, Dresden, Germany, ACM (Mar 2018) 1–7
12. Reinbrecht, C., Susin, A., Bossuet, L., Sepulveda, J.: Gossip noc - avoiding timing side-channel attacks through traffic management. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI 16), Pittsburgh, USA, IEEE (July 2016) 601–606

13. Sepúlveda, J., Gross, M., Zankl, A., Sigl, G.: Exploiting Bus Communication to Improve Cache Attacks on Systems-on-Chips. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI '17). (July 2017)
14. Forlin, B., Reinbrecht, C., Sepúlveda, J.: Attacking real-time mpsocs: Preemptive nocs are vulnerable. In: 2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC). (Oct 2019) 204–209
15. Stefan, R., Goossens, K.: Enhancing the security of time-division-multiplexing networks-on-chip through the use of multipath routing
16. Chaves, C.G., Azad, S.P., Hollstein, T., Sepulveda, J.: A distributed dos detection scheme for noc-based mpsocs. In: 2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC). (Oct 2018) 1–6
17. Reinbrecht, C., Susin, A., Bossuet, L., Sigl, G., Sepulveda, J.: Side channel attack on noc-based mpsocs are practical: Noc prime+probe attack. In: 29th Symposium on Integrated Circuits and Systems Design (SBCCI), Belo Horizonte, Brazil, IEEE (Aug 2016) 1–6
18. Osvik, D., et al.: Cache attacks and countermeasures: The case of aes. In: Topics in Cryptology - CT-RSA 2006. (2006) 1–20
19. Indrusiak, L.S., Harbin, J.R., Reinbrecht, C., Sepúlveda, M.J.: Side-channel protected mpsoc through secure real-time networks-on-chip. Microprocessors and Microsystems (July 2019) 34–46 ©2019 Elsevier B.V. All rights reserved. This is an author-produced version of the published paper. Uploaded in accordance with the publisher?s self-archiving policy.
20. Yao, W., Suh, E.: Efficient timing channel protection for on-chip networks. In: Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on. (2012) 142–151
21. Wassel, H., Ying, G., Oberg, J., Huffmire, T., Kastner, R., Chong, F., Sherwood, T.: Networks on chip with provable security properties. Micro, IEEE **34**(3) (May 2014) 57–68
22. Sepulveda, J., Florez, D., Soeken, M., Diguet, J., Gogniat, G.: ddynamic noc buffer allocation for mpsoc timing side channel attack protection. In: Network-on-Chip, timing, side channel attack (LASCAS), Florianópolis, Brazil, IEEE (Mar 2016) 91–94
23. Traber, A., Stucki, S., Zaruba, F., Gautschi, M., Pullini, A., Benini, L.: Pulpino: A risc-v based single-core system (2015) ORCONF2015; Geneva, Switzerland; October 9-11, 2015; .
24. Xiong, Q., Wu, F., Lu, Z., Xie, C.: Extending real-time analysis for wormhole nocs. IEEE Transactions on Computers **66**(9) (Sept 2017) 1532–1546
25. Reinbrecht, C., Susin, A., Bossuet, L., Sigl, G., Sepúlveda, J.: Timing attack on noc-based systems: Prime+probe attack and noc-based protection. Microprocessors and Microsystems **51** (Jan 2017)
26. Reinbrecht, C., Forlin, B., Sepúlveda, J.: Cache timing attacks on noc-based mpsocs. Microprocessors and Microsystems **66** (April 2019) 1–9