CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

IPDeN: Real-Time deflection-based NoC with in-order flits delivery

Yilian Ribot*
Geoffrey Nelissen
Eduardo Tovar*

*CISTER Research Centre

CISTER-TR-220610

2022/08/23

IPDeN: Real-Time deflection-based NoC with in-order flits delivery

Yilian Ribot*, Geoffrey Nelissen, Eduardo Tovar*

*CISTER Research Centre

Polytechnic Institute of Porto (ISEP P.Porto)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: ribot@isep.ipp.pt, gnn@isep.ipp.pt, emt@isep.ipp.pt

https://www.cister-labs.pt

Abstract

In deflection-based Network-on-Chips (NoC), when several flits entering a router contend for the same output port, one of the flit is routed to the desired output and the others are deflected to alternatives outputs. The approach reduces power consumption and silicon footprint in comparison to virtual channels (VCs) based solutions. However, due to the non-deterministic number of deflections that flits may suffer while traversing the network, flits may be received in an out-of-order fashion at their destinations. In this work, we present IPDeN, a novel deflection-based NoC that ensures in-order flit delivery. To avoid the use of costly reordering mechanisms at the destination of each communication flow, we propose a solution based on a single small buffer added to each router to prevents flits from over taking other flits belonging to the same communication flow. We also develop a worst-case traversal time (WCTT) analysis for packets transmitted over IPDeN. We implemented IPDeN in Verilog and synthesized it for an FPGA platform. We show that a router of IPDeN requires "483-times less hardware resources than routers that use VCs. Experimental results shown that the worst-case and average packets communication time is reduced in comparison to the state-of-the-art.

IPDeN: Real-Time deflection-based NoC with in-order flits delivery

1st Yilian Ribot González
CISTER Research Centre
ISEP, Polytechnic Institute of Porto
Porto, Portugal
ribot@isep.ipp.pt

2nd Geoffrey Nelissen
Eindhoven University of Technology
Eindhoven, the Netherlands
g.r.r.j.p.nelissen@tue.nl

3rd Eduardo Tovar
CISTER Research Centre
ISEP, Polytechnic Institute of Porto
Porto, Portugal
emt@isep.ipp.pt

***Abstract*—In deflection-based Network-on-Chips (NoC), when several flits entering a router contend for the same output port, one of the flit is routed to the desired output and the others are deflected to alternatives outputs. The approach reduces power consumption and silicon footprint in comparison to virtual-channels (VCs) based solutions. However, due to the non-deterministic number of deflections that flits may suffer while traversing the network, flits may be received in an out-of-order fashion at their destinations. In this work, we present IPDeN, a novel deflection-based NoC that ensures in-order flit delivery. To avoid the use of costly reordering mechanisms at the destination of each communication flow, we propose a solution based on a single small buffer added to each router to prevents flits from over taking other flits belonging to the same communication flow. We also develop a worst-case traversal time (WCTT) analysis for packets transmitted over IPDeN. We implemented IPDeN in Verilog and synthesized it for an FPGA platform. We show that a router of IPDeN requires ≈3-times less hardware resources than routers that use VCs. Experimental results shown that the worst-case and average packets communication time is reduced in comparison to the state-of-the-art.**

***Index Terms*—Real-Time Embedded Systems, Systems-on-Chips, Network-on-Chips, Worst-Case Traversal Time**

I. Introduction

As more powerful embedded systems are required to perform advanced functionalities, the number of processing elements (PEs) integrated in Systems-on-Chips (SoCs) has increased significantly over the past decade. In this context, shared communication buses, which used to connect all the PEs together, suffer from several drawbacks, such as limited bandwidth and scalability with respect to the growing number of messages exchanged between PEs. They have been identified as one of the major bottlenecks for the performance of SoCs. NoCs are router-based packet switching networks that offer an alternative to the traditional communication architectures based on buses [2], [8]. NoCs allow the integration of a large number of computation nodes on a single SoC, as they allow multiple PEs to transmit messages in parallel. In this paper, we are interested about real-time systems, that is, systems that have timing requirements associated to their tasks. The hardware platform on which such systems execute should preferably be analyzed from a timing perspective. Therefore, NoCs used in real-time systems should provide certain guarantees, namely, packets that travel through the network must reach their destinations within previously defined deadlines.

The two most popular types of NoCs architectures revolve around the concepts of wormhole switching with VCs, and deflection-based routing. VCs are a popular solution to improve NoCs throughput but significantly increase silicon footprint and power consumption. They mainly target high-end applications.

NoCs built upon the concept of deflection are a good alternative, as they have lower implementation cost and power consumption. Deflection-based NoCs are more suitable to systems with energy and/or area constraints and limited resource availability.

In deflection-based NoCs, flits of the same communication flow may be deflected to different routes with different lengths while traversing the network. Hence, the flits may not be received at their destinations in the same order as they were injected into the network at their sources.

In-order packet delivery is a required feature in an extensive variety of applications, such as control applications, where the control actions depend on a specific sequence of inputs, or in applications where large data blocks need to be transmitted over multiple flits, which thus require the complete data to be reconstructed at the destination. To guarantee in-order packet delivery, reordering mechanisms are frequently used at the destination nodes. However, those strategies add an additional layer of complexity, thereby slowing down the application, and considerably increasing hardware resources consumption.

Contribution. We propose IPDeN, a deflection-based NoC that ensures in-order flit delivery at the destination. Specifically, we propose a new router architecture associated to a new routing policy that guarantees in-order flit delivery. It introduces small constant size buffers in the router's design to avoid the use of costly reordering logic at the destination. We derive a WCTT analysis for packets transmitted over IPDeN, That is, we derive upper-bounds on the maximum number of clock cycles taken by the flits injected into the network to reach their destinations. We implement our NoC on an FPGA platform, and compare against the state-of-the-art.

II. Related work

Time-triggered routing arbitration-based NoCs [1], [4], [17] isolate the timing properties of the communication flows by allocating pre-calculated transmission slots to them. If slots are properly assigned, in-order packet delivery is ensured. However, they require to know the complete system specification at configuration time. Therefore, they difficultly support changes in the system workload at run-time and are complex to configure.

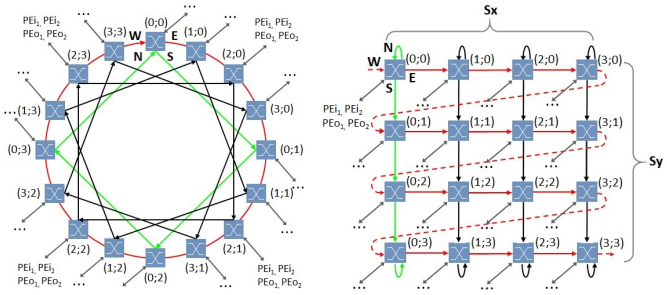
For those reasons, wormhole switching NoCs with VCs [12], [13], [20] are the most popular NoC solutions. Wormhole switching with VCs ensures in-order packets delivery if all packets of a same flow take the same route. [5] proposes a solution that guarantees in-order packet delivery while packets are routed through multiple paths in the network, and [6] proposes a solution that supports in-order packet delivery under adaptive routing by reserving an alternate virtual path during the VC allocation process. Yet, such approaches are expensive to implement in terms of silicon footprint and have high power consumption due to the extensive use of buffers.

PaterNoster [11] and CAERUS [15] are two in-order deflection NoCs. The solutions rely on minimally buffered deflection routing to preserve flit order between sending and receiving nodes. [10] exposes the weaknesses of the solutions (e.g., large and costly reception buffers; the buffer overflow handling mechanism may destroys the order of flits; and livelocks may occur) and solves all the issues.

[9] presents a Butterfly Fat Tree NoC that ensures in-order delivery at medium implementation cost and provides bandwidth configuration flexibility. HopliteBuf [7] provides in-order packet delivery by adding stall-free buffers in routers. The fact that, a large buffer is added in each router leads for increased resource and power requirements for the NoC.

III. System model

We assume a system composed of N programming elements (PEs) $\{\pi_1, ..., \pi_N\}$ interconnected with a NoC of N routers $\{R_1, ..., R_N\}$. Each PE π_q is connected to a router R_q with coordinates (x_q, y_q) in a grid representation of the NoC. Each PE π_q injects a set of n_q communication flows $F_q = \{f_{q,1}, f_{q,2}, ..., f_{q,n_q}\}$ into the network. A communication flow $f_{q,i}$ generates a potentially infinite number of packets with a minimum inter-arrival time $T_{q,i}$, i.e., the minimum duration between the generation of two packets by $f_{q,i}$ is $T_{q,i}$. A packet is split in $C_{q,i}$ flits sequentially injected in the router at coordinates $(x^o_{q,i}, y^o_{q,i})$. The destination of the flits of $f_{q,i}$ is the PE at coordinates $(x^d_{q,i}, y^d_{q,i})$. We assume that there is no distinction between header, body or tail flits. Specifically, we assume that the destination coordinate of a packet is encoded in all flits. Note that, the number of bits in a flit that are used to encode its destination coordinate only depends on the size of the network, e.g., 64-bits width flits traversing a 4x4 IPDeN NoC must use 4-bits to encode their destination coordinates. Each flit contains S_{flit} bits. We define the utilization of flow $f_{q,i}$ as $U_{q,i} = \frac{C_{q,i}}{T_{q,i}}$. The utilization of a PE π_q is thus $U_q = \sum_{\forall f_{q,i} \in F_q} U_{q,i}$.



(a) Circulant topology (16;1,4). (b) Equiv. grid-based network.

Fig. 1. IPDeN NoC topology.

IV. IPDeN NoC

In this section, we present IPDeN; the network topology, router architecture, and routing policy. In the next section, we present a timing analysis and discuss some of IPDeN's main properties.

A. NoC topology

In IPDeN, the routers are connected together according to a ring circulant topology [16] with parameters $(N; g_1, g_2)$ (where $g_1 = 1$, N is the total number of routers, and g_1, g_2 are the *generatrices* of the network). We assume that g_2 divides N. Under this circulant topology (see Fig. 1(a)), each router of IPDeN has four inputs and four output ports (see Fig. 2(d)). Two inputs ports referred to by the letters N and W (for North and West), and two output ports referred to as E and S (for East and South) are used for inter-routers communication. The remaining two inputs ports, which are named PEi_1 and PEi_2, are used by the PE to inject packets into the network. Similarly, when a flit reaches its destination router, it is transmitted to the PE via either of the remaining two output ports named PEo_1 and PEo_2.

All the routers are connected by a single unidirectional ring using the ports W and E of each router (see red line in Fig. 1). Each router is also connected to the routers that are g_2 hops away on the ring using the N and S ports (see green and black lines in Fig. 1).

As an alternative that facilitates later discussion about coordinates of PEs and routers in the network, the circulant network $(N; 1, g_2)$ may be represented as a $S_x \times S_y$ grid-based network (see Fig. 1(b)) where S_x and S_y correspond to the number of routers on the horizontal and vertical dimensions, respectively, and where $S_x = g_2$ and $S_y = \frac{N}{g_2}$. Note that, the grid-based network does not illustrate how the routers should be mapped at the moment of creating the layout of IPDeN. Figs.1(a) and 1(b) are abstract representations of the topology. We provide the grid-based representation as a supporting element to simplify the explanation and therefore the understanding of the IPDeN's routing policy in Section IV-B. As further discussed in the experimental section, there is no long wire in the actual implementation of the network.

Therefore, there is no transmission latency problems impacting the maximum clock frequency of the network. Long wires (e.g., the wire that connects the port E of router (3,3) with the port W of router (0,0) in Fig. 1(b)) are simply artifacts of the grid-based representation due to representing a 3D structure in a 2D drawing. In fact, all wires connecting consecutive routers on the ring have the same length (see Fig. 1(a)). The length of the wires can be reduced by mapping the routers appropriately during the placement and routing step on the final chip. Note that the same happens with a torus topology.

Example: Fig. 1(a) shows the circulant network with parameters $(16; 1, 4)$. In Fig. 1(b), we provide its equivalent representation as a 4x4 grid-based network. The main unidirectional ring (red link in Fig. 1(a)) corresponds to the rows of the grid (see Fig. 1(b)), where the E port of the last router in row number y is connected to the W port of the first router in row number $(y+1) \mod S_y$. Similarly, the bypasses (green and black in Fig. 1(a)) correspond to the links on the columns of the grid where the last router in a column is connected to the first router in that same column.

B. Routing policy

A PE can inject packets on the ports E and S by using the input ports PEi_1 and PEi_2, respectively (Fig. 2(d)). Note that to reduce waiting time, packets may be injected via PEi_1 and PEi_2 in parallel. Therefore, a packet that is waiting to be injected into the network only conflicts with the subset of packets that must be injected to the same input port PEi_u.

Property 1: In this paper, we assume that a flit of a flow $f_{q,i}$ with origin coordinates $(x^o_{q,i}, y^o_{q,i})$ and destination coordinates $(x^d_{q,i}, y^d_{q,i})$, is injected in the network using port PEi_1 **if and only if** $x^o_{q,i} \neq x^d_{q,i}$ and using port PEi_2 **if and only if** $x^o_{q,i} = x^d_{q,i}$.

From Property 1, we get that all the flits of the same flow will be injected in the network using the same input port.

IPDeN's routing policy is mostly similar to that proposed in [21], which is, a modified version of X-Y routing. A flit of flow $f_{q,i}$ would first travel horizontally along the X-axis of the network until it reaches a router with the same X coordinate as its destination router. Once in that router, it requests the S port and travels vertically along the Y-axis until reaching its destination. IPDeN's routing policy differs from X-Y routing in that if two flits request the same port S at the same time, it "deflects" one of the flits to the port E. A deflected flit must then travel along the X-axis until reaching the same router as it would have if it had not been deflected. It will then request the port S again and continue traveling along the Y-axis.

The routing policy is summarized in Table I. Flits entering by the port W may request the ports E or S, while flits entering by the port N may only request the port S. Flits entering by the port W and requesting E never suffer contention, i.e., they travel freely through the X-axis (row 1 in Table I). In addition, a flit entering a router by the port W always has the highest priority to use the port S when it conflicts with a flit entering by the port N. Under this scenario, the flit coming from the port N is deflected to the port E inconsiderately of its final destination (row 2 in Table I). The deflected flit will now travel along the X-axis. It will enter the next routers via the port W and will thus have the highest priority the next time it will require an S port. Finally, flits entering by the ports PEi_1 and PEi_2 always have the lowest priority to use the ports E and S, respectively, and must wait for those ports to become free (rows 3 to 9 in Table I).

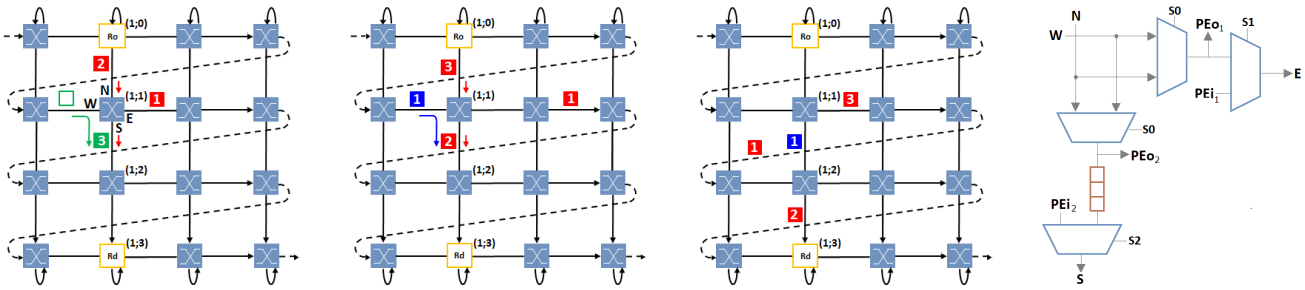
The routing policy and network topology discussed so far are identical to those presented in [16]. One issue with this policy is that the number of deflection the flit suffers, and thus the length of its route, depend on the congestion it encounters during its travel. Therefore, flits from the same packet that have been injected sequentially into the network may take different routes while traveling until their destinations and may not be received at the destination in the same order as they were injected into the network, i.e., they may reach the destination in an out-of-order fashion.

Example 1. Consider a packet p (red packet in Fig. 2) split in 3 flits sequentially injected at router with coordinates $(1; 0)$. The destination router of p is router $(1; 3)$. Flit 1 of red packet enters the router $(1; 1)$ by the input port N. Now, assume that it contends to access port S with another flit (green flit in Fig. 2(a)) coming from the port W. Then, according to the arbitration policy (Table I), the green flit wins access to port S and Flit 1 the of red packet is deflected to the port E. Next cycle (see Fig. 2(b)), flit 2 of the red packet is routed $N \rightarrow S$ at router $(1; 1)$, since it does not conflict with other flits for the port S. One cycle later, as we observe in Fig. 2(c), flit 2 of the red packet is the first flit from p to reach its destination $(1; 3)$, i.e., flits are received in an out-of-order fashion.

C. New router architecture

Example 1 shows the following shortfall of the routing policy. If a flit fl_p of a communication flow $f_{q,i}$ is deflected in a router R_q at time t, the length of its route to its destination is increased by $S_x - 1$ hops. Thus, if the next flit, say fl_{p+1} of $f_{q,i}$ passes through router R_q d clock cycles later (with $d < S_x - 1$) without being deflected, it will be $S_x - 1 - d$ hops ahead of fl_p. Therefore, to avoid this issue, we added a small buffer with $(S_x - 1)$ slots and some clever logic in each router of IPDeN (see Fig. 2(d)). The buffer delays the transmission of flits through the port S by as many clock cycles as needed to prevent a flit fl_{p+1} to be ahead of a flit fl_p that was previously deflected in the same router.

Each router π_q of IPDeN contains a buffer with $S_x - 1$ slots numbered from 1 to $S_x - 1$ and a variable B_q that points to one of the slots of the buffer. After each deflection, B_q is set to $S_x - 1$, thereby pointing to the last slot of the buffer. At each clock cycle, if a flit is routed to port S, it is saved at the buffer position pointed by B_q. The value of B_q is then kept the same. Otherwise, i.e., if no flit is routed to port S, the value of B_q decreases by 1. At each clock cycle, the flits in the buffer are shifted one slot closer to the head of the queue. A flit reaching the buffer's head is then transmitted through port S. This approach ensures that flits wait exactly B_q clock cycles before being transmitted through port S, and B_q is



(a) (b) (c) (d)

Fig. 2. (a), (b), and (c) Example of an out-of-order reception. (d) New router architecture of IPDeN that ensures in-order-packet delivery.

TABLE I
ROUTING TABLE OF IPDEN.

Input requests	Routing decisions	Explanation
$W \rightarrow E + N \rightarrow S$	$W \rightarrow E + N \rightarrow S$	No contention.
$W \rightarrow S + N \rightarrow S$	$W \rightarrow S + N \rightarrow E$	Conflict over the S port. $W \rightarrow S$ requests wins. N packet is deflected.
$N \rightarrow S + PEi_1 \rightarrow E$	$N \rightarrow S + PEi_1 \rightarrow E$	No contention.
$N \rightarrow S + PEi_2 \rightarrow S$	$N \rightarrow S$	PEi_2 cannot inject its packet.
$W \rightarrow S + PEi_1 \rightarrow E$	$W \rightarrow S + PEi_1 \rightarrow E$	No contention.
$W \rightarrow S + PEi_2 \rightarrow S$	$W \rightarrow S$	PEi_2 cannot inject its packet.
$W \rightarrow E + PEi_1 \rightarrow E$	$W \rightarrow E$	PEi_1 cannot inject its packet.
$W \rightarrow E + PEi_2 \rightarrow S$	$W \rightarrow E + PEi_2 \rightarrow S$	No contention.
$PEi_1 \rightarrow E + PEi_2 \rightarrow S$	$PE_1 \rightarrow E + PEi_2 \rightarrow S$	No contention.

always larger than $S_x - 1 - d$ where d is the delay between the arrival of a new flit in port S of router R_q, and the last time a flit was deflected in R_q, thus avoiding the problem discussed above. When B_q is equal to 0, flits are not buffered and are directly transmitted through port S.

Example 2. Consider a 4×4 IPDeN NoC (see Fig. 3) and a packet p (red packet) that traverses the network from its origin router $(1;0)$ to its destination $(1;3)$. Let's assume that initially the value of B_q is 0 in all routers. The packet p is split in three flits sequentially injected to the port S of its origin router (since $x^o_{q,i} = x^d_{q,i}$). Assume that in router $(1;1)$, the flit 1 of p requests the port S at the same time as the flit of a green packet (see Fig. 3(a)). In this contention scenario, according to Table I, the first red flit is deflected to the port E and the green flit is routed to the port S (see Fig. 3(a)). Because there is a deflection, the variable B_q of router $(1;1)$ is set to $S_x - 1 = 3$. Thus, at the next clock cycle, the flit 2 of p, which is routed to the port S as it does not conflict with any other flow, is buffered in the last position of the buffer and the value of B_q remains unchanged (see Figs. 3(a) and 3(b)). One clock cycle later, flit 3 of p requests also the port S of router $(1;1)$. However, if a flit of another packet (blue packet) also requests the port S at the same time as in Fig. 3(c). The flit 3 of p is deflected to the port E and the blue flit is buffered in the last position of the buffer (Figs. 3(b) and 3(c)). The value of B_q is thus still equal to 3 since a flit has been buffered at each clock cycle since the last deflection. If no flit requests the port S at the next clock cycle as in Fig. 3(c), the value of B_q becomes 2. Therefore, the next flit to be routed to the port S of router $(1;1)$ (pink flit), is buffered in position number 2 of the buffer (Figs. 3(d) and 3(e)). Note that, thanks to the fact that flit 2 of red packet p was buffered in router $(1;1)$, the flits 1, 2, and 3 of p reach the destination router $(1;3)$ in an orderly fashion (see Figs. 3(d)-3(f)).

V. IPDEN'S ANALYSIS

A. Buffer size analysis

One of the biggest advantage of IPDeN is that the buffers are small and their sizes do not depend on the set of flows that traverse the network. It is a clear difference with most NoCs that implement VCs or other type of buffering mechanisms meant to avoid back-pressure or losing flits during the transmission. For instance, in HopliteBuf buffers are also added to routers to enforce in-order flit delivery, but those buffers have size that depends on the bandwidth requested by the communication flows transmitted through the NoC as well as their source and destination. In practice buffers tend to be oversized to ensure they can cope with unforeseen circumstance. In IPDeN, no such resources waste is needed. The size of the buffers depends solely on the network topology, i.e., it is equal to $S_x - 1$, ensuring that the NoC's power and silicon footprint remains small and independent of the specific applications running on the network.

B. Livelock and deadlock situations

Livelock occurs when a packet travels indefinitely through the network and never reaches its destination. IPDeN's routing policy guarantees livelock-free transmissions. The circulant topology ensures that flits always progress towards their destination routers even when they are deflected. Once at the destination routers, flits always exit the network by using the output ports PEo_1 and PEo_2.

IPDeN also guarantees freedom from deadlock. Deadlock occurs when a packet cannot progress in the network because it is waiting permanently for an event that cannot happen. In IPDeN, any flit entering a router and requesting a certain

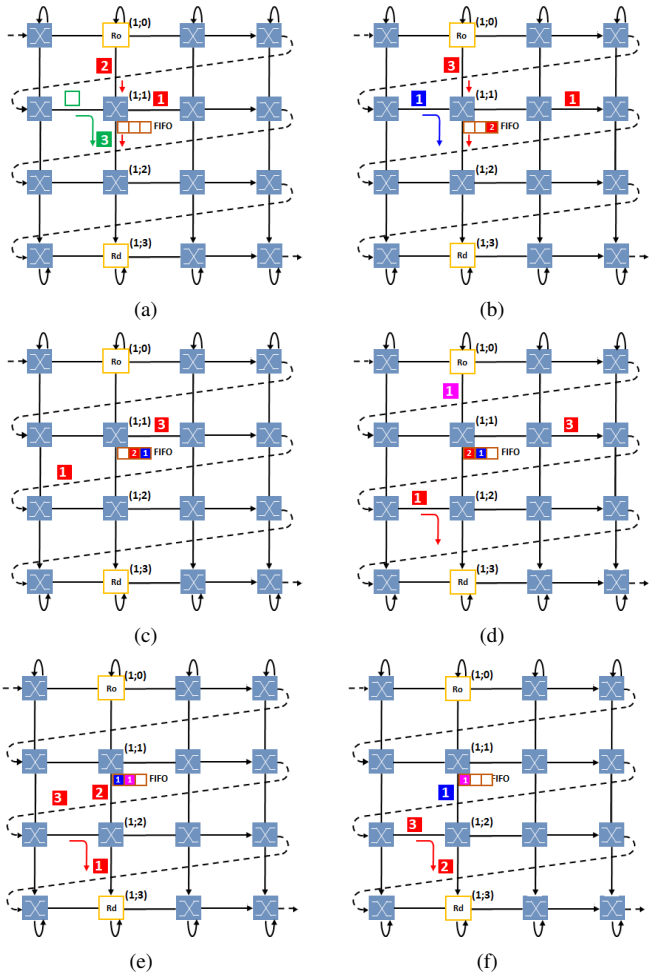


Fig. 3. Example of how buffers are used in IPDeN.

output port may be: (1) routed the desired port, (2) deflected to an alternative port, (3) or buffered. Under scenarios 1 and 2, the flit never stops progressing towards its destination. Conversely, in scenario 3, the flit is forced to wait inside a buffer before exiting the router. Nonetheless, as discussed in section IV-C, it is shifted one slot every clock cycle until reaching the head of the buffer no matter what. Once at the buffer's head, the flit is always routed to the port S of the router and is thus allowed to resume its journey to its destination. A flit is therefore never blocked indefinitely.

C. Priority levels

IPDeN's routing policy does not make any distinction between different priority levels, i.e., all flits are treated identically. The main reason behind this design choice in that the worst-case transmission time of all flits will always be the same, whatever their priority. Indeed, in each router of IPDeN, the transmission of a flit may be delayed by a deflection to an alternative port or, if it is not deflected, may be buffered inside the router for a number of clock cycles equal to the extra-cost introduced by a deflection. Note that buffering must happen without any consideration of flit's priorities in order to ensure in-order flit delivery. Therefore, the worst case traversal time will remain the same when it does or does not support priority-aware routing. Although our solution does not provide different quality-of-service to different classes of traffic, our network architecture does ensure predictable traversal time for all flits.

D. Bound on the worst-case communication time

In this section, we present an analysis for the worst-case communication time (WCCT) between two PEs connected with IPDeN. The WCCT of any flit of a packet is composed of two parts, the worst-case injection time (WCIT) and the WCTT. The WCIT is defined as the maximum amount of clock cycles between the generation of a packet by a PE and the injection of its last flit into the network. The WCIT depends on the load of the network and can be computed as in [16]. The WCTT is defined as the maximum amount of clock cycle taken by a flit to traverse the network and reach its destination once it has been injected. The WCTT of a flit of a packet of flow $f_{q,i}$ is computed as

$$wctt_i = h_r^i(x_{q,i}^d, y_{q,i}^d) + h_b^i(x_{q,i}^d, y_{q,i}^d) \times (1 + cost) + 2 \quad (1)$$

where $h_r^i(x_{q,i}^d, y_{q,i}^d)$ is the number of hops made by the flit on the ring until reaching its destination, assuming a zero contention (i.e., that no flit is ever deflected in the network). Similarly, $h_b^i(x_{q,i}^d, y_{q,i}^d)$ is the number of hops on the bypasses until its destination, assuming no contention. Simply speaking, $h_r^i(x_{q,i}^d, y_{q,i}^d)+h_b^i(x_{q,i}^d, y_{q,i}^d)$ is the Manhattan distance between the origin and destination routers, where $h_r^i(x_{q,i}^d, y_{q,i}^d)$ and $h_b^i(x_{q,i}^d, y_{q,i}^d)$ are the X- and Y-distance, respectively. $cost$ is the maximum number of cycles by which the journey of a flit may be increased due to a single deflection or buffering in one of the routers. The two hops added at the end of Equation (1) account for the one cycle needed to inject the flit into the network and the one cycle needed by the destination PE to read that flit. We now explain how Equation (1) was derived.

It was proved in [16] that $h_r^i(x_{q,i}^d, y_{q,i}^d) + h_b^i(x_{q,i}^d, y_{q,i}^d) + 2$ is the maximum number of hops a flit must make between its origin and destination if it suffers no deflection and is never buffered in any router along its route. We must thus still prove (using Property 2 and Lemma 1 below) that the maximum total delay that may be suffered by any flit of $f_{q,i}$ due to deflections or buffering is upper bounded by $h_b^i(x_{q,i}^d, y_{q,i}^d) \times cost$. We then prove in Lemma 2 that $cost = S_x - 1$.

***Property** 2:* A flit of flow $f_{q,i}$ may only be deflected in routers in which it requests the port S.

Proof: As discussed in previous sections, a flit of flow $f_{q,i}$ can suffer a deflection if and only if it requests the same output port as other coming flit at the same time. According to IPDeN's routing policy (see Table I), a flit of flow $f_{p,i}$ entering a router R_k by the port W may request the port E or S, while a flit of flow $f_{q,i}$ entering R_k by the port N can only request the port S. Then, flits coming from the port W and requesting the port E travel freely along the X-axis as they do not conflict over the port E with other flits traversing the network. However, if the flit of flow $f_{p,i}$ requests the port S at the same time as the flit of flow $f_{q,i}$, one of the flit wins

access to port S and the another one is deflected to the port E. Hence, a flit may only be deflected in a router in which it requests the port S and thus, the property is proved. ■

Let $n^i_{def}(x^d_{q,i}, y^d_{q,i})$ and $n^i_{buf}(x^d_{q,i}, y^d_{q,i})$ be the maximum number of times a flit of $f_{q,i}$ suffers a deflections and the maximum number of times it enters a buffer, respectively. The following lemma holds

***Lemma** 1:*

$$n^i_{def}(x^d_{q,i}, y^d_{q,i}) + n^i_{buf}(x^d_{q,i}, y^d_{q,i}) = h^i_b(x^d_{q,i}, y^d_{q,i}), \quad (2)$$

Proof: According to Property 2, a flit of flow $f_{q,i}$ may be deflected or buffered as many times as it requests the port S of a router on its route, that is, as many time as the Y-distance between the origin and destination router, i.e., $h^i_b(x^d_{q,i}, y^d_{q,i})$. Since, a flit is either deflected or buffered (but not both) in each such router, we have $n^i_{def}(x^d_{q,i}, y^d_{q,i}) + n^i_{buf}(x^d_{q,i}, y^d_{q,i}) = h^i_b(x^d_{q,i}, y^d_{q,i})$. ■

The additional cost (in terms of clock cycles) added by each deflection or buffering is defined in Lemma 2.

***Lemma** 2:* The maximum cost of a deflection or buffering is upper bounded by $cost = S_x - 1$.

Proof: First, when a flit is deflected in a router R_q, it must do S_x hops instead of 1, before reaching the same router as it would have if it could have used the S port instead. Therefore, each deflection involves an additional cost of $S_x - 1$ hops. Second, the size of the buffer in each router is $S_x - 1$. Hence, the cost of a buffering is *at most* $S_x - 1$. Then, the maximum cost of a deflection or buffering is $S_x - 1$. ■

Combining Lemmas 1 and 2, we get that the total additional delay suffered by a flit of $f_{q,i}$ due to both deflections and buffering is upper bounded $h^i_b(x^d_{q,i}, y^d_{q,i}) \times cost$, thus proving Equation (1).

E. Discussion about the network latency

The timing behavior of IPDeN is deterministic. The timing analysis proposed in Section V-D only utilizes information of the flit under analysis and does not use any information from other flows that may traverse the network. Hence, this analysis is useful for applications with dynamically varying workload and timing properties. IPDeN provides predictable performances to those applications without any need to over-provision or dynamically reconfigure the network. Injection time however may vary with the workload. Nonetheless, this could be addressed by adding leaky buckets at each network input to limit the maximum bandwidth that may be claimed by each PE and/or communication flow.

VI. Experimental results

A. Implementation of IPDeN

We implemented IPDeN with the hardware description language Verilog. We synthesized a single router of IPDeN for flits of 64 bits. The target platform was a Xilinx Virtex-7 485T FPGA. A 64-bits IPDeN router required 471 LUTs and 715 Flip-Flops (FFs). This corresponds to only 0.16% and 0.12% of the total number of LUTs and FFs available in the target FPGA, respectively.

TABLE II
Resources utilization of one router.

NoC	VC-based	HopliteBuf	IPDeN
LUTs	1300-1574	402	471

We compared the hardware resource utilization with HopliteBuf [7], our closest competitor (as it is, as far as we know, the most recent solution for in-order packet delivery inspired by a deflection based routing policy), and three VC-based NoCs: CONNECT [13], IDAMC [20], and ProNoC [12] (see Table II). According to [13] and [20], a single router of CONNECT and IDAMC requires approximately 1500 and 1300 LUTs, respectively. A ProNoC router with two VCs requires 1574 LUTs. Comparatively, a router of HopliteBuf requires 402 LUTs. Therefore, IPDeN needs three times less resources than VC-based solutions. It is, however, 20% more expensive than HopliteBuf (in terms of LUTs utilization), but IPDeN uses smaller buffers, which is an advantage in terms of silicon footprints and power consumption. The size of IPDeN's buffers is also fixed, while those of HopliteBuf depend on the properties of the application using the network.

B. RTL simulations

In this section, we provide experimental results by performing cycle-accurate simulations. We used an HDL Verilog implementations of a 4x4 IPDeN and a 4x4 HopliteBuf NoC. We generated sets of flows according to a random traffic pattern. We configured each PE to inject one, two, or three flows into the network. The destination coordinates of each flow was randomly generated using a uniform probability distribution. The inter-arrival time $T_{q,i}$ of each flow was randomly selected within the set $\{100, 200, ..., 900\}$. The utilization of each PE was randomly chosen within the interval $[U_{bound} - 5\%, U_{bound}]$, where U_{bound} is a parameter of the experiment. The utilization $U_{q,i}$ of each flow was generated using the method described in [3]. The number of flits $C_{q,i}$ in each packet is determined by multiplying the flow's utilization $U_{q,i}$ by $T_{q,i}$. The flit size was set to 64 bits. Each value reported in Fig. 4 and 5 is the result of running 20 experiments.

In Fig. 4, we show the average measured communication time for an increasing number of communication flows per PE (i.e., one, two, and three flows per PE) when $U_{bound} = 20\%$. As discussed in Sec. V-D, the average measured communication time is given by the sum of the average measured traversal time and average injection time. We observe that IPDeN performs better than HopliteBuf in terms of traversal time. That can be explained by the fact that in HopliteBuf, flits can wait long periods of time in the buffer inside each router. Indeed, the number of cycles that flits must wait inside buffers depends on the set of flows that travel trough the network. That is, flits may be blocked in the buffers as many cycles as other flows use the port S. In IPDeN, however, the worst-case buffering time in each router is limited to $S_x - 1$ clock cycles, i.e., it is independent of the traffic and only depends on the network topology. As another consequence of the aforementioned property, the average traversal time

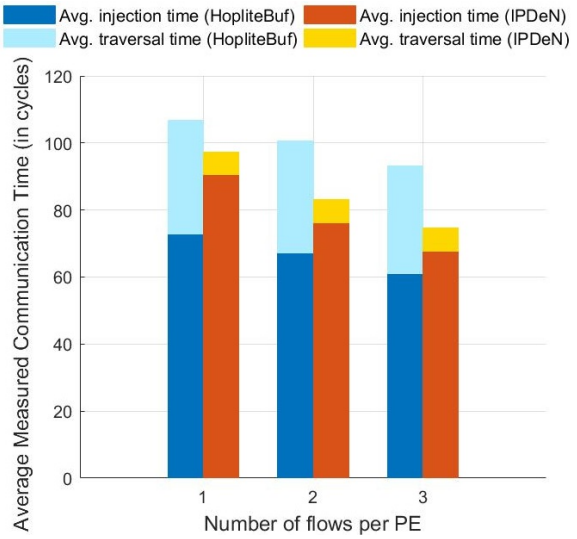


Fig. 4. Average measured communication time for random traffic with $U_{bound} = 20\%$.

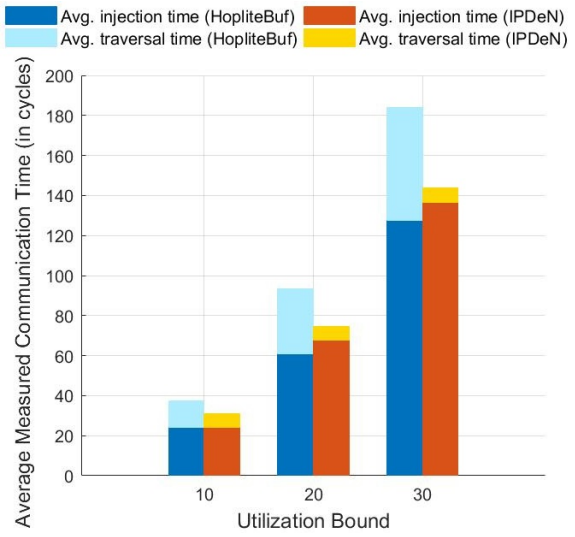


Fig. 5. Average measured communication time for random traffic with 3 flows per PE.

keeps practically the same when the number of flows that are traversing IPDeN increases. On the other hand, HopliteBuf performs better than IPDeN in terms of average injection time. We suspect that can be explained by the fact that, HopliteBuf replaces the need for deflection by using bigger buffers in each router instead. Therefore, less flits use the port E of the routers, and those can be used by PEs to inject new flits in the network instead. Note that, the average injection time decreases when the number of flows injected by PEs increases. It can be explained by the fact that, if the number of flows increases for a same value of U_{bound}, the average utilization of each flow decreases, thereby meaning that the number of flits in the packets in each flow decreases. Therefore, the injection time of all flits of a packet decreases too. Finally, we see that the average measure communication time of packets is more or less 20% smaller than with HopliteBuf.

In Fig. 5, we show the average measured communication time for an increasing utilization bound by measuring the average traversal time and injection time. We varied the utilization bound U_{bound} from 10 to 30% by steps of 10. We configured each PE to inject three communication flows into the network. Note that, greater PEs utilization means that each packet contains more flits. Therefore, it is expected that the injection time of a packet increases too when the number of flits it must inject increase (as seen in Fig. 5). However, as in the previous experiment, and for the same reasons discussed there, we observe that the average traversal time stays the same in IPDeN when U_{bound} increases. This is not the case for HopliteBuf which sees its traversal time increasing when U_{bound} increases. Therefore, overall, IPDeN has an smaller average communication time and IPDeN performs better than HopliteBuf when the volume of data transmitted increases.

C. Test Cases

In this section, we use a synthetic test case [19] and a real case study (Orion Crew Exploration Vehicle) [14] to evaluate our proposed NoC architecture. Both test cases are commonly used to evaluate Ethernet-based systems, but we adapted them to compare the performances of IPDeN to other types of NoCs when those networks must support realistic communication loads. To perform our experiments, we assume that the entire system is consolidated in a single chip, and that, the computation nodes rely on a NoC to communicate instead of using an Ethernet network.

The configuration details of the test cases are presented in Table III. The origin and destination PEs of each communication flow were randomly generated in both test cases. For the synthetic test case, the traffic flows' parameters were taken from [22]. For the realistic test case, upper and lower bounds on packet size and inter-arrival time were set as in [18]. Then, the packet size and the inter-arrival time associated to each communication flow were randomly selected within the intervals $[S_{min}, S_{max}]$ and $[T_{min}, T_{max}]$, respectively. For both test cases, the flit size was set to 64 bits.

We evaluate IPDeN against HopliteBuf and a generic VCs-based NoC. For HopliteBuf, we consider 128-deep buffers in the routers. For the VCs-based NoC, we consider one-slot VCs . We measured the traversal and communication times of flits of the flow sets defined by the test cases when traversing IPDeN, HopliteBuf and the VCs-based NoC. We identified for each flow the worst measured traversal time (WMTT) and the worst measured communication traversal (WMCT). Moreover, we computed the average measured traversal time (AMTT) and the average measured communication time (AMCT).

Tables IV-VII show the number of flows, for which IPDeN performs better, the same, or worse than HopliteBuf or the VCs-based NoC. The results are presented for both test cases.

In Tables IV and V, we observe that the WMTT and AMTT of between 59% and 66% of flows are better with IPDeN as compared to HopliteBuf. HopliteBuf performs better than IPDeN for ≈20% of flows, and between 14% and 20% of flows have similar traversal times with both NoCs. In terms

TABLE III
CONFIGURATION DETAILS FOR THE TEST CASES.

Test case	Network size	Number of PEs	Number of PEs generating communication flows	Number of flows	S_{min} (Bytes)	S_{max} (Bytes)	T_{min} (Clock cycles)	T_{max} (Clock cycles)
Synthetic	4x4	16	12	46	20	1443	2	25
Orion	6x6	36	31	187	20	1460	4	375

TABLE IV
IPDEN VS HOPLITEBUF (SYNTHETIC TEST CASE).

	IPDeN<HopliteBuf	IPDeN=HopliteBuf	IPDeN>HopliteBuf
WMTT	27 out of 46 (58.7%)	9 out of 46 (19.6%)	10 out of 46 (21.7%)
WMCT	19 out of 46 (41.3%)	0 out of 46 (0%)	27 out of 46 (58.7%)
AMTT	28 out of 46 (60.9%)	8 out of 46 (17.4%)	10 out of 46 (21.7%)
AMCT	23 out of 46 (50%)	0 out of 46 (0%)	23 out of 46 (50%)

TABLE V
IPDEN VS HOPLITEBUF (ORION TEST CASE).

	IPDeN<HopliteBuf	IPDeN=HopliteBuf	IPDeN>HopliteBuf
WMTT	124 out of 187 (66.3%)	27 out of 187 (14.4%)	36 out of 187 (19.3%)
WMCT	52 out of 187 (27.8%)	0 out of 187 (0%)	135 out of 187 (72.2%)
AMTT	120 out of 187 (64.2%)	27 out of 187 (14.4%)	40 out of 187 (21.4%)
AMCT	49 out of 187 (26.2%)	0 out of 187 (0%)	138 out of 187 (73.8%)

TABLE VI
IPDEN VS VCS-BASED NOC (SYNTHETIC TEST CASE).

	IPDeN<VCs-NoC	IPDeN=VCs-NoC	IPDeN>VCs-NoC
WMTT	19 out of 46 (41.3 %)	8 out of 46 (17.4%)	19 out of 46 (41.3%)
WMCT	26 out of 46 (56.5%)	0 out of 46 (0%)	20 out of 46 (43.5%)
AMTT	14 out of 46 (30.4%)	8 out of 46 (17.4%)	24 out of 46 (52.2%)
AMCT	24 out of 46 (52.2%)	0 out of 46 (0%)	22 out of 46 (47.8%)

TABLE VII
IPDEN VS VCS-BASED NOC (ORION TEST CASE).

	IPDeN<VCs-NoC	IPDeN=VCs-NoC	IPDeN>VCs-NoC
WMTT	80 out of 187 (42.78%)	14 out of 187 (7.48%)	93 out of 187 (49.73%)
WMCT	104 out of 187 (55.6%)	0 out of 187 (0%)	83 out of 187 (44.4%)
AMTT	47 out of 187 (25.1%)	14 out of 187 (7.5%)	126 out of 187 (67.4%)
AMCT	96 out of 187 (51.3%)	0 out of 187 (0%)	91 out of 187 (48.7%)

of communication time (which is the sum of traversal time and injection time), between 50% and 74% of flows have better results with HopliteBuf, while IPDeN performs better for between 26% and 50% of flows. Note that, 73220 flits out of 225995 flits never reach their destination when HopliteBuf was used for the synthetic test case, since the buffers in the routers overflowed even when each router contains a 128-deep buffer. For the realistic test case, the number of lost flits was 79583 out of 301746 flits. This shows that HopliteBuf must be oversized to support realistic communication loads. This is not the case for IPDeN, which only needs a single 3-deep or 5-deep buffer in each router (depending on test case).

In Tables VI and VII, we evaluate IPDeN against the VCs-based NoC. The WMTT, WMCT, and AMCT of more than 50% of the flows are better or the same with IPDeN in contrast to the VCs-based NoC. In term of AMTT, the VCs-based NoC performs better than IPDeN. That is, between 52% and 67% of flows have better results with the VCs-based NoC. This is however at the cost of much complex logic since VC-based routers require three times more hardware resources than a IPDeN router (see Section VI-A).

Tables VIII and IX show the number of flows for which IPDeN performs better, the same, or worse than both HopliteBuf and the VCs-based NoC for the synthetic test case and the real-life case study, respectively.

Figures 8 - 11 present plots showing the WMTT, AMTT, WMCT, and AMCT of some of the flows for both test cases. The segments were selected to show when IPDeN performs better, the same o worse than HopliteBuf and the VCs-based NoC. The results for all communication flows can be found in an Appendix available at https://www.dropbox.com/s/hwvm56f7clbpvaw/Appendix.pdf?dl=0 .

VII. SUMMARY AND CONCLUSION

In this paper, we presented IPDeN, a deflection-based NoC that ensures that flits reach their destinations in the same order as they were injected into the network using a new adaptive buffering mechanism in each router. We showed that the WCTT of flits across the network is deterministic and only depends on the properties of the communication flow under the analysis. Cycle-accurate simulations on a Xilinx Virtex-7 FPGA showed that IPDeN performs better in terms of network communication latency than HopliteBuf, which is, as far as the authors know, the most recent solution for in-order packet delivery inspired by a deflection based routing policy.

VIII. ACKNOWLEDGMENTS

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDP/UIDB/04234/2020); by FCT and the ESF (European Social Fund) through the Regional Operational Programme (ROP) Norte 2020, under PhD grant 2020.06898.BD.

TABLE VIII
IPDeN vs HopliteBuf and IPDeN vs VCs-based NoC (Synthetic test case).

	IPDeN < HopliteBuf and IPDeN < VCs-NoC	IPDeN = HopliteBuf = VCs-NoC	IPDeN > HopliteBuf and IPDeN > VCs-NoC
WMTT	13 out of 46 (28.3%)	4 out of 46 (8.7%)	8 out of 46 (17.4%)
WMCT	13 out of 46 (28.3%)	0 out of 46 (0%)	14 out of 46 (30.4%)
AMTT	11 out of 46 (23.9%)	4 out of 46 (8.7%)	10 out of 46 (21.7%)
AMCT	14 out of 46 (30.4%)	0 out of 46 (0%)	13 out of 46 (28.3%)

TABLE IX
IPDeN vs HopliteBuf and IPDeN vs VCs-based NoC (Orion test case).

	IPDeN < HopliteBuf and IPDeN < VCs-NoC	IPDeN = HopliteBuf = VCs-NoC	IPDeN > HopliteBuf and IPDeN > VCs-NoC
WMTT	62 out of 187 (33.2%)	11 out of 187 (5.9%)	28 out of 187 (15%)
WMCT	40 out of 187 (21.4%)	0 out of 187 (0%)	71 out of 187 (38%)
AMTT	34 out of 187 (18.2%)	11 out of 187 (5.9%)	37 out of 187 (19.8%)
AMCT	37 out of 187 (19.8%)	0 out of 187 (0%)	79 out of 187 (42.2%)

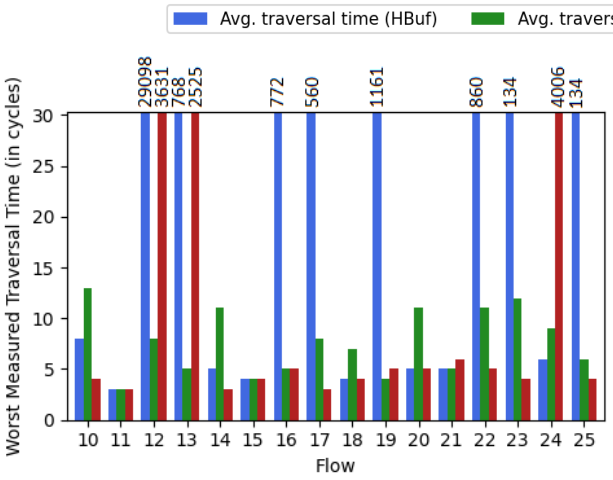


Fig. 6. Worst measured traversal time (Synthetic test case).

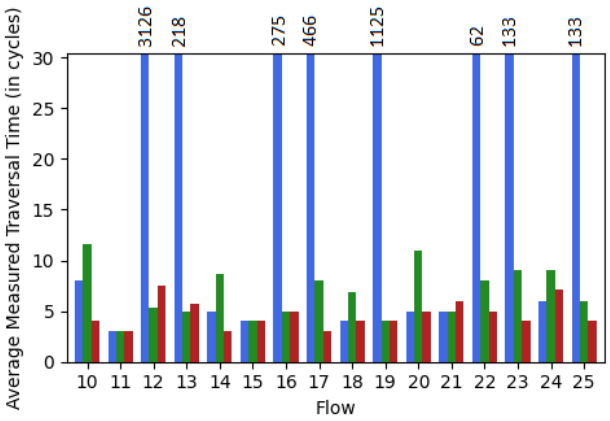


Fig. 7. Average measured traversal time (Synthetic test case).

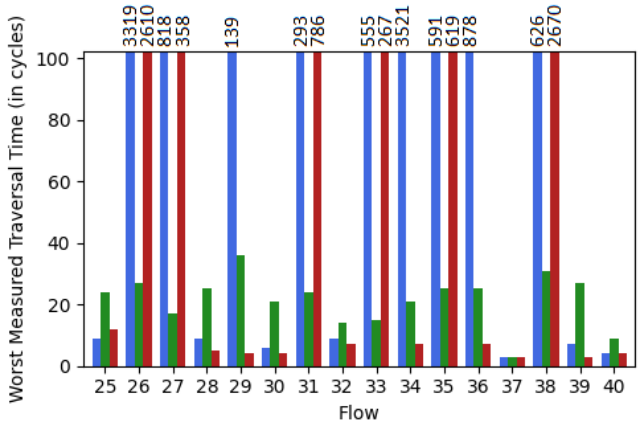


Fig. 8. Worst measured traversal time (Orion test case).

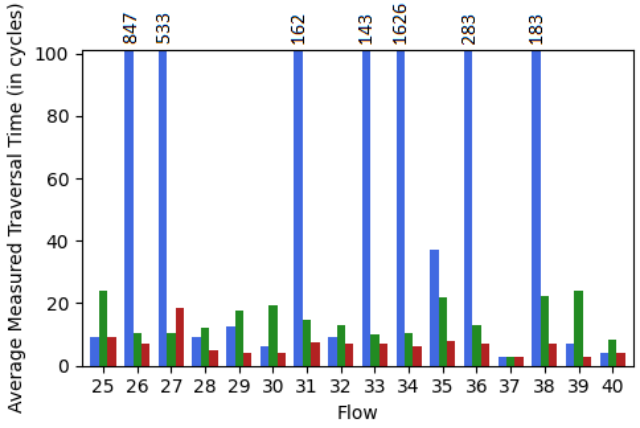


Fig. 9. Average measured traversal time (Orion test case).

References

[1] M. G. Alonso, J. Flich, M. Turki, and D. Bertozzi, "A low-latency and flexible TDM NoC for strong isolation in security-critical systems," in *IEEE 13th Int. Symp. on Emb. Multi/Many-core SoC*, 2019, pp. 149–156.

[2] L. Benini and G. De Micheli, "Networks on chip: a new paradigm for systems on chip design," in *Design, Autom. and Test in Europe Conf. and Exhibition*, March 2002, pp. 418–419.

[3] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129–154, 2005.

[4] T. Bjerregaard and J. Sparsø, "Implementation of guaranteed services in the MANGO clockless network-on-chip," *Computers and Digital Techniques*, vol. 153, no. 4, pp. 217–229, 2006.

[5] M. Daneshtalab, M. Ebrahimi, S. Dytckov, and J. Plosila, "In-order delivery approach for 2D and 3D NoCs," *The Journal of Supercomputing*, vol. 71, no. 8, pp. 2877–2899, 2015.

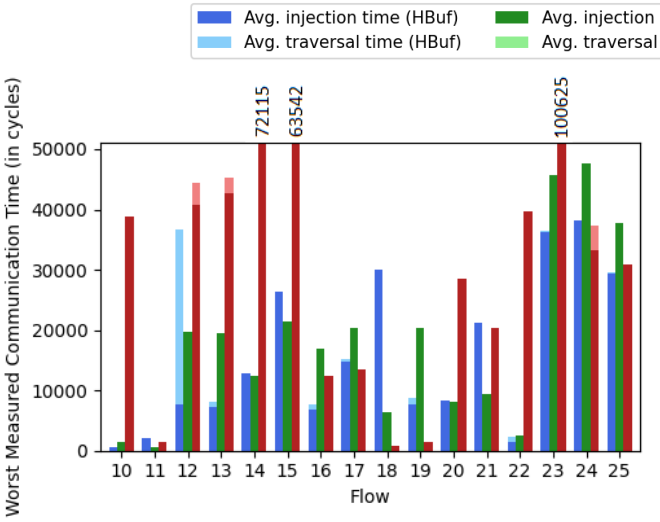


Fig. 10. Worst measured communication time (Synthetic test case).

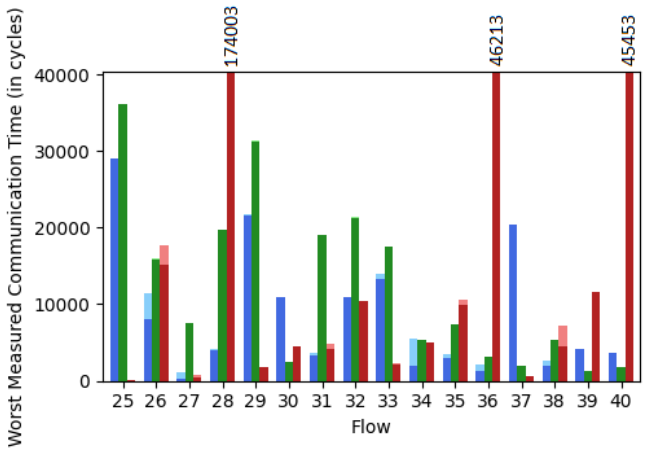


Fig. 12. Worst measured communication time (Orion test case).

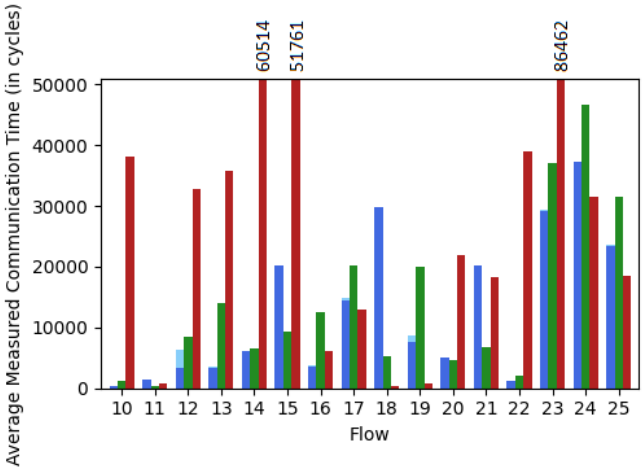


Fig. 11. Average measured communication time (Synthetic test case).

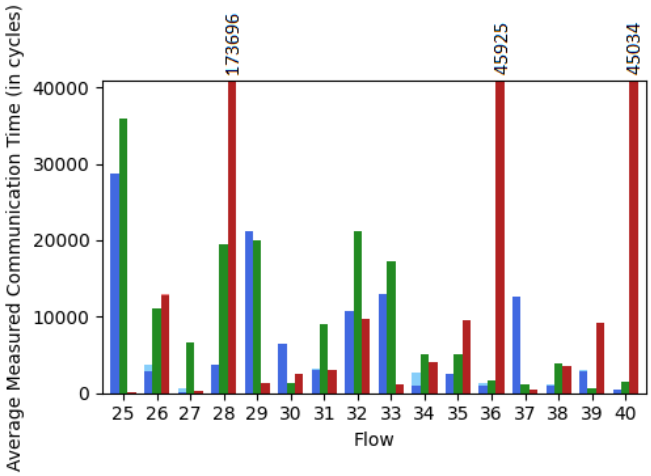


Fig. 13. Average measured communication time (Orion test case).

[6] T. S. Das, P. Ghosal, and N. Chatterjee, "Virtual circuit switch based orderly delivery of packets in adaptive NoC routing," in *12th Int. Workshop on Network on Chip Architectures*, 2019, pp. 1–6.

[7] T. Garg, S. Wasly, R. Pellizzoni, and N. Kapre, "HopliteBuf: FPGA NoCs with provably stall-free FIFOs," in *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays*, 2019, pp. 222–231.

[8] J. Henkel, W. Wolf, and S. Chakradhar, "On-chip networks: A scalable, communication-centric embedded system design paradigm," in *17th Int. Conf. on VLSI Design.*, 2004.

[9] G. S. Malik and N. Kapre, "Enhancing Butterfly Fat Tree NoCs for FPGAs with Lightweight Flow Control," in *IEEE 27th Int. Symp. on Field-Programmable Custom Computing Machines*, 2019, pp. 154–162.

[10] J. Mische, C. Mellwig, A. Stegmeier, M. Frieb, and T. Ungerer, "Minimally buffered deflection routing with in-order delivery in a torus," in *11th IEEE/ACM Int. Symp. on Networks-on-Chip*, 2017, pp. 1–8.

[11] J. Mische and T. Ungerer, "Low power flitwise routing in an unidirectional torus with minimal buffering," in *5th Int. Workshop on NoC Arch.*, 2012, pp. 63–68.

[12] A. Monemi, J. Tang, M. Palesi, and M. N. Marsono, "ProNoC: A Low Latency Network-on-Chip based Many-Core System-on-Chip Prototyping Platform," *Microprocessors and Microsystems*, vol. 54, 09 2017.

[13] M. K. Papamichael and J. C. Hoe, "CONNECT: Re-Examining Conventional Wisdom for Designing Nocs in the Context of FPGAs," in *ACM/SIGDA Int. Symp. on FPGAs*, 2012, pp. 37–46.

[14] M. Paulitsch, E. Schmidt, C. Scherrer, and H. Kantz, "Industrial applications," in *Time-Triggered Communication*. CRC Press, 2018, pp. 331–388.

[15] N. Rathod, S. Balachandran, and N. Gala, "CAERUS: An effective arbitration and ejection policy for routing in an unidirectional torus," in *8th Int. workshop on interconnection network architecture: On-chip, multi-chip*, 2014, pp. 1–4.

[16] Y. Ribot González and G. Nelissen, "HopliteRT*: Real-Time NoC for FPGA," *IEEE Trans. on Comp.-Aided Des. of Integ. Circ. and Sys.*, vol. 39, no. 11, pp. 3650–3661, 2020.

[17] R. A. Stefan, A. Molnos, and K. Goossens, "dAElite: A TDM NoC supporting QoS, multicast, and fast connection set-up," *IEEE Trans. on Computers*, vol. 63, no. 3, pp. 583–594, 2012.

[18] D. Tămaş-Selicean, P. Pop, and W. Steiner, "Design optimization of ttethernet-based distributed real-time systems," *Real-Time Systems*, vol. 51, no. 1, pp. 1–35, 2015.

[19] D. TamasSelicean, P. Pop, and W. Steiner, "Timing analysis of rate constrained traffic for the ttethernet communication protocol," in *IEEE 18th Int. Symp. on Real-Time Dist. Comp.* IEEE, 2015, pp. 119–126.

[20] S. Tobuschat, P. Axer, R. Ernst, and J. Diemer, "IDAMC: A NoC for mixed criticality systems," in *IEEE 19th Int. Conf. on Emb. and Real-Time Computing Systems and Applications*, 2013.

[21] S. Wasly, R. Pellizzoni, and N. Kapre, "HopliteRT: An efficient FPGA NoC for real-time applications," in *2017 Int. Conf. on Field Prog. Tech.*, 2017, pp. 64–71.

[22] L. Zhao, P. Pop, Q. Li, J. Chen, and H. Xiong, "Timing analysis of rate-constrained traffic in ttethernet using network calculus," *Real-Time Systems*, vol. 53, no. 2, pp. 254–287, 2017.