nDimNoC: Real-Time D-dimensional NoC

Yilian Ribot González
CISTER Research Centre, ISEP, Polytechnic Institute of Porto, Portugal

Geoffrey Nelissen
Eindhoven University of Technology, The Netherlands

Eduardo Tovar
CISTER Research Centre, ISEP, Polytechnic Institute of Porto, Portugal

Abstract

The growing demand of powerful embedded systems to perform advanced functionalities led to a large increase in the number of computation nodes integrated in Systems-on-chip (SoC). In this context, network-on-chips (NoCs) emerged as a new standard communication infrastructure for multi-processor SoCs (MPSoCs). In this work, we present nDimNoC, a new D-dimensional NoC that provides real-time guarantees for systems implemented upon MPSoCs. Specifically, (1) we propose a new router architecture and a new deflection-based routing policy that use the properties of circulant topologies to ensure bounded worst-case communication delays, and (2) we develop a generic worst-case communication time (WCCT) analysis for packets transmitted over nDimNoC. In our experiments, we show that the WCCT of packets decreases when we increase the dimensionality of the NoC using nDimNoC's topolgy and routing policy. By implementing nDimNoC in Verilog and synthesizing it for an FPGA platform, we show that a 3D-nDimNoC requires ≈5-times less silicon than routers that use virtual channels (VC). We computed the maximum operating frequency of a 3D-nDimNoC with Xilinx Vivado. Increasing the number dimensions in the NoC improves WCCT at the cost of a more complex routing logic that may result in a reduced operating clock frequency.

2012 ACM Subject Classification Computer systems organization → Real-time systems; Networks → Network on chip

Keywords and phrases Real-Time Embedded Systems, Systems-on-Chips, Network-on-Chips, Worst-Case Communication Time

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2021.5

Funding 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); also by FCT and the ESF (European Social Fund) through the Regional Operational Programme (ROP) Norte 2020, under PhD grant 2020.06898.BD.

1 Introduction

These days, SoCs include more and more heterogeneous processing elements that execute dedicated functions in parallel. Traditional shared communication buses, which used to connect all the computation nodes together, are a major performance bottleneck of modern SoCs. Therefore, NoCs emerged as a new standard communication infrastructure for SoC as they present a scalable and versatile solution for systems with a high level of parallelism [2, 15].

The literature on NoCs is extensive. However, real-time systems add new constraints on the NoC infrastructures. In addition to ensure that messages arrive at their destination in a correct fashion, real-time NoCs must guarantee that packet transmissions respect strong timing constraints [16]. Over the years, there have been several attempts to design real-time NoCs by considering different approaches. A large body of solutions consider a mesh topology and rely on wormhole switching with VCs. That strategy leads to powerful NoC infrastructures with bounded WCCT but they rely extensively on buffers and virtual channels to provide timing guarantees. This makes them expensive to implement in terms of silicon footprint, and increases their power consumption.

© Yilian Ribot González, Geoffrey Nelissen, and Eduardo Tovar;
licensed under Creative Commons License CC-BY 4.0

These last years, buffer-less NoCs have gain popularity as an alternative to VC-based NoCs. Buffer-less NoCs are compact; their implementation cost and power consumption are lower than traditional approaches. Therefore, they are more suitable to (embedded) systems with area and/or power consumption constraints. In [39] and [31] two novel buffer-less deflection-based real-time NoCs called HopliteRT and HopliteRT* were proposed. They ensure predictable timing behaviors, accommodates dynamic workload and have an extremely low hardware consumption footprint. Noticeably, HopliteRT* uses the characteristics of a circulant topology to ensure bounded worst-case communication delays.

NoCs are an attractive and promising alternative for the traditional shared-buses. Yet, most of the existing literature for real-time systems focuses on 2-Dimensional NoCs (2D-NoCs), i.e., where routers are connected according to a mesh or torus topology for example. However, in a non-real-time setting, Romanov [32] shows that circulant topologies possess better characteristics over traditional mesh and torus topologies. Circulant topologies are a type of n-dimensional topologies for networks. Thus, in this work, we explore the design of n-dimensional NoCs architectures compatible with real-time systems requirements.

This line of research is also motivated by the recent evolution in the integrated circuit (IC) industry. Indeed, three-dimensional integrated circuits (3D-ICs) seem to be the future of ICs [19, 5, 20, 33, 29]. 3D-ICs achieve higher performance, while reducing average interconnection length; provide higher packing density thanks to the added third dimension; reduce power consumption; and enhance computation bandwidth. Hence, there is currently a drive towards creating new powerful NoCs solutions that meet the requirements of future large-scale MPSoCs by combining the advantages of 3D integration and NoC architecture.

Contribution. We propose nDimNoC, a new D-dimensional NoC that provides real-time guarantees for systems implemented upon MPSoCs, reduces average network communication latency and provides greater flexibility compared to more traditional 2D NoCs. The main contributions of our work are: (1) to design a new buffer-less router architecture that allows synthesizing D-dimensional NoCs; (2) to propose a new deflection-based routing policy that uses the characteristics of D-dimensional circulant topologies to ensure bounded worst-case communication delays; (3) to develop a generic WCCT analysis for packets transmitted over nDimNoC; (4) to implement a 3D version of nDimNoC in Verilog (a hardware description language) that can be instantiated on a real FPGA platform; and (5) to assess our new design against related works in terms of computed WCCT bounds and hardware requirements.

2 Related work

Most 2D-NoC solutions rely on wormhole switching with virtual channels (VCs) (e.g., CONNECT [27], IDAMC [37]). In [34], Shi et al. propose an analysis of the worst-case network latency for a new real-time fixed priority preemptive wormhole NoC in which each priority level is assigned its own VC. Several variations of that approach were proposed over the years [36, 7, 37, 6, 30], for instance, handling the case where several flows share the same priority [21], changing the routing policy to EDF [25] or supporting communication flows with different criticality levels [3, 18]. The complexity of those designs and their routing policies led to complex WCCT analyses inspired by both the classic real-time system theory [41, 42, 17, 26] and Network Calculus [10, 11].

In [24], a new type of NoC called SBT-NoC was proposed. In this work, Nikolic et al. introduced a global arbitration protocol inspired by the CAN protocol. Theoretical results are promising but this NoC solution has not been implemented in a real platform yet.

Recently, Wasly et al. in [39] proposed a new buffer-less NoC for real-time systems. Their NoC is called HopliteRT. The design of HopliteRT ensures that the WCCT of packets is upper-bounded. HopliteRT* is an evolution of HopliteRT proposed in [31]. It introduces a notion of quality of service in the routing policy and uses a circulant topology in order to improve the packets' WCCT in comparison to HopliteRT.

In [32], various routing strategies, i.e., table routing, Clockwise routing and Adaptive routing, were studied for two-dimensional ring circulant networks. The author shows that several characteristics of NoCs are improved in comparison to mesh and torus topologies when circulant graphs are used as a topological basis.

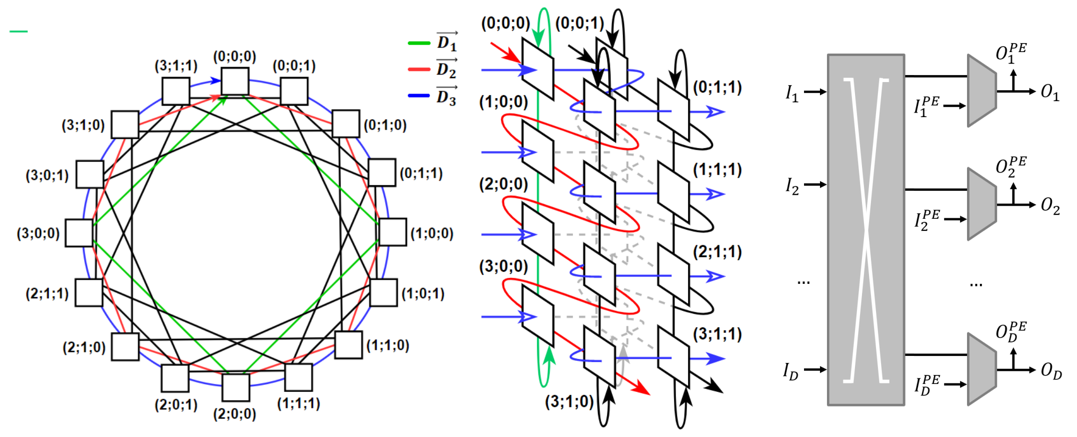
From a 3D-NoC perspective, Park et al. [28] proposed a Multi-layered on-chip Interconnect Router Architecture (MIRA). Their approach assumes 3D processor designs (i.e., processor cores partitioned into multiple layers), and is therefore inadequate for existing highly optimized 2D processor designs. In [9], Ghidini et al. presented a 3D-NoC mesh architecture called Lasio relying on wormhole switching with FIFO queues. In order to minimize packet communication latency and NoC area, Tiny 3D mesh NoC was later proposed in [22]. Tiny NoC reduces the number of routers and links in the network by connecting multiple programming elements to the same router. This solution minimizes the total NoC area as compared to Lasio NoC, however, average packets latency improves only when there are few flows and/or under a low packet injection rate. In [4], a 3D NoC architecture based on De-Bruijn graph was proposed. Tree-based interconnect architectures have been also considered in some works [13, 14, 12]. However, they are very complex to implement due to their irregular and complex network topologies. In [8], a NoC/Bus-based hybrid 3D architecture was proposed, but the approach suffers from low throughput due to inefficient hybridization between the NoC and bus media.

To the best of our knowledge, none of the 3D-NoC solutions developed so far targets real-time systems. Therefore, they do not provide guaranteed upper-bounds on the packets WCCT, and do not come with a WCCT analysis. In this work, we develop a new real-time D-dimensional NoC (with $D \geq 2$) and its associated timing analysis.

3 System model

In this paper, we assume a system composed of N programming elements $\{\pi_1, ..., \pi_N\}$. Each programming element π_q is connected to a different router R_q of a D-dimensional NoC. The coordinates of a router R_q in the D-dimensional network are noted $(r_1^q, r_2^q, ..., r_D^q)$. Each programming element π_q injects a set of n^q communication flows $F^q = \{f_1^q, f_2^q, ..., f_n^q\}$ into the network. A communication flow f_j is defined by the parameters $\{(s_1^j, s_2^j, ..., s_D^j), (d_1^j, d_2^j, ..., d_D^j), C_j, T_j\}$. A communication flow f_j generates a potentially infinite number of packets that are injected at coordinates $(s_1^j, s_2^j, ..., s_D^j)$ of the NoC and must reach the programming element at coordinates $(d_1^j, d_2^j, ..., d_D^j)$. f_j respects a minimum inter-arrival time T_j between the generation of every two packets. Each packet sent by flow f_j is divided in C_j flits that are sequentially injected in the network. Each flit has a size S_{flit} (in bits). We assume that all the routing information is encoded in each flit of the packet, i.e., there is no distinction between header, body or tail flits. The routing information is the coordinates of the destination programming element of the associated flow.

In the rest of this paper, we use the notations $R_{orig}(f_j)$ and $R_{dest}(f_j)$ to refer to the origin and destination router of flow f_j, respectively. That is, $R_{orig}(f_j)$ has coordinates $(s_1^j, s_2^j, ..., s_D^j)$ and $R_{dest}(f_j)$ has coordinates $(d_1^j, d_2^j, ..., d_D^j)$.



(a) Circulant topology $C(16; 1, 2, 4)$. **(b)** Equiv. 4x2x2 grid-based 3D-network. **(c)** nDimNoc router architecture.

Figure 1 nDimNoc's topology and router architecture.

4 nDimNoC architecture

In this section, we present nDimNoC. More specifically, we describe: (1) the network topology, (2) the router architecture, and (3) the routing policy. We later provide the timing analysis for nDimNoC in Section 5.

4.1 NoC topology

Consider a network composed of N routers R_0 to R_{N-1}. In nDimNoC, the routers are connected together according to a ring circulant topology $C(N; g_1, g_2, ..., g_D)$ where $g_1 = 1$, N is the total number of routers, D indicates the dimensionality of the network, and $g_1, g_2, ..., g_D$ are the *generatrices* of the network. We assume that the generatrices follow the following properties: $1 = g_1 < g_2 < ... < g_D < N$, and that their values are harmonic (i.e., for any pair of generatrices g_i and g_j such that $i < j$, g_i is a divider of g_j). Under the circulant topology $C(N; 1, g_2, ..., g_D)$, all routers have D inputs $I_1, I_2, ..., I_D$ and D outputs $O_1, O_2, ..., O_D$ for inter-routers communications. All routers are connected by a single unidirectional ring using one of their inputs and one of their outputs (see blue line in Figure 1a). Then, each router is also connected to the routers that are $g_2, g_3, ..., g_D$ hops away on the ring (see red, green and black lines in Figure 1a). Formally stated, for each router R_q (with $0 \leq q < N$), its u^{th} output port O_u $(1 \leq u \leq N)$ is connected to the u^{th} input port I_u of the router $R_{(q+g_u) \bmod N}$.

A circulant network $C(N; 1, g_2, ..., g_D)$ may also be represented as a S_1xS_2x...xS_D grid-based D-dimensional network, where $S_1, S_2, ...S_D$ correspond to the number of routers on the dimension $\overrightarrow{D_1}$,$\overrightarrow{D_2}$,...,$\overrightarrow{D_D}$, respectively. The size of the network on each dimension can be computed as follows $S_1 = \frac{N}{g_D}$, $S_2 = \frac{g_D}{g_{D-1}}$, $S_3 = \frac{g_{D-1}}{g_{D-2}}$, ..., $S_D = \frac{g_2}{g_1}$. The coordinates $(r_1^q, r_2^q, ..., r_D^q)$ of a router R_q defines the position of the router R_q in the grid representation.

As an example, Fig. 1a shows the circulant network $C(16; 1, 2, 4)$. In Fig. 1b, we provide the equivalent representation as a 4x2x2 grid-based 3-Dimensional network of the circulant network shown in Fig. 1a. The red, green, and blue links in Fig. 1a correspond to the red, green, and blue links in Fig. 1b, respectively.

In the rest of this paper, we often reason about the position pos^q of a router R_q on the main unidirection ring of the circulant topology. That position can be inferred from the coordinates of the router in the grid topology as follows

$$pos^q = \sum_{k=1}^{D} r_k^q \times g_{D-k+1} \tag{1}$$

To simplify some of our further discussions, we define the helping function $\text{dist}(R_q, R_m)$ as the distance between routers R_q and R_m on the main ring, i.e.,

$$\text{dist}(R_q, R_m) = (pos^m - pos^q + N) \bmod N \tag{2}$$

Note that the following properties hold for circulant topologies.

▶ **Property 1.** *Let R_l be a router at which flit p is located. After one hop on dimension $\overrightarrow{D_u}$ of the network, flit p reaches a router R_m located g_{D-u+1} steps further on the main ring of the network, i.e.,* $\text{dist}(R_l, R_m) = g_{D-u+1}$.

Finally, we define $\text{ring}_u(R_q)$ as the set of routers that are on the same ring of dimension $\overrightarrow{D_u}$ as R_q. That is,

$$\text{ring}_u(R_q) = \{R_l \mid \forall b \in [u+1, D],\, r_b^l = r_b^q\} \tag{3}$$

As an example, let R_0 be the router at coordinates $(0; 0; 0)$ in Figure 1a, then all the routers connected by the green links are in $\text{ring}_1(R_0)$, and all the routers connected by red links are in $\text{ring}_2(R_0)$.

4.2 Router architecture

In order to reduce implementation cost in terms of hardware resources utilization and network analysis complexity, nDimNoC does not use VCs and does not rely on extensive buffer.

As we discuss in the previous section, nDimNoC routers have D inputs (i.e., I_1, I_2, ..., I_D) and D output ports (i.e., O_1, O_2, ..., O_D) connected to neighboring routers to allow for inter-routers communication (see Fig. 1c). In addition, all routers also have D input ports (i.e., I_1^{PE}, I_2^{PE}, ..., I_D^{PE}) that may be used by the programming element to inject packets into the network. Therefore, in total, each router has $2 \times D$ input ports and D output ports. A programming element can inject packets on any of the D dimensions $\overrightarrow{D_1}$,$\overrightarrow{D_2}$,...,$\overrightarrow{D_D}$ of the network by using the input ports $I_1^{PE}, I_2^{PE}, ... I_D^{PE}$, respectively. Therefore, several packets may be injected to different dimensions simultaneously. Thus, the waiting times suffered by the packets inside the programming elements decreases. Indeed, in solutions that support a single input port to inject packets into the network, all packets compete for the same input port. In nDimNoc, however, 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 I_u^{PE}.

▶ **Property 2.** *In this paper, we assume that a flit of a flow f_j with origin and destination coordinates $(s_1^j, s_2^j, ..., s_D^j)$ and $(d_1^j, d_2^j, ..., d_D^j)$ is injected in the network using port I_u^{PE}* ***if and only if*** *$s_u^j \neq d_u^j$ and $\forall x \mid u < x \leq D, s_x^j = d_x^j$.*

From Property 2, we get that all the packets of a given flow will be injected using the same input port.

The ports O_1, O_2,..., O_D of a router are connected to the ports I_1, I_2,..., I_D of its neighboring routers, but also serve as inputs to the programming elements. That is, the programming element connected to a router can reads packets from all the output ports

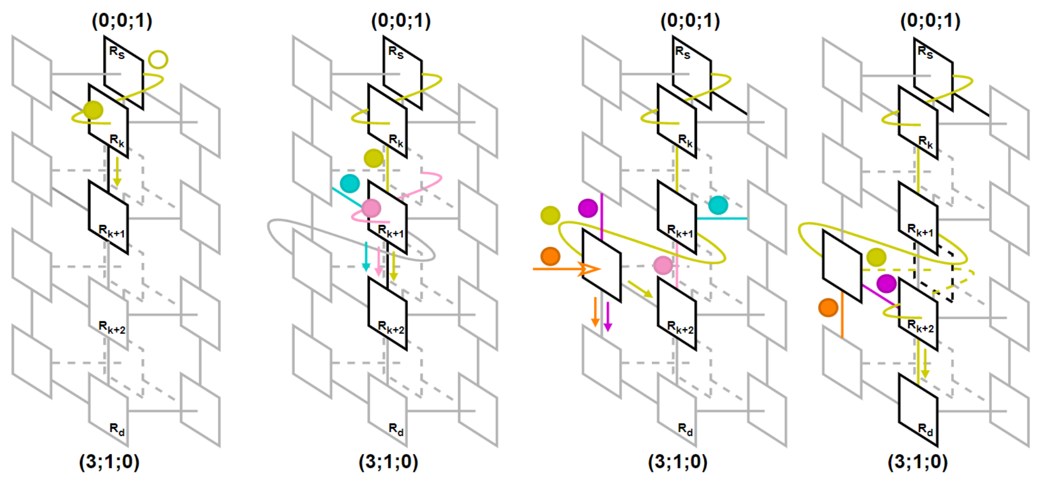
Table 1 Generic routing policy table of nDimNoC with D dimensions.

Rule	**Flows requests**	**Conflicting requests**	**Routing decisions**	**Explanation**
1	$I_D \rightarrow O_D$	None	$I_D \rightarrow O_D$	No contention over O_D.
2	$I_D \rightarrow O_1$	Any	$I_D \rightarrow O_1$	$I_D \rightarrow O_1$ always wins.
3	$I_u \rightarrow O_u$	None	$I_u \rightarrow O_u$	No contention over O_u.
4		I_{u-1} deflected to O_u	$I_u \rightarrow O_{u+1}$	Flows coming from the I_{u-1} and I_u ports conflict over O_u. $I_{u-1} \rightarrow O_u$ always wins over $I_u \rightarrow O_u$. The flow coming from the I_u port is deflected to the O_{u+1} port.
5	$I_u \rightarrow O_1$	None **or** $I_{v<u} \rightarrow O_1$	$I_u \rightarrow O_1$	No flow entering by a port on a higher dimension than I_u requests O_1. $I_u \rightarrow O_1$ wins.
6		$I_{v>u}$	$I_u \rightarrow O_{u+1}$	A flow entering by a port on a higher dimension than I_u wins O_1. The flow coming from the I_u port is deflected to the O_{u+1} port.
7	$I_u^{PE} \rightarrow O_u$	None	$I_u^{PE} \rightarrow O_u$	There is no flow coming from another port that requests O_u. The flow on I_u^{PE} is injected in the network via O_u.
8		$I_v \rightarrow O_u$ and/or I_{u-1} deflected to O_u	None	The flow waiting on the I_u^{PE} port conflicts over the O_u port with flows coming from neighboring routers. Since flows from I_u^{PE} have the lowest priority, the flow waiting on the I_u^{PE} port is not injected in the network.

O_1, O_2,..., O_D. We show this property (i.e., that the programming element has read-access to all output ports of the router) using the notations O_1^{PE}, O_2^{PE}, ..., O_D^{PE} in Fig. 1c. We assume that a programming element can read packets from several different output ports simultaneously. This may be done by considering that each programming element has a FIFO queue connected to each port O_u^{PE} (with $1 \leq u \leq D$). We assume that those FIFO queues are large enough to prevent back pressure in the network. Although this design solution may lead to increased router programming logic complexity, it avoids the extra cost of implementing expensive exit multiplexers.

We consider that each programming element has also a FIFO queue connected to each port I_u^{PE}. These queues store flits that are pending to be injected in the network. Note that, the FIFO queues connected to each O_u^{PE} and I_u^{PE} port could be implemented in software or hardware. Their specific implementation is irrelevant to the matter discussed in this work.

We assume that no traffic injection regulator exists at the programming elements. Therefore, they can inject flits into the network as fast as possible. Nonetheless, we assume that each flow f_j can have a maximum of one packet in the FIFO queue pending to be injected in the network at any moment in time. That is, only after a packet is injected, a new packet from the same flow f_j can be stored in the FIFO queue. The implicit assumption is that the minimum inter-arrival time T_j between the generation of every two packets of f_j is larger or equal than the worst-case packet injection time $wcit_j$ of that flow, i.e., $\forall f_j, T_j \geq wcit_j$. Note that, the restriction is only related to the content of the FIFO queues at the injection ports



(a) Packet request to use I_1.

(b) Situation after routing arbitration and new packet requests.

(c) Situation after routing arbitration and new packet requests.

(d) Situation after routing arbitration.

Figure 2 nDimNoc's routing policy example.

and does not limit the number of in-flight packets in the network. That is to say, several packets from the same flow f_j can be traveling around the network at the same time. Also note that this assumption is less constraining than those made in many works on real-time NoCs that assume periods larger than the worst-case communication time (of which the injection time is just one component).

4.3 Routing policy

Consider a flit that must travel from router (0;0;0) to router (2;0;0) in the example network of Figure 1a. It will reach its destination faster if it travels on the green link than if it hops through the red or blue links. Since the green, red and blue links correspond to dimensions $\overrightarrow{D_1}$,$\overrightarrow{D_2}$, and $\overrightarrow{D_3}$, respectively, it is equivalent to say that it is faster for the flit to travel on a dimension of lower order. nDimNoC's routing policy simply builds upon that property. Additionally, it uses the idea of deflection routing [1] to avoid the cost of packet buffering. The approach is as follows.

Consider a flit of a flow f_j that has been injected at the origin router $(s_1^j, s_2^j, ..., s_D^j)$ and with destination $(d_1^j, d_2^j, ..., d_D^j)$. As mentioned in Section 4.2, the programming element injects that flit on port I_u^{PE} such that $s_u^j \neq d_u^j$ and $\forall x \mid u < x \leq D,\ s_x^j = d_x^j$.

If the flit was transmitted in isolation (i.e., without any interfering flow), it would travel along the dimension $\overrightarrow{D_u}$ of the network by entering in each router by input port I_u and requesting output port O_u. Then, when it reaches the first router R_k with the same coordinates r_2^k, r_3^k,...,r_D^k as its destination (i.e., $r_b^k = d_b^j, \forall b \in [2, D]$), it would request the output port O_1^k and travel along dimension $\overrightarrow{D_1}$ until reaching its destination. It results that flits entering by input port I_u (such that $2 \leq u \leq D$) may only request the output port O_u or O_1. Flits entering by the input port I_1 may only request the output port O_1.

If there is interfering traffic, nDimNoC's routing policy allows flits to be "deflected" to make place for "higher priority" traffic. Two such scenarios may happen:

1. If multiple flits entering by different input ports request the output port O_1 at the same time, nDimNoC always gives the highest priority to the flit that entered by the input port with highest dimension (i.e., I_D wins over I_{D-1}, which wins over I_{D-2}, etc.). Consider

two flits entering by ports I_u and I_v such that $u < v$ and that request output port O_1. Then, the flit entering by I_v exists through O_1, and the flit entering by I_u exists through O_{u+1}. We say that the flit that entered by I_u is deflected to dimension $\overrightarrow{D_{u+1}}$.

2. A flit entering by port I_u that was deflected to the output port O_{u+1} may now conflict for port O_{u+1} with a flit coming from I_{u+1} and that requests O_{u+1} at the same time. Under this contention scenario, the flit coming from I_u and that was deflected towards O_{u+1} wins the right to use O_{u+1} and the flit coming from I_{u+1} is deflected towards the output port O_{u+2}.

Note that deflections redirect deflected flits on longer paths towards their destination. However, the topology presented in Section 4.1 ensures that it still progresses towards its destination router. Therefore, nDimNoC's routing policy is deadlock-free and livelock-free. Furthermore, after each deflection, a flit's priority to request output port O_1 in a future router increases (since flits traveling on higher dimensions have higher priorities). Therefore, its probability to be able to later travel on a shorter route increases too.

Finally, flits injected by the programming element (i.e., flits entering by any port I_u^{PE}), always have the lowest priority and must wait for the respective port O_u to be free. Table 1 summarizes the routing policy of a D-dimensional nDimNoC.

Example. Consider a 4x2x2 3-dimensional nDimNoC (i.e., $D = 3$) (see Figure 2a-2d). Each 3D-nDimNoC router has six input ports (I_1, I_2, I_3, I_1^{PE}, I_2^{PE}, and I_3^{PE}) and three output ports (O_1, O_2, and O_3). Consider also a flit of a flow f_j (yellow flit in Figure 2a) with origin and destination coordinates $(0; 0; 1)$ and $(3; 1; 0)$, respectively. Since $s_3^j \neq d_3^j$, the flit is injected via input port I_3^{PE} (see Figure 2a). The flit then travels along the dimension $\overrightarrow{D_3}$ until it reaches router R_k with the same coordinates r_2^k, r_3^k,...,r_D^k as its destination, i.e., $r_2^k = d_2^j = 1$ and $r_3^k = d_3^j = 0$ (see Figure 2a). In R_k, the flit enters by input port I_3 and requests output port O_1 to travel along the dimension $\overrightarrow{D_1}$ until its destination (see Figure 2b). According to rule 2 of nDimNoC's routing policy (see Table 1), it has the highest priority to use O_1 and therefore enters the router $(1; 1; 0)$ (next router to R_k on dimension $\overrightarrow{D_1}$) by its port I_1, and requests port O_1 (see Figure 2b). If a flit enters by the input I_2 (blue flit in Figure 2b) and/or I_3 port (pink flit in Figure 2b) and request O_1 at the same time as the yellow flit, then the yellow flit is deflected to the output port O_2 (see Figure 2c and rule 6 in Table 1). Thus, it must now travel along dimension $\overrightarrow{D_2}$ until it reaches the same router as it would have if it could have used the O_1 port instead. Note that the yellow flit may still suffer additional deflections to dimension $\overrightarrow{D_3}$ in any router it reaches while traveling along dimension $\overrightarrow{D_2}$ as it is the case on Figure 2c where both a flit entering by the I_1 port (violet flit) and a flit entering by the I_3 port (orange flit) request the O_1 port. Then, the request $I_3 \rightarrow O_1$ wins over the other requests and the flits entering by the I_1 and I_2 ports are deflected to the O_2 and O_3 ports, respectively (see Fig. 2d and rule 4 in Table 1).

5 Bound on the worst-case communication time

In Section 4, we presented nDimNoC's design. In this section, we present an analysis for the worst-case communication time (WCCT) between two processing elements connected with nDimNoC. The WCCT of a packet is defined as the sum of the maximum amount of time *wcit* during which the last flit of the packet must wait in the programming element before to be injected into the network, and the maximum amount of time *wctt* taken by any flit of the packet to traverse the network and reach its destination. We refer to those as the worst-case injection time ($wcit_j$) and the worst-case traversal time ($wctt_j$) of flow f_j, respectively. Then, the WCCT of a packet of a flow f_j is defined as:

$$wcct_j = wcit_j + wctt_j, \tag{4}$$

5.1 Worst-case and best-case traversal time

In this section, we compute bounds on the worst- and best-case traversal time of a flit p (abbreviated WCTT and BCTT, respectively). A bound on the WCIT is later derived in Section 5.2.

As discussed in the previous section, a flit p of flow f_j that travels through nDimNoC can be deflected in any router on its path to its destination, but there is only a limited set of routers in which it can *actively* request to change the dimension it travels along. Those routers are (i) the origin router of the flit with coordinates $(s_1^j, s_2^j, ..., s_D^j)$, and (ii) every router R_k on the path of p such that its coordinates respect $r_b^k = d_b^j$, $\forall b \in [2, D]$. We formally denote this set of routers by $\mathcal{R}$ where

$$\mathcal{R} = \{R_k \mid \forall l \in [1, D],\ r_l^k = s_l^j\ \lor\ \forall b \in [2, D], r_b^k = d_b^j\}. \tag{5}$$

Note that the destination router of flow f_j is obviously in $\mathcal{R}$ since that router has the coordinates $(d_1^j, d_2^j, ..., d_D^j)$.

As will be shown later in this section, the routing decisions in the routers in $\mathcal{R}$ are the only ones that must be analyzed to get a bound on the BCTT and WCTT of a flit p.

We use a directed acyclic graph (DAG) G to compute the WCTT and BCTT of a flit of a flow f_j that traverses an D-dimensional nDimNoC. A DAG $G = (V, E)$ is formed by a set of vertices V and a set of edges E. Each edge $e \in E$ connects two vertices u and v in E. We note $e = (u, v)$. Each edge is assigned a weight $w(u, v)$.

The DAG compactly represents all the routes that the flit p may potentially follow (from its origin to its destination) when it traverses nDimNoC. Each vertex v in the DAG G represents one input port of a router in $\mathcal{R}$. Let $R(u)$ and $I(u)$ be the router and the input port of the router represented by vertex u in the graph, then we note $u = (R(u), I(u))$. Each edge $e = (u, v) \in E$ connecting vertices u and v represents a possible path taken by the flit from input port $I(u)$ of router $R(u)$ to the input port $I(v)$ of another router $R(v)$ on its path. The weight of the edge $e = (u, v)$ is the maximum number of hops from $I(u)$ to $I(v)$ according to that path. Additionally, we label each edge $e = (u, v)$ with the specific output port taken by the flit in router $R(u)$.

Example. Figure 3 shows the DAG of the example of Section 4.3 (Figure 2). It shows the potential paths that a flit of a flow f_j may follow from the origin router $(0; 0; 1)$ to the destination router $(3; 1; 0)$ when it traverses a 4x2x2 3-dimensional nDimNoC. The source vertex v_0 at level 0 of the graph represents the input port I_3^{PE} of the origin router R_s at which the flit is injected by the programming element. Since the flit p is injected by I_3^{PE}, it can only exist by output port O_3 of R_s. R_k is the first router p reaches after leaving R_s where p may request output port O_1. Flit p may only enter R_k by the input port I_3. Vertex v_1 on Level 1 represents input port I_3 of R_k. The weight w_1 is the number of hops the flit p does from the O_3 port of R_s to the I_3 port of R_k. In router R_k, the flit enters by the I_3 port and requests the O_1 port to travel along dimension $\overrightarrow{D_1}$. According to rule 2 of nDimNoC's routing policy (see Table 1), the routing decision for that request is always $I_3 \rightarrow O_1$ (because any flit entering by the I_3 port has the highest priority to use the O_1 port in a 3-dimensional

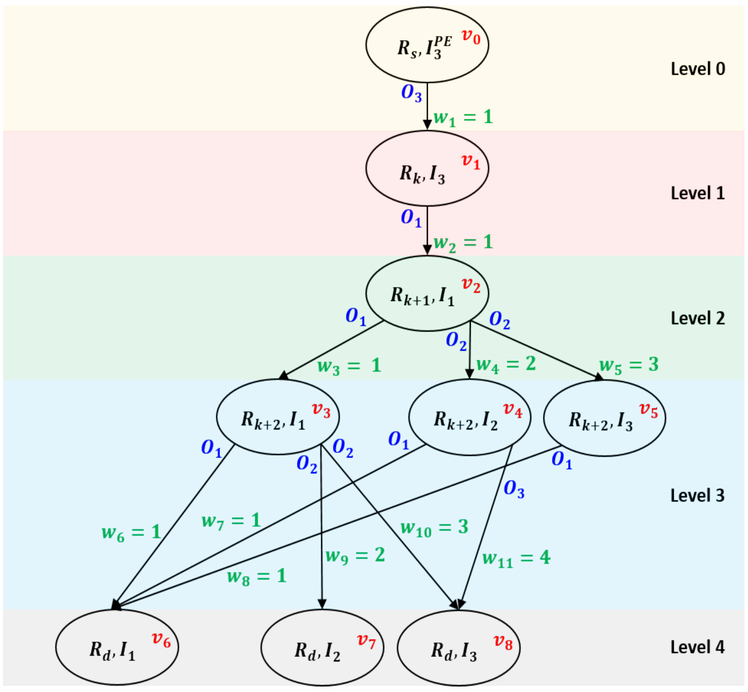


Figure 3 DAG of the potential trajectories that a flit may take from the origin $(0;0;1)$ to the destination $(3;1;0)$ when it traverses a 4x2x2 3-D nDimNoC.

nDimNoC). Therefore, p reaches router R_{k+1} in one hop, and certainly enters R_{k+1} by port I_1. Since R_{k+1} is also in $\mathcal{R}$, it is represented by vertex v_2. According to Table 1, two different routing decisions may be taken in R_{k+1}: (1) p is routed to the O_1 port if there is no conflict over O_1 (see rule 5 in Table 1); or (2) p is deflected to the O_2 port if there is one or more flows coming from other ports that request the O_1 port at the same time as p (see rule 6 in Table 1). If situation (1) happens (i.e., the flit under analysis is routed to the O_1 port), it enters the router R_{k+2} using port I_1. We represent the I_1 port of R_{k+2} as vertex v_3 in Level 3. If situation (2) occurs (i.e., the flit is deflected to the O_2 port), it may enter router R_{k+2} from: (1) the I_2 port if it suffer no further deflection to reaching R_{k+2} (see rule 3 in Table 1) or (2) the I_3 port if it suffers more deflections on its path to R_{k+2} (see rule 4 in Table 1). We represent the I_2 and I_3 ports of R_{k+2} as vertices v_4 and v_5 in the level 3 of the graph, respectively. Considering the potential routing decisions to which the flit p may be subjected after it enters R_{k+2} by the ports I_1, I_2 or I_3, p may reach its destination router R_d by input ports I_1, I_2 or I_3 (vertices v_6, v_7, and v_8 on Level 4) in a maximum number of hops represented by the weights of the edges connecting the vertices of level 3 to those of level 4. Note that, the flit will always be received by the programming element regardless of the routing decision taken in the destination router.

After building the graph G as exemplified above, the WCTT of flit p is the longest weighted path in graph G, and its BCTT is the shortest weighted path in G. For the example of Figure 3, the WCTT is thus equal to 8 and corresponds to the case where the flit p follows the path represented by vertices v_0, v_1, v_2, v_4 and v_8. Similarly, taking the shortest weighted path, we get that the BCTT of p is 4 in that example. Note that the WCTT may not always be obtained when the flit experiences its maximum number of deflection, hence the need for building the full graph G.

Following the reasoning above, the graph G can systematically be built using Algorithm 1. Algorithm 1 uses Lemmas 1 to 5 to compute the set of input and output ports to which the flit p may be routed in each router in $\mathcal{R}$, and to compute the weight of each edge. We now present and prove those lemmas.

In the following, we denote by R_{cur} and R_{next} any two routers in $\mathcal{R}$ such that R_{next} is the first router in $\mathcal{R}$ reached by p after leaving R_{cur}.

▶ **Lemma 1.** *A flit p of a flow f_j that enters router R_{cur} by port I_u^{PE} will be routed to the output port O_u.*

Proof. By rule 7 of nDimNoC's routing policy (Table 1). ◀

Algorithm 1 Building the DAG of the potential trajectories.

```
Input: flow $f_j$;
Output: $V$, $E$;
$V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
Build set $\mathcal{R}$ according to Equation (5);
$R_{cur} \leftarrow$ source router of $f_j$;
$I_u \leftarrow$ input port by which $f_j$ is injected in its source router according to Property 2;
Create vertex $v_{cur} = (R_{cur}, I_u)$;
$V \leftarrow V \cup \{v_{cur}\}$;
$\Gamma^I \leftarrow \{I_u\}$;
$\Gamma^I_{new} \leftarrow \emptyset$;
while $R_{cur}$ is not the destination router of $f_j$ do
    $R_{next} \leftarrow$ first router in $\mathcal{R}$ reached by any flit of $f_j$ after it leaves $R_{cur}$;
    foreach $I_{cur} \in \Gamma^I$ do
        $v_{cur} \leftarrow$ get vertex $(R_{cur}, I_{cur})$ in $V$;
        $\Gamma^O \leftarrow$ Set of output ports to which the flit may be routed if it enters $R_{cur}$ by the
          input port $I_{cur}$ ;                                  // use Lemmas 1 and 2
        foreach $O_{cur} \in \Gamma^O$ do
            $\Gamma^I_{next} \leftarrow$ Set of input ports by which the flit may enter $R_{next}$ if it exits $R_{cur}$ by
              the output port $O_{cur}$ ;                         // use Lemmas 3 and 4
            foreach $I_{next} \in \Gamma^I_{next}$ do
                if $I_{next} \notin \Gamma^I_{new}$ then
                    $\Gamma^I_{new} \leftarrow \Gamma^I_{new} \cup \{I_{next}\}$ ;
                    Create vertex $v_{next} = (R_{next}, I_{next})$;
                    $V \leftarrow V \cup \{v_{next}\}$;
                else
                    $v_{next} \leftarrow$ get vertex $(R_{next}, I_{next})$ in $V$;
                end
                Create edge $e = (v_{cur}, v_{next})$ with weight $h^{R_{cur} \to R_{next}}_{O_{cur} \to I_{next}}$ ;   // use Lemma 5
                $E \leftarrow E \cup \{e\}$;
            end
        end
    end
    $\Gamma^I \leftarrow \Gamma^I_{new}$ ;
    $\Gamma^I_{new} \leftarrow \emptyset$ ;
    $R_{cur} \leftarrow R_{next}$;
end
```

▶ **Lemma 2.** *A flit p that enters router R_{cur} by port I_u may be routed to any of the output ports belonging to the set Γ_u^O, such that*

$$\Gamma_u^O = \begin{cases} \{O_1\} & \textit{when } u = D \\ \{O_1\} \cup \{O_{u+1}\} & \textit{when } u \neq D \end{cases} \tag{6}$$

Proof. According to rule 2 in Table 1, a flit entering by the I_D port has the highest priority to use the O_1 port and will never be deflected to any other output port. This proves the first case of Equation (6). If the flit enters the router by an input port I_u such that $u < D$ and requests output port O_1, two scenarios may happen according to Table 1: (1) it wins port O_1 (see rule 5 in Table 1), or (2) it is deflected to port O_{u+1} (see rule 6 in Table 1). This proves the second case of Equation (6). ◀

▶ **Lemma 3.** *Let O_u be the output port by which flit p exits the router R_{cur}. If R_{next} is only one hop further on dimension $\overrightarrow{D_u}$, then flit p enters R_{next} by its port I_u.*

Proof. Since, according to the network topology defined in Section 4.1, the output port O_u of R_{cur} is connected to the input port I_u of R_{next}, and because flit p exits R_{cur} by port O_u, the lemma follows. ◀

▶ **Lemma 4.** *Let O_u be the output port by which flit p exits the router R_{cur}. If R_{next} is more than one hop away from R_{cur} on dimension $\overrightarrow{D_u}$, then the flit p will enter R_{next} by one of the input ports belonging to the set Γ_u^I, such that,*

$$\Gamma_u^I = \{I_v \mid u \leq v \leq D\} \tag{7}$$

Proof. If R_{next} is more than one hop away from R_{cur} on dimension $\overrightarrow{D_u}$, flit p must hop through at least one other router between R_{cur} and R_{next}. First, we note that by definition, R_{next} is the first router after R_{cur} on flit p's path where p may request output port O_1. Thus, according to nDimNoC's routing policy (Section 4.3), p may only continue to travel along the same dimension (see rule 3 in Table 1) or be deflected to a higher order dimension while traveling between R_{cur} and R_{next} (see rule 4 in Table 1).

If no deflection happens in the routers located between R_{cur} and R_{next}, flit p will enter R_{next} by input port I_u. However, according to rule 4 of nDimNoC's routing policy (Table 1), if $u < D$, the flit p may also be deflected to dimension $\overrightarrow{D_{u+1}}$ in one of those intermediate routers. If no further deflection happen until reaching R_{next}, p will then enter R_{next} by the input port I_{u+1}. Yet, if $u + 1 < D$, Table 1 states that the flit p may still be deflected to dimension $\overrightarrow{D_{u+2}}$ while traveling along dimension $\overrightarrow{D_{u+1}}$. Repeating this reasoning, we get that flit p may enter R_{next} by any input port I_v such that $u \leq v \leq D$. ◀

▶ **Lemma 5.** *The maximum number of hops from the output port O_u of R_{cur} to the input port I_v of R_{next} (with $u \leq v$) is upper bounded by*

$$h_{O_u \to I_v}^{R_{cur} \to R_{next}} = (v - u) + \frac{(pos^{next} - pos^{cur'} + N) \bmod N}{g_{D-v+1}} \tag{8}$$

where

$$pos^{cur'} = (pos^{cur} + \sum_{k=u}^{v-1} g_{D-k+1}) \tag{9}$$

and pos^{cur} and pos^{next} are computed with Equation (1).

Proof. According to nDimNoC's routing policy and following the same explanation as in the proof of Lemma 4, a flit that exits R_{cur} by port O_u and enters R_{next} by port I_v must have been deflected exactly $(v-u)$ times.

According to Property 1, flit p bypasses g_{D-k+1} routers on the main ring of the network with every hop it does on dimension $\overrightarrow{D_k}$. Because, by definition of our circulant topology, we have $g_{D-k+1} > g_{D-k}$ for all k, the flit p will make its maximum number of hops when it suffers its $(v-u)$ deflections as early as possible and thus travels as long as possible along the highest order dimension, i.e., along $\overrightarrow{D_v}$.

In such scenario, the flit does exactly one hop on each dimension $\overrightarrow{D_u}$, $\overrightarrow{D_{u+1}}$, $\overrightarrow{D_{u+2}}$, ..., $\overrightarrow{D_{v-1}}$ and bypasses $\sum_{k=u}^{v-1} g_{D-k+1}$ routers on the network's main ring. Thus, after the $(v-u)$ initial hops, the flit reaches router $R_{cur'}$ situated $\sum_{k=u}^{v-1} g_{D-k+1}$ steps further than R_{cur} on the main ring. That is, the position of $R_{cur'}$ on the main ring is given by Equation (9).

Since the network contain N routers on its main ring, the router R'_{cur} and R_{next} are still $(pos^{next} - pos^{cur'} + N) \bmod N$ steps away from each other on that ring. However, since the flit p only travels along dimension $\overrightarrow{D_v}$ after it reached R'_{cur}, it bypasses g_{D-v+1} routers of the main ring at each hop. Thus, it needs $\frac{(pos^{next} - pos^{cur'} + N) \bmod N}{g_{D-v+1}}$ hops from port O_v of $R_{cur'}$ to port I_v of R_{next}. Therefore, in total, flit p does $(v-u)$ hops to reach R'_{cur} and $\frac{(pos^{next} - pos^{cur'} + N) \bmod N}{g_{D-v+1}}$ additional hops to reach R_{next}, hence proving Equation (8). ◀

▶ **Corollary 6.** *If $u = v$, then $h_{O_u \to I_v}^{R_{cur} \to R_{next}}$ is an exact bound on the number of hops between the output port O_u of R_{cur} and the input port I_v of R_{next}.*

Proof. According to nDimNoC's routing policy, any deflection of a flit p between R_{cur} and R_{next} would result in p entering R_{next} by an input port I_v such that $v > u$. Therefore, if $u = v$, flit p must not have suffered any deflection and must have travel along dimension $\overrightarrow{D_u}$ only. Because $(pos^{next} - pos^{cur} + N) \bmod N$ is the distance between R_{cur} and R_{next} on the main ring of the network, and because for every hop on dimension $\overrightarrow{D_u}$, packet p bypasses g_{D-u+1} routers on the main ring (by Property 1), we have that p reaches R_{next} in exactly $\frac{(pos^{next} - pos^{cur} + N) \bmod N}{g_{D-v+1}}$ hops. Note that this last equation is equal to $h_{O_u \to I_v}^{R_{cur} \to R_{next}}$ when $v = u$, which proves the claim. ◀

We now prove that the WCTT and BCTT of a flit of flow f_j are bounded by the longest and the shortest weighted path of the graph G returned by Algorithm 1, respectively. To do so, we first prove that the graph G built using Algorithm 1 contains all routes that may be taken by packets of flow f_j between its origin and destination.

▶ **Lemma 7.** *The DAG built using Algorithm 1 contains one edge for each possible path between any two routers in $\mathcal{R}$ that may be successively traversed by any flit of flow f_j.*

Proof. Algorithm 1 iterates over all routers in $\mathcal{R}$ that are on the path of f_j from its origin to its destination router (Lines 3, 9, 10 and 31). For each pair of routers R_{cur}, R_{next}, the algorithm computes the set Γ^I of all input ports by which f_j may enter R_{cur}. For each such input, it uses Lemmas 1 and 2 to compute the set Γ^O of all output ports by which f_j may exit R_{cur} (Line 13). For each output port $O_{cur} \in \Gamma^O$, it then uses Lemmas 3 and 4 to compute the set Γ^I_{next} of all input ports by which f_j may enter R_{next} (Line 15). It finally creates an edge for every path between O_{cur} and the input ports in Γ^I_{next} (Line 24). Since Lemmas 1 to 4 were all proven correct, we have that Algorithm 1 creates an edge for every possible path between any two routers R_{cur} and R_{next} in $\mathcal{R}$, i.e., one edge for any combination of output and input port of R_{cur} and R_{next} that may be successively traversed by a packet of f_j. ◀

Lemma 7 has the following corollary as direct consequence.

▶ **Corollary 8.** *The DAG built using Algorithm 1 contains all possible paths taken by flow f_j from its origin to its destination router.*

▶ **Theorem 9.** *The longest weighted path of the DAG built with Algorithm 1 is an upper bound on the WCTT of any flit of flow f_j.*

Proof. By Lemma 7 and Corollary 8, the DAG built with Algorithm 1 contains all possible routes from the origin to the destination of f_j encoded as a different path in the graph. Furthermore, by Lemma 5 and Line 24 of Algorithm 1, the weight of every edge in the graph is an upper bound on the number of hops on the longest path between the output and input ports of the two routers represented by the vertices connected by that edge. Thus, the longest weighted path in the graph is an upper bound on the number of hops between all routers on the path of f_j from its origin to its destination. This proves the Theorem. ◀

▶ **Theorem 10.** *The shortest weighted path of the DAG built with Algorithm 1 is the BCTT of any flit of flow f_j.*

Proof. According to nDimNoC's routing policy and its discussion in Section 4.3, a flit p of flow f_j performs its minimum number of hops between its origin and destination router when it does not suffer any deflection.

Since the DAG built with Algorithm 1 contains all possible routes from the origin to the destination of f_j encoded as a different path in the graph (by Lemma 7 and Corollary 8), it also contains the path where the flit of f_j does not suffer any deflection. Furthermore, by Corollary 6 and Line 24 of Algorithm 1, the weight of every edge corresponding to a path where p does not suffer deflection is equal to the exact number of hops performed by p on that path. Therefore, the shortest weighted path in the graph is an exact bound on the BCTT of f_j from its origin to its destination. This proves the Theorem. ◀

5.2 Worst-case injection time

In the previous section, we explained how to compute bounds on the BCTT and WCTT of any flit of a packet injected by a flow f_j. In this section, we derive a bound on the worst-case injection time WCIT of any packet of f_j (see Theorem 12).

First, we recall a bound on the maximum number of packets that may be injected in the network by any flow f_j. This bound was already proven in [31].

▶ **Lemma 11.** *In any time interval of length Δt, the flow f_j can inject in the network at most $\lambda_j(\Delta t) = \min\left\{\Delta t,\ \left\lceil \frac{\Delta t + wcit_j}{T_j} \right\rceil C_j\right\}$ flits.*

Proof. The proof is similar to that of the maximum workload that can be executed by a task with minimum inter-arrival time T_j and release jitter $wcit_j$. The complete proof is provided in Lemma 14 of [31]. ◀

Then, we prove an upper bound on the WCIT of any packet of f_j using Theorem 12. To prove that theorem, we use the following notation: flow f_j is injected in router R_{inj} via input port I_{inj}^{PE} (i.e., R_{inj} is the origin router of f_j). We define $\mathcal{F}_{inj}$ as the set of all flows (including f_j) injected in the same input port I_{inj}^{PE} of the same router R_{inj} as f_j. Note that this set of flows is a property of the system and thus we assume that it is given as an input to the analysis. We also define Γ_{inj}^{conf} as the set of all flows originating from other routers than R_{inj} and that may conflict with the injection of flow f_j in router R_{inj}. The content of Γ_{inj}^{conf} is computed using Lemmas 13 and 17 proven later in this section.

▶ **Theorem 12.** *The WCIT $wcit_j$ of any packet of flow f_j is given by the smallest positive solution to the recursive equation*

$$wcit_j \geq \left(\sum_{\forall f_i \in \mathcal{F}_{inj}} C_i \right) - 2 + \sum_{\forall f_l \in \Gamma_{inj}^{conf}} \lambda_l(wcit_j + 1 + J_l) \tag{10}$$

where $J_l = wctt'_l - bctt'_l$ is the difference between the worst-case and the best-case traversal time of flow f_l until router R_{inj} (computed with Theorems 9 and 10).

Proof. Let p be the last flit of any packet of flow f_j. According to nDimNoC's routing policy, the flit p will be injected in the network as soon as : 1) all flits ahead of p in the FIFO queue of the input port by which it is injected in the network have been injected, and 2) there is one clock cycle during which no packet entering R_{inj} from other input ports conflicts for the same output port as p.

Let n_{flits} be the maximum number of flits ahead of p in the FIFO queue, and let $W_u(\Delta t)$ be the maximum number of flits entering into R_{inj} by another input port than p and requesting the same output port as p in a time interval of length Δt. Then, conditions 1) and 2) are met as soon as

$$\Delta t + 1 \geq n_{flits} + W_u(\Delta t + 1) \tag{11}$$

for $\Delta t \geq 0$. The solution to Equation (11) is thus equal to the WCIT $wcit_j$ of flow f_j.

To solve the above equation, we first derive a bound on n_{flits}, and then derive a bound on $W_u(\Delta t + 1)$.

Bound on n_{flits}: Section 4.2 explains that each flow in $\mathcal{F}_{inj}$ may have at most one packet in the FIFO queue of the input port by which they are injected in the network. Therefore, in the worst-case scenario, there is one packet of each other flow injected via the same port I_{inj}^{PE} as f_j ahead of p in the FIFO queue. Since p is the last flit of f_j's packet, there must also be $(C_j - 1)$ other flits of f_j ahead of p in the FIFO queue. That is,

$$n_{flits} \leq \left(\sum_{\forall f_i \in \mathcal{F}_{inj}} C_i \right) - 1 \tag{12}$$

Bound on $W_u(\Delta t + 1)$: Let f_l be a flow entering the router R_{inj} by another input port than p but requesting the same output port as p (i.e., $f_l \in \Gamma_{inj}^C$). In the best and worst case scenarios, a flit from f_l takes $bctt'_l$ and $wctt'_l$ clock cycles to reach R_{inj}, respectively. Therefore, the first flit of f_l that may conflict with the injection of p must have been injected no earlier than $wctt'_l$ clock cycles before the beginning of the period during which p is interfered with. Conversely, the last flit of f_l that may conflict with the injection of p must have been injected no later than $bctt'_l$ clock cycles before the end of the interference with p. Therefore, the length of the interval during which f_l may inject flits that conflict with the injection of p is

$$\Delta t + 1 + wctt'_l - bctt'_l = \Delta t + 1 + J_l. \tag{13}$$

Lemma 11 states that a flow f_l may inject at most $\lambda_l(\Delta t + 1 + J_l)$ flits in that time interval. Therefore, the total number of flits from all conflicting flows $f_l \in \Gamma_{inj}^C$ is upper bounded as

$$W_u(\Delta t + 1) \leq \sum_{\forall f_l \in \Gamma_{inj}^C} \lambda_l(\Delta t + 1 + J_l) \tag{14}$$

Injecting Equations (12) and (14) in Equation (11), we prove the lemma. ◀

Theorem 12 requires to know the set of all flows Γ_{inj}^{C} that may conflict with the injection of f_j in router R_{inj}. The content of that set depends on the specific output port requested by the packets of f_j. Lemmas 13 and 17 below provide a means to compute the content of Γ_{inj}^{C} when f_j request output port O_1 or O_u (with $u \neq 1$), respectively.

▶ **Lemma 13.** *The set of flows coming from other routers than R_{inj} and that may be routed to output port O_1 of R_{inj} is given by*

$$\Gamma_{inj}^{1} = \{f_l \mid \forall b \in [2, D],\ d_b^l = r_b^{inj} \wedge \mathrm{dist}(R_{orig}(f_l), R_{inj}) \leq \mathrm{dist}(R_{orig}(f_l), R_{dest}(f_l))\} \quad (15)$$

Proof. According to nDimNoC's routing policy, a flow f_l may request the output port O_1 of R_{inj} only if (i) f_l may hop through router R_{inj}, and (ii) $\forall b > 1,\ d_b^l = r_b^{inj}$. Condition (i) requires that R_{inj} is located between the origin and destination router of f_l, i.e., $\mathrm{dist}(R_{orig}(f_l), R_{inj}) \leq \mathrm{dist}(R_{orig}(f_l), R_{dest}(f_l))$. Combining both (i) and (ii) proves the lemma. ◀

To compute the set Γ_{inj}^{C} for the case where f_j requests output port O_u (with $u \neq 1$), we must first prove some intermediate results using Lemmas 14 to 16.

To prove those lemmas, we define $\mathcal{F}_{inj}^{u}$ as the set of all flows that may enter router R_{inj} by input port I_u. That set can easily be built by checking all paths that may be taken by each flow in the network according to Table 1. All those that have at least one path in which they enter R_{inj} by input port I_u are then added to the set $\mathcal{F}_{inj}^{u}$.

▶ **Lemma 14.** *The set of flows that may enter R_{inj} by input port I_u and request output port O_u is given by $\Gamma_{inj}^{u \to u} = \mathcal{F}_{inj}^{u} \setminus \{f_l \mid \forall b \in [2, D], r_b^{inj} = d_b^l\}$*

Proof. According to Table 1, a flow entering by input port I_u and requesting output port O_1 cannot be routed to output port O_u (only to O_1 or O_{u+1}) (see rules 2, 5, and 6 in Table 1). Therefore, the set of flows entering by I_u that may be routed to O_u is the set of flows that enters R_{inj} by I_u (i.e., $\mathcal{F}_{inj}^{u}$) minus those that request O_1, i.e., all the flows f_l that have a destination such that $\forall b \in [2, D], r_b^{inj} = d_b^l$ (see routing policy explained in Section 4.3). This proves the lemma. ◀

▶ **Lemma 15.** *Let def_q be a boolean equal to true if a deflection may happen in router R_q, and equal to false otherwise. Then, we have*

$$\mathrm{def}_q = \begin{cases} \textbf{\textit{true}} & \textit{if } \exists u, v \textit{ with } u \neq v \mid \exists f_l \in \mathcal{F}_q^u, \exists f_m \in \mathcal{F}_q^v \textit{ s.t. } l \neq m \wedge \forall b \in [2, D],\ d_b^l = d_b^m = r_b^q \\ \textbf{\textit{false}} & \textit{otherwise.} \end{cases} \quad (16)$$

Proof. According to Table 1, a deflection may happen in router R_q only if at least two different flows compete to access the output port O_1. For that situation to happen, there must exist at least two different flows f_l and f_m entering by two different input ports (i.e., $\exists u, v$ with $u \neq v \mid \exists f_l \in \mathcal{F}_q^u, \exists f_m \in \mathcal{F}_q^v$ s.t. $l \neq m$) that both request output port O_1. According to the routing policy explained in Section 4.3, this happens only if $\forall b \in [2, D],\ d_b^l = d_b^m = r_b^q$. This proves the lemma. ◀

▶ **Lemma 16.** *The set of flows that may enter R_{inj} by input port I_{u-1} and be routed to output port O_u is given by*

$$\Gamma_{inj}^{u-1 \to u} = \begin{cases} \mathcal{F}_{inj}^{u-1} & \textit{if } \mathrm{def}_{inj} = \textbf{\textit{true}} \\ \emptyset & \textit{if } \mathrm{def}_{inj} = \textbf{\textit{false}} \end{cases} \quad (17)$$

Table 2 Resources utilization of different NoCs in Kirtex-7 FPGAs.

NoC	LUTs	% Resource utilization of the platform
8x8 ProNoC	100000	20%-150%
8x8 IDAMC	83000	18%-127%
8x8 CONNECT	96000	20%-147%
8x8 HopliteRT*	5632	1.1%-8.5%
4x4x4 3D-nDimNoC	18560	3.9%-28%

Proof. According to Table 1, all flows that enter router R_{inj} by input port I_{u-1} (i.e., those in $\mathcal{F}_{inj}^{u-1}$) may be deflected to output port O_u (see rules 4 and 6 in Table 1). Thus, the set of flows entering by I_{u-1} and routed to O_u is given by all flows in $\mathcal{F}_{inj}^{u-1}$ if a deflection may happen in R_{inj}, i.e., if $\text{def}_{inj} = \textbf{true}$. If no deflection may happen in R_{inj} (i.e., $\text{def}_{inj} = \textbf{false}$), then Table 1 states that none of the flows entering by I_{u-1} may be routed to O_u. This proves both cases of Equation (17). ◀

▶ **Lemma 17.** *The set of flows coming from other routers than R_{inj} and that may be routed to output port O_u of R_{inj} (with $u \neq 1$) is given by*

$$\Gamma_{inj}^{u} = \Gamma_{inj}^{u \to u} \cup \Gamma_{inj}^{u-1 \to u} \tag{18}$$

Proof. According to Table 1, only flows that enter a router by its input ports I_u or I_{u-1} can be routed to output port O_u. Since, according to Lemmas 14 and 16, $\Gamma_{inj}^{u \to u}$ and $\Gamma_{inj}^{u-1 \to u}$ contain all the flows entering R_{inj} by input ports I_u and I_{u-1}, respectively, that may be routed to output port O_u, their union contains all flows that may come from other routers than R_{inj} and may be routed to output port O_u of R_{inj}. ◀

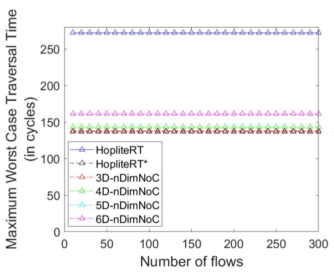
The content of Γ_{inj}^{C} is thus equal to Γ_{inj}^{1} if f_j requests output port O_1 (Lemma 13), and to Γ_{inj}^{u} if it requests any other output port (Lemma 17). Using Γ_{inj}^{C} in Theorem 12 we can now bound the WCIT of f_j.

6 Experimental results

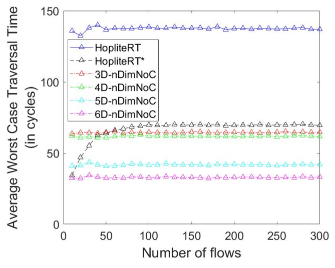
6.1 Implementation of nDimNoC

We implemented a 3D-nDimNoC with the hardware description language Verilog. We synthesized a single router of 3D-nDimNoC for flits of 64 bits. The target platform was a Xilinx Virtex-7 485T FPGA. It required 290 LUTs and 202 Flip-Flops (FFs) in total. This corresponds to only 0.1% and 0.03% of the total number of LUTs and FFs available in the target FPGA, respectively.

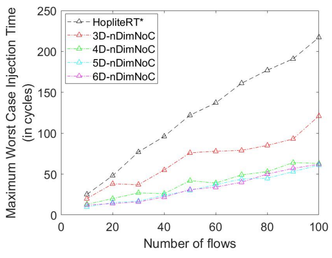
We compared the hardware cost of a 3D-nDimNoC with HopliteRT* [31], as well as to some other NoCs based on virtual channels (VCs): ProNoC [23], IDAMC [37], and CONNECT [27]. The target platform was a Xilinx Kirtex-7 FPGA. Kirtex-7 is a mid-range family of FPGAs that contains approximately between 65,600 and 477,760 LUTs depending on which one you pick. Table 2 shows the synthesis results. A ProNoC router with two VCs required 1574 LUTs, a HopliteRT* router required 88 LUTs, and according to [38] and [27], an IDAMC and a CONNECT router require approximately 1300 and 1500 LUTs,



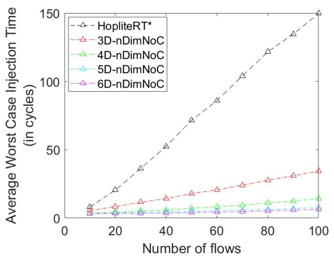
(a) Max WCTT NoC.



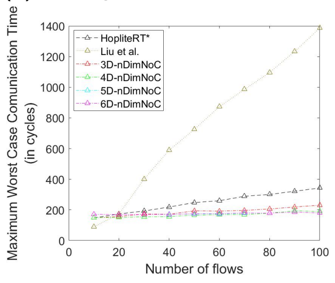
(b) Average WCTT NoC.



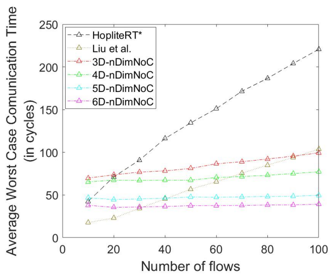
(c) Max WCIT NoC.



(d) Average WCIT NoC.



(e) Max WCCT NoC.



(f) Avg WCCT NoC.

Figure 4 Experimental results for a random traffic pattern.

respectively. Then, as reported in Table 2, an 8x8 ProNoC, IDAMC, and CONNECT NoCs require ≈100,000, ≈83,000, and ≈96,000 LUTs, respectively, eating up a big portion (if not all in some cases) of the logic available in the FPGA. This leaves limited resources available for any computation logic. Therefore, those solutions are not really suitable for systems implemented over mid-range FPGAs. On the other hand, a 4x4x4 3D-nDimNoC requires 18,560 LUTs, i.e., between 3.9% to 28% of the Xilinx Kirtex-7 resources. It is three times more expensive than HopliteRT* (which requires 5632 LUTs) but approximately 5-times cheaper than ProNoC, IDAMC, and CONNECT NoCs in terms of LUTs utilization. We thus conclude that 3D-nDimNoC is a suitable solution for such FPGA platforms.

Finally, we connected the nDimNoC router to a Microblaze soft-core and synthesized a 4x2x2 3D-network for a Virtex-7 485T using Xilinx Vivado. We computed the maximum operating frequency of the network with Xilinx Vivado. We obtained ≈210 MHz for a 4x2x2 3D-nDimNoC against ≈275 MHz for an equivalent 4x4 HopliteRT* NoC. This degradation in terms of maximum operating frequency may be explained by the fact that (1) an nDimNoC router requires more complex logic to route packets from its input to its output ports, and (2) the additinal dimensions increase the number of wires between routers, which increases the complexity of the placement and routing during the logic synthesis.

6.2 Analyses results

In this section, we provide experimental results by computing the WCTT, WCIT, and WCCT of sets of communication flows that traverse NoCs of different dimensionalities.

As a starting point, we generated sets of communication flows for a 16x16 2D-NoC according to a random traffic pattern. The origin and destination coordinates of each flow were randomly generated using a uniform probability distribution. The number of flits of packets released by a communication flow was randomly chosen between 1 and 5, and their inter-arrival times were generated as in [36]. Then, we made a one-to-one mapping of the routers in the 16x16 2D-NoC to the routers of a 4x8x8 3D-nDimNoC, a 4x4x4x4 4D-nDimNoC, a 2x2x4x4x4 5D-nDimNoC, and a 2x2x2x2x4x4 6D-nDimNoC. The origin and destination of each flow were accordingly updated for each network topology.

In Figs. 4a and 4b, we show the maximum and average packets WCTT for an increasing number of flows in NoCs of different dimensionalities. The results were computed by using the analysis of HopliteRT [39, 40] and HopliteRT* [31] (assuming a 16x16 2D-NoC), and the analysis presented in Section 5.1 for the 2D, 3D, 4D, 5D and 6D-nDimNoC topology. To establish a fair comparison, we assume one priority level (i.e., all flows were assigned the highest priority) for the analysis proposed in [31]. Each point in the plot is the result of 100 repetitions (100 different random flow sets). We varied the number of generated flows from 10 to 300 by steps of 10.

In 4a, we observe that the maximum WCTT is slightly worse with nDimNoC as compared to HopliteRT*. Nonetheless, Fig 4b) shows that the average traversal time improves with nDimNoC as the dimensionality of the network increases. This can easily be explained by the fact that new routes, possibly shorter and faster, are made available between pairs of routers when a new dimension is added to the network. Moreover, the number of interfering flows, and therefore, the number of deflections that flows may suffer on each link decreases since the number of routers on each dimension decreases. Note that, the average packets WCTT is reduced by ≈40% and ≈60% with a 5D-nDimNoC and a 6D-nDimNoC, respectively, against HopliteRT*. We also show that the maximum and average worst-case traversal times are noticeably reduced with nDimNoC as compared to HopliteRT.

In Fig. 4c and 4d, we show the maximum and average WCIT of flows using the analysis of HopliteRT* and nDimNoC. We also computed the maximum and average WCIT by using the analysis of HopliteRT, but we do not show them on the graphs as they are extremely pessimistic and would render the plots unreadable by cluttering all other lines together. As shown, the packets see their WCIT drastically reduced in nDimNoC in comparison to HopliteRT*. This is expected since nDimNoC allows the programming element connected to a router to inject packets simultaneously via as many input ports as there are dimensions in the network. A router of HopliteRT*, on the other hand, can inject at most one flit per cycle in the network (on either of the router output ports). Furthermore, the number of communication flows that may interfere with the injection of a packet at a router decreases since more routes are available in the network, and thus less traffic uses each individual route.

In Fig. 4e and 4f, we show the maximum and average packets WCCT (which we recall to be equal to the sum of the WCTT and WCIT of those packets). We varied the number of generated flows from 10 to 100 by steps of 10. The results were obtained by using the analysis of nDimNoC, and the analyses proposed in [31] and [21] for HopliteRT* and a VC-based real-time NoC, respectively. The analysis presented in [21] by Liu et al. is an improved analysis of that proposed in [36, 35] by Shi and Burns. To establish a fair comparison, we assume one VC (i.e., one priority level) for the analysis presented in [21]. As shown in Figure 4e, for almost all configurations, the WCCT returned by the analysis of nDimNoC outperforms that returned by the analysis of [31] and [21]. The average WCCT is only better with the analysis by Liu et al. when the network is completely underloaded and very few flows are traversing the network (i.e., less than 30 flows). Note also that, [21] considers that each flow may only have one packet of each flow traveling across the network at the same time, whilst nDimNoC supports the transmission of several packets from the same communication flow simultaneously.

The average WCCT with nDimNoC improves when the network's dimensionality increases and is barely impacted by the number of flows. Therefore, we conclude that increasing the dimensionality of nDimNoC has a positive impact from an average performance perspective for a limited impact on the worst-case performance of the flows.

7 Summary and conclusion

In this paper, we presented nDimNoC, a new and flexible real-time D-dimensional NoC that uses the properties of circulant topologies to provide real-time guarantees to the flows transmitted over that NoC. We proposed a timing analysis for nDimNoC. We also did a complete implementation of 3D-nDimNoC in HDL Verilog. Experimental results show improvements in terms of network communication latency in comparison to existing 2D solutions.

References

1 P. Baran. On distributed communications networks. *IEEE Transactions on Communications Systems*, 12(1):1–9, 1964. `doi:10.1109/TCOM.1964.1088883`.

2 L. Benini and G. De Micheli. Networks on chip: a new paradigm for systems on chip design. In *Design, Automation and Test in Europe Conference and Exhibition*, pages 418–419, March 2002.

3 Alan Burns, James Harbin, and Leandro Soares Indrusiak. A wormhole NoC protocol for mixed criticality systems. In *IEEE Real-Time Systems Symposium*, pages 184–195, 2014.

4 Yiou Chen, Jianhao Hu, Xiang Ling, and Tingting Huang. A novel 3d noc architecture based on de bruijn graph. *Computers & Electrical Engineering*, 38(3):801–810, 2012.

5 Shamik Das, Andy Fan, Kuan-Neng Chen, Chuan Seng Tan, Nisha Checka, and Rafael Reif. Technology, performance, and computer-aided design of three-dimensional integrated circuits. In *Proceedings of the 2004 International Symposium on Physical Design*, ISPD '04, page 108?115, New York, NY, USA, 2004. Association for Computing Machinery. `doi:10.1145/981066.981091`.

6 Dakshina Dasari, Borislav Nikoli'c, Vincent N'elis, and Stefan M Petters. NoC contention analysis using a branch-and-prune algorithm. *ACM Transactions on Embedded Computing Systems*, 13(3s):113, 2014.

7 Jonas Diemer, Jonas Rox, Mircea Negrean, Steffen Stein, and Rolf Ernst. Real-time communication analysis for networks with two-stage arbitration. In *9th ACM International Conference on Embedded Software*. IEEE, 2011.

8 Feihui Li, C. Nicopoulos, T. Richardson, Yuan Xie, V. Narayanan, and M. Kandemir. Design and management of 3d chip multiprocessors using network-in-memory. In *33rd International Symposium on Computer Architecture (ISCA'06)*, pages 130–141, 2006. `doi:10.1109/ISCA.2006.18`.

9 Yan Ghidini, Thais Webber, Edson Moreno, Fernando Grando, Rubem Fagundes, and César Marcon. Buffer depth and traffic influence on 3d nocs performance. In *2012 23rd IEEE International Symposium on Rapid System Prototyping (RSP)*, pages 9–15. IEEE, 2012.

10 Frédéric Giroudot and Ahlem Mifdaoui. Buffer-aware worst-case timing analysis of wormhole NoCs using network calculus. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2018.

11 Frederic Giroudot and Ahlem Mifdaoui. Tightness and computation assessment of worst-case delay bounds in wormhole networks-on-chip. In *27th International Conference on Real-Time Networks and Systems*, 2019.

12 C. Grecu, P. P. Pande, A. Ivanov, and R. Saleh. A scalable communication-centric soc interconnect architecture. In *International Symposium on Signals, Circuits and Systems. Proceedings, SCS 2003. (Cat. No.03EX720)*, pages 343–348, 2004. `doi:10.1109/ISQED.2004.1283698`.

13 R. I. Greenberg and Lee Guan. An improved analytical model for wormhole routed networks with application to butterfly fat-trees. In *Proceedings of the 1997 International Conference on Parallel Processing (Cat. No.97TB100162)*, pages 44–48, 1997. `doi:10.1109/ICPP.1997.622554`.

14 P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000 (Cat. No. PR00537)*, pages 250–256, 2000. `doi:10.1109/DATE.2000.840047`.

15 Jörg Henkel, Wayne Wolf, and Srimat Chakradhar. On-chip networks: A scalable, communication-centric embedded system design paradigm. In *17th International Conference on VLSI Design*. IEEE, 2004.

16 S. Hesham, J. Rettkowski, D. Goehringer, and M. A. Abd El Ghany. Survey on real-time networks-on-chip. *IEEE Transactions on Parallel and Distributed Systems*, 28(5):1500–1517, May 2017. `doi:10.1109/TPDS.2016.2623619`.

17 Leandro Soares Indrusiak, Alan Burns, and Borislav Nikolić. Buffer-aware bounds to multi-point progressive blocking in priority-preemptive nocs. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 219–224. IEEE, 2018.

18 Leandro Soares Indrusiak, James Harbin, and Alan Burns. Average and worst-case latency improvements in mixed-criticality wormhole networks-on-chip. In *27th Euromicro Conference on Real-Time Systems*. IEEE, 2015.

19 J. W. Joyner, P. Zarkesh-Ha, and J. D. Meindl. A stochastic global net-length distribution for a three-dimensional system-on-a-chip (3d-soc). In *Proceedings 14th Annual IEEE International ASIC/SOC Conference (IEEE Cat. No.01TH8558)*, pages 147–151, 2001. `doi:10.1109/ASIC.2001.954688`.

20 C. C. Liu, I. Ganusov, M. Burtscher, and Sandip Tiwari. Bridging the processor-memory performance gap with 3d ic technology. *IEEE Design Test of Computers*, 22(6):556–564, 2005. `doi:10.1109/MDT.2005.134`.

21 Meng Liu, Matthias Becker, Moris Behnam, and Thomas Nolte. Tighter time analysis for real-time traffic in on-chip networks with shared priorities. In *10th IEEE/ACM International Symposium on Networks-on-Chip*, 2016.

22 César Marcon, Ramon Fernandes, Rodrigo Cataldo, Fernando Grando, Thais Webber, Ana Benso, and Letícia B Poehls. Tiny noc: A 3d mesh topology with router channel optimization for area and latency minimization. In *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*, pages 228–233. IEEE, 2014.

23 Alireza Monemi, Jia Tang, Maurizio Palesi, and Muhammad Nadzir Marsono. ProNoC: A low latency network-on-chip based many-core system-on-chip prototyping platform. *Microprocessors and Microsystems*, 54, September 2017. `doi:10.1016/j.micpro.2017.08.007`.

24 B. Nikolic, Robin Hofmann, and R. Ernst. Slot-based transmission protocol for real-time nocs - sbt-noc. In *ECRTS*, 2019.

25 B. Nikolić and S. M. Petters. Edf as an arbitration policy for wormhole-switched priority-preemptive nocs-myth or fact? In *International Conference on Embedded Software*, pages 1–10, October 2014.

26 Borislav Nikolić, Sebastian Tobuschat, Leandro Soares Indrusiak, Rolf Ernst, and Alan Burns. Real-time analysis of priority-preemptive nocs with arbitrary buffer sizes and router delays. *Real-Time Systems*, 55(1):63–105, 2019.

27 M. K. Papamichael and J. C. Hoe. CONNECT: Re-examining conventional wisdom for designing Nocs in the context of FPGAs. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '12, pages 37–46, New York, NY, USA, 2012. ACM.

28 D. Park, S. Eachempati, R. Das, A. K. Mishra, Y. Xie, N. Vijaykrishnan, and C. R. Das. Mira: A multi-layered on-chip interconnect router architecture. In *2008 International Symposium on Computer Architecture*, pages 251–261, 2008. `doi:10.1109/ISCA.2008.13`.

29 Vasilis F Pavlidis, Ioannis Savidis, and Eby G Friedman. *Three-dimensional integrated circuit design*. Newnes, 2017.

30 Eberle A Rambo and Rolf Ernst. Worst-case communication time analysis of networks-on-chip with shared virtual channels. In *Design, Automation & Test in Europe Conference & Exhibition*, 2015.

31 Y. Ribot González and G. Nelissen. Hoplitert*: Real-time noc for fpga. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3650–3661, 2020. `doi:10.1109/TCAD.2020.3012748`.

32 Aleksandr Yu Romanov. Development of routing algorithms in networks-on-chip based on ring circulant topologies. *Heliyon*, 5(4):e01516, 2019.

33 Abbas Sheibanyrad, Frédéric Pétrot, Axel Jantsch, et al. *3D integration for NoC-based SoC Architectures*. Springer, 2011.

34 Zheng Shi and Alan Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Second ACM/IEEE International Symposium on Networks-on-Chip*, 2008.

35 Zheng Shi and Alan Burns. Improvement of schedulability analysis with a priority share policy in on-chip networks. In *17th International Conference on Real-Time and Network Systems*, pages 75–84, 2009.

36 Zheng Shi and Alan Burns. Real-time communication analysis with a priority share policy in on-chip networks. In *21st Euromicro Conference on Real-Time Systems*, pages 3–12. IEEE, 2009.

37 S. Tobuschat, P. Axer, R. Ernst, and J. Diemer. IDAMC: A NoC for mixed criticality systems. In *IEEE 19th International Conference on Embedded and Real-Time Computing Systems and Applications*, 2013. `doi:10.1109/RTCSA.2013.6732214`.

38 Sebestian Tobuschat. *Predictable and Runtime-Adaptable Network-On-Chip for Mixed-critical Real-time Systems*. PhD thesis, TU Braunschweig, 2019.

39 S. Wasly, R. Pellizzoni, and N. Kapre. HopliteRT: An efficient FPGA NoC for real-time applications. In *International Conference on Field Programmable Technology*, pages 64–71, December 2017.

40 Saud Wasly, Rodolfo Pellizzoni, and Nachiket Kapre. Worst case latency analysis for hoplite FPGA-based NoC. Technical report, University of Waterloo, 2017.

41 Qin Xiong, Zhonghai Lu, Fei Wu, and Changsheng Xie. Real-time analysis for wormhole noc: Revisited and revised. In *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*, pages 75–80, 2016.

42 Qin Xiong, Fei Wu, Zhonghai Lu, and Changsheng Xie. Extending real-time analysis for wormhole nocs. *IEEE Transactions on Computers*, 66(9):1532–1546, 2017.