First-Last: A Cost-Effective Adaptive Routing Solution for TSV-Based Three-Dimensional Networks-on-Chip

Amir Charif, Alexandre Coelho, *Student Member, IEEE,* Masoumeh Ebrahimi, *Member, IEEE,* Nader Bagherzadeh, *Fellow, IEEE,* and Nacer-Eddine Zergainoh, *Member, IEEE*

Abstract—3D integration opens up new opportunities for future multiprocessor chips by enabling fast and highly scalable 3D Network-on-Chip (NoC) topologies. However, in an aim to reduce the cost of Through-silicon via (TSV), partially vertically connected NoCs, in which only a few vertical TSV links are available, have been gaining relevance. To reliably route packets under such conditions, we introduce a lightweight, efficient and highly resilient adaptive routing algorithm targeting partially vertically connected 3D-NoCs named First-Last. It requires a very low number of virtual channels (VCs) to achieve deadlock-freedom (2 VCs in the East and North directions and 1 VC in all other directions), and guarantees packet delivery as long as one healthy TSV connecting all layers is available anywhere in the network. An improved version of our algorithm, named Enhanced-First-Last is also introduced and shown to dramatically improve performance under low TSV availability while still using less virtual channels than state-of-the-art algorithms. A comprehensive evaluation of the cost and performance of our algorithms is performed to demonstrate their merits with respects to existing solutions.

Index Terms—Network-on-Chip, 3D NoC, 3D Integration, Adaptive routing, TSV.

1 Introduction

Networks-on-Chip (NoCs) [1] have effectively become the go-to paradigm for on-chip interconnections in modern manycore systems, offering a high-performance communication infrastructure for Chip Multiprocessors (CMPs), Multiprocessor Systems-on-Chip (MPSoCs) and even Graphics Processing Units (GPUs) [2], [3]. If NoCs were already perceived as a highly scalable and efficient alternative to the traditional bus, they are even more so with the recent emergence of 3D integration. The stacking of several silicon layers allows for inherently low-latency three-dimensional NoC topologies (3D-NoCs) to be considered [4], [5].

Through-Silicon Via (TSV) has been accepted as one of the most viable technologies to enable vertical communication between different NoC layers [6]. However, due to its non-negligible cost, the number of vertical links must be kept to a minimum, resulting in irregular 3D-NoC topologies commonly referred to as Vertically-Partially-Connected NoCs. In addition, due to the vulnerability of TSVs to manufacturing defects as well as runtime failures [7], the number of available TSVs may end up being reduced even further.

Under such extreme conditions, a flexible routing algorithm that guarantees packet delivery with a limited number of vertical links is necessary. Perhaps the most challenging aspect in designing such algorithms is ensuring correct operation (deadlock-freedom, livelock-freedom, connectivity) at a reasonable cost, without heavily limiting the flexibility of the algorithm and the number of fault scenarios it can tolerate. More specifically, deadlock-avoidance often requires adding a certain number of Virtual Channels (VCs) in each router, which consist of disjoint flit FIFOs used to separate different flows. As these FIFOs occupy the largest part of a NoC router's area [8], an algorithm that can operate using a small number of VCs is strongly desirable.

While several algorithms requiring no or few virtual channels have been recently proposed [9], [10], they often follow specific routing rules that pose restrictions on the location and the selection of vertical links, hindering both reliability and performance. A routing algorithm capable of relaxing these restrictions while keeping the implementation cost to a minimum is yet to be introduced.

In this paper, we address this challenge by introducing an efficient, adaptive and highly resilient routing algorithm targeting partially vertically connected 3D-NoCs named First-Last. Our algorithm requires a very low number of virtual channels to achieve deadlock-freedom (2 VCs in the East and North directions and 1 VC in West, South, and Local port directions). This unique distribution of virtual channels, which does not assume any symmetry along dimensions, greatly increases the number of supported topologies. It guarantees full connectivity for all regular (a.k.a. pillar based) partially connected topologies with no assumptions on the placement of the pillars or their assignment to nodes. Furthermore, we will prove that First-Last is the optimal routing algorithm for such topologies in terms of the number of required VCs (cost) and supported

- *A. Charif is with the Computing and Design Environment Laboratory, CEA LIST, F-91191 Gif-sur-Yvette, France. E-mail: mohamedelamir.charif@cea.fr*
- *Alexandre Coelho and Nacer-Eddine Zergainoh are with the TIMA laboratory. E-mail: firstname.lastname@univ-grenoble-alpes.fr*
- *Masoumeh Ebrahimi is with University of Turku, Finland and the KTH Royal Institute of Technology, Sweden. E-mail: masebr@utu.fi*
- *Nader Bagherzadeh is with the Department of Electrical Engineering and Computer Science, University of California Irvine, Irvine, CA 92697. E-mail: nader@uci.edu*

irregular topologies (resilience).

A preliminary version of our contribution was presented in [11]. This paper extends and improves upon our previous work in the following ways:

- We formally identify the condition to be met by TSV placement in order to guarantee full connectivity. The optimality of the algorithm for a given set of topologies is also proven in this paper.
- An improved variant of our algorithm called Enhanced-First-Last is presented. It uses an additional VC along the Z dimension, not only to improve performance but also to increase resiliency. The elevator assignment strategy is also improved to allow for more runtime adaptability.
- A more thorough evaluation of the proposed algorithms, including real application benchmarks, hardware synthesis results, and comparison with more recent routing solutions is performed to demonstrate the effectiveness of our approach.

The remainder of this paper is organized as follows: In Section 2 we explore existing solutions in the context of 3D routing, with an emphasis on the works that are closest related to our contribution. Section 3 describes the target system architecture. In Section 4, we construct the First-Last algorithm as well as its improved version named Enhanced-First-Last, and provide a high level view of how the algorithm operates. Deadlock-freedom, livelock-freedom, connectivity and optimality are addressed in detail in Section 5. Section 6 provides details about a scalable and distributed implementation of the proposed algorithms consisting of an offline selection algorithm and a hardware-friendly distributed online routing algorithm. In Section 7 we evaluate the proposed routing solution in terms of cost, performance, temperature and power, before concluding in Section 8.

2 Related works

In the context of 3D-NoCs, several routing algorithms have been proposed. From simple deterministic algorithms such as XYZ, to fully adaptive algorithms such as 3D-FAR [12] and DyXYZ [8]. [12] also introduces 3D-FT, which is capable of tolerating the absence of vertical or horizontal links. However, like 3D-FAR, it requires a very large number of virtual channels (2, 2 and 4 along the Z, X and Y dimensions, respectively). In [13], the authors extend the turn model for 2D meshes [14] to the third dimension and propose an algorithm that tolerates faults by replicating each packet and sending it in two different virtual networks, one using the 3D negative-first algorithm and the other using the 3D positive-first algorithm. AFRA [15] is another algorithm that can tolerate a certain number of faulty vertical links in fully connected NoCs.

Only a few proposals have been made in the context of partially vertically connected 3D-NoCs. In [16], the authors propose to use any deterministic deadlock-free 2D mesh routing algorithm to deliver a packet to an elevator (vertical link), which will be used to deliver the packet to its destination layer, then to continue routing using the planar routing algorithm until the packet reaches its destination. It was proven to be deadlock-free using 2 virtual channels along the X and Y dimensions. This approach, named Elevator-First, is appealing because of its simplicity, its support for any layer topology, and because it does not impose any constraints on the position of healthy vertical links. Routing a packet towards an elevator requires the insertion of a temporary header containing the elevator's address. Addresses of the up and down elevators are stored inside each router [8], requiring an amount of storage that increases with the network size. In order to reduce the requirements of Elevator-First in terms of virtual channels, authors in [9] add certain constraints on the usage of the elevators and show that routing is possible without the use of virtual channels. In [17], another algorithm that does not require the use of virtual channels is presented, but it requires the presence of one vertical link at the north-east corner.

The ETW (East-then-West) routing algorithm [10] aims at reducing the required number of virtual channels of Elevator-First, while maintaining a certain routing flexibility and offering partial adaptiveness to mitigate congestion. ETW uses 1, 2 and 1 virtual channels along the X, Y and Z dimensions respectively. The authors have also proposed some solutions to tolerate runtime failures using the dynamic elevator assignment in [18] or the propagation of TSV status in [19]. Unfortunately, ETW poses some limiting constraints on both the location and the selection of the elevators. It requires the existence of at least one pillar in the east-most column, and for packets heading downwards, an elevator located east to the destination must be taken, leading to inefficient routes in some cases. In addition, because the choice of the elevator depends on the destination, 3 elevator addresses need to be stored in each router.

Our algorithm uses the same total number of VCs as ETW [19] but does not require the presence of TSV pillars, i.e. the position of TSVs may differ from one layer to the other. Moreover, unlike ETW, our algorithm makes it possible for packets to reach their destination node regardless of which TSV they decide to use, allowing for much shorter routes than those imposed by ETW.

Redelf [20] is a routing algorithm for partially connected topologies that does not require any additional virtual channels. To make this possible, Redelf adds restrictions on the topology as well as the selection of the elevators. It is connected and deadlock-free under the important assumption that all the vertical connections are bidirectional [20], which means that two unidirectional TSVs must be used for each vertical connection. This is not a requirement for our algorithm. Redelf also forces part of the packets to take one specific elevator, named the Pivot elevator. In some topologies, especially if the tiers are quite large, this can result in many packets taking longer routes than necessary, which heavily impacts performance as we will see in Section 7.

Finally, the 3D variant of the LBDR (Logic Based Distributed Routing) [21] was recently presented in [22]. As is the case of LBDR, LBDR3D supports a variety of partially adaptive routing algorithms and is fully reconfigurable to tolerate faulty horizontal and vertical links. It was proven deadlock- and livelock- free using the same method as Elevator-First and requires the same minimum number of virtual channels to separate between Upward and Downward flows. However, in LBDR3D, only a fixed number of bits are stored within each router to locate healthy elevators.

Like LBDR3D, to keep track of healthy elevators without having to store router addresses, in Section 6 we propose a scalable method that uses a fixed number of bits per router (12 bits) to guide packets to the nearest elevator. However, we will show that the method adopted in [22] for assigning elevators to nodes is not inherently deadlock-free. We propose an offline selection algorithm that effectively guarantees deadlock-freedom without changing the hardware.

3 Target architecture

We consider a network comprised of several 2D Mesh layers (or tiers) connected vertically using TSV, as shown in Fig. 1. Each layer consists of a mixture of classic 2D routers including only 5 ports (East, West, South, North, Local), and 3D routers having either an Up port, a Down port, or both. 3D routers will also be referred to as ”Elevators” [16]. An upward (downward) elevator is one that connects to the upper (lower) tier.

While we assume that the channels connecting routers of the same tier are bidirectional, we make no such assumption on the vertical connections. In effect, with a limited number of TSVs available, the system designer may prefer to place an upward and a downward TSV in two different routers instead of placing them in the same router, so as to better balance the load. For instance, in Fig. 1, router A and router B are connected using a bidirectional channel (2 TSVs), whereas routers C and D are connected using only an upward TSV. The problem of finding the best placement strategy for TSVs is beyond the scope of this paper, and has already been addressed in [23]. However, the solution proposed in this paper has some specific requirements on the placement of TSVs, although not very restrictive. These requirements are discussed in detail in Section 5.

Due to the limited number of vertically connected nodes, routers need to locate the elevators of their tier in order to be able to communicate with other tiers. We also consider the TSVs to be prone to manufacturing defects, as well as permanent failure. The information stored in each router regarding the location of TSVs must be updated to reflect the new state of the network upon failure. In Section 6, we propose a strategy that requires a fixed number of bits in each router to locate the nearest elevators, and present the adequate algorithms for reconfiguring and using these bits during the routing process.

4 The First-Last routing algorithm

4.1 General approach

A recently proposed method for constructing a deadlock-free algorithm is to divide the network channels into several virtual networks (or subnetworks) that are cycle-free, and to make sure that these virtual networks (VNs) are visited by packets in a predefined order. This approach, called EbDa, was introduced in [24] and was used to construct algorithms such as [19] and [25].

However, by ensuring that the header of the flit is always at the head of the queue, more flexibility can be obtained. In this work, we take a two-step approach for constructing

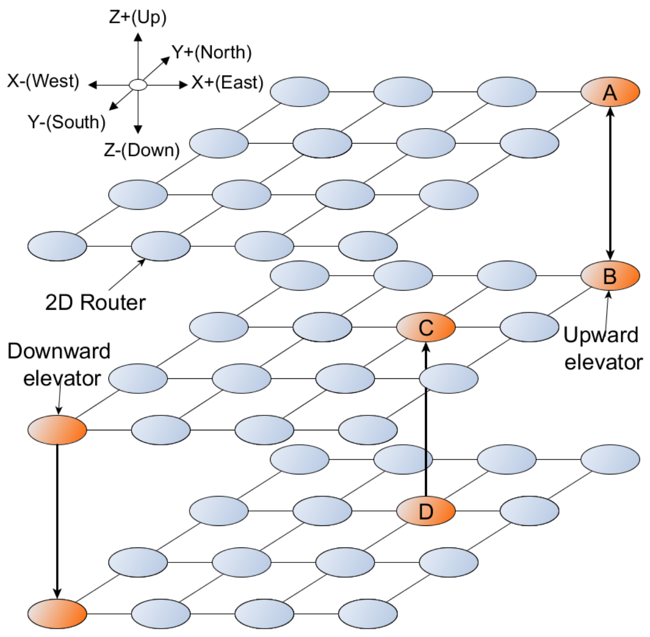


Fig. 1. Target NoC topology.

our routing algorithm such that virtual channels are used as efficiently as possible. First, we identify the virtual networks as well as the required number of virtual channels, and then we determine the set of virtual channels that can be used by each virtual network.

4.1.1 *Step 1: Defining cycle-free virtual networks*

We define a virtual network (VN) as a set of **physical** channels (output ports) that the packet can use to make progress towards its destination. The virtual network number is stored in the packet header and is used by the route computation logic to determine the set of candidate directions that can be taken at every hop.

Moving from one virtual network to the next is simply performed by updating the virtual network number in the packet, as will be shown in Section 6.

The first-last routing algorithm defines 3 virtual networks as shown in Fig. 2. The first and third virtual networks (VN0, VN2) only include the X+ and Y+ physical channels. The second virtual network (VN1) includes all the remaining directions (X-, Y-, Z+, Z-). Note that it is not possible to form a cycle within any of these virtual networks, as none of them spans two full dimensions [24].

Because the X+ and Y+ physical channels are shared between two virtual networks, two virtual channels are required in these directions. The two virtual channels in the X+ direction are noted (X0+, X1+), whereas the virtual channels in the Y+ direction are noted (Y0+, Y1+). The other directions, which are only used by VN1, do not necessitate additional virtual channels.

Packets may traverse virtual networks only in increasing order ($VN0 \rightarrow VN1 \rightarrow VN2$). The algorithm is named First-Last, because the physical channels that are used first (Positive channels) are also used last.

The routing algorithm can be simply described as follows: If a packet has reached the destination tier, i.e. the

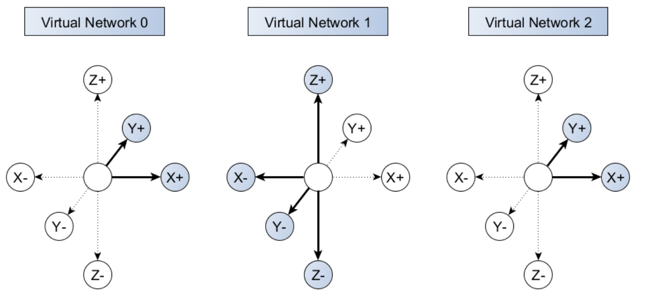


Fig. 2. Virtual network decomposition for First-Last.

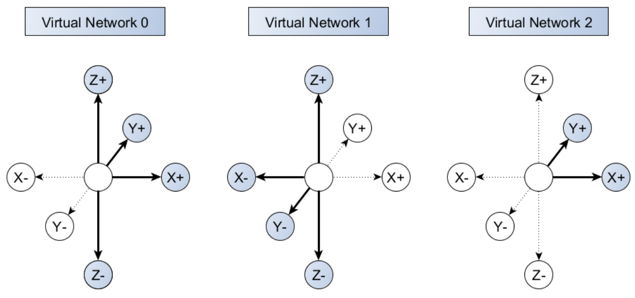


Fig. 3. Virtual network decomposition for Enhanced-First-Last.

tier where the destination node is located, then it should be routed towards the destination node using the negative directions first (VN1 then VN2). Otherwise, route the packet towards an elevator using the positive directions first (VN0 then VN1) and use the channels of VN1 to elevate the packet to the destination layer. Whenever a packet is headed East-North or West-South, routing can be performed adaptively and the least congested route is selected. Details about a possible implementation are provided in Section 6.

4.1.2 Step 2: Assigning virtual channels

The (X-, Y-, Z+, Z-) channels are only used by packets of VN1 and the corresponding single virtual channel in each of these directions is fully dedicated to packets of VN1.

In our design, packets of VN0 are only allowed to use virtual channels (X0+, Y0+), whereas packets of VN2 are allowed to use all the virtual channels associated with the physical channels of VN2 (X0+, X1+, Y0+, Y1+). In other words, virtual channels (X1+, Y1+) are only used by VN2 packets, whereas virtual channels (X0+, Y0+) are shared between VN0 and VN2 packets.

Note that unlike the traditional approaches that prohibit cycles by dedicating each VC to a single virtual network, here we allow extra freedom on the acquisition of virtual channels for packets of VN2. This implies that packets that have reached their destination layers are allowed to occupy any VC in the network to reach their destination node. Deadlock-freedom is preserved by ensuring that packets of VN2 that are occupying VCs (X0+, Y0+) are at all times able to escape to their dedicated virtual channels [26].

This can be achieved by slightly altering the flow control mechanism, such that VCs (X0+, Y0+) are only allocated to a VN2 packet if they are empty. This guarantees that VN2 packets are always at the head of these VCs, and can therefore request their dedicated (escape) VCs at any time.

4.2 Enhanced-First-Last: Boosting network performance and resilience with vertical VCs

With a reduced number of vertical channels, elevator nodes are very likely to turn into hotspots as several flows may need to share a single TSV.

A simple way to mitigate this issue is to add a virtual channel along the Z dimension to help reduce the pressure on vertically connected nodes. This VC can be used by all VN1 packets without any restrictions.

Because the VC is added in the vertical ports, only 3D routers need to be modified. This means that although 3D routers now include as many VCs as Elevator-First [8], 2D routers still include fewer VCs, resulting in a lower overall cost than Elevator-First, especially for designs with a low TSV density. Under heavy inter-layer communication, VCs along the Z dimension will have a much higher impact on performance than VCs in the planar ports, making them well worth the extra cost.

Furthermore, in addition to throughput enhancement, this extra VC, if wisely used, can also contribute to the resilience of the routing algorithm. Consider the virtual network definition presented in Fig 3. Compared to the original First-Last algorithm, VN0 now also includes the vertical dimension. This means that packets of VN0 can now reach other layers without having to transit to VN1, provided that an elevator could be reached using only the positive directions. In this case, after reaching the next layer, any elevator can be taken as packets can still use VN0 and VN1. Fig. 4 illustrates an example network in which layer 0 cannot reach layer 2 using the original First-Last algorithm. However, Enhanced-First-Last is capable of partially connecting layer 0 to layer 2.

Here again, we allow VN1 packets to use both vertical VCs (Z0+, Z1+, Z0-, Z1-) while ensuring they can always escape to (Z1+, Z1-), whereas packets of VN0 may only ever use (Z0+, Z0-) in the vertical dimension. The assignment of planar VCs remains the same as that of the original First-Last algorithm.

5 Deadlock-freedom, livelock-freedom, connectivity and optimality

5.1 Proof of deadlock-freedom

Because each virtual network is cycle-free, it is not possible for a deadlock configuration to involve only packets of one virtual network. Therefore, any deadlock configuration must involve at least two virtual networks.

Let us assume that the network is deadlocked, and that the deadlock set includes a packet P of VN2. At any point in time, blocked packets of VN2 are either waiting for VCs (X0+, X1+) or (Y0+, Y1+). However, X1+ and Y1+ can only be occupied by packets of VN2. Therefore, there is a path in the packet wait-for graph starting from P and involving only packets of VN2 that are occupying channels X1+ and Y1+. This path cannot include cycles due to the absence of negative directions in VN2, and therefore packet P has an escape path, which contradicts with the deadlock

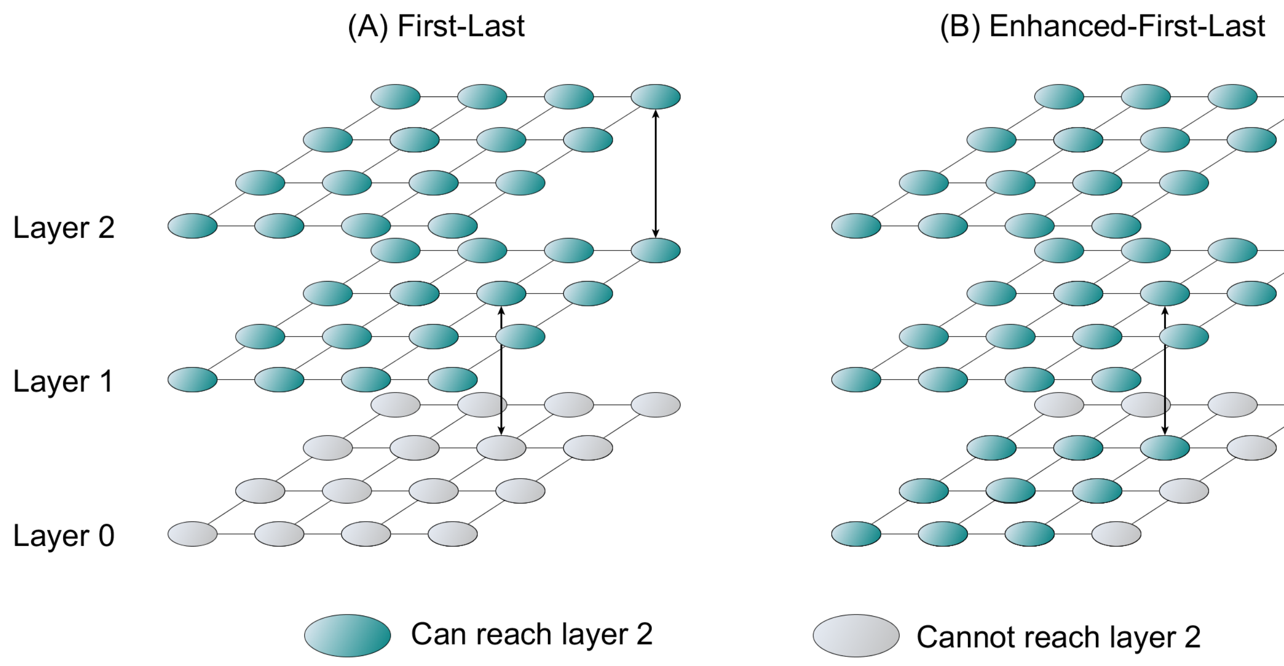


Fig. 4. An example in which Enhanced-First-Last improves the connectivity of the network. (A) Packet originating at layer 0 and destined for layer 2 need to reach layer 1 in VN1, and can therefore only use negative directions in layer 1. (B) Packets can reach layer 1 in VN0, making it possible to reach any elevator in layer 1.

assumption. That is, no deadlock configuration can involve VN2 packets.

With VN2 excluded, the same reasoning can now be applied to exclude VN1 simply by replacing X1+ and Y1+ by Z1+ and Z1-.

Because at least two virtual networks are required to form a deadlock, the network is deadlock-free.

5.2 Livelock-freedom

Since the individual virtual networks do not allow packets to loop indefinitely, and because they are traversed in an increasing order, the algorithm is also livelock-free. In the worst case, a packet reaches the north-east corner in VN0 at the source layer (top or bottom layer in the case of Enhanced-First-Last), then the west-south corner of either the top or the bottom layer in VN1 and then end up in the north-east corner of the same layer in VN2.

5.3 Condition of connectivity

The network is connected if the routing algorithm is able to provide a path for all source-destination pairs. We will identify the condition that must be met by the TSV placement strategy in order for the network to be connected using the First-Last algorithm.

First, we know that routing from any source to any elevator in the same layer, which is done using VN0 and VN1, is always possible. Similarly, routing from any elevator to any destination on the same layer is possible using VN1 and VN2. This means that for the network to be connected, it is enough to ensure that every layer in the network can reach every other layer.

Although Enhanced-First-Last makes it possible to traverse layers in VN0 for some packets, in the general case, packets may need to go to VN1. So to guarantee connectivity, we consider the worst case, wherein packets need to move to VN1 to reach other layers.

Let us consider a network consisting of L layers, with U_l and D_l being the set of upward elevators and the set of downward elevators of layer l, respectively. The condition can then be expressed as follows:

The network is connected if and only if

$\forall l \in]0, L-1[$,

$\exists u_-(x_-, y_-) \in U_{l-1}$, such that $\exists u(x,y) \in U_l$ with $x \leq x_-, y \leq y_-$

and

$\exists d_+(x_+, y_+) \in D_{l+1}$, such that $\exists d(x,y) \in D_l$ with $x \leq x_+, y \leq y_+$.

That is, each layer must be able to forward packets coming from a previous layer through a certain elevator E_{in}, to the next layer through an elevator that is reachable using only the negative channels (VN1), i.e. that is located south-east to the incoming elevator E_{in}.

Fig. 5 illustrates a few examples of connected networks. In the first example (Fig. 5 (a)), two TSVs are used to connect each consecutive layers. The network is connected because the bottom layer can reach the top layer through elevators $1 \rightarrow 2$, and the top layer can reach the bottom layer through elevators $A \rightarrow B$.

The second example (Fig. 5 (b)) shows that it is possible to connect the network using only one TSV ”pillar”. Note that the network is connected regardless of the position of the pillar.

5.4 Optimality

We can further show that, for a certain set of partially connected 3D topologies, in particular, the topologies where

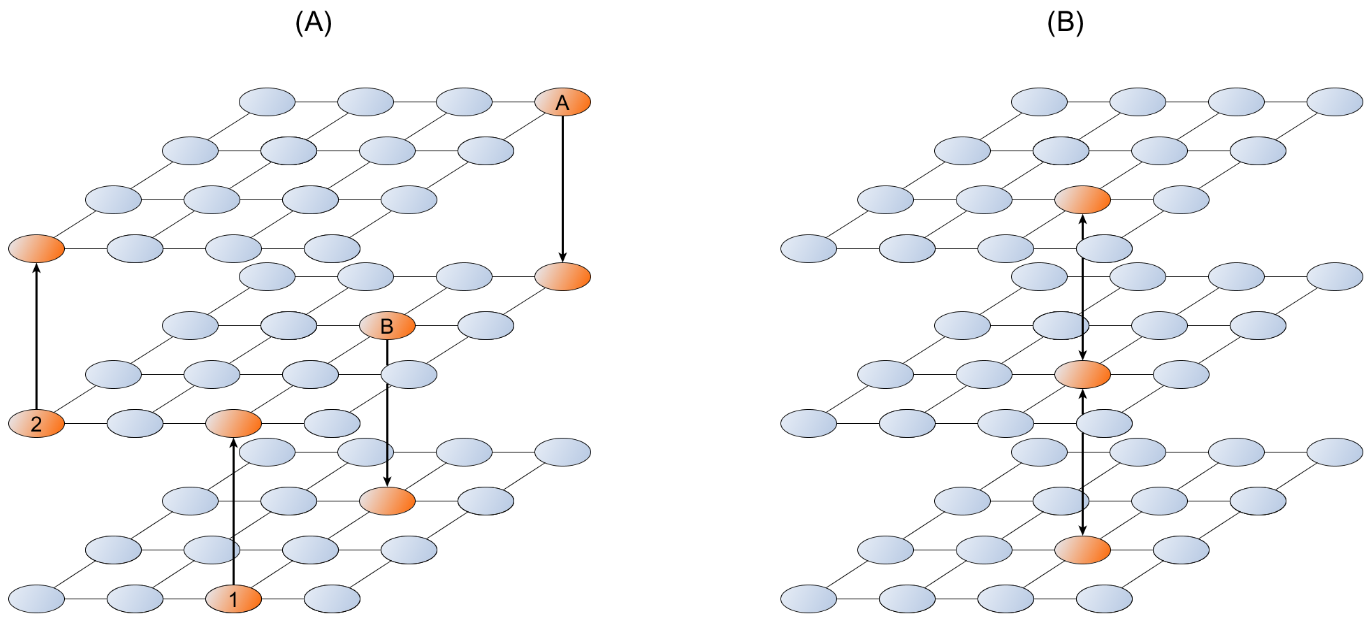


Fig. 5. Examples of connected networks using First-Last.

TSVs are placed as pillars (as in Fig. 5 (B)), and where the vertical channels are not necessarily bidirectional, First-Last is the optimal routing algorithm. That is:

1) In the presence of one upward (downward) pillar placed at any X,Y coordinates, any node can reach any node in the upper (lower) tiers. One upward pillar and one downward pillar guarantee full connectivity, regardless of the position of the two pillars in the network, and of their positions with respect to each other.
2) During the routing process, any pillar in the network can be used to reach other tiers. No restrictions apply on the assignment of elevators.
3) Ignoring symmetries, it is the only possible routing algorithm that satisfies 1) and 2) with this number of virtual channels. That is, removing or displacing one virtual channel would result in either 1) or 2) or both being false.
4) It is the most adaptive at the source and destination layers.
5) It is the most resilient to TSV failures.

From 2) we can deduce two important requirements on the routing algorithm:

(a) At the source tier, any elevator must be reachable from any node. This implies that the routing function used to route a packet from the source towards an elevator must use all the planar directions {East, West, South, North}.
(b) At the destination tier, any destination node must be reachable from any elevator. This means that the routing function used to route a packet from the source elevator to the final destination also uses all the planar directions {East, West, South, North}.

Lemma 5.1. *If at least one dimension of the plane (X, Y) contains no virtual channels, then the routing algorithm is not deadlock-free.*

Proof. Given (a) and (b), it is always possible to form a cycle using Up, Down and the two directions from the dimension that contains no virtual channels. □

From Lemma 5.1, it follows that at least one VC is required per planar dimension. This proves that First-Last is indeed using the minimum required number of VCs to satisfy 2), and subsequently 1).

We now demonstrate that under this minimal VC configuration (a VC in one of the directions of each planar dimension), First-Last is optimal in terms of adaptiveness as well as the number of supported irregular topologies, i.e. resilience to TSV failures.

The First-Last algorithm is adaptive in two out of the four quadrants that constitute the mesh tiers, both when routing at the source tier and the destination tier.

Theorem 5.2 (Maximum adaptiveness)**.** *Under this VC configuration, it is not possible for a routing algorithm to be adaptive in more than two quadrants.*

Proof. Without loss of generality, let us assume that the 2 VCs are included in the North and East directions.

If the routing algorithm is adaptive in more than two quadrants at the **destination tier**, then one of these two quadrants must be adaptive: South-East, North-West. If we consider symmetries, the two cases are equivalent, so let us consider the case where the South-East quadrant is adaptive.

Then, one of the two following cases must be true:

- Case 1) The North-East quadrant is not adaptive, which means that the North-West quadrant is adaptive.
- Case 2) The North-West quadrant is not adaptive, which means that the North-East quadrant is adaptive.

In case 1, a cycle can be formed as follows:

$East \rightarrow South \rightarrow Down \rightarrow North \rightarrow West \rightarrow Up \rightarrow East$.

We now explain why each of the turns involved in this cycle is always possible:

- $East \rightarrow South$ because we are under the assumption that the South-East quadrant is adaptive. We assume the packet making this turn is at its destination tier.
- $South \rightarrow Down$ because any elevator should be reachable following minimal distance at the source tier. The packet making the turn is therefore at its source tier. It is worth reminding that the West and South channels only include one VC, shared between packets at their source tier and packets at intermediate and destination tiers.
- $Down \rightarrow North$ is possible for a packet in its destination tier, to ensure all destinations are reachable. The packet making the turn is at its destination tier.
- $North \rightarrow West$ because the North-West quadrant is assumed adaptive in this case.
- $West \rightarrow Up$ because packets at their source tier can reach any elevator. Again, the packet must be at its source tier.
- $Up \rightarrow East$ because packets at their destination tier can take any direction.

Following a similar reasoning, in case 2, the following cycle can be formed:

$North \rightarrow East \rightarrow South \rightarrow Down \rightarrow West \rightarrow Up \rightarrow North$.

Which is always possible since the North-East and South-East quadrants are both adaptive.

The East and North channels are occupied by packets at their destination tier. Note that because we made sure that the up and $down$ directions are only taken after the $south$ and $west$ directions, which is always possible due to requirement (a) presented above, this cycle can be formed regardless of which restrictions are applied on the usage of the two VCs in the East and North directions.

This proves that at the destination tier, the algorithm cannot be adaptive in more than two quadrants.

At the source tier, different cycles can be formed for case 1) and case 2):

Case 1)

$West \rightarrow North \rightarrow Down \rightarrow South \rightarrow East \rightarrow East \rightarrow Up \rightarrow West$.

Case 2)

$Down \rightarrow South \rightarrow East \rightarrow Up \rightarrow West \rightarrow North \rightarrow Down$.

Here, the East and North channels are occupied by packets at their source tiers. Note that the rules imposed on the use of VCs is irrelevant here, as we are exploiting the dependencies towards channels with single VCs (the vertical directions).

The existence of such cycles independently of the rules applied to acquire VCs proves the theorem.

Therefore, First-Last has the maximum possible adaptiveness at the source and destination tiers. □

The difference between regular and irregular topologies is that the routing algorithm needs to use the planar channels not only at the source and destination tiers, but at the intermediate tiers as well. The number of supported topologies therefore depends on how many channels can be used in the intermediate tiers. First-last uses two channels (West and South) to route in the intermediate layers. We demonstrate that under this VC configuration, it is the maximum number of channels that can be used for routing in intermediate layers.

Theorem 5.3 (Maximum resilience)**.** *When routing towards an elevator at a tier located between the source tier and the destination tier, then at most two physical channels can be used.*

Proof. Let E be the set of directions that include escape VCs, and let $\bar{E}$ be the set of directions not including VCs. If more than two directions are used for routing in intermediate layers then there must be at least two directions that are opposite to each other (same dimension). Let d one of these two directions such that $d \in E$, and let $\bar{d}$ be the second direction such that $\bar{d} \in \bar{E}$.

A cycle can be formed between two intermediate layers using d, $down$, $\bar{d}$ and up. Note that directions $\bar{d}$, up and $down$ only include one VC, so the cycle can be formed regardless of how many VCs are present in direction d.

This proves that First-Last is the most resilient algorithm that satisfies the previously described criteria. □

6 Implementation Details

In order to reliably forward packets to other layers, every router in the network must contain some information about the location of elevators in the same layer. While it is possible to provide each router with the addresses of all elevators and have it select the best one at runtime, it will incur a very hardware overhead. A more feasible approach is to perform the selection offline, and to provide limited information about the location of elevators to the routers, so that they can use it online.

In this section, we present a mechanism for performing TSV selection for the First-Last and Enhanced-First-Last routing algorithms, that uses a fixed number of bits regardless of the tier sizes and selects an elevator at runtime in a fully distributed manner.

6.1 Manhattan-Distance-Based TSV selection

One sensible criterion of selection is the distance of the elevator from the current router. That is, each router forwards a packet towards an elevator located at minimum Manhattan distance. The main idea behind this choice is to make sure packets spend as little time as possible at intermediate layers, and are able to reach their destination layer as quickly as possible.

6.1.1 Storage requirements

Packets that enter a new layer in virtual network VN0 are able to reach an elevator located at any position. Therefore, each router stores the location of the upward and downward nearest elevators as two 4-bit vectors named $Elevator\_Up$ and $Elevator\_Down$, respectively. The 4 bits $\{East, South, West, North\}$ indicate whether the selected elevator is located East, South, West or North, respectively.

However, for packets that enter a layer in VN1, an elevator that is reachable only in VN1 must be used. Because $Elevator\_Up$ and $Elevator\_Down$ may point to arbitrary positions, they cannot be relied on for routing packets that have arrived from other layers in VN1. Therefore, we also need to provide information about the nearest upward and downward elevators reachable only using the negative channels (West, South). Since VN1 only includes two planar directions, two bits $\{South, West\}$ per elevator are required. The two 2-bit vectors are named $Elevator\_Up\_Neg$ and $Elevator\_Down\_Neg$.

In sum, each router must include 12 reconfigurable bits to locate the nearest eligible elevator. These bits are configured offline for each router to point to its nearest elevator using the method described next.

6.1.2 Offline selection algorithm

It should be taken into consideration that the distributed nature of this selection method allows different routers to point to different nearest elevators, and consequently, one router that forwards a packet in the direction of its nearest elevator cannot guarantee that it will reach that same elevator after traversing the next hops.

Since there can be several elevators with an equal Manhattan distance from a given node, we show that the method used to break the ties can actually have an impact on reachability.

In effect, if ties are broken in a fully randomized manner as suggested in [22], the inconsistencies between several nodes can lead to the violation of the routing rules, potentially leading to deadlocks.

Consider the example shown in Fig. 6. Here, elevators $E1$ and $E2$ are at equal distance from both nodes A and B. If the selection algorithm assigns elevator $E1$ to node A and elevator $E2$ to B as shown in the example, node B will route packets in VN1 to reach $E2$. After reaching node A, however, these packets are routed closer to $E1$ in VN0. Taking VN0 channels after visiting VN1 is a violation of the routing rules, and will eventually result in deadlocks.

Alternatively, the routing logic could be altered to explicitly ensure that the routing rules do not get violated. In this case however, in addition to the extra hardware complexity, if a situation occurs where an illegal turn is required to reach the nearest elevator, then the routing logic would be incapable of selecting a new elevator on the fly due to its limited knowledge about the TSV locations.

For instance, going back to the example in Fig. 6, even if router A were designed to never forward a packet coming from the East port to the North, it would not be able to determine if the packet was meant for an elevator in the West, or in the South, as both are valid VN1 channels.

One way to solve this issue offline without changing the hardware is to break the ties by giving priority to the elevators that are reachable using only VN1 channels. The offline selection algorithm used to set the $Elevator\_Up$ and $Elevator\_Down$ vectors is presented in Algorithm 1.

According to Algorithm 1, an elevator requiring VN0 channels is only selected if there are no nearest elevators reachable using only VN1 channels. If a node A sends a packet to another node B to reach an elevator E_A using the negative directions (VN1), then the receiving router B is guaranteed to also point to an elevator E_B reachable in VN1. If B had another elevator E'_B that requires VN0, then E'_B must be closer to B than E_B, which means it is also closer to A than E_A and would have been selected as the elevator of A instead of E_A. Therefore, conflicts can never arise.

Algorithm 1 Setting the Elevator Bits

```
Output:
    Elevator[LayerNodes] : Elevator location bits
    ($Elevator\_Up$ or $Elevator\_Down$)
 1: for all node i do
 2:     Initialize Elevator[i] to $\{0, 0, 0, 0\}$
 3: end for
 4: for all node i of coord (x, y) do
 5:     $E \leftarrow$ Set of eligible upward (or downward) elevators
 6:     Sort E By distance from i
 7:     $Min \leftarrow e \in E / distance(e, i) = distance(E[0], i)$
 8:     $Neg \leftarrow (x', y') \in Min / x' \leq x$ and $y' \leq y$
 9:     if $Neg$ is empty then
10:         $(xE, yE) \leftarrow$ random elevator from Min
11:     else
12:         $(xE, yE) \leftarrow$ random elevator from $Neg$
13:     end if
14:     $Elevator[i].North \leftarrow (yE > y)$
15:     $Elevator[i].South \leftarrow (yE < y)$
16:     $Elevator[i].West \leftarrow (xE < x)$
17:     $Elevator[i].East \leftarrow (xE > x)$
18: end for
```

Setting the $Elevator\_Up\_Neg$ and $Elevator\_Down\_Neg$ vectors for packets in VN1 is done using Algorithm 2.

Algorithm 2 Setting the intermediate (VN1) Elevator Bits

```
Output:
    Elevator[LayerNodes] : Elevator location bits
    ($Elevator\_Up\_Neg$ or $Elevator\_Down\_Neg$)
 1: for all node i do
 2:     Initialize Elevator[i] to $\{0, 0\}$
 3: end for
 4: for all node i of coord (x, y) do
 5:     $E \leftarrow$ Set of eligible upward (or downward) elevators
 6:     $N \leftarrow (x', y') \in E / x' \leq x$ and $y' \leq y$
 7:     Sort $N$ By distance from i
 8:     $Min \leftarrow e \in N / distance(e, i) = distance(V[0], i)$
 9:     assert($Min$ is not empty)
10:     $(xE, yE) \leftarrow$ random elevator from $Min$
11:     $Elevator[i].South \leftarrow (yE < y)$
12:     $Elevator[i].West \leftarrow (xE < x)$
13: end for
```

6.2 Route computation logic

The route computation logic is presented in Algorithm 3. As an input, the algorithm takes a bit vector $Dest$ describing the position of the destination (Up, Down, East, West, South, North), and the current virtual network number v_{in}. The $Dest$ vector is obtained by comparing the position of the current router to that of the destination indicated in the packet header. The packet header also stores the virtual network number v_{in}, as mentioned in section 4. The algorithm outputs the set of possible output ports, and possibly a new virtual network number, if moving to the next virtual network is necessary.

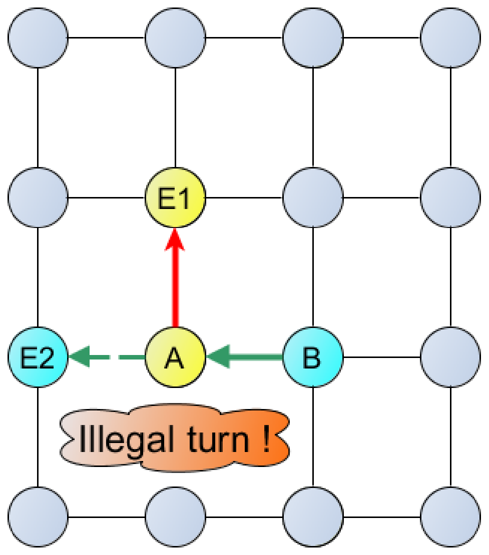


Fig. 6. Example of making an illegal turn due to an incorrect selection algorithm.

If the destination is on the same layer as the current router (Algorithm 3 - Line 32), routing is performed following the $Positive\_Last$ routing algorithm presented in Algorithm 5. Otherwise, the algorithm first computes the position of the appropriate elevator according to the destination layer (up or down) and whether the packet has entered virtual network 1 or not (Algorithm 3 - Lines 2 to 17).

If the current router is an elevator (Algorithm 3 - Line 18), then the packet is forwarded appropriately either to the up or down port. With Enhanced-First-Last, moving up and down is possible in both VN0 and VN1, so changing the virtual network is not necessary. However, in the First-Last algorithm (Algorithm 3 - Line 20), moving vertically can only be done in VN1, so the VN number must be updated.

If the current router is not an elevator, the packet is routed towards the selected elevator following the $Positive\_First$ routing algorithm presented in Algorithm 4. It is worth noting that the routing algorithm is very simple and does not check where the packet comes from to enforce the turn restrictions. The routing rules are guaranteed to never be violated thanks to the offline selection algorithms presented previously.

7 Experimental results

In this section, we compare First-Last to two routing algorithms from the literature, namely Elevator-First [8] and Redelf [20]. Elevator-First uses two virtual channels per planar direction, i.e. two more virtual channels per router compared to First-Last. However, it offers partial adaptiveness in all the tiers, including intermediate tiers. It poses no restrictions on the placement or selection of TSVs and is general enough to support any layer topology. Redelf, by contrast, uses only one VC per direction but restrictions are imposed on the selection of TSVs. We will be comparing First-Last to variants of these algorithms both in terms of implementation cost and performance.

Algorithm 3 Route computation logic

```
Input:
    $Dest$: Destination position bits
    $v\_in$: Current virtual network
Output:
    $R$ : Set of possible output channels
    $v\_out$ : Output virtual network
1: if $Dest.Up$ or $Dest.Down$ then
2:     if $v\_in = 0$ then                ▷ Have not moved to V1 yet
3:         if $Dest.Up$ then
4:             $Elevator \leftarrow Elevator\_Up$
5:         else
6:             $Elevator \leftarrow Elevator\_Down$
7:         end if
8:     else
9:         $Elevator \leftarrow \{0, 0, 0, 0\}$
10:        if $Dest.Up$ then
11:            $Elevator.West \leftarrow Elevator\_Up\_Neg.West$
12:            $Elevator.South \leftarrow Elevator\_Up\_Neg.Sout$
13:        else
14:            $Elevator.West \leftarrow Elevator\_Dn\_Neg.West$
15:            $Elevator.South \leftarrow Elevator\_Dn\_Neg.Sout$
16:        end if
17:    end if
18:    if $Elevator = \{0, 0, 0, 0\}$ then          ▷ Self is elevator
19:        $v\_out \leftarrow v\_in$
20:        if not using Enhanced-First-Last then
21:            $v\_out \leftarrow 1$ ▷ Must move to VN1 to go vertical
22:        end if
23:        if $Dest.Up$ then
24:            $R \leftarrow \{Z+\}$
25:        else
26:            $R \leftarrow \{Z-\}$
27:        end if
28:    else
29:        $(R, v\_out) \leftarrow Positive\_First(Elevator, v\_in)$
30:    end if
31: else
32:    $(R, v\_out) \leftarrow Positive\_Last(Dest, v\_in)$
33: end if
```

7.1 Hardware synthesis

To analyze the hardware cost of the proposed solutions, we have extended the Netmaker library [27] to support 3D router architectures. Netmaker is a library of parameterizable and synthesizable NoC routers written in SystemVerilog. This library was used to implement the First-Last and Enhanced-Fist-Last routing algorithms described in this paper, as well as the Elevator-First algorithm described in [8] and Redelf presented in [20].

All routers include 4-flit deep virtual channel FIFOs and perform virtual channel allocation followed by switch allocation (2 cycles). To evaluate the area overhead, we synthesize the Elevator-First, First-Last, Enhanced-First-Last, and Redelf routers using Synopsys Design Compiler. The designs were setup to work with an operating frequency of 1GHz, a power supply of 0.8V, and a NanGate Open Cell $15nm$ Library [28]. The resulting area and power estimates for each router are summarized in Table 1. The three types

Algorithm 4 The Positive_First routing function

```
Input:
    $Dest$ : Destination position bits
    $v\_in$ : Current virtual network
Output:
    $R$ : Set of possible output channels
    $v\_out$ : Output virtual network
1: $v\_out \leftarrow v\_in$
2: if $Dest.East$ and $Dest.North$ then
3:     $R \leftarrow \{X+, Y+\}$
4: else if $Dest.East$ then
5:     $R \leftarrow \{X+\}$
6: else if $Dest.North$ then
7:     $R \leftarrow \{Y+\}$
8: else
9:     $v\_out \leftarrow 1$
10:    if $Dest.West$ and $Dest.South$ then
11:        $R \leftarrow \{X-, Y-\}$
12:    else if $Dest.West$ then
13:        $R \leftarrow \{X-\}$
14:    else if $Dest.South$ then
15:        $R \leftarrow \{Y-\}$
16:    else
17:        $R \leftarrow \emptyset$
18:    end if
19: end if
```

Algorithm 5 The Positive_Last routing function

```
Input:
    $Dest$: Destination position bits
    $v\_in$: Current virtual network
Output:
    $R$ : Set of possible output channels
    $v\_out$ : Output virtual network
1: $v\_out \leftarrow v\_in$
2: if $Dest.West$ and $Dest.South$ then
3:     $R \leftarrow \{X-, Y-\}$
4: else if $Dest.West$ then
5:     $R \leftarrow \{X-\}$
6: else if $Dest.South$ then
7:     $R \leftarrow \{Y-\}$
8: else
9:     $v\_out \leftarrow 2$
10:    if $Dest.East$ and $Dest.North$ then
11:        $R \leftarrow \{X+, Y+\}$
12:    else if $Dest.East$ then
13:        $R \leftarrow \{X+\}$
14:    else if $Dest.North$ then
15:        $R \leftarrow \{Y+\}$
16:    else
17:        $R \leftarrow \{L\}$                    ▷ Local port
18:    end if
19: end if
```

of routers described in Section 3 were considered: 5 port 2D routers, 6 port 3D routers with one vertical connection, and 7 port 3D routers with two vertical connections. Furthermore, the 6 and 7 port 3D router have serial block circuit in the up and down ports, considering a sender and a receiver, serializing and deserializing data, respectively. Based on the work given in [29] and [30], we have adopted a 4:1 Serializer-Deserializer block, where 4 data bits are transmitted on one serial line, reducing the number of interconnect TSVs in each 3D router by four.

First, we compare the area overhead for a 5-port router. The area for First-Last and Enhanced-First-Last is the same because, in a 2D router, both algorithms require the same number of virtual channels and use the same routing logic. On the other hand, although Elevator-First uses a simple XY routing function for routing, it includes one extra virtual channel in the South and West directions, which increases the area and power overhead by 12% and 18% respectively. Since Redelf does not require extra virtual channel, the synthesis results present less area and power than others routing algorithms.

A similar comparison can be done for a 6-port router. In this case, Elevator-First has two more virtual channels than First-Last and one more virtual channel than Enhanced-First-Last. Those additional virtual channels can explain the area and power overhead observed for Elevator-First, which are of approximately 10% and 13% compared with First-Last, and 4% and 6% with respects to Enhanced-First-Last. However, when comparing Elevator-First and Enhanced-First-Last in the case of a 7-port router, where the total number of virtual channels is the same, we note a slightly larger area (less than 1%) and power (less than 2%) when using Enhanced-First-Last, due to its more complex routing logic when compared to the simple XY algorithm used by Elevator-First.

In sum, the Enhanced-First-Last algorithm can be considered an appealing alternative to Elevator-First, as it not only reduces the area and power, especially for designs that use more 5-port and 6-port routers than 7-port routers, but also is capable of attaining higher levels of performance, as will be shown in the rest of this section.

7.2 Performance evaluation

We use an in-house cycle-accurate NoC simulator based on [31] to compare First-Last and Enhanced-First-Last to Elevator-First and Redelf. To make sure the tested topologies are compatible with all three algorithms, we only generate regular partially connected topologies. All the topologies consist of only TSV pillars, i.e. TSVs are placed on the same X,Y coordinates across all layers and all the vertical channels are bidirectional, as in Fig. 5 (B). The latter restriction is required for Redelf to be deadlock-free. The X,Y coordinates of the TSVs are selected randomly, so that we do not advantage any of the algorithms. We consider 4 different TSV densities (12.5%, 25%, 50% and 75%), and 30 random topologies are tested for each TSV density. We evaluate the algorithms under two different network sizes: 4x4x4, 8x8x4 with three types of synthetic traffic (Uniform, Complement, Shuffle). Adaptive algorithms select the least congested output port among available options based on a local congestion metric.

TABLE 1
Hardware synthesis results

Type	**Elevator-First**		**First-Last**		**Enhanced-First-Last**		**Redelf**	
# Ports	Area (μm^2)	Power (mW)	Area (μm^2)	Power (mW)	Area (μm^2)	Power (mW)	Area (μm^2)	Power (mW)
5 ports	11982	9.3	10646	7.9	10646	7.9	8793	6.3
6 ports	16283	12.1	14813	10.6	15659	11.4	13041	9.6
7 ports	20792	14.8	19303	13.4	20958	15.1	17607	12.7

A congestion metric is associated with each output port and is computed as follows: Whenever a packets requests the associated output port, the metric is incremented by twice the number of flits in the packet. Then, whenever a flit leaves from the output port, the metric is decremented by one. Finally, the metric is decremented by one when the flit leaves the downstream router, i.e. when a credit is received. The metric therefore accounts for the number of flits requesting a given output port as well as the number of flits contained in the downstream buffers. It is worth mentioning that all the simulations end with a drain phase, during which all the packets remaining in the network are ejected. This is to make sure that no deadlocks have occurred in any of the simulated scenarios.

For First-Last and Redelf, two versions are considered. The first version (First-Last and Redelf) uses the minimum number of VCs required for the algorithm to be deadlock-free. In the second version (First-Last-2VC and Redelf-2VC), the same number of virtual channels as Elevator-first is used (2 VCs at each planar port). Any extra VCs that are not required for deadlock-avoidance are used to reduce head-of-line blocking and improve performance. For Enhanced-Elevator-First, on the other hand, we do not add extra VCs in the planar ports due to its particular distribution of VCs.

Results are presented in Fig. 9 and Fig. 10. An interesting first observation is that Enhanced-Elevator-First offers the best performance in most of tested scenarios. This confirms our prediction on the importance of relaxing the pressure on vertical channels by adding an extra VC along the Z dimension (see Section 4.2). The only cases where Enhanced-Elevator-First is outperformed by First-Last-2VC and/or Elevator-First are with the Shuffle traffic pattern. In shuffle traffic, many nodes communicate with nodes on the same tier, which means that the number of VCs in the planar channels is more important in this case, especially when the tiers are quite large (8x8). Both Elevator-First and First-Last-2VC include 2 VCs in all the planar directions, unlike Enhanced-Elevator-First which includes only 1VC in the South and West directions.

We notice that Elevator-First sometimes performs better than First-Last-2VC under uniform and complement traffic. These types of traffic are known to favor direction order routing over adaptive routing.

The second thing to take note is that Redelf exhibits, most of the time, a higher latency than the other solutions, even when using extra VCs (Redelf-2VC). This highlights the importance of the two restrictions imposed by Redelf to avoid using virtual channels:

- Packets can only take south-eastern elevators if any. When several elevators are available, Elevator-First or First-Last are able to select an elevator in any direction, resulting in a better load balance between elevators. This translates to a large difference in performance for large layer sizes (8x8).
- When no elevators are available in the south-east direction, the pivot elevator must be taken. This typically results in both a high pressure on the pivot elevator, and longer paths for many packets. The other algorithms can not only select elevators in any directions, but choose the closest elevator to minimize the traveled distance.

To back this up, we perform an extra set of measurements to obtain more information about the distribution of packets over elevators. For each simulated algorithm, and

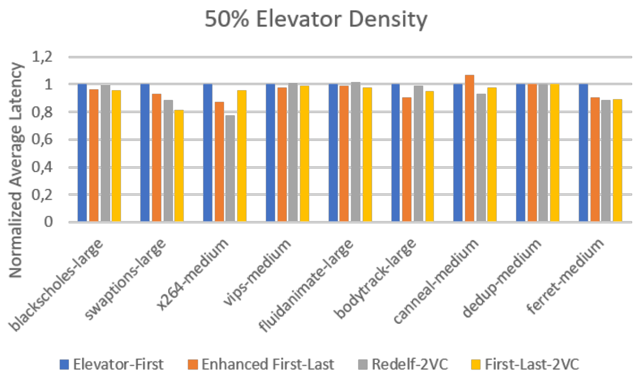


Fig. 7. Average packet latency in a 4x4x4 network.

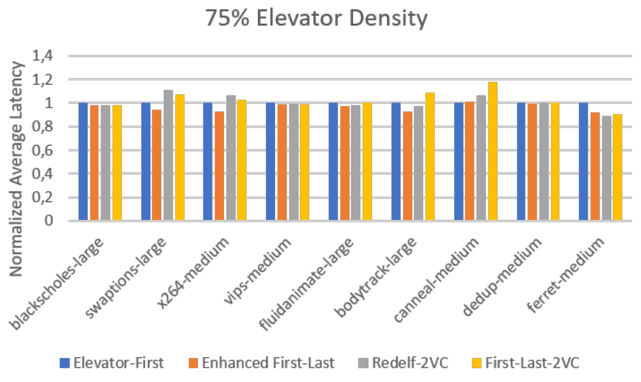


Fig. 8. Average packet latency in a 4x4x4 network.

for a given TSV density, we generate 1000 random partially connected topologies of size 8x8x2 (two tiers are enough to perform this experiment). We then inject a fixed number of packets (300 packets/node) with random destinations and count the number of times that each elevator was used, then we estimate the load imbalance between elevators based on two metrics: Percent imbalance (υ) and Standard deviation (σ). Such that:

$$\upsilon = \frac{U_{max}}{\bar{U}} - 1 \tag{1}$$

where U_{max} is the maximum elevator usage across all elevators, and U_{max} the mean elevator usage.

$$\sigma = \sqrt{\frac{\sum_{i=0}^{E-1}(U_i - \bar{U})^2}{E-1}} \tag{2}$$

where E is the total number of elevators, $\bar{U}$ is the mean elevator usage, and U_i the number of times elevator i was used.

Table 2 shows the results obtained with different algorithms. First, we observe that none of the algorithms is able to achieve good balance between the different elevators. This is because even in Elevator-First and First-Last, elevators are assigned according to their distance from the source, therefore it is not possible to ensure that the load is distributed evenly among the elevators since they are placed at arbitrary positions. Interestingly, we notice that under high elevator availability, First-Last does not distribute the load among elevators as well as Elevator-First. This is due to the algorithm used to assign elevators (Algorithm 1), which gives priority to elevators located in a given direction when breaking ties between several nearest elevators. However, the difference is minor and should not have a significant impact on performance.

In all cases, imbalance is much more severe in the case of Redelf. The most used elevator has nearly 100% more load than the most used elevator in Elevator-First and First-Last, thereby confirming our claim that the performance of Redelf is most likely impacted by high traffic concentration around the pivot elevator.

There are a few cases where Redelf is expected to perform better than other algorithms. This is the case when the layer size is sufficiently small, and few elevators are available. Chances are that all the algorithms will utilize the same set of elevators, and take the same paths in these cases. We observe that for a 4x4 layer size, Redelf-2VC sometimes performs slightly better than First-Last-2VC and Elevator-First. In most of these cases, however, First-Last-2VC performs as well as Redelf-2VC, even in presence of 1 elevator (12.5% TSV density). Two factors come into play here. First, although First-Last-2VC uses the VCs in the North and East direction for deadlock-avoidance, it does not strictly partition the VCs as it is the case in Elevator-First. At the destination tier, all VCs can freely be used just like Redelf-2VC. Second, unlike Redelf-2VC which strongly relies on deterministic routing, First-Last is adaptive and is capable of reducing congestion.

These results clearly demonstrate the benefits of the proposed routing solutions, as well as the outstanding performance they can offer while using the strict minimum number of VCs to guarantee connectivity and total freedom on TSV selection in the simulated topologies (See Section 5.4).

TABLE 2
Elevator load balance

Algorithm	**Elevator-First**		**First-Last**		**Redelf**	
# Elevators	σ	υ	σ	υ	σ	υ
4	1169.89	0.60	1137.99	0.54	2429.72	1.41
8	637.38	0.86	637.30	0.88	1156.19	1.99
16	395.33	1.62	399.59	1.64	552.37	2.57
24	197.59	1.17	210.03	1.26	302.64	2.41

7.3 Evaluation under realistic traffic

To confirm these conclusions under more realistic scenarios, we also evaluate the algorithms with real application traffic from the PARSEC benchmark suite. We use the traces provided by Netrace [32] to capture packet dependencies.

For each algorithm, the first 20 million cycles of the region of interest for each benchmark are simulated. This corresponds to the time stamp contained in the traces themselves, not to the actual simulation time, as the latter may differ from one algorithm to the other. We therefore ensure that the same portion of the benchmark is executed for all algorithms.

The average latencies are captured by simulating 10 Random topologies for each application. Normalized results are presented in Fig. 7 and Fig. 8.

One observation that can be made is that the relative performance of the different routing solutions heavily depends on the type of benchmark and the TSV availability. For the topologies we tested with 50% TSV density, we observe that First-Last-2VC, Enhanced-First-Last, and Redelf perform better than Elevator-First. As pointed out previously, because the number of TSVs is low, all three algorithms are likely to take similar paths, and the difference mainly relies on the intra-layer performance (number of planar VCs, freedom of VC selection and adaptivity), which is why First-Last-2VC and Redelf perform the best for most benchmarks.

An interesting observation is that in benchmarks like swaptions and x264, Redelf-2VC performs better than First-Last-2VC and/or Enhanced-First-Last at low TSV density. However, note that the relative performance of our solutions changes as soon as the number of TSVs is increased. This can be explained by the fact that our solutions are able to distribute the traffic better among the available TSVs, and benefit from a more significant gain in performance.

8 Conclusion

We have presented a novel algorithm targeting partially vertically connected 3D-NoCs named First-Last, that guarantees packet delivery as long as one TSV pillar is available anywhere in the network. Through a unique assignment of VCs, we have managed to avoid all the restrictions that related works impose on either the placement of the pillars, their selection during runtime, or both. Moreover, we have shown, through Enhanced-First-Last, that by adding one VC in the vertical dimension, it was possible to dramatically

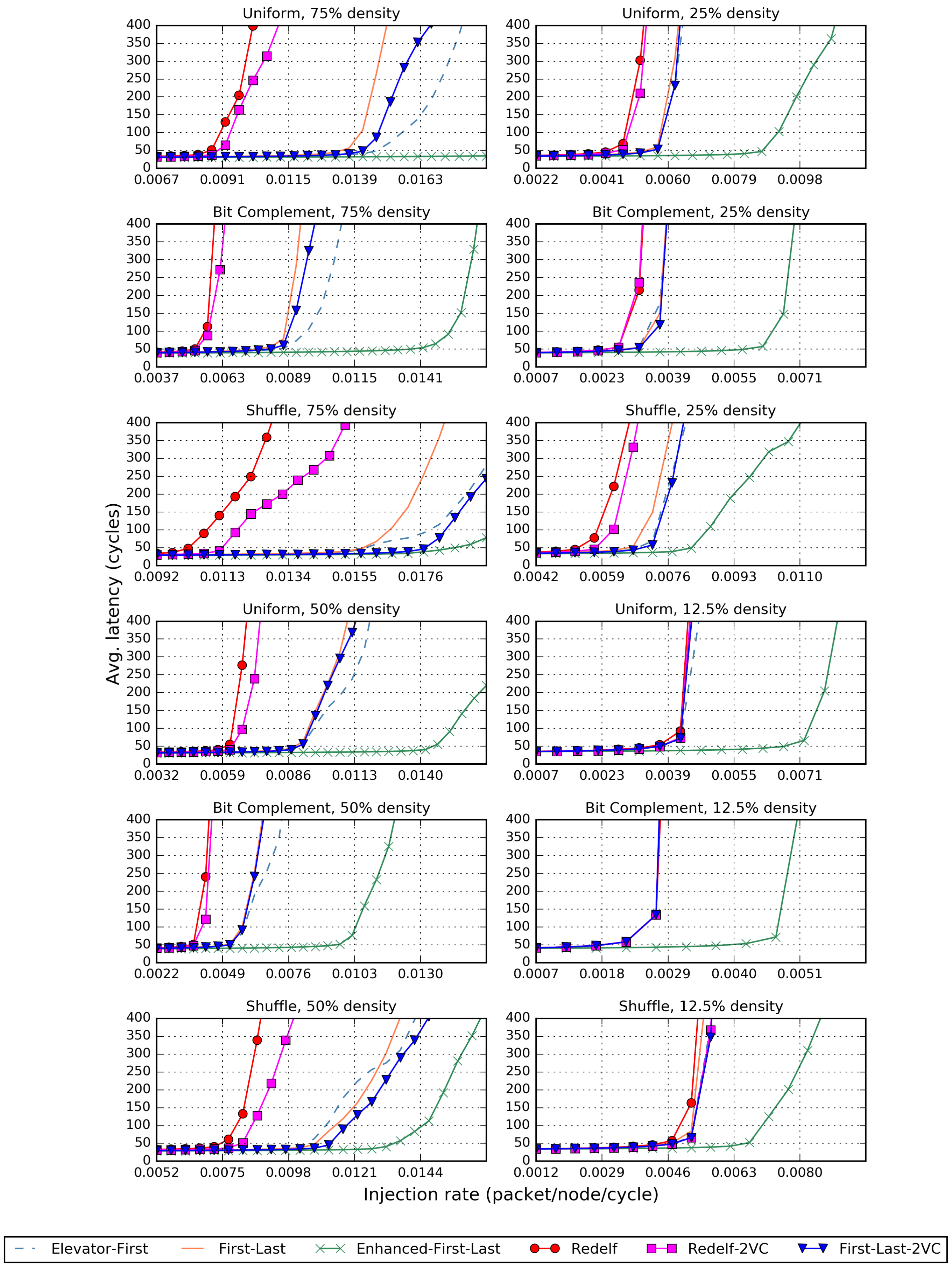


Fig. 9. Average packet latency in a 4x4x4 network.

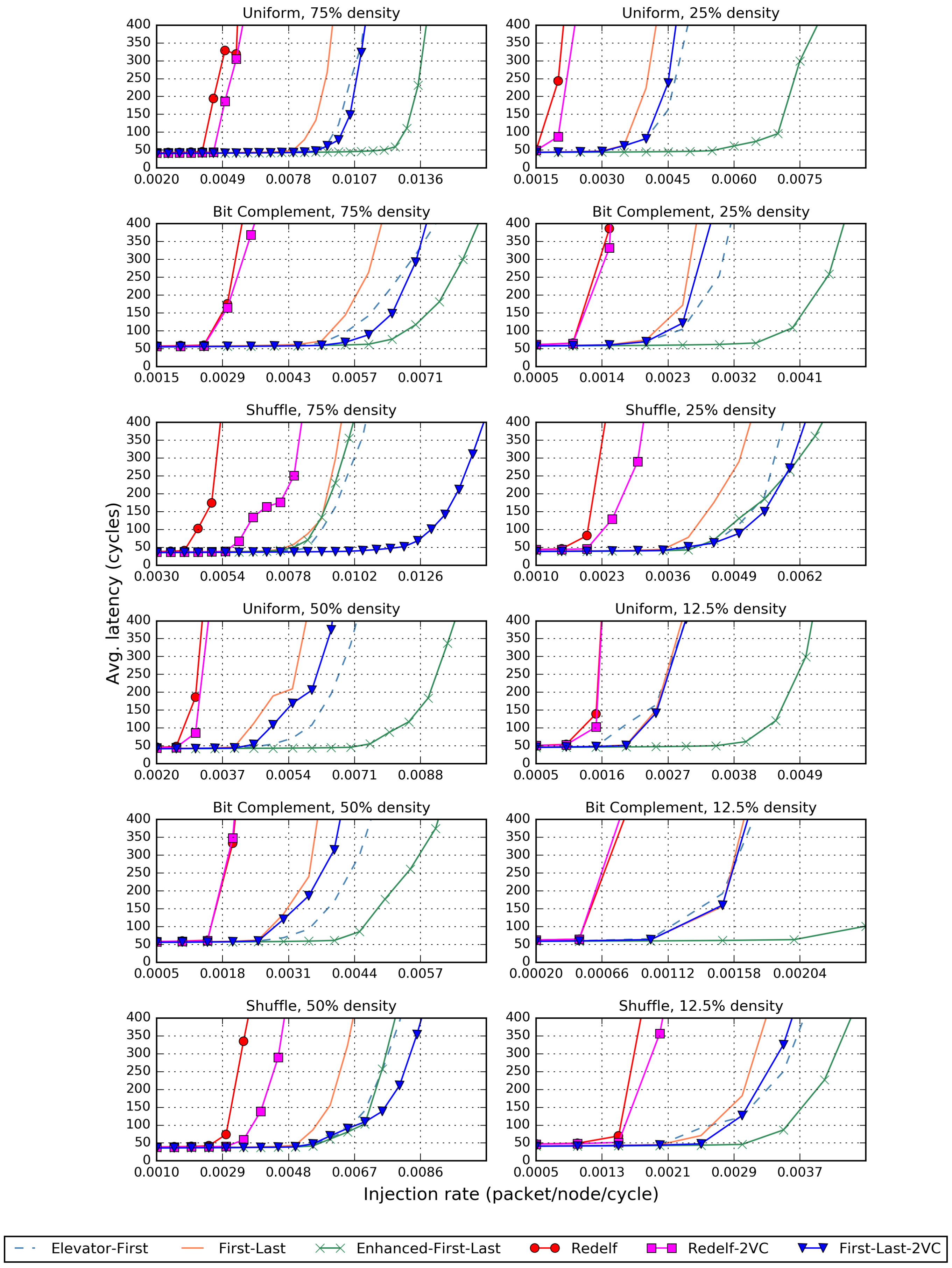


Fig. 10. Average packet latency in a 8x8x4 network.

boost the NoC's performance in presence of few vertical connections, while still ensuring a lower implementation cost than state-of-the-art algorithms. Both hardware synthesis and comprehensive cycle-accurate simulations were performed to demonstrate the merits of our routing approach.

References

[1] W. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference*, 2001, pp. 684–689.

[2] A. Bakhoda, J. Kim, and T. M. Aamodt, "Throughput-effective on-chip networks for manycore accelerators," in *Proceedings of the Annual International Symposium on Microarchitecture, MICRO*. IEEE, dec 2010, pp. 421–432.

[3] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C. C. Miao, J. F. Brown, and A. Agarwal, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, sep 2007.

[4] B. S. Feero and P. P. Pande, "Networks-on-chip in a three-dimensional environment: A performance evaluation," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32–45, jan 2009.

[5] V. Pavlidis and E. Friedman, "3-D Topologies for Networks-on-Chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 285–288, sep 2006.

[6] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon, "Demystifying 3D ICs: The pros and cons of going vertical," pp. 498–510, jun 2005.

[7] A. Eghbal, P. M. Yaghini, N. Bagherzadeh, and M. Khayambashi, "Analytical Fault Tolerance Assessment and Metrics for TSV-Based 3D Network-on-Chip," *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3591–3604, dec 2015.

[8] M. Bahmani, A. Sheibanyrad, F. Pétrot, F. Dubois, and P. Durante, "A 3D-NoC router implementation exploiting vertically-partially-connected topologies," *Proceedings - 2012 IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2012*, pp. 9–14, 2012.

[9] J. Lee and K. Choi, "A deadlock-free routing algorithm requiring no virtual channel on 3D-NoCs with partial vertical connections," *2013 7th IEEE/ACM International Symposium on Networks-on-Chip, NoCS 2013*, pp. 0–1, 2013.

[10] R. Salamat, M. Ebrahimi, and N. Bagherzadeh, "An Adaptive, Low Restrictive and Fault Resilient Routing Algorithm for 3D Network-on-Chip," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, mar 2015, pp. 392–395.

[11] A. Charif, N.-E. Zergainoh, A. Coelho, and M. Nicolaidis, "Rout3d: A lightweight adaptive routing algorithm for tolerating faulty vertical links in 3d-nocs," in *22th IEEE European Test Symposium (ETS'17)*. ACM IEEE, 2017, pp. 1–6.

[12] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "Fault-tolerant method with distributed monitoring and management technique for 3D stacked meshes," in *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)*. IEEE, oct 2013, pp. 93–98.

[13] S. Pasricha and Y. Zou, "A low overhead fault tolerant routing scheme for 3D networks-on-chip," *Proceedings of the 12th International Symposium on Quality Electronic Design, ISQED 2011*, pp. 204–211, 2011.

[14] C. Glass and L. Ni, "The Turn Model for Adaptive Routing," in *Proceedings the 19th Annual International Symposium on Computer Architecture*, vol. 11, no. 7. New York, NY, USA: IEEE, 1992, pp. 278–287.

[15] S. Akbari, A. Shafiee, M. Fathy, and R. Berangi, "AFRA: A low cost high performance reliable routing for 3D mesh NoCs," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, mar 2012, pp. 332–337.

[16] F. Dubois, A. Sheibanyrad, F. Petrot, and M. Bahmani, "Elevator-First: A Deadlock-Free Distributed Routing Algorithm for Vertically Partially Connected 3D-NoCs," *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 609–615, mar 2013.

[17] H. Ying, K. Hofmann, and T. Hollstein, "Dynamic quadrant partitioning adaptive routing algorithm for irregular reduced vertical link density topology 3-Dimensional Network-on-Chips," in *2014 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, jul 2014, pp. 516–522.

[18] R. Salamat, M. Ebrahimi, N. Bagherzadeh, and F. Verbeek, "CoBRA: Low cost compensation of TSV failures in 3D-NoC," in *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. IEEE, sep 2016, pp. 115–120.

[19] R. Salamat, M. Khayambashi, M. Ebrahimi, and N. Bagherzadeh, "A Resilient Routing Algorithm with Formal Reliability Analysis for Partially Connected 3D-NoCs," *IEEE Transactions on Computers*, vol. 13, no. 9, pp. 1–1, 2016.

[20] J. Lee, K. Kang, and K. Choi, "Redelf: An energy-efficient deadlock-free routing for 3d nocs with partial vertical connections," *J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 3, pp. 26:1–26:22, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2751560

[21] J. Flich and J. Duato, "Logic-based distributed routing for nocs," *IEEE Computer Architecture Letters*, vol. 7, no. 1, pp. 13–16, Jan 2008.

[22] B. Niazmand, S. P. Azad, J. Flich, J. Raik, G. Jervan, and T. Hollstein, "Logic-based implementation of fault-tolerant routing in 3D network-on-chips," in *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*. IEEE, sep 2016, pp. 1–8.

[23] S. Foroutan, A. Sheibanyrad, and F. Petrot, "Assignment of vertical-links to routers in vertically-partially-connected 3-d-nocs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 8, pp. 1208–1218, 2014.

[24] M. Ebrahimi and M. Daneshtalab, "Ebda: A new theory on design and verification of deadlock-free interconnection networks," in *ISCA*, 2017, pp. 1–13.

[25] M. Ebrahimi, X. Chang, M. Daneshtalab, J. Plosila, P. Liljeberg, and H. Tenhunen, "DyXYZ: Fully adaptive routing algorithm for 3D NoCs," in *Proceedings of the 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2013*. IEEE, feb 2013, pp. 499–503.

[26] A. Charif, A. Coelho, N. E. Zergainoh, and M. Nicolaidis, "A dynamic sufficient condition of deadlock-freedom for high-performance fault-tolerant routing in networks-on-chips," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[27] R. Mullins. (2009) Netmaker. [Online]. Available: http://www-dyn.cl.cam.ac.uk/~rdm34/wiki

[28] Nangate. (2017) Nangate open cell library 15nm. [Online]. Available: http://www.nangate.com/?page_id=2328

[29] G. Beanato, A. Cevrero, G. D. Micheli, and Y. Leblebici, "Impact of data serialization over tsvs on routing congestion in 3d-stacked multi-core processors," *Microelectronics Journal*, vol. 51, no. Supplement C, pp. 38 – 45, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0026269215003110

[30] S. Pasricha, "Exploring serial vertical interconnects for 3d ics," in *Proceedings of the 46th Annual Design Automation Conference*, ser. DAC '09. New York, NY, USA: ACM, 2009, pp. 581–586. [Online]. Available: http://doi.acm.org/10.1145/1629911.1630061

[31] A. Charif, A. Coelho, N. E. Zergainoh, and M. Nicolaidis, "Detailed and highly parallelizable cycle-accurate network-on-chip simulation on GPGPU," in *Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC*. IEEE, jan 2017, pp. 672–677.

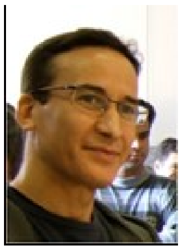
[32] J. Hestness, B. Grot, and S. W. Keckler, "Netrace: Dependency-driven trace-based network-on-chip simulation," in *Proceedings of the Third International Workshop on Network on Chip Architectures*, ser. NoCArc '10. New York, NY, USA: ACM, 2010, pp. 31–36. [Online]. Available: http://doi.acm.org/10.1145/1921249.1921258

Amir Charif received his engineer's degree in Electronics and Computer Technology from the Polytechnic school of the University of Grenoble in 2014, and his PhD in Nanoelectronics and Nanotechnology from the National Institute of Technology, Grenoble in 2017. He has published several contributions in the field of Networks-on-Chip in key conferences, including ETS, ASP-DAC, and DFT. His research interests include deadlock-free routing in NoCs and parallel simulation.

Alexandre Coelho received his B.S. degree in Electrical Engineering and M.S. degree in Teleinformatic Engineering from University Federal of Ceara, Brazil, in 2005 and 2010, respectively. He is currently working towards his Ph.D. degree at the TIMA laboratory in Grenoble, France. His research interests include Stochastic Bayesian Machines and fault tolerant Network-On-Chips. Also, his research is focused on the study of Single Event Effects (SEE) tolerance of digital circuits implemented on FPGAs and ASICs.

Masoumeh (Azin) Ebrahimi received a PhD degree with honours from University of Turku, Finland in 2013. She is currently a senior researcher at KTH Royal Institute of Technology, Sweden. Her scientific work contains more than 80 publications including book chapters, journal articles and conference papers. The majority of work has been performed on interconnection networks, fault-tolerant methods, multicast communication, and congestion-aware techniques. She actively acts as a guest editor, organizer, and program chair in different workshops and conferences. She is a member of the IEEE.

Nader Bagherzadeh received a Ph.D. degree from the University of Texas at Austin in 1987. He is a professor of computer engineering in the department of electrical engineering and computer science at the University of California, Irvine, where he served as a chair from 1998 to 2003. Dr. Bagherzadeh has been involved in research and development in the areas of: computer architecture, reconfigurable computing, VLSI chip design, network-on-chip, 3D chips, sensor networks, computer graphics, memory and embedded systems. He has published more than 300 articles in peer-reviewed journals and conferences. His former students have assumed key positions in software and computer systems design companies in the past thirty years. He has been a PI or Co-PI of research grants for developing all aspects of next generation computer systems for embedded systems as well as general purpose computing. He is a Fellow of the IEEE.



Nacer-Eddine Zergainoh received the state engineering degree in electrical engineering from National Telecommunication School and the M.S. and Ph.D. degrees in computer engineering from the University of Paris Orsay, France, in 1992 and 1996, respectively. He is currently an associate professor of computer engineering in PolytechGrenoble at the University of Grenoble Alpes and a Member of the Research Staff of TIMA labs. Prior to that, he was a research fellow at France Telecom Paris, France. Prof. Zergainoh has been involved in research and development in the areas of: System Level design, VLSI chip design, real-time system, parallel architecture, parallel programming, network-on-chip, 3D chips, fault-tolerance and reliability. He has published more than 100 articles in peer-reviewed journals and conferences. He was a recipient of two Best Paper Awards (DATE2011 and RTCSA 2007). He has trained hundreds of students who have assumed key positions in computer companies in the past twenty years. He is a member of IEEE.