

Rout3D: A Lightweight Adaptive Routing Algorithm for Tolerating Faulty Vertical Links in 3D-NoCs

Amir Charif, Nacer-Eddine Zergainoh, Alexandre Coelho, Michael Nicolaidis

Univ. Grenoble Alpes, TIMA, F-38000 Grenoble, France

CNRS, TIMA, F-38000 Grenoble, France

{Amir.Charif, Nacer-Eddine.Zergainoh, Alexandre.Coelho, Michael.Nicolaidis}@imag.fr

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. In addition, the number of vertical paths can be expected to be further reduced due to defects and runtime failures. 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 “Rout3D”. 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. We combine our algorithm with a novel offline reconfiguration method requiring only 4 bits per router to maintain connectivity upon the occurrence of faults while minimizing the implementation cost. Simulation results reveal that our algorithm is capable of sustaining a very good level of performance compared to related works, in spite of using less virtual channels.

Keywords—*network-on-chip; fault-tolerance; 3D-NoC; TSV failure; vertically partially connected 3D-NoC;*

I. 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], [4]. 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, which enables the stacking of several silicon layers and allowing for inherently low-latency three-dimensional NoC topologies (3D-NoCs) to be considered [3], [6].

Through-Silicon Via (TSV) has been accepted as one of the most viable technologies to enable vertical communication between different NoC layers [7]. However, due to its non-negligible cost, the number of vertical links must be kept to a minimum, resulting in incomplete 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 [18], the number of available TSV links 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 [17], 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 [16], [8], 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 and highly resilient routing algorithm targeting partially vertically connected 3D-NoCs named “Rout3D” (reads “Routed”). 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 all other directions), and guarantees packet delivery as long as one healthy TSV connecting all layers is available in the network, regardless of its position. Moreover, it is capable of routing packets adaptively within each layer, so as to avoid congested areas. A novel offline reconfiguration method requiring only 4 bits per router is also introduced and combined with our flexible algorithm to maintain connectivity upon the occurrence of faults. Despite the reduced number of virtual channels, we report a very good level of performance compared to related works thanks to an optimized utilization of the available resources.

The remainder of this paper is organized as follows: In section II we explore existing solutions in the context of 3D routing, with an emphasis on the works that are closest related to our contribution. In section III, the target system architecture as well as the proposed routing algorithm are described in detail. In section IV, we compare our algorithm by simulation to other works from the literature before concluding in section V.

II. RELATED WORKS

In the context of 3D-NoCs, several routing algorithms have been proposed. From simple deterministic algorithms such as ZXY, to fully adaptive algorithms such as 3D-FAR [19] and DyXYZ [21]. Fully adaptive algorithms were shown to perform well but they only operate in fully connected meshes. An extension of 3D-FAR, called 3D-FT was introduced in [19], 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 [20], the authors extend the turn model for 2D meshes [5] 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 [22] 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 [9], 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 [17], 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 [16] add certain constraints on the usage of the elevators and show that routing is possible without the use of virtual channels. In [15], 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 [8] employs a clever subnetwork decomposition to limit the virtual channel requirements, while 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 [12] or the propagation of TSV status in [10]. Unfortunately, ETW poses some very limiting constraints on both the location, and the selection of the elevators. It requires the existence of at least one elevator in the east-most column, and for packets heading south, an elevator located east to the destination must be taken, leading to highly 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.

Finally, the 3D variant of the LBDR (Logic Based Distributed Routing) was recently presented in [11]. 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 similarly to Elevator-First and requires the same **minimum** number of virtual channels as Elevator-First 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.

The algorithm that we propose in this paper also targets partially vertically connected 3D NoCs and aims at using as few virtual channels as possible while offering adaptivity to avoid

congestion. To help the reader clearly position our contribution, a comparison between some of the aforementioned algorithms and Rout3D is summarized in table 1. Our algorithm uses the same total number of VC FIFOs as ETW but does not have any requirements with regards to the position of the elevators. Like ETW and unlike Elevator-First and LBDR3D, it requires that the vertical connections be pillars, i.e. for Rout3D, a healthy TSV is one that connects all layers. To keep track of healthy elevators without having to store router addresses, we propose a scalable method that uses a small number of bits per router (only 4 bits) to guide packets to their nearest healthy elevator.

III. THE ROUTING ALGORITHM

In this section, we introduce the proposed routing solution after describing the target NoC architecture.

A. Preliminary setup

In this work, we consider a NoC consisting of stacked mesh layers connected to each other using TSV pillars called «elevators», as shown in fig. 1. At design time, for cost reduction purposes, we assume that only some of the routers are connected by vertical links. Therefore, the NoC consists of *2D routers* including only 5 input/output ports (Local, East/X+, West/X-, North/Y+, South/Y-), and *3D routers* which include 2 additional ports (Up/Z+, Down/Z-) [17].

In addition, we consider that some TSV (Through-Silicon Via) connections may become unavailable either due to manufacturing defects, aging or wear-out, rendering some elevators unusable. An elevator is considered healthy only if it connects all layers (i.e. if it is still a pillar). For instance, the vertical link at router B in fig. 1 is not considered healthy.

B. Locating Healthy Elevators with the Elevator Compass

In partially vertically connected 3D-NoCs, routers need a way to locate the available elevators in the network. To avoid storing complete node addresses inside each router, we suggest using the following approach. Each router stores 4 reconfigurable bits ($C_{east}, C_{west}, C_{north}, C_{south}$) that are used as a compass to point to the nearest elevator, hence the name *Elevator Compass*. As an example, in fig. 1, if router A wants to communicate with a node in a different layer, it has to take the nearest elevator located at router D, which is south-east to node A. Therefore, its compass bits are set to $(C_{east}, C_{west}, C_{north}, C_{south}) = (1, 0, 0, 1)$. Similarly, routers B and C store the following values to also point to router D: $(C_{east}, C_{west}, C_{north}, C_{south}) = (0, 0, 0, 1)$ and $(C_{east}, C_{west}, C_{north}, C_{south}) = (1, 0, 0, 0)$.

TABLE I. A COMPARISON BETWEEN ROUT3D AND OTHER ALGORITHMS

Routing algorithm	VCS / router	Elevator position	Storage / router	Pillar	Adaptive
Elevator-first [9]	10	Any	2 IDs	No	No
ETW [12]	8	East-most column	3 IDs	Yes	Yes
LBDR-3D [11]	10	Any	22bits	No	Yes
Proposed	8	Any	4bits	Yes	Yes

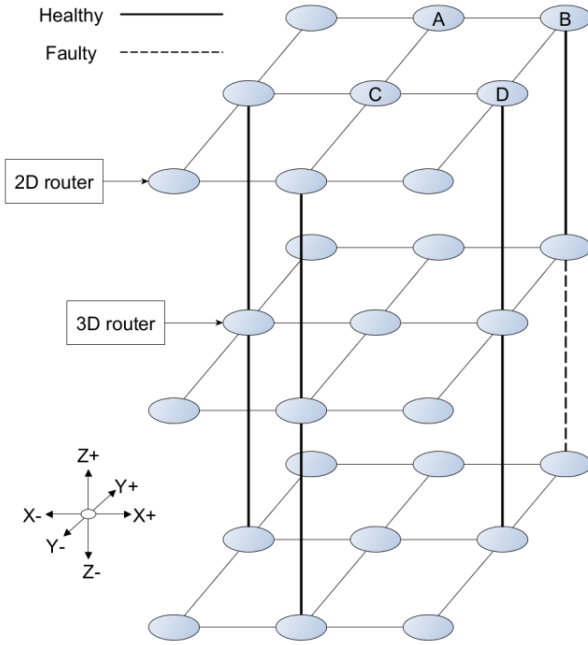


Fig. 1. Overview on the NoC architecture

This is analogous to the *Vertical Bits* used by LBDR3D [11]. However, the algorithm used to set these bits in LBDR3D does not guarantee that following vertical bits starting from an initial node leads to the intended Elevator, as different hops along the way may have a different view of where the closest elevator is, which is why additional signals had to be introduced to prevent livelocks [11]. We introduce an algorithm for setting the *Elevator Compass* bits that does not suffer from this limitation. The algorithm is presented in Algorithm 1. Instead of iterating over the nodes and finding the nearest elevator for each node independently as was done in [11], we iterate through each healthy elevator in turn and check if it is the nearest elevator to every node in the network. Even if the distance from some node to the new elevator is the same as its previously assigned nearest elevator, the new elevator is still preferred as the nearest elevator node. This ensures that starting from any initial node A, whose compass is pointing to node E, following the compass in the direction of E reaches a node B that points to the same nearest elevator E. If B had a nearest elevator node F different from E, then according to Algorithm 1, F would also be the nearest elevator to A. Therefore, our algorithm inherently guarantees that packets always reach their nearest elevator. Algorithm 1 is executed offline every time a new TSV fault is detected.

C. Constructing the Rout3D algorithm

The general approach we take to construct our routing algorithm is similar to that taken in several previous works [19], [12], which consists in identifying a set of virtual networks that ensure connectivity and deadlock-freedom. Our proposal starts by defining 3 virtual networks as shown in fig. 2. The first and third virtual networks (V0, V2) only use the X+ and Y+ physical channels. The second virtual network (V1) includes all the remaining directions (X-, Y-, Z+, Z-).

Algorithm 1: Setting Elevator Compass bits

```

00: Function set_compass {
01:
02: Variables:
03:   Dist[NumNodes] : Distance to closest elevator
04: Output:
05:   Compass[NumNodes] : Compass bits
06:
07: For each node i do
08:   Initialize Dist[i] to infinity
09:   Initialize Compass[i] to all zeros
10: End for
11:
12: For each healthy elevator (xE, yE) do
13:   For each node i of coord (x, y, z) do
14:     If |xE-x| + |yE-y| <= Dist[i] then
15:       Dist[i] = |xE-x| + |yE-y|
16:       Compass[i].North = (yE > y)
17:       Compass[i].South = (yE < y)
18:       Compass[i].West = (xE < x)
19:       Compass[i].East = (xE > x)
20:     End if
21:   End for
22: End for
23: }
```

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 at 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 V1, do not necessitate additional virtual channels.

Packets may traverse virtual networks only in increasing order (V0 → V1 → V2). Moreover, in order to enhance the utilization of virtual channels, packets of the highest virtual network (V2) are allowed to request **both** virtual channels in the X+ and Y+ directions (X0+, X1+, Y0+, Y1+) as long as they wait for both virtual channels simultaneously, whereas packets of the lowest virtual network (V0) can only request one of the two virtual channels (namely X0+ and Y0+).

The routing algorithm can be simply described as follows: If a packet has reached (or has originated in) the destination layer, route it towards its destination using the negative directions first (V1 then V2). Otherwise, route the packet towards the chosen elevator using the positive directions first (V0 then V1) and use V1 to elevate the packet to the destination layer. When a packet is headed East-North or West-South, routing can be performed adaptively and the least congested route is selected.

Two examples are shown in fig. 3 to illustrate this routing process. In fig. 3 (a), the source “S” and the destination “D” are located on the same layer so the packet is delivered adaptively using the Negative-First algorithm [5]. In the second example the source and destination are located at different layers so the closest elevator is taken first following the positive directions first before routing towards the destination in Negative-first order.

D. Deadlock-freedom, livelock-freedom and connectivity

It is easy to prove that cycles cannot form within each virtual network as none of the defined virtual networks spans two full dimensions, which is a requirement for completing a cycle. However, because some virtual channels are shared among two virtual networks, we cannot rely on the absence of cyclic

dependencies to prove freedom from deadlocks. Instead, we elaborate our proof based on two facts. (1) First, packets in the highest virtual network (V2) are always able to request virtual channels $X1+$ and $Y1+$, which are dedicated to V2 and can never be acquired by packets of other virtual networks. This implies that packets of V2 cannot be blocked indefinitely because these virtual channels can either be free or occupied by other packets of V2, which, as was previously mentioned, cannot form cycles between them due to the absence of negative directions. Second, our algorithm ensures that virtual networks are traversed in increasing order. This means that packets of V0 may be waiting for other packets of V0, in which case no cycles can form, or for packets in V1, which in turn can only wait for other packets in V1 or packets in V2. From (1) we know that packets in V2 always have an escape VC available. Therefore, deadlocks can never occur. It is worth noting that the absence of cycles in the channel dependency graph has been proven unnecessary for deadlock-freedom and overly restrictive in many previous studies [13], [14]. Rout3D has no such limitation and allows the presence of cycles in order to enhance the utilization of VCs and increase network throughput.

Because the individual virtual networks do not allow packets to loop indefinitely, and because they are traversed in increasing order, the algorithm is also livelock-free. In the worst case a packet reaches the north-east corner in V0, then the west-south corner of either the top or the bottom layer in V1 then end up in the north-east corner of that same layer in V2.

One thing to notice is that routing within each layer with Rout3D is equivalent to routing using the minimal Negative/Positive-first turn model [5], which is connected. This means that routing from any source node to any elevator, and from any elevator to any destination node is always possible. Therefore, as long as one elevator exists, regardless of its position, packet delivery is guaranteed.

E. Rout3D+: Enhancing Rout3D with additional VCs

As a way of improving our algorithm in terms of performance, we propose another variant, called **Rout3D+**, which includes one additional virtual channel per vertical port compared to Rout3D. These VCs are used without restrictions by all V1 packets for the sole purpose of boosting performance. Assigning these extra VCs to the Z dimension has two major benefits: First, these additional VCs are not present in 2D routers. This means that Rout3D+ still uses less virtual channels in 2D routers than Elevator-First, but now uses the same number of VCs in 3D routers, leading to a lower total cost in terms of VC FIFOs. The second reason for selecting the Z dimension is because there is likely to be a high contention on the elevators as few vertical connections may be shared between numerous flows. Therefore, adding virtual channels can reduce the pressure on the vertical links.

F. Complexity analysis

To have an approximate idea about the expected cost of Rout3D, we compare its implementation requirements to those of the Elevator-First implementation that uses XY routing [17]. Because Rout3D uses a very simple routing logic, we expect the cost of the routing function, which is purely combinational, to be only slightly more complex than the XY logic used by

Elevator-First. However, while the use of XY routing removes the need to connect all input-output pairs in the switch allocator and the crossbar (turns from Y to X are not required), Rout3D allows turns from any dimension to the other, which means that more connections need to be added if adaptiveness must be achieved. In addition, Elevator-First does not require input VC arbiters because every packet can only request one same VC during its entire transit. Conversely, because Rout3D and Rout3D+ make it possible for some packets to request several VCs (section III-C and section III-D), arbiters are necessary to select among free output VCs. Finally, because our algorithm is adaptive, extra hardware may be necessary for congestion management if using the locally available information does not offer the desired level of performance [24]. However, this choice is up to the application designer and is not intrinsic to our routing algorithm. Despite these additions, because the router's area is largely dominated by the number of virtual channel FIFOs [17], we expect both Rout3D and Rout3D+ to be less costly than Elevator-First overall. The elevator selection policy (amount of storage + logic) also contributes to the router's area requirements and using the Elevator Compass method proposed in this section guarantees that the cost does not increase proportionally with the network size. However, similarly to congestion management, this method is not specific to Rout3D and can be used in conjunction with the Elevator-first algorithm as well. Also, in cases where the Nearest Elevator is not the optimal criterion of selection, any other selection method can also be used in conjunction with the Rout3D algorithm.

IV. EXPERIMENTAL RESULTS

In this section, we compare on the same platform the performance of Rout3D and Rout3D+ to Elevator-first [9]. Although the original Elevator-first is associated with deterministic routing (XY), the approach can be generalized to support other 2D deadlock-free algorithms as was done in [11]. We compare our algorithm with two variants of Elevator-first, the original one using XY ('Elevator-First-XY'), and an adaptive version that uses the Negative-first algorithm ('Elevator-First-NF'). We should point out that because we want to test topologies with random fault scenarios, we will not include algorithms that have special requirements about the location of TSVs in this evaluation. Also, because LBDR3D is not a routing algorithm per se but rather a scalable and reconfigurable method for implementing Elevator-First algorithms in hardware, a comparison with Elevator-first is equivalent to a comparison with LBDR3D as long as we are only comparing performance at the system level.

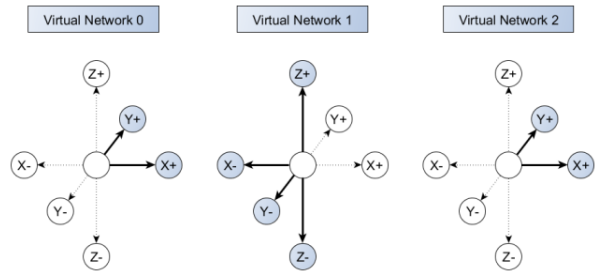


Fig. 2. Physical channels used by the three virtual networks

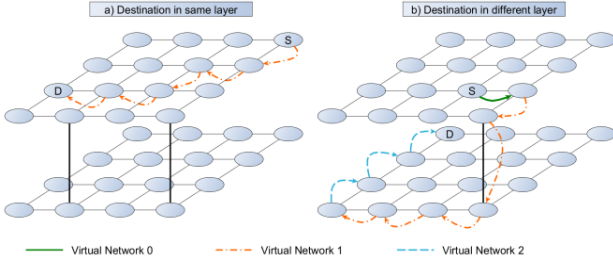


Fig. 3. An example demonstrating the operation of Rout3D

For simulations, we use an in-house GPU-based parallel cycle-accurate NoC simulator that we validated against an RTL NoC implementation [23]. We simulate a 256-node network consisting of 4 stacked 8x8 meshes. For all four algorithms, we use 4-flit deep FIFOs, and packets have a fixed size of 5 flits. Each algorithm uses the minimum number of VCs it requires to operate. All algorithms use the Manhattan Distance –based elevator selection as described in section III-B. Adaptive algorithms select the least congested output port, based on a local congestion metric, when several candidates are available.

At the beginning of each simulation, a specific number of TSV pillars are assigned randomly. We perform two series of tests: one with 48 pillars (75% vertical connections) and another with only 16 pillars (25% vertical connections). For each configuration, 100 iterations lasting 100000 cycles each are executed and the average results are presented.

For performance evaluation, 4 types of synthetic traffic modes are considered:

Uniform Random: Each source sends its packets to a random destination following a uniform distribution.

Hotspot: One node in the network is designated as a hotspot node and 10% of the packets generated by each source is destined to this hotspot node. The rest of the packets are sent to a random destination. In our case, we chose router $(X, Y, Z) = (7, 7, 2)$ to be the hotspot node. This choice is justified later.

Bit complement: Node (X, Y, Z) sends all of its packets to node $(\bar{X}, \bar{Y}, \bar{Z})$ where \bar{X} is the two's complement of X .

Shuffle: If N is the number of nodes, a node of ID $id < N/2$ sends packets to node of ID $2 \times id$. A node of $id \geq N/2$ sends its packets to the node of ID $2 \times id + 1 \bmod N$.

As a performance metric we consider the average packet latency, which is the average time it takes a packet to reach its destination, including the time it spends in the source queue. Our results are presented in fig. 4.

We first examine the results when 75% of TSVs are available. Although Rout3D would be expected to be outperformed by Elevator-First due to its lower number of VCs, we can see that in some cases (hotspot and bit complement traffic profiles) it can yield better performance than Elevator-first. This is explained by the fact that Rout3D is more flexible in its usage of VCs. Because packets having reached their destination layer are able to request both VCs in the north and east directions, they experience less blocking as they move

forward if their destination is located in the north east. By contrast, in Elevator-first, packets may only request one VC during their entire journey. This explanation is confirmed by the large difference in latency under hotspot traffic, where we expressly chose the hotspot to be located in the north east corner to highlight this property. Another thing to take note of is that although Rout3D performs better than Elevator-First-NF under complement traffic, it is outperformed by Elevator-First-XY. This result is not surprising because both uniform and bit complement traffics are known to favor deterministic routing over adaptive routing.

One interesting but perhaps counterintuitive final observation, is that although Rout3D+ (Section III-E) expectedly outperforms Rout3D in uniform, hotspot and complement traffics, we can see cases where adding a VC in the Z dimension does not have the desired effect on performance. This is the case, for instance, in Shuffle traffic. We explain this based on two factors. One is the fact that the tested network consists of very large layers (64 nodes), and the second is the availability of a relatively large number of elevators. Because packets are able to quickly reach their destination layer, due to a high elevator availability, packets spend most of their time in the destination layer. In such scenarios, performance is not limited by the vertical bandwidth, but rather by the multiplexing taking place within each layer. Because adding a VC in the vertical ports means that packets in their destination layer are multiplexed not with one, but two packets coming from other layers, packets in their destination layer experience a larger delay. Of course, this is only valid for traffic scenarios where many nodes communicate with other nodes of the same layer, which is the case of Shuffle traffic. When the number of TSVs is reduced, we can see that, as predicted in Section III-E, the use of these vertical VCs yields a significant performance improvement.

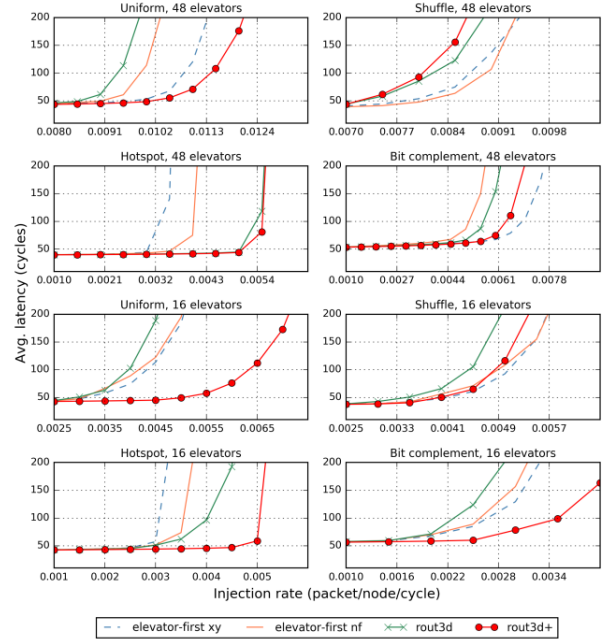


Fig. 4. Performance evaluation results

V. CONCLUSION

In this paper, we have presented a novel algorithm targeting partially vertically connected 3D-NoCs named "Rout3D" that guarantees packet delivery as long as one TSV pillar is available anywhere in the network. Although our algorithm uses the same number of virtual channels as other low-cost algorithms, it is in the way these virtual channels are exploited that lies all of its strength. What we have demonstrated in this paper is that by carefully placing these virtual channels and appropriately defining the routing rules, it was possible to eliminate all the restrictions that other solutions impose on both the supported fault patterns and the runtime elevator selection, which can negatively impact the resilience and performance of the NoC. Moreover, we have shown, through Rout3D+, that by adding one VC in the vertical dimension, it was possible to dramatically boost the NoC's performance in presence of a few vertical connections, while still ensuring a lower implementation cost than state-of-the-art algorithms.

In partially vertically connected NoCs, routers need a way to locate the available elevators in the network. In order to avoid storing complete node addresses within each router, we have proposed a low-overhead **scalable** technique named the "Elevator Compass", requiring only 4 bits of storage per router to point to the nearest elevator. These bits can be reconfigured every time the state of the network changes because of faults. In addition, this method is not specific to Rout3D and can be combined with any other routing algorithm.

Beyond the low implementation cost, it is also the elegance and simplicity of the proposed routing solution that make it a highly attractive option to implement into future many-core chips that need to support resiliency against defects with minimum effort.

REFERENCES

- [1] Dally, W.J.; Towles, B., "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference, 2001. Proceedings*, vol., no., pp.684-689, 2001.
- [2] A. Bakhoda, J. Kim and T. M. Aamodt, "Throughput-Effective On-Chip Networks for Manycore Accelerators," *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, Atlanta, GA, 2010, pp. 421-432.
- [3] B. S. Feero and P. P. Pande, "Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation," in *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32-45, Jan. 2009.
- [4] D. Wentzlaff *et al.*, "On-Chip Interconnection Architecture of the Tile Processor," in *IEEE Micro*, vol. 27, no. 5, pp. 15-31, Sept.-Oct. 2007.
- [5] C. Glass, L. Ni, "The turn model for adaptive routing", in: *Proceedings of the 19th Annual International Symposium on Computer architecture (ISCA '92)*, New York, NY, USA, 1992, pp. 278-287.
- [6] V. F. Pavlidis and E. G. Friedman, "3-D Topologies for Networks-on-Chip," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1081-1090, Oct. 2007.
- [7] W. R. Davis *et al.*, "Demystifying 3D ICs: the pros and cons of going vertical," in *IEEE Design & Test of Computers*, vol. 22, no. 6, pp. 498-510, Nov.-Dec. 2005.
- [8] R. Salamat, M. Ebrahimi and N. Bagherzadeh, "An Adaptive, Low Restrictive and Fault Resilient Routing Algorithm for 3D Network-on-Chip," *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Turku, 2015, pp. 392-395.
- [9] F. Dubois, A. Sheibanyrad, F. Pétrot and M. Bahmani, "Elevator-First: A Deadlock-Free Distributed Routing Algorithm for Vertically Partially Connected 3D-NoCs," in *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 609-615, March 2013.
- [10] R. Salamat, M. Ebrahimi, N. Bagherzadeh and F. Verbeek, "CoBRA: Low cost compensation of TSV failures in 3D-NoC," *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Storrs, CT, 2016, pp. 115-120.
- [11] 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," *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, Nara, 2016, pp. 1-8.
- [12] R. Salamat, M. Khayambashi, M. Ebrahimi and N. Bagherzadeh, "A Resilient Routing Algorithm with Formal Reliability Analysis for Partially Connected 3D-NoCs," in *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3265-3279, Nov. 1 2016.
- [13] Verbeek, F.; Schmaltz, J., "On Necessary and Sufficient Conditions for Deadlock-Free Routing in Wormhole Networks," in *Parallel and Distributed Systems, IEEE Transactions on*, vol.22, no.12, pp.2022-2032, Dec. 2011.
- [14] Pinkston, T.M.; Wamakulasuriya, S., "Characterization of deadlocks in k-ary n-cube networks," in *Parallel and Distributed Systems, IEEE Transactions on*, vol.10, no.9, pp.904-921, Sep 1999.
- [15] 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," *2014 International Conference on High Performance Computing & Simulation (HPCS)*, Bologna, 2014, pp. 516-522.
- [16] J. Lee and K. Choi, "A deadlock-free routing algorithm requiring no virtual channel on 3D-NoCs with partial vertical connections," *2013 Seventh IEEE/ACM International Symposium on Networks-on-Chip (NoCS)*, Tempe, AZ, 2013, pp. 1-2.
- [17] M. Bahmani, A. Sheibanyrad, F. Pétrot, F. Dubois and P. Durante, "A 3D-NoC Router Implementation Exploiting Vertically-Partially-Connected Topologies," *2012 IEEE Computer Society Annual Symposium on VLSI*, Amherst, MA, 2012, pp. 9-14.
- [18] A. Eghbal, P. M. Yaghini, N. Bagherzadeh and M. Khayambashi, "Analytical Fault Tolerance Assessment and Metrics for TSV-Based 3D Network-on-Chip," in *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3591-3604, Dec. 1 2015.
- [19] M. Ebrahimi, M. Daneshdalan, P. Liljeberg and H. Tenhunen, "Fault-tolerant method with distributed monitoring and management technique for 3D stacked meshes," *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADSD 2013)*, Tehran, 2013, pp. 93-98.
- [20] S. Pasricha and Y. Zou, "A low overhead fault tolerant routing scheme for 3D Networks-on-Chip," *2011 12th International Symposium on Quality Electronic Design*, Santa Clara, CA, 2011, pp. 1-8.
- [21] M. Ebrahimi, X. Chang, M. Daneshdalan, J. Plosila, P. Liljeberg and H. Tenhunen, "DyXYZ: Fully Adaptive Routing Algorithm for 3D NoCs," *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, Belfast, 2013, pp. 499-503.
- [22] S. Akbari, A. Shafiee, M. Fathy and R. Berangi, "AFRA: A low cost high performance reliable routing for 3D mesh NoCs," *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, 2012, pp. 332-337.
- [23] A. Charif, A. Coelho, N-E. Zergainoh, M. Nicolaidis, "Detailed and Highly Parallelizable Cycle-Accurate Network-on-Chip Simulation on GPGPU," *The 22nd IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*, Chiba, 2017.
- [24] P. Gratz B. Grot, S.W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," *IEEE 14th International Symposium on High Performance Computer Architecture*, 2008. *HPCA 2008.*, vol., no., pp.203,214, 16-20 Feb. 2008.