Novel flexible buffering architectures for 3D-NoCs

Mostafa Said, Amin Sarihi, Ahmad Patooghy, Awny M. El-Mohandes, Abdel-Hameed A. Badawy, & Fayez Gebali

March 2021

Sustainable Computing: Informatics and Systems 29 (2021) 100472

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

Novel flexible buffering architectures for 3D-NoCs

Mostafa Said [a,b,*], Amin Sarihi [a], Ahmad Patooghy [c], Awny M. El-Mohandes [e], Abdel-Hameed A. Badawy [a], Fayez Gebali [d]

[a] New Mexico State University, USA
[b] California State University at Bakersfield, USA
[c] University of Central Arkansas, USA
[d] University of Victoria, Canada
[e] McMaster University, Canada

ARTICLE INFO

Keywords:
3D-NoCs
Router
Buffer allocation
Network delay
Packet routing

ABSTRACT

In the conventional router architecture of Network-on-Chips (NoCs), each input port employs a set of dedicated flit buffers to store incoming flits. This mechanism unevenly distributes flits among router buffers, which in turn leads to higher packet blocking rates and under-utilization of buffers. In this paper, we address this problem by proposing two novel buffering mechanisms and their corresponding architectures to share flit buffers among several ports of a router efficiently. Our first proposed mechanism is called *Minimum-First* buffering. This mechanism distributes flits among buffers of input ports based on the number of free buffer slots available in each port, giving priority to minimum occupied buffers. This approach increases the utilization of underutilized buffers by allowing them to store flits of other input ports. The second mechanism (so-called *Inverse-Priority* buffering) is a lighter yet efficient, flexible buffering technique. This mechanism employs a simple priority order for each buffer. According to our analysis, prioritizing specific ports over others balances the traffic loads between router buffers, and thus yields higher throughput. Both mechanisms lead to lower waiting times in the router and higher utilization in hardware resources. After studying all possible scenarios and analyzing corner cases, we have optimally designed two router architectures equipped with the proposed buffering mechanisms. Moreover, a hardware optimization technique is introduced to reduce the area overhead of the Minimum-First router architecture. The proposed architectures show significant improvements in the performance of 3D-NoCs in terms of the average network throughput and average delay as well as the total number of blocked packets compared to different state-of-the-art and baseline router architectures.

1. Introduction

Network-on-Chip (NoC) has been widely adopted as a scalable communication architecture for multi- and many-core systems. However, with recent massive 1000-cores chips [1,2], the average distance between NoC cores increases substantially counteracting the benefits of technology shrinking. To tackle this problem, 3D-NoCs were recently introduced as a result of the emerging 3D integration technologies. 3D stacked silicon layers in 3D-ICs allow designers to place frequently communicating cores on top of each other in different layers. This shortens the length of interconnects and considerably reduces the hop-count. Higher connectivity of 3D-NoCs contributes to higher performance in comparison with 2D-NoCs when hosting the same number of cores [3]. Buffers in 2D-NoC consume a large area of the router layout [4] and dissipate a massive portion of energy [5]. The situation gets more severe in 3D-NoC routers as they need two additional ports to connect with the upper and the lower layer routers to forward inter-layer packets. This urges a call for optimal designs and efficient utilization of buffers in these types of networks.

Per the conventional buffering policy, when a flit requests for an input port, the port's controller stores the flit in a pre-associated FIFO buffer. For example, flits requesting the West port are stored in the West port buffer (W_B). When the corresponding buffer is full, a requesting flit will be blocked until a buffer slot of the queue is released. This policy disregards the state of other buffers, which leads to undesirable waiting times. As shown in Fig. 1a, the conventional buffering router (CBR) blocks the requesting packet at the West port because W_B is full, although other buffers have free slots. This leads to uneven utilization of

* Corresponding author at: California State University at Bakersfield, USA.
E-mail address: mossaied2@gmail.com (M. Said).

https://doi.org/10.1016/j.suscom.2020.100472
Received 21 June 2020; Received in revised form 3 October 2020; Accepted 18 October 2020
Available online 18 November 2020

routers' buffers and increases the network delay by introducing congested regions [6]. Many researchers have proposed solutions to avoid this problem in 2D-NoCs [4,7–9]. All these mechanisms have side effects and drawbacks that make them either power-hungry designs with a significant area overhead or inefficient in specific types of applications or traffic patterns.

To solve the blocking problem, we introduce two novel buffering mechanisms, namely Minimum-First Flexible Buffering Router (MFFBR) and Inverse-Priority Flexible Buffering Router (IPFBR). MFFBR mechanism chooses the minimum occupied buffer to store the incoming flits. IPFBR, on the other hand, prioritize the choice of candidate buffers in a certain order, i.e., Z-port buffers (U_B, D_B), Y-port buffers (N_B, S_B), then finally X-port buffers (E_B, W_B). This order ensures a balanced load among buffers and leads to lower blocking time/rates. The contributions of this paper can be summarized as follows:

- Introducing two novel router architectures for 3D-NoCs, namely MFFBR and IPFBR.
- Detailed analysis of the flexibility efficiency in different network dimensions leading to a lightweight version of MFFBR, yet maintaining its high performance.
- Deep deadlock analysis for the new routers proposing a deadlock prevention strategy for them.
- A modified task-to-core mapping optimization strategy to boost the performance of the proposed architectures.
- Addressing the hardware requirements of the proposed routers and introducing an optimized sorting circuit for MFFBR.

This paper is organized as follows. Related papers are reviewed in Section 2. Section 3 discusses the deadlock in 2D versus 3D NoCs by extending Turn-Model analysis. In Section 4, the proposed MFFBR and IPFBR buffering techniques are presented and their advantages and challenges are discussed in details. A wide range of evaluations is done and presented in Section 5. Finally, Section 6 concludes the paper.

2. Related work

Buffer management in NoC routers highly impacts network performance metrics including the waiting time of flits in routers, average network delay, and average network throughput. In [7], the so-called ViChaR router architecture is proposed to enhance efficiency among virtual channels (VCs) related to the same port. When a flit requests an input port, any available flit buffer among all VCs of the port is allowed to store the flit. This work suffers from (i) a blocking situation that may happen when all VCs of a port are full, (ii) not being able to redirect newcomer flits to available VCs of other ports, and (iii) incurring a very high power consumption overheard [10] so that the method might not be preferred in thermally-limited applications [11,12]. Marculescu et al. [4,8] allocate the size of buffer queues at the design time based on the characteristics of the target application. Despite the substantial budget reduction in the buffering area, this technique is application-dependent and cannot be used for a general-purpose design or applications with uncertainty regarding their load volumes.

In [9], the Round-Robin Flexible Buffering Router (RRFBR) handles the incoming flits in a dynamic way leading to performance improvement in the network. In any blocking situation, RRFBR picks among the available buffers in a Round-Robin fashion to store the requesting flits. As we will later show in Section 4, the Round-Robin mechanism does not ensure efficiency in all cases. Applying the flexibility only to the blocked packets is another source of inefficiency let alone its hardware cost. The latter is due to the required all-to-all accessibility between the ports and the Round-Robin criterion in selecting the candidate buffer.

Driven by the inefficiency and the low performance of the CBR and RRFBR, we propose two novel buffering mechanisms; MFFBR and IPFBR, which address the blocking problem of CBR more efficiently than RRFBR.

3. Deadlock analysis and prevention

Applying buffering flexibility with no restrictions makes the network susceptible to deadlock even if the operating routing algorithm is deadlock-free, e.g., XY routing [9]. Based on the study of [9], the Turn-Model for the XY plane in Fig. 2a, inherits some buffering restrictions for each buffer to prevent deadlock. Forbidden turns $\{2, 4, 5, 7\}$ cause some restrictions to the router buffers, e.g., turn 2 bans packets moving South to turn West. This implies that the North Buffer[1] (N_B) cannot store packets heading West. So each buffer has to have a set of restrictions in terms of its allowed/forbidden next hop directions. The set of allowed and forbidden packet directions for each buffer in a 2D-NoC is summarized in Table 1, which is inferred from the Turn-Model of Fig. 2a [9].

The set of restrictions introduced in Table 1 prevents the deadlock in 2D-NoCs; although, these rules are not enough to avoid the deadlock in 3D-NoCs. Because 3D-NoCs have two more directions along the Z dimension, namely Up and Down, along with their flit buffers (U_B and D_B). However, simply extending the same restrictions (as of shown in Table 1) to U_B and D_B buffers will raise deadlock situations in 3D-NoCs. Fig. 3 shows some deadlock examples in XZ and YZ planes. To extend Table 1 to support 3D-NoCs, the Turn-Model itself should be extended. This is achieved by making analogies between the new created planes, i.e., XZ and YZ planes, and the original XY plane. As shown in Fig. 2b and c, Z dimension in XZ plane is considered analogous to Y, and both Y and Z in YZ plane are analogous to X and Y, respectively. This is inherited

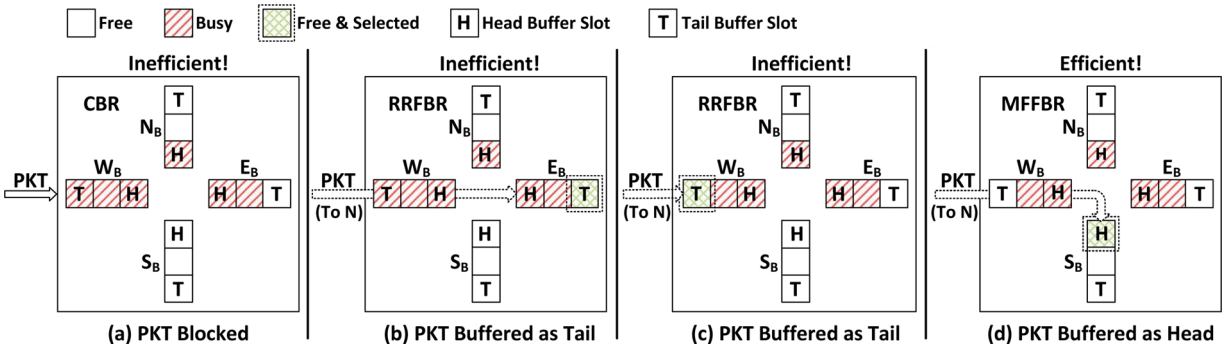
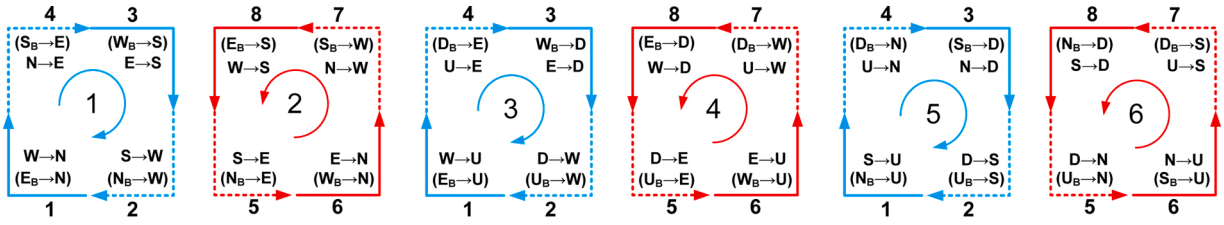


Fig. 1. An example showing operational differences between CBR, RRFBR, and MFFBR routers. The main difference lies in the criteria of choosing a buffer to store an incoming flit. Note: (To N) means that the next hop of PKT is North.

[1] Packets moving South stored all the time in N_B under XY routing in CBR.



(a) XY plane Turn-Model. (b) XZ plane Turn-Model. (c) YZ plane Turn-Model.

Fig. 2. Extended Turn-Model to 3D shows possible turns of XYZ routing in different planes. For example, in (b), cycles $(3, 4)$ occur between head flits of E_B, W_B, U_B, and D_B. The notation S → E, for example, means a flit is moving South then East, while S_B → E means a flit inside the South buffer is heading East.

Table 1
Buffering restrictions in 2D-NoCs [9].

Buffer	Allowed next hops	Forbidden next hops
East (E_B)	N, S, W, L	E
West (W_B)	N, S, E, L	W
North (N_B)	S, L	N, E, W
South (S_B)	N, L	S, E, W

from the nature of XYZ routing algorithm where X dimension has a priority over Y and Z, and further, Y dimension has a priority over Z. Our extended buffering restrictions are summarized in Table 2, where they ensure deadlock-freedom in 3D-NoCs.

A remarkable observation in Table 2 is that buffers have different degrees of flexibility. As shown, X-port buffers are the most flexible. For example, E_B is allowed to store packets heading North, South, West, Up, Down, and Local. Nevertheless, E_B is not allowed to store packets heading East. On the other hand, Z-port buffers like U_B only has permission to store packets heading Down or Local. Other output directions could cause deadlocks, e.g., North. Such property can be used to achieve load fairness between buffers by IPFBR as shown later in Section 4.2.

4. The proposed buffering architectures

In this section, we introduce our proposed buffering mechanisms and their router designs. We start with MFFBR buffering mechanism and its architecture. We also present a lightweight version of the Minimum-First buffering. Finally, the mechanism of IPFBR and its load fairness are introduced and analyzed in detail.

4.1. Minimum-First Flexible Buffering Router (MFFBR)

4.1.1. Idea and buffering efficiency

RRFBR of [9] does not guarantee buffering efficiency in all situations. Fig. 1b and c explain cases of RRFBR limitations. The first case is shown in Fig. 1b. To serve the incoming packet request, RRFBR checks the buffer of the associated request (W_B), which in this case is full. Based on that, RRFBR selects another free slot based on the next hop direction of the incoming packet (North) and the availability of other buffers. Referring to Table 2, RRFBR excludes N_B even if it has free slots to avoid deadlock. The available set of buffers now reduces to S_B and E_B. Assuming that it is the turn of the E_B (RRFBR uses Round-Robin selection mechanism), the packet is stored in E_B.

Another inefficient case is shown in Fig. 1c. In this case, there is a free slot in W_B. RRFBR stores the requesting packet in W_B regardless of its queuing occupancy versus other buffers. In both cases of Fig. 1b and c the request is not blocked, however, the choice of either E_B (Fig. 1b) or W_B (Fig. 1c) is not efficient. This is because the packet is stored as a tail and accordingly, it will experience a long queuing time. This inefficiency occurs because the Round-Robin mechanism of RRFBR is oblivious of the buffer occupancy or because of prioritizing the associated port buffer over other less occupied buffers.

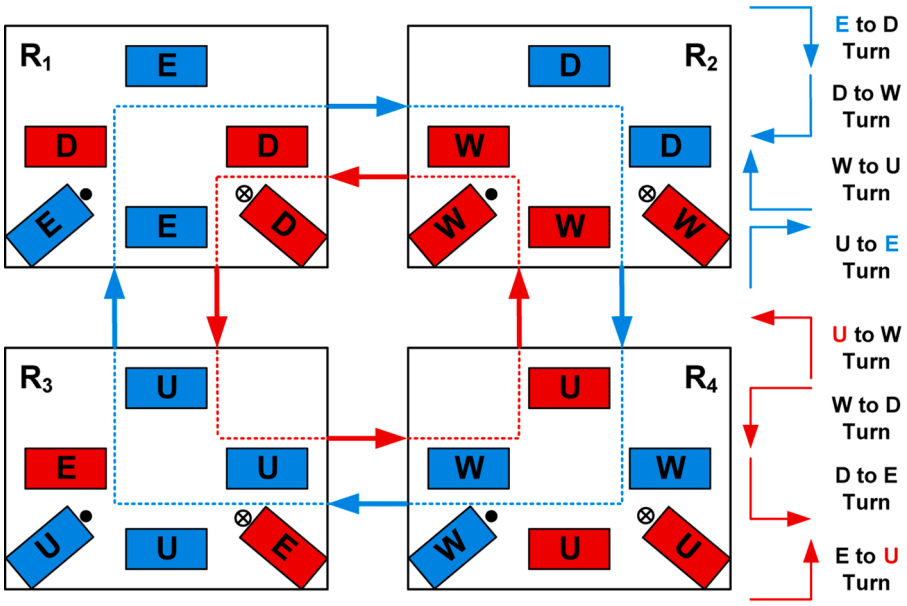
On the other hand, Minimum-First Flexible Buffering Router (MFFBR) avoids all these inefficiency cases using a smart buffering criterion. MFFBR gives higher priority to less occupied buffers, i.e., buffers with a higher number of free slots have a higher priority to be selected. This way, the queuing time of flits in routers is shortened, which in turn improves the overall performance. In Fig. 1d, the buffer occupancies at the time of the request are $2/3$ in W_B, $0/3$ in S_B, $1/3$ in N_B, and $2/3$ in E_B where 3 is assumed to be the maximum buffer size. MFFBR first excludes N_B from the list to avoid deadlock and selects the buffer with minimum occupancy from the rest. MFFBR selects S_B because it is the minimum occupied buffer. Redirecting the requesting packet to S_B makes it the head of the queue and decreases its waiting time before it leaves the router. This simple Minimum-First buffering solution has a notable impact on improving the network delay (see Section 5 for details) and helps MFFBR to outperform both CBR and RRFBR routers.

4.1.2. Candidate buffer selection in MFFBR

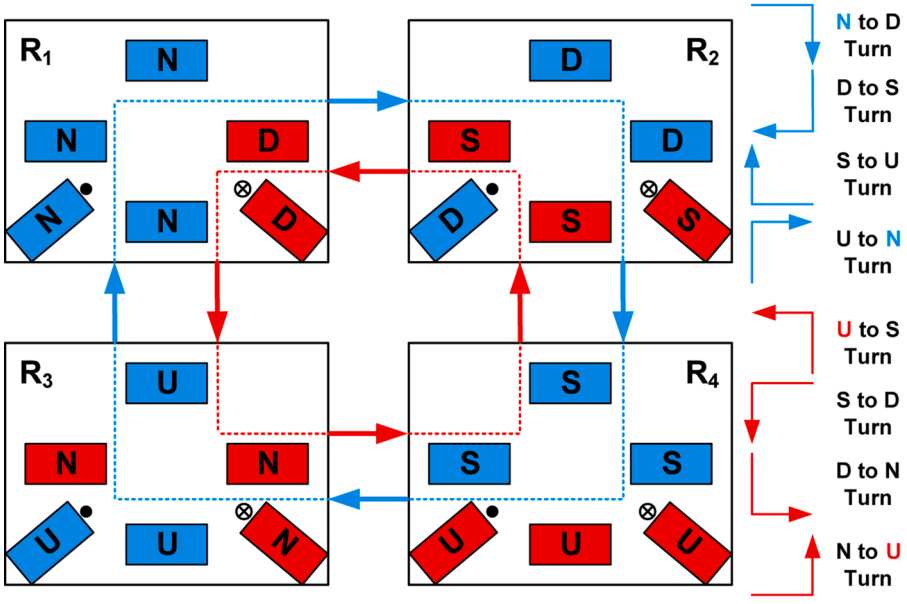
When multiple buffers are found to have minimum occupancy, it is crucial to find a criterion to choose between the allowed options. Fig. 4a shows two packets, PKT1 and PKT2, requesting for buffer storage at East and West ports, respectively, at the same clock cycle. Let us assume at this clock cycle: (1) all buffers have a maximum size of one, (2) West port will be served first, and (3) PKT2 and PKT1 are heading Down and West, respectively, in their next hop.

When MFFBR serves PKT2, the main criterion is to choose a buffer with the minimum occupancy. Since PKT2 is heading Down in its next hop, there are multiple buffers (E_B, W_B, N_B, S_B, and U_B) that can store it. Assuming that MFFBR chooses any minimum occupied buffer arbitrarily, its selection might be E_B that makes E_B full (Fig. 4b). When it comes to serving PKT1, which is heading West in its next hop, the only allowed buffer to store packets heading West is E_B, which has already been reserved by PKT2. Therefore, PKT1 will be blocked. It is worth mentioning that no matter which port is served first, this problem can still occur.

To address this problem, we design MFFBR such that it prioritizes some buffers over others when it comes to choosing among multiple available buffers with minimal occupancy. A careful look at Table 2 reveals two extreme cases based on the packet direction. While packets moving in Z direction, i.e., Up or Down, can be stored in five buffers. Packets flowing in X direction can be stored only in either W_B or E_B. Therefore, we design the selection criterion of MFFBR to choose among available buffers in the following order: Z-port buffers (U_B & D_B), Y-port buffers (N_B & S_B), and finally, X-port buffers (E_B & W_B). The underlying reason is to make buffering in E_B and W_B less probable than other buffers. By applying this criterion to the example in Fig. 4a, MFFBR chooses U_B for PKT2, which leaves E_B free for PKT1. This prevents PKT1 from being blocked (Fig. 4c). Since buffers are ordered in Z, Y, and X, we call this criterion ZYX buffering.



(a) Deadlock cycles involving four routers in XZ plane.



(b) Deadlock cycles involving four routers in YZ plane.

Fig. 3. Deadlock cases in (a) XZ and (b) YZ planes.

Table 2
Extended buffering restrictions in 3D-NoCs.

Buffer	Allowed next hops	Forbidden next hops
East (E_B)	N, S, W, U, D, L	E
West (W_B)	N, S, E, U, D, L	W
North (N_B)	S, U, D, L	N, E, W
South (S_B)	N, U, D, L	S, E, W
Up (U_B)	D, L	N, S, E, W, U
Down (D_B)	U, L	N, S, E, W, D

4.1.3. Hardware optimization for MFFBR

In this section, we propose a method to reduce the area overhead of MFFBR architecture without affecting the performance gain. The general idea is to design the hardware to prioritize minimum occupied buffers and, at the same time, keep the area overhead minimal. As shown in Table 2, there are two extreme cases for storing incoming flits under XYZ routing: (1) packets traversing Z direction are allowed to be buffered in five out of the router's six buffers, (2) on the other extreme, packets moving in the X direction are only allowed to be buffered in one out of six buffers. However, if an X-direction packet is opting to take Y or Z turn at a router, it would be eligible to stay at either three out of six (turn to Y) or five out of six (turn to Z) buffers.

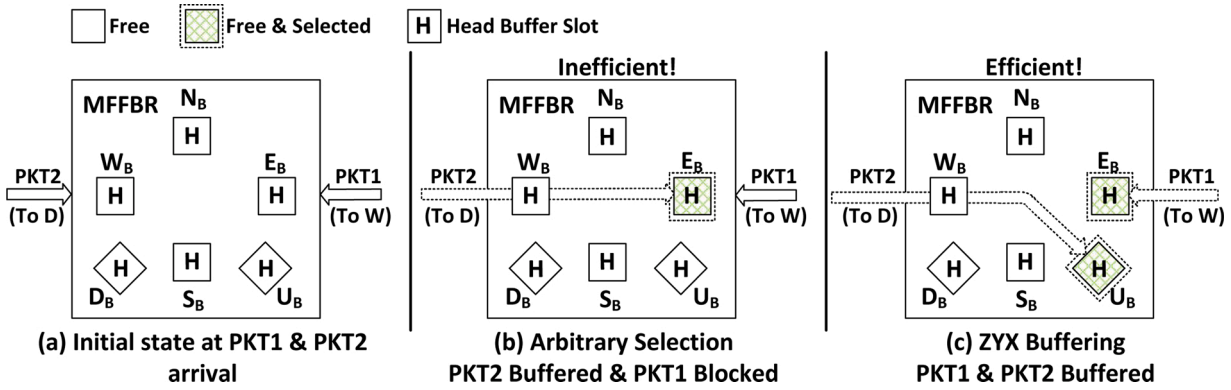


Fig. 4. A case showing how the ZYX buffering mechanism chooses the candidate buffer without affecting traffic of other ports.

Based on these observations, we propose a lighter version of MFFBR; YZ-Port MFFBR (MFFBR-YZ). In the original MFFBR, we add flexibility controllers to all ports of the router. In MFFBR-YZ, we do not include any flexibility control modules in X ports because the chance of applying flexibility to packets moving in this dimension is minimal. One crucial fact to note is that the amount of reduction in the achieved performance gain is traffic dependent and, as shown later in Section 5, some traffic patterns lose a minor performance gain compared to the amount of overhead saved.

4.2. Inverse-Priority Flexible Buffering Router (IPFBR)

4.2.1. Idea and congestion mitigation

Inverse-Priority Flexible Buffering Router (IPFBR) is the second flexible router we propose in this paper. IPFBR achieves load balancing between buffers, and thus, higher throughput at a minimal amount of hardware cost. The buffering mechanism is based on a fixed order of search which can be adapted based on the underlying routing algorithm. Upon receiving a packet request under XYZ routing, IPFBR searches for a free slot in the following order of buffers: Z-port buffers, Y-port buffers, and finally X-port buffers which is similar to ZYX buffering in Section 4.1.2 yet the reason here is different. Based on that, we set the search order as follows: U_B, D_B, N_B, S_B, E_B, then finally W_B.

Fig. 5 demonstrate the efficiency of IPFBR over CBR. As shown in Fig. 5a, in the initial situation PKT1, PKT2, and PKT3 are heading to R1 from R0. In the case of CBR (Fig. 5b), PKT1 moves to W_B of R1 after one clock cycle. The next hop direction of PKT1 is North toward R2 which currently has a busy South buffer. Therefore PKT1 will stay in R1 for another cycle. When PKT2 arrives, CBR stores it in the second slot of W_B behind the blocked packet PKT1. We will assume the network has congestion for at least 4 cycles ahead (blockings might even stay longer than this when congestion starts). In the same manner, PKT3 will be stored in the third slot of W_B after 3 cycles. After 4 clock cycles and since the South buffer of R2 is busy blocking PKT1 from advancing, PKT2 and PKT3 are blocked as well despite the next hops for both PKT2 and PKT3 have free buffer slots; W_B of R4 and N_B of R3, respectively.

IPFBR mechanism completely bypasses the consequences of the congestion situation of R2 (Fig. 5c). When PKT1 arrives, IPFBR first determines the direction of it next hop. In this example, PKT1 is heading North in the next hop. Since N_B cannot accept packets heading North, it is stored in S_B. Similarly, after 2 cycles in R2, PKT1 will advance to E_B of R2, bypassing the blocking in its S_B. During the same cycle, PKT2 is stored in the same way in N_B in R1. After 3 cycles, PKT2 advances to N_B of R3 and PKT3 is buffered in W_B of R1. Finally, after 4 clock cycles, PKT3 moves to E_B of R4. Hence, while the packets in CBR were blocked in R1 after 4 cycles, IPFBR flexibly forwarded the packets to their next hops bypassing the blocking situation. Note that IPFBR does not search for the minimum occupied buffer like MFFBR, instead, a fixed order of search is applied each time, reducing its hardware requirements compared to MFFBR.

4.2.2. Effect of search order on load balance

The suggested search order for IPFBR is not arbitrary. This search order is based on Table 2. As shown in Table 2, X-port buffers are the most flexible, while Z-port buffers are the least. We set the search in the *inverse* direction of buffer flexibility, i.e., starting from the least flexible; U_B and D_B to the most flexible; E_B and W_B. Although the search priority is set as inverse of buffer flexibility, this order guarantees a high degree of buffering fairness while enhancing the router performance. To justify that, we propose a hypothetical buffering architecture, which we call Forward-Priority Flexible Buffering Router (FPFBR). It has the exact opposite search order of IPFBR (E_B, W_B, N_B, S_B, U_B, then D_B); or

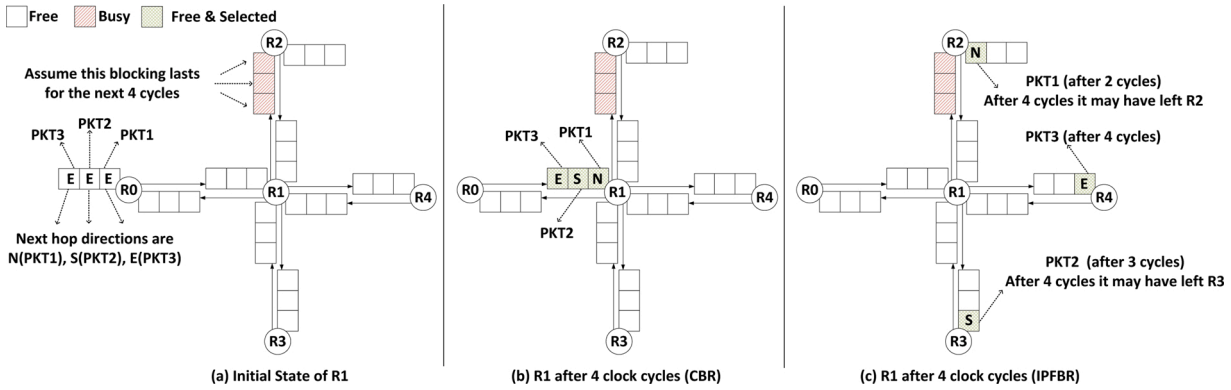


Fig. 5. A case showing how IPFBR bypasses the regional blocking to boost network performance against CBR. The initial situation of the routers is shown in (a). After 4 cycles, all packets are still blocked in CBR router (b), while they all passed R1 in the case of the IPFBR router (c).

downwardly from the highest to lowest flexible buffers.

In Fig. 6, PKT1 in R4, PKT2 in R3, and PKT3 in R2 are all heading to R1. Assuming East port request is served first (PKT1), FPFBR checks next-hop direction, which is West in this case. FPFBR searches for a free buffer slot in the following order: E_B, W_B, N_B, S_B (this example is a 2D-NoC for simplicity). In that example, E_B has three free slots so PKT1 is stored in E_B and can accept packets heading West. Then FPFBR will serve South port packet (PKT2) which is heading North. Since FPFBR has a fixed order and E_B still has two free slots and can accept packets heading North, PKT2 is stored in E_B. Finally, the same situation is applicable to PKT3. Since E_B is checked first and it still has a free slot can accept packets heading South, PKT3 is also stored in the tail slot of E_B. It is clear, that searching for a free slot starting from the most flexible buffers, e.g., E_B, will increase the probability of unbalanced loading.

IPFBR guarantees a considerable amount of fairness by just flipping the search order. In the same example and assuming same order of serving requests, for PKT1, IPFBR will check for a free buffer slot in the following order N_B, S_B, E_B, and W_B. Since PKT1 next hop direction is West, neither N_B nor S_B can store it to avoid deadlock. Therefore, PKT1 is stored in E_B. Then for PKT2 and searching in the same order, since PKT2 is heading toward North, N_B cannot store it, but S_B can, thus IPFBR stores PKT2 in S_B. Finally, for PKT3, same search order is followed, and since PKT3 is heading south and N_B can store packets heading South, E_B is selected to buffer PKT3. As shown in Fig. 6 the search order that IPFBR follows, guarantees load balancing between router buffers. If the next hop buffers of PKT1, PKT2, and PKT3 are all available to receive new packets, all the packets will leave R1 in one cycle. On the other hand, FPFBR needs at least three clock cycles because all the packets are queued in the same buffer.

4.2.3. *Effect of buffer depth on load balance*

Buffer depth can affect the load balancing advantage of IPFBR and its buffering efficiency. Fig. 7 shows a simple example of the problem. In Fig. 7a when the buffer depth is only one slot, PKT1 and PKT2 are buffered in separate buffers; i.e., N_B and S_B, respectively. This would give a chance for both packets to exit the router in the next cycle in case there is no blocking ahead. On the other hand, when the buffer depth is two as in Fig. 7b, the situation is less efficient. Since N_B can buffer packets heading South or Local, both PKT1 and PKT2 are queued in N_B. In that case, at least two cycles are required for both packets to continue their journey out of the router. It is worth mentioning that, the buffer depth effect on IPFBR is limited. Based on our results at different buffer depths, we noticed that effect only at buffer depth of 16, as will be shown later in Section 5.

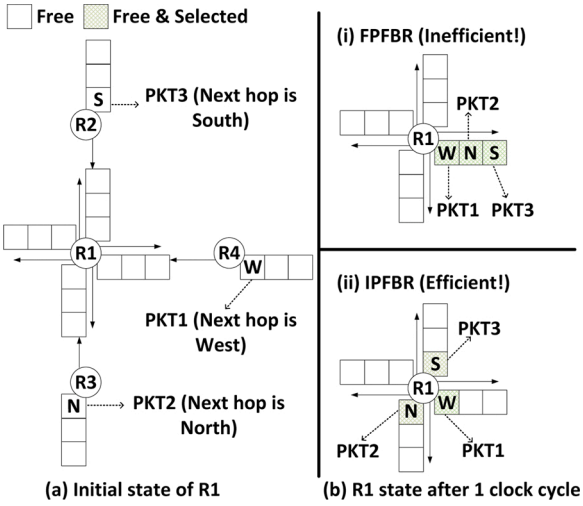


Fig. 6. An example showing how the Inverse-Priority buffering criteria of IPFBR achieves load fairness.

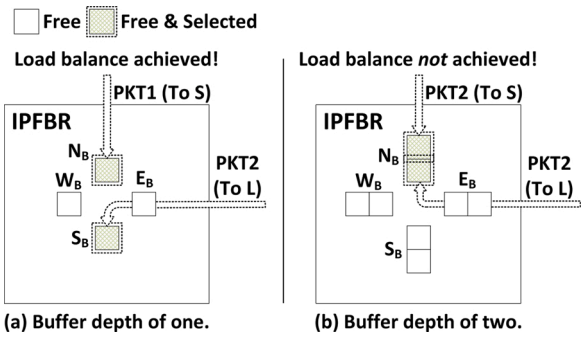


Fig. 7. The effect of buffer depth on IPFBR load fairness. With shallower buffers (a), IPFBR achieves better buffering fairness.

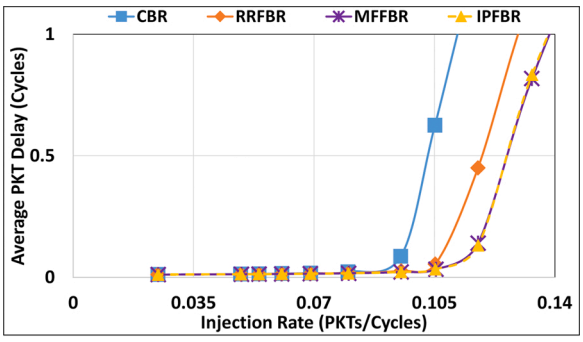
5. Simulation platform and results

In this section, we introduce various scenarios to evaluate the achievements of our proposed architectures in terms of network performance as well as hardware cost of 3D-NoCs. We evaluate MFFBR and IPFBR architectures under four different traffic scenarios. The first scenario is a synthetic traffic in which all of the generated traffic is directed in one dimension, i.e., exclusively either X, Y, or Z (Single-Dimension Traffic). This experiment is conducted to investigate which port yields the highest performance gain in the proposed flexible routers. The results of these experiments are used to support our analysis regarding hardware optimized architecture MFFBR-YZ (Section 4.1.3). The second scenario is a more realistic traffic to model a general-purpose many-core processor with evenly distributed L2 cache slices between 3D-NoC tiles. In the third scenario, the Transpose-I (Tr-I) traffic is adopted which is very similar to the matrix-transpose real applications [13]. Finally, in the last scenario, we adopt a real-world dVOPD video benchmark traffic in our simulations.

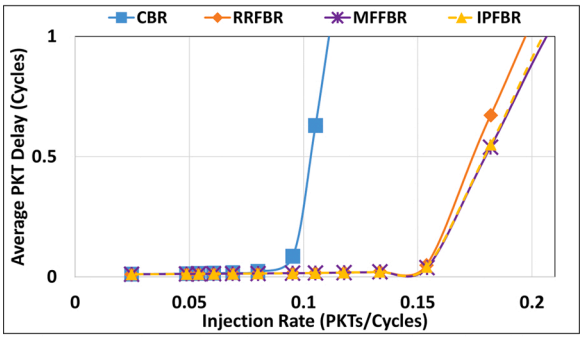
In the first three scenarios, an $8 \times 8 \times 8$ 3D mesh is used in which every core injects a total of 1000 packets into the network. In the dVOPD case, the mesh size is $3 \times 3 \times 4$ to match the size of dVOPD task graph, i.e., 32 tasks. Simulations of Single-Dimension traffic scenario are done with the buffer depth of one. For inter-tile and dVOPD traffic scenarios, the buffer depth is set to four and eight, respectively. However, in the case of Tr-I we adopt three sizes; 4, 8, and 16, to study the effect of buffer depth on the efficiency of IPFBR as presented earlier in Section 4.2.3. Finally, in inter-tile, Tr-I and dVOPD traffic scenarios, we show the performance of MFFBR in two cases: all-Port flexibility in which flexibility is applied at all ports of the router, and YZ-Port flexibility in which we apply flexibility only in Y and Z ports to stress the efficiency of MFFBR-YZ architecture.

5.1. *Single-Dimension Traffic*

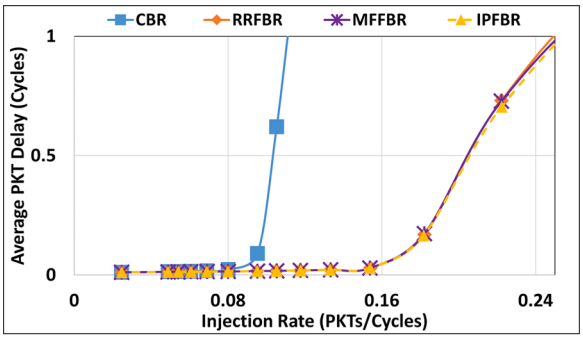
Fig. 8a–c shows the average packet delay under All-X, All-Y, and All-Z traffic patterns, respectively, in which packets source and destination are in the same dimension. In the three cases, MFBRR and IPFBR show the best performance with the least packet delay. Though the blocking delay of CBR is almost constant, it is continuously decreasing for MFBRR, IPFBR, and RRFBR cases. We can observe that the maximum blocking delays of MFBRR, IPFBR, and RRFBR occur under All-X traffic, while their minimum blocking delays are seen in All-Z traffic case. This behavior supports our observation presented in Section 4.1.3. In fact, under All-X traffic, packets are moving either toward East or West, and the allowed buffers are only either W_B or E_B, respectively. So the flexibility is only applied when packets reach their destinations, whereas packets heading to the Local port can be stored in any buffer. In the case of All-Y traffic, the delay is the middle ground between All-Z and All-X because packets heading North or South has potentially three out of six buffers available to be stored in.



(a) Average packet delay vs. packet injection rate (All-X).



(b) Average packet delay vs. packet injection rate (All-Y).



(c) Average packet delay vs. packet injection rate (All-Z).

Fig. 8. Average delay comparison under (a) All-X, (b) All-Y, and (c) All-Z traffic patterns. The All-Z indicates the best performance due to the greater flexibility in buffering incoming packets.

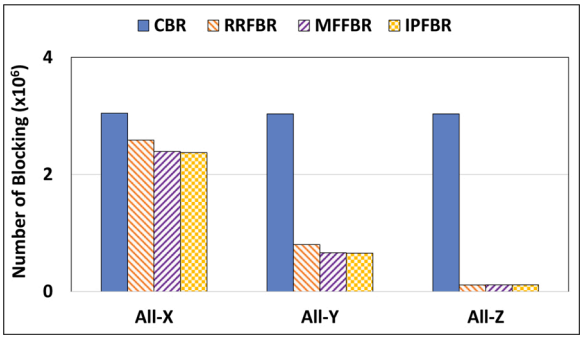
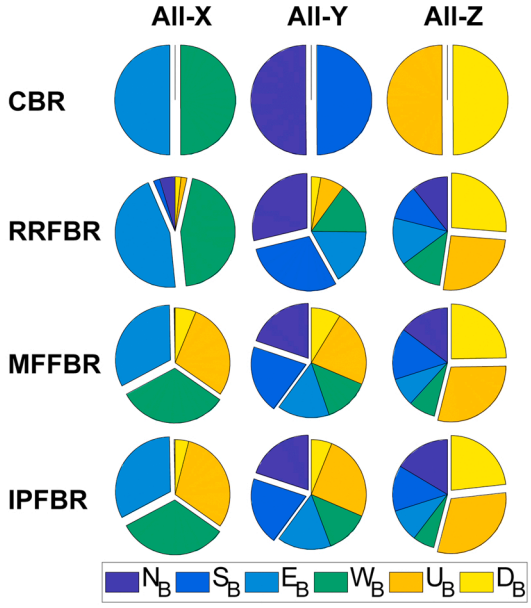


Fig. 9. Total packet blocking counts at the highest packet injection rate possible under Single-Dimension Traffic scenarios.

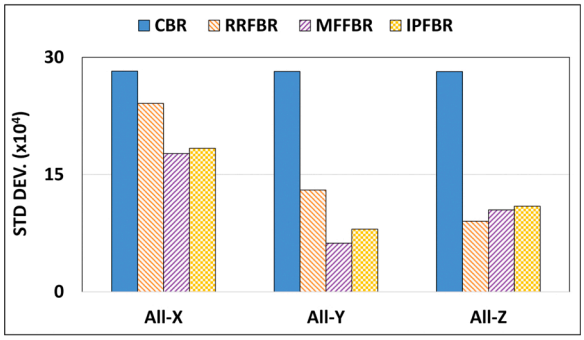
Fig. 9 highlights the advantage of MFFBR and IPFBR in reducing the total number of blockings. In our simulation, a blocking counter at each port is incremented once an incoming request fails to be granted due to the lack of available buffer slots. As the blocking count is directly proportional to the delay, CBR has the highest number of blockings while adopting MFFBR and IPFBR leads to the least number of blocked packets, which is minimized under the All-Z traffic pattern.

To further explore the achievement of MFFBR and IPFBR, we reveal their fairness capabilities in balancing the load between router buffers. Fig. 10a shows the percentage number of packets stored in each buffer in each router architecture at the highest injection rate possible. For example, under All-X traffic, CBR only uses E_B and W_B (≈50% of the packets for each one), and other buffers receive nothing. Despite utilizing the flexibility of RRFBR, E_B and W_B are still prioritized over other buffers. Other buffers are only used when there is a blocking. This causes significant shares of packets in E_B and W_B (≈45% stored in each one), while other buffers receive only ≈10%. Finally, the buffering criteria of MFFBR and IPFBR ensures a higher degree of fairness, i.e., packets are divided more evenly between buffers. For example, 35% of the packets are directed to Y and Z port buffers.

In All-Z case, while only U_B and D_B of CBR are used, the flexible buffering architectures utilize other buffers more efficiently than the case of All-X and All-Y traffic patterns. Respectively, 47.7%, 46.2%, and



(a) Packet share stored in each buffer.



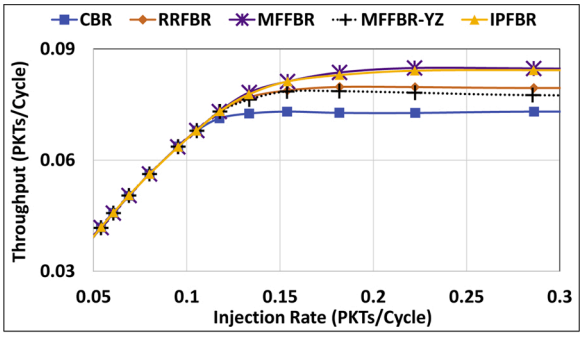
(b) Standard deviation of buffer loads.

Fig. 10. The pie-chart (a) shows the share of packets stored in each buffer in different Single-Dimension traffic scenarios. Standard deviation figures between these shares are shown in (b). As shown, MFFBR has the minimum STD-DEV (best buffering fairness).

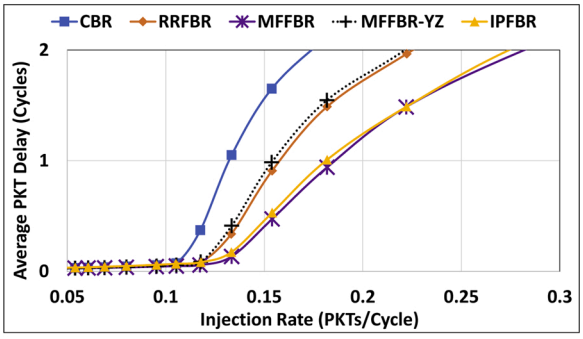
46% of the received packets are stored in N_B, S_B, E_B, and W_B in the case of RRFBR, MFFBR, and IPFBR. Fig. 10b demonstrates the balancing fact of MFFBR and IPFBR in terms of the standard deviation (STD-DEV) between the loads of each buffer. STD-DEV in the cases of MFFBR and IPFBR is approximately equal to and less than both CBR and RRFBR (more evenly distribution of packets between buffers) in All-X and All-Y traffic patterns respectively. Under All-Z traffic, the STD-DEV values of all flexible architectures are comparable. Though, they are also far bellow CBR with respective figures of 3.71%, 3.81%, and 3.79%, for RRFBR, MFFBR, and IPFBR, respectively.

5.2. Uniform inter-tile traffic

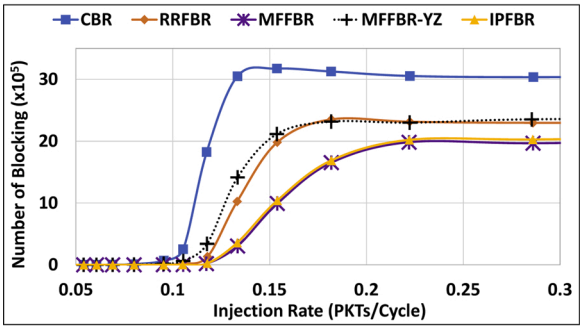
In this experiment, each processing tile is assumed to enclose a single processing core, a private L1 cache, and a slice of shared L2 cache. The whole tile is assumed to be connected to its local router. In the case of L1 cache misses, inter-tile communications are needed to fetch the required data from L2 slices, possibly in other tiles rather than the original requesting tile [14]. Distributing L2 slices evenly between tiles results in a uniform distribution of traffic over the network.



(a) Throughput comparison in PKTs/Cycles.



(b) Average packet delay comparison in Cycles.



(c) Total packet blockings comparison.

Fig. 11. Throughput (a), delay (b), and blocking count (c) versus packet injection rate under Uniform inter-tile traffic.

As shown in Fig. 11a and b, the network performance is always the best in terms of throughput and delay for MFFBR and IPFBR compared to that of RRFBR and CBR. Throughput speedup is 6.05% and 15.36% compared to RRFBR and CBR, respectively. The percentage delay reductions at the saturation injection rate (≈0.133 PKTs/Cycle) of both MFFBR and IPFBR, are 48.69% and 83.48% compared to RRFBR and CBR, respectively. Although the all-Port flexibility of MFFBR yields the highest performance gain, the performance gain of MFFBR-YZ is very close to RRFBR and still better than CBR. However, the hardware cost of MFFBR-YZ is minimal compared to RRFBR, as we will show later in Section 5.5. In terms of throughput, the speedup is 6.1% compared to CBR. MFFBR-YZ delay at the saturation injection rate is 60.79% less than CBR. The same conclusions can also be made by looking at the number of packets being blocked for the under-study architectures in Fig. 11c. A summary of the percentage blocking reduction is shown in Table 3.

To give more insights, we looked at the order of different types of packets in the buffers, i.e., to see how packets are distributed among the buffer slots of each port. This is a fundamental advantage of MFFBR and MFFBR-YZ. "1st" bars in Fig. 12 indicate the number of packets that are buffered as heads when they were first stored in the router. The more packets stored as head, the less they experience queuing time. As shown, the number of head packets in MFFBR and MFFBR-YZ is considerably higher than those of other routers. At the same time, MFFBR and MFFBR-YZ have the minimum number of packets stored as tails ("4th"). A summary of the percentage differences in Tail/Head packet counts over CBR is shown in Table 4. While MFFBR and MFFBR-YZ outperform CBR, both RRFBR and IPFBR are worse than CBR in that regard. Both have a smaller number of head packets and a higher number of tail packets.

5.3. Transpose-I Traffic

The average delay under Tr-I is shown in Fig. 13, where the buffer depth (BD) is assumed four in 13 a and 16 in Fig. 13b. As shown, the saturation injection rate of IPFBR worsens slightly at the larger buffer depth of 16. This is because it loses some of its balancing advantages at large buffer depths, as explained earlier in Section 4.2.3. At the same time, as concluded from Fig. 13a and b, MFFBR is immune against that negative effect. This is because the Minimum-First criterion guarantees a lower number of blocked packet regardless of the buffer depth.

Fig. 13c shows the percentage reduction in delay versus CBR at three different buffer depths: 4, 8, and 16. As noticed from the trend lines, the percentage delay reduction deteriorates as we increase the buffer depth in the case of RRFBR and IPFBR. IPFBR percentage delay reduction changes as follows: (17.39% (BD = 4), 17.03% (BD = 8), and 16.29% (BD = 16)). The opposite trend can be seen for MFFBR (17.5% (BD = 4), 18.07% (BD = 8), and 18.44% (BD = 16)) confirming its superior Minimum-First policy.

5.4. dVOPD video benchmark traffic

The communication traffic between dVOPD benchmark tasks [15] is illustrated in the graph of Fig. 14. The problem of mapping each task in Fig. 14 to a core in the $3 \times 3 \times 4$ NoC is modeled in MiniZinc discrete optimization language [16,17]. The objective is to minimize

Table 3
Summary of the percentage reduction (Red.) in the number of blockings versus CBR under inter-tile traffic.

Router	% Blocking Red.
RRFBR	24.1%
MFFBR	35%
MFFBR-YZ	22.44%
IPFBR	33%

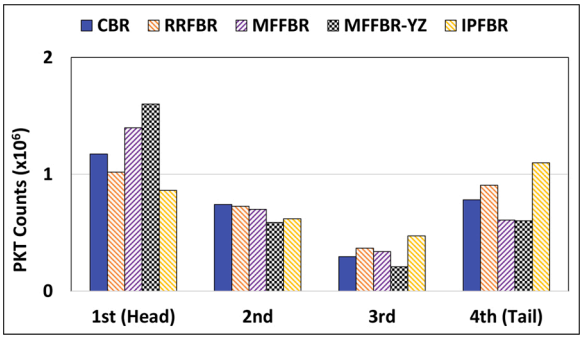


Fig. 12. Packet-order counts for router architectures (1st denotes Head, 2nd & 3rd denote Body, and 4th denotes Tail packets).

Table 4
Summary of increase/reduction (Inc./Red.) in Head/Tail (H/T) packet counts versus CBR.

Router	% H Inc.	% T Red.
RRFBR	− 13.19%[A]	− 15.85%[A]
MFFBR	19.10%	22.20%
MFFBR-YZ	36.46%	22.96%
IPFBR	− 26.49%[A]	− 40.41%[A]

[A] Negative sign means reduction/increase in Head/Tail packets.

communication cost, which is defined as the sum of the communication bandwidths between every two tasks (BW_{ij}) multiplied by the number of hops between the their corresponding cores. The hop count between any two cores is calculated as ($Z_{Hops-ij} + Y_{Hops-ij} + X_{Hops-ij}$), where Z_{Hops}, Y_{Hops}, and X_{Hops} are the total hop counts in Z, Y, and X dimensions, respectively.

Since hops along the Z and Y dimensions in the case of the flexible architectures (Fig. 8c and b) incur less delay (due to the higher flexibility in buffering the packets moving in the Z and Y directions), we add the constraints of $Z_{Hops} \geq Y_{Hops}$ and $Y_{Hops} \geq X_{Hops}$ into our mapping process. However, we do not apply such constraints into the mapping process for CBR as hops of all directions have the same delay properties. We define the objective function as:

$$min \quad \sum_i \sum_j BW_{ij} \cdot (Z_{Hops-ij} + Y_{Hops-ij} + X_{Hops-ij})$$

$$\text{s.t.} \quad Z_{Hops} \geq Y_{Hops}$$
$$Y_{Hops} \geq X_{Hops}$$

that obtains the better performance for the flexible architectures.

The delay plots in Fig. 15b show that the proposed architectures significantly outperform CBR as the saturation injection rate increases from 0.128 to 0.185 PKTs/Cycles (44.53% increase). Nevertheless, MFFBR-YZ shows less improvement in delay after reaching the saturation injection rate. It still significantly outperforms CBR. Also, MFFBR-YZ shows a negligible delay similar to other flexible routers between 0.128 and 0.185 injection rates, while it has less hardware overhead. As can be seen in Fig. 15a, the speedup of MFFBR, IPFBR, and RRFBR over CBR is 22.58% while the respective figure for MFFBR-YZ is 18.54%. It is worth mentioning that the comparable performance results of MFFBR and IPFBR versus RRFBR are in harmony with the conclusions drawn from Fig. 8c. With high traffic in Z dimension, these architectures show very close performance results.

5.5. Hardware design and evaluation

5.5.1. Overall router architectures

As shown in Fig. 16, the difference between CBR and flexible architectures lies in the input stage. In CBR architecture, each input controller has only accessibility to only one buffer that is associated with the port (Fig. 16a), whereas the Flexibility Controller in the flexible architectures can access all buffers to apply flexibility. However, the Flexibility Controller module design varies from a flexible architecture to another based on the specific buffering criterion used.

5.5.2. Flexibility Controller

The internal modules of the Flexibility Controller are shown in Fig. 17. The Buffer Allocator circuit takes an input packet and stores its flits in one of the six buffers. The six arbiters in Fig. 17, receive requests from buffer allocators, e.g., N-N, N-S, etc. Since more than one flit can be forwarded to the same FIFO at the same clock cycle, arbiters have been used to prioritize these requests. We chose a fixed priority arbiter scheme for its simplicity and low hardware cost [18].

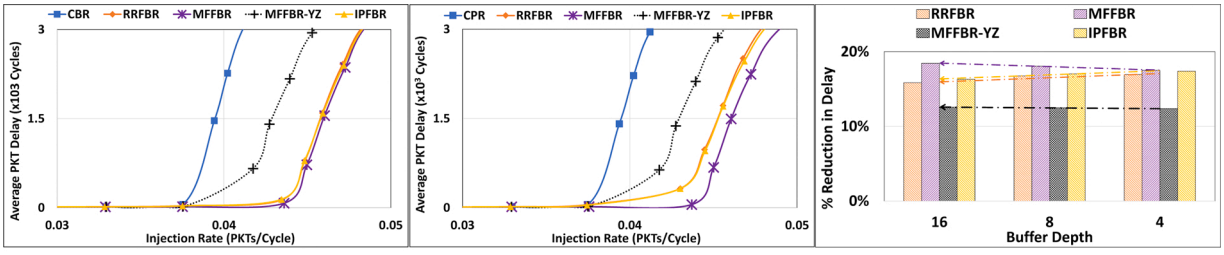
5.5.3. Buffer Allocator

The details of the Buffer Allocator unit are explained in Fig. 18. In the first step, the added circuitry computes the destination address in the Routing Logic module. For MFFBR, a sorter module is needed to sort the number of occupied buffer slots (buffer sizes in Fig. 18) from the lowest to the highest. The sorted numbers should be appended with their respective buffer labels since we need to know which number belongs to which buffer. The sorted figures are then passed to the Request Handler to place the packet in one of the six buffers. A dictionary has been used in this module to properly choose the right FIFO according to the packet destination which is originally based on Table 2.

The Buffer Allocator of IPFBR has no sorting circuit, instead, the request handler schedules the requests based on IPFBR's priority order. For RRFBR, the Request Handler uses Round-Robin to accomplish the job. Although, in the case of free buffer slots, the entered packet will be placed in its own buffer. More details of RRFBR implementation can be found in [9].

5.5.4. Sorting module optimization

As presented in Section 5.5.3, we need a sorting module to sort the number of occupied buffer slots from the lowest to the highest, to apply



(a) Average delay (Buffer Depth 4). (b) Average delay (Buffer Depth 16). (c) Delay % reduction vs. buffer depth.

Fig. 13. Average delay comparison under Tr-I traffic at buffer depths of 4 (a) and 16 (b). The effect of buffer depth on the delay reduction of the proposed routers versus CBR is in (c).

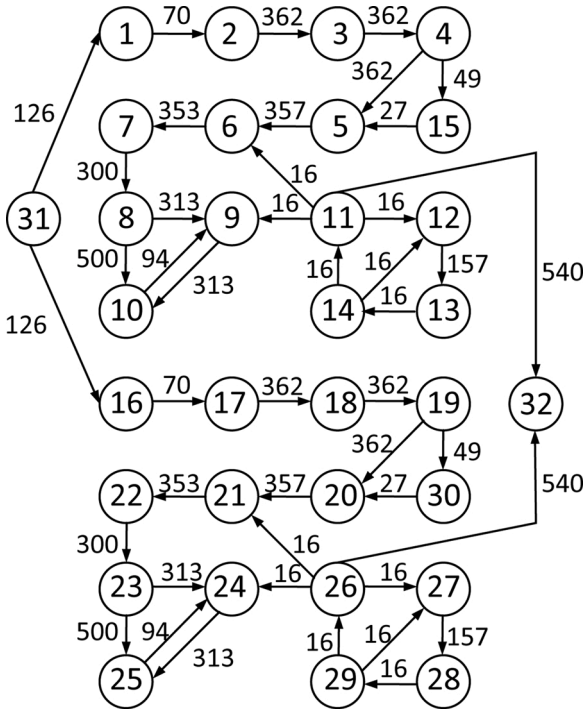
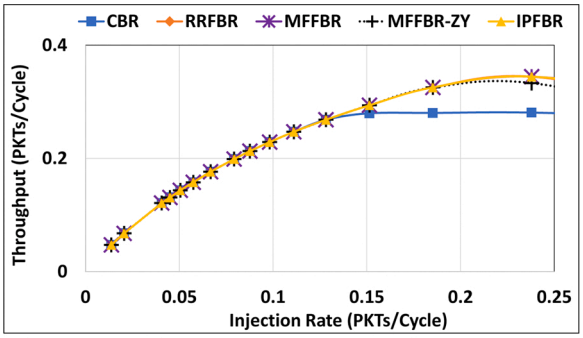
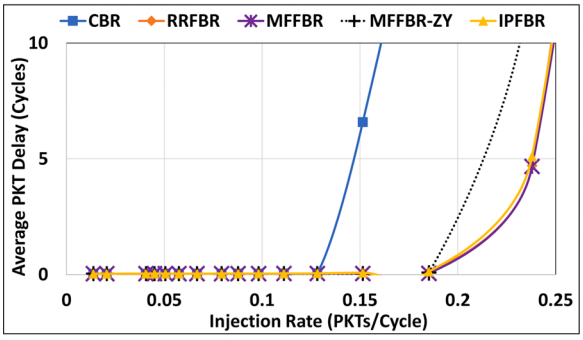


Fig. 14. dVOPD communication task graph with communication bandwidth stated in MB/s on each edge [15].



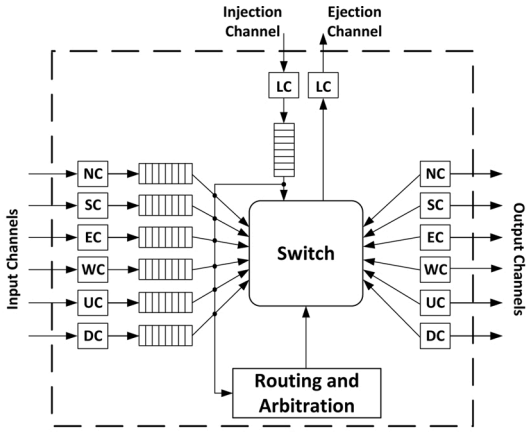
(a) Throughput comparison in PKTs/Cycles.



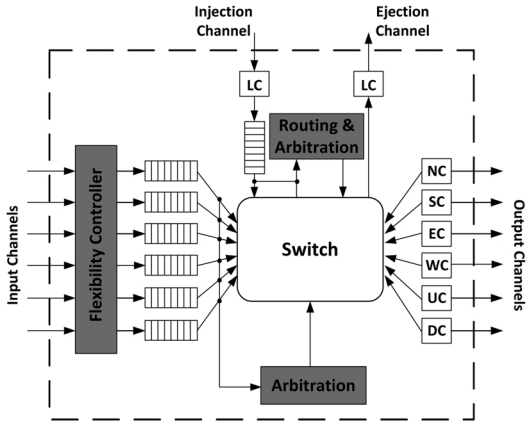
(b) Average packet delay comparison in Cycles.

Fig. 15. Throughput (a) and average delay (b) at different packet injection rates under dVOPD benchmark traffic pattern.

Minimum-First mechanism. Efficient sorting in hardware can be done using many available sorting networks [19–21]. Among them, Odd-Even sorting, and Bitonic sorting networks [19] were chosen to accomplish the task. The primary element in any sorting network is the



(a) CBR architecture.



(b) Conceptual architecture of all flexible buffering routers.

Fig. 16. Architectures of CBR (a) versus flexible buffering (b) that employs the Flexibility Controller to apply flexibility. The major differences between the two architectures are shaded in grey.

Compare&Swap (C&S) [20]. This element compares two input numbers and swaps them provided that they are not sorted in minimum-maximum order. This element can be built in hardware by a comparator and two multiplexers [21].

We customized Bitonic and Odd-Even Sorters in [19] to adapt to our situation where we only need six inputs. The modified networks are shown in Fig. 19. As can be noticed, Odd-Even simplified version is more optimal than Bitonic circuit with three fewer C&S elements, which in turn leads to less hardware resources needed. Hence, the Odd-Even sorter is selected for our designs.

5.5.5. *Routing of the buffered packet*

Forwarding packets to an output port in CBR requires feeding the head flit of the packet to the routing computation module. As shown in Fig. 18, for any input packet, the output port is computed before being buffered. To avoid doubling the routing logic, we add the output port information, which was computed in the Buffer Allocator, in the head flit of any packet. Since this only needs adding three bits (seven possible output ports), this is a negligible overhead compared to the total flit widths, e.g., 128 in [22] and 256 in [23]. Another factor that makes such overhead negligible, is the high number of flits per packets, e.g., eight flits [12], while those three bits are only added in head flit. However, the Local port still needs routing computation since the flexibility is not applied to it. To address the issue properly, we provide a dedicated routing and arbitration logic module for this port only. At the same time, other ports only need arbitration, as shown in Fig. 16b.

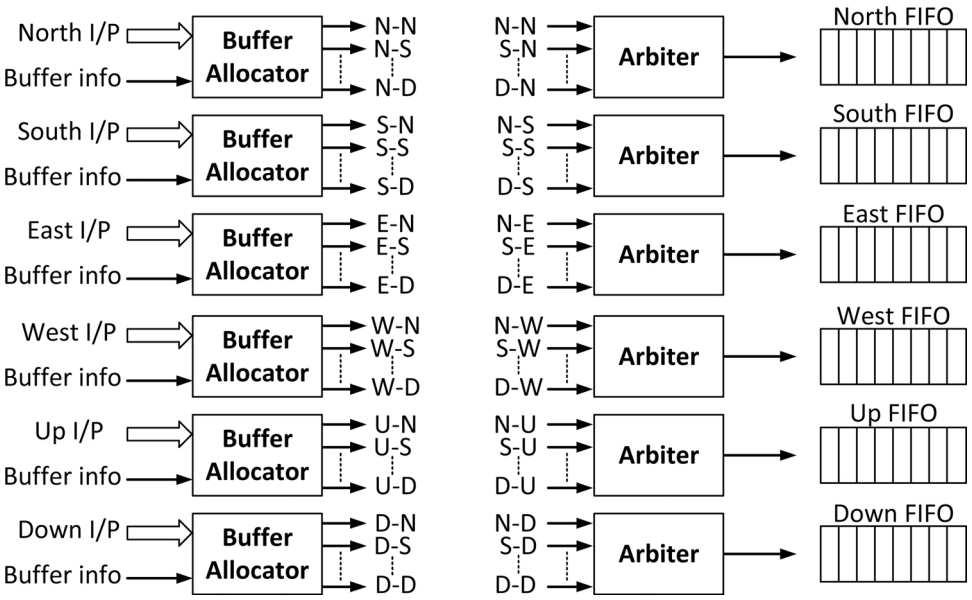


Fig. 17. The internal structure of the Flexibility Controller module.

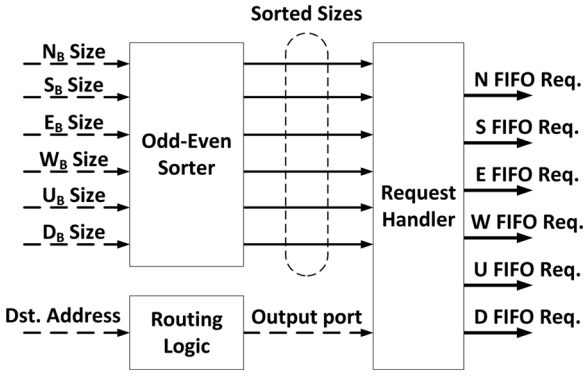


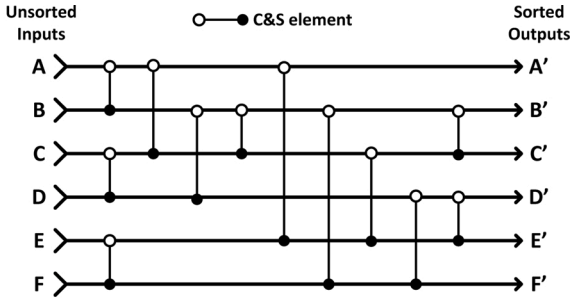
Fig. 18. Details of the buffer allocator in MFFBR. Removing the sorter module in IPFBR leads to less hardware complexity.

5.5.6. *Area cost evaluation*

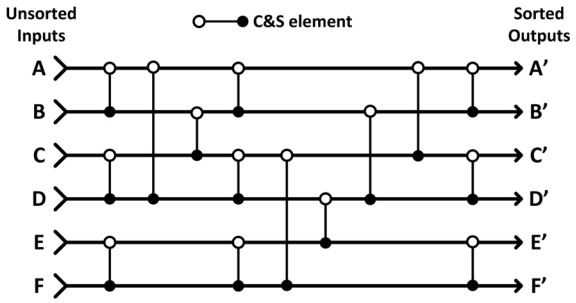
All the designs were implemented using VHDL and synthesized using Quartus Prime Pro. with the flit width of 256 bits and buffer depth of 8 similar to [23,24]. Area overhead of each buffering scheme over the baseline router (CBR) is given in Table 5. As shown, MFFBR-YZ has the lowest overhead (9.81%) since it does not need the X dimension flexibility modules. It is followed by IPFBR and RRFBR with 13.09% for both. However, IPFBR outperforms RRFBR in most of our experiments as shown earlier. MFFBR has the highest overhead (20.84%). The main contributor is the sorter circuit which occupies a significant area. At the same time, MFFBR has only ≈8% area cost over RRFBR, although it considerably outperforms RRFBR in most of our performance evaluations.

6. Conclusions

In this paper, two novel buffering mechanisms and the corresponding router architectures for 3D-NoCs were proposed and evaluated. The first mechanism prioritizes the minimum occupied buffers for storing new flits to minimize the queuing time of packets in the NoC routers. The second mechanism assures balancing the load between router buffers to increase the throughput. The detailed deadlock analysis was provided to guarantee the deadlock-freedom of the network. Based on this analysis, we concluded that not all ports of a 3D NoC have the same impacts on



(a) Odd-Even sorting needs 12 C&S elements.



(b) Bitonic sorting needs 15 C&S elements.

Fig. 19. Odd-Even (a) and Bitonic (b) six-inout modified sorting networks regarding. Odd-Even has three fewer C&S elements compared to Bitonic.

Table 5
Area comparison of the proposed flexible router architectures as compared to the baseline router.

Buffering scheme	% Area overhead
MFFBR	20.84%
MFFBR-YZ	9.81%
IPFBR	13.09%
RRFBR	13.09%

the overall load-balance of the network.

The proposed architectures were evaluated under synthetic traffic, such as cache-miss uniform traffic, as well as dVOPD video benchmark real-world traffic pattern. Gathered statistics justify the expected outcomes of the proposed buffering mechanisms and their architectures at both network and router levels. Finally, hardware optimization strategies were proposed to reduce the area overhead while maintaining the prescribed performance gains. The synthesized design shows a reasonable area cost compared to conventional buffering and other state-of-the-art routers.

Declaration of Competing Interest

The authors report no declarations of interest.

Appendix A. Abbreviations

N_B	North port buffer
S_B	South port buffer
E_B	East port buffer
W_B	West port buffer
U_B	up port buffer
D_B	down port buffer
L_B	local port buffer
RRFBR	Round-Robin flexible buffering router
MFFBR	Minimum-First flexible buffering router
IPFBR	Inverse-Priority flexible buffering router
FPFBR	Forward-Priority flexible buffering router
All-X	Single-Dimension Traffic in X dimension
All-Y	Single-Dimension Traffic in Y dimension
All-Z	Single-Dimension Traffic in Z dimension

References

[1] B. Bohnenstiehl, A. Stillmaker, J. Pimentel, T. Andreas, B. Liu, A. Tran, E. Adeagbo, B. Baas, KiloCore: a fine-grained 1,000-processor array for task-parallel applications, IEEE Micro 37 (2) (2017) 63–69.

[2] Z. Jia, B. Tillman, M. Maggioni, D.P. Scarpazza, Dissecting the Graphcore IPU Architecture via Microbenchmarking, 2019.

[3] B.S. Feero, P.P. Pande, Networks-on-chip in a three-dimensional environment: a performance evaluation, IEEE Trans. Comput. 58 (1) (2009) 32–45.

[4] J. Hu, U.Y. Ogras, R. Marculescu, System-level buffer allocation for application-specific networks-on-chip router design, IEEE Trans. CAD ICs Syst. 25 (12) (2006) 2919–2933.

[5] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, S. Borkar, A 5-GHz mesh interconnect for a teraflops processor, IEEE Micro 27 (5) (2007) 51–61.

[6] L. Mingche, G. Lei, S. Wei, W. Zhiying, Escaping from blocking: a dynamic virtual channel for pipelined routers, Int. Conf. on Complex, Intelligent and Software Intensive Syst. (2008) 795–800.

[7] C.A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M.S. Yousif, C.R. Das, Vichar: a dynamic virtual channel regulator for network-on-chip routers, MICRO'06 (2006) 333–346.

[8] T. Huang, U.Y. Ogras, R. Marculescu, Virtual channels planning for networks-on-chip, ISQED'07 (2007) 879–884.

[9] M.S. Sayed, A. Shalaby, M. El-Sayed, V. Goulart, Flexible router architecture for network-on-chip, Comps. Math. Appl. 64 (5) (2012) 1301–1310.

[10] S. Park, A Verilog-HDL Implementation of Virtual Channels in A Network-on-Chip Router, Texas A&M University, 2008 (Master's thesis).

[11] N. Dahir, R. Al-Dujaily, T. Mak, A. Yakovlev, Thermal optimization in network-on-chip-based 3D chip multiprocessors using dynamic programming networks, ACM TECS 13 (4s) (2014).

[12] K. Chen, Game-based thermal-delay-aware adaptive routing (GTDAR) for temperature-aware 3D network-on-chip systems, IEEE TPDS 29 (9) (2018) 2018–2032.

[13] A. Rahmani, A. Afzali-Kusha, M. Pedram, A novel synthetic traffic pattern for power/performance analysis of network-on-chips using negative exponential distribution, J. Low Power Electron. 5 (2009) 1–10.

[14] C. Kim, D. Burger, S.W. Keckler, An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches, ASPLOS'02 (2002) 211–222.

[15] P.K. Sahu, S. Chattopadhyay, A survey on application mapping strategies for network-on-chip design, J. Syst. Arch. 59 (1) (2013) 60–76.

[16] N. Nethercote, P.J. Stuckey, R. Becket, S. Brand, G.J. Duck, G. Tack, Minizinc: towards a standard cp modelling language, 13th Int. Conf. on Principles and Practice of Constraint Programming, ser. CP'07 (2007) 529–543.

[17] http://www.minizinc.org.

[18] W.J. Dally, B.P. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[19] R. Mueller, J. Teubner, G. Alonso, Sorting networks on FPGAs, VLDB J. 21 (1) (2012) 1–23.

[20] T.C. Chen, K.P. Eswaran, V.Y. Lum, C. Tung, Simplified odd-even sort using multiple shift-register loops, Int. J. Comput. Inform. Sci. 7 (3) (1978) 295–314.

[21] D. Koch, J. Torresen, FPGASort: a high performance sorting architecture exploiting run-time reconfiguration on FPGAs for large problem sorting, ACM/SIGDA FPGA'11, ser. FPGA'11 (2011) 45–54.

[22] T.C. Xu, P. Liljeberg, H. Tenhunen, A study of through silicon via impact to 3D network-on-chip design, Int. Conf. on Electronics and Inform. Eng. 1 (2010) pp. V1-333–V1-337.

[23] D.H. Woo, N.H. Seong, D.L. Lewis, H.S. Lee, An optimized 3D-stacked memory architecture by exploiting excessive, high-density TSV bandwidth, HPCA'10 (2010) 1–12.

[24] K. Chen, S. Lin, H. Hung, A.A. Wu, topology-aware adaptive routing for nonstationary irregular mesh in throttled 3D NoC systems, IEEE TPDS 24 (10) (2013) 2109–2120.