LEF: An Effective Routing Algorithm for Two-Dimensional Meshes

Thiem Van CHU†a), Nonmember and Kenji KISE†b), Member

SUMMARY We design a new oblivious routing algorithm for two-dimensional mesh-based Networks-on-Chip (NoCs) called LEF (Long Edge First) which offers high throughput with low design complexity. LEF’s basic idea comes from conventional wisdom in choosing the appropriate dimension-order routing (DOR) algorithm for supercomputers with asymmetric mesh or torus interconnects: routing longest dimensions first provides better performance than other strategies. In LEF, we combine the XY DOR and the YX DOR. When routing a packet, which DOR algorithm is chosen depends on the relative position between the source node and the destination node. Decisions of selecting the appropriate DOR algorithm are not fixed to the network shape but instead made on a per-packet basis. We also propose an efficient deadlock avoidance method for LEF in which the use of virtual channels is more flexible than in the conventional method. We evaluate LEF against O1TURN, another effective oblivious routing algorithm, and a minimal adaptive routing algorithm based on the odd-even turn model. The evaluation results show that LEF is particularly effective when the communication is within an asymmetric mesh. In a 16 × 8 NoC, LEF even outperforms the adaptive routing algorithm in some cases and delivers from around 4% up to around 64.5% higher throughput than O1TURN. Our results also show that the proposed deadlock avoidance method helps to improve LEF’s performance significantly and can be used to improve O1TURN’s performance. We also examine LEF in large-scale NoCs with thousands of nodes. Our results show that, as the NoC size increases, the performance of the routing algorithms becomes more strongly influenced by the resource allocation policy in the network and the effect is different for each algorithm. This is evident in that results of middle-scale NoCs with around 100 nodes cannot be applied directly to large-scale NoCs.

key words: Network-on-Chip, two-dimensional (2D) mesh, routing algorithm, Long Edge First (LEF)

1. Introduction

Many modern computing hardware platforms such as many-core processors, Multi-Processor Systems-on-Chip (MP-SoCs), and hardware accelerators are composed of a large number of processing cores that communicate and cooperate with each other when executing a task. As the core count increases, the inter-core communication becomes a critical factor that determines the overall performance. Unfortunately, traditional on-chip interconnects like buses and crossbars do not scale beyond a small number of cores. Therefore, a new class of on-chip interconnects called Networks-on-Chip (NoCs) that use routers for the inter-core communication has been developed.

In any NoC, the routing of messages sent between cores plays a key role in achieving high performance. It is one of the major factors that determine how closely the network operates to the performance boundary set by the choice of topology. This article focuses on routing algorithms for two-dimensional (2D) meshes which constitute an important class of NoCs that have been widely used in both commercial and research systems [1]–[4].

Existing routing algorithms can be classified into two categories: oblivious routing and adaptive routing. Oblivious routing algorithms do not use the network’s state information in their routing decisions. On the other hand, adaptive routing algorithms consider the state of the network when determining a path for a packet from the source node to the destination node. Thus, they are potentially better in terms of performance. Despite this, oblivious routing algorithms are easier to implement, have much shorter routing computation delay, and do not incur much hardware overhead. Therefore, they are commonly used in practice.

The simplest and most popular oblivious routing algorithm for 2D meshes is the DOR algorithm. There are two variations of this algorithm: XY and YX. As shown in Fig. 1 (a), in the XY DOR algorithm, packets are routed first in the X dimension and then in the Y dimension to reach their destinations. On the other hand, in the YX DOR algorithm (Fig. 1 (b)), packets are routed first in the Y dimension and then in the X dimension. Besides simplicity, the DOR algorithm has the major advantage of being deadlock-free and thus has been employed in many practical systems.

1.1 DOR Algorithm in Asymmetric 2D Meshes

Most of the routing algorithms proposed for 2D meshes have been evaluated on only symmetric networks of size $n \times n$. However, in practice, the network size can be $m \times n$ where $m \neq n$. For example, the 2D mesh network used in the Intel Xeon Phi Knights Landing architecture [1] is 6 × 9. Moreover, when multiple parallel applications are mapped into a 2D-mesh-based system, each application does not necessarily have to be mapped into a symmetric sub-mesh region. Thus, it is crucial to evaluate routing algorithms on both symmetric and asymmetric networks.

Previous studies of supercomputers with asymmetric mesh or torus interconnects [6]–[8] have revealed that routing longest dimensions first typically provides better perfor-
mance than other strategies; that is, if the interconnect is an asymmetric 2D mesh with the X dimension longer than the Y dimension, then the XY DOR is often preferred over the YX DOR. This wisdom has also been adopted in some commercial many-core processors. For instance, the 2D-mesh-based Intel Xeon Phi Knights Landing architecture[1] employs the YX DOR because the Y dimension is longer than the X dimension.

To confirm the difference in performance of the two varieties of the DOR algorithm, XY and YX, in asymmetric 2D meshes, we have performed simulations on a 16 × 8 network (a 2D mesh with the X dimension longer than the Y dimension). The detailed parameters are described in Sect. 4. Figure 2 shows the throughput results under four different traffic patterns: uniform, hotspot, bit reverse, and shuffle. Under the uniform traffic, the destination of each packet is randomly selected with equal probability among all nodes in the network. The hotspot traffic is basically similar to the uniform traffic but there are four hotspot nodes (a 2 × 2 cluster) located at a corner of the network. These hotspot nodes receive four times more traffic than the other nodes. Under the bit reverse traffic, node $s_{b-1}s_{b-2} \ldots s_1s_0$ ($b$-bit binary representation) only communicates with node $d_{b-1}d_{b-2} \ldots d_1d_0$ where $d_i = s_{(i-1)} \mod b$. Finally, under the shuffle traffic pattern, which can be found in fast Fourier Transform (FFT) and sorting applications, node $s_{b-1}s_{b-2} \ldots s_1s_0$ ($b$-bit binary representation) only communicates with node $d_{b-1}d_{b-2} \ldots d_1d_0$ where $d_i = s_{(i-1)} \mod b$.

Figure 2 shows that the XY DOR typically outperforms the YX DOR, which is in line with previous studies. Below we explain the reason for this difference.

In an asymmetric 2D mesh, a channel on the longer dimension is typically shared by more flows than a channel on the shorter dimension. In the example here, either the XY DOR or YX DOR is used, channel C1 is shared by 8 flows while channel C2 is shared by only 4 flows.

Table 1 shows the utilization of X and Y channels in the 16 × 8 network under the uniform traffic when the XY DOR is used in our experiment. The same data are obtained when the routing algorithm is YX DOR. We can see that most X channels are shared by more flows and hence potentially more congested than Y channels. Only 64 X channels
Table 1  The utilization of X and Y channels in the 16 × 8 network under the uniform traffic when the XY DOR is used in our experiment. The same data are obtained when the routing algorithm is YX DOR. Load here indicates the number of flows that share a channel

| X channels | Load | #Channels |
|------------|------|-----------|
|            | 120  | 32        |
|            | 224  | 32        |
|            | 312  | 32        |
|            | 384  | 32        |
|            | 440  | 32        |
|            | 480  | 32        |
|            | 504  | 32        |
|            | 512  | 32        |

| Y channels | Load | #Channels |
|------------|------|-----------|
|            | 112  | 64        |
|            | 192  | 64        |
|            | 240  | 64        |
|            | 256  | 64        |

In LEF, we combine XY DOR and YX DOR. Decisions of selecting the right DOR algorithm are not fixed to the network shape but instead made on a per-packet basis. The first dimension of traversal of a packet is the one in which the packet needs to traverse more hops. In this way, LEF distributes the load over both XY and YX paths and thus can outperform the DOR algorithm when the traffic pattern is non-uniform. Moreover, in the cases of global communication in an asymmetric network and local communication within an asymmetric region, there are more source-destination pairs of which the distance in the longer dimension is greater than that in the shorter dimension. Thus, in LEF, the majority of packets traverse the longer dimension, which is more heavily loaded, first. Therefore, the pressure on the channel buffers is lower than in other oblivious routing algorithms like O1TURN [5] which also distributes the load over both XY and YX paths but 50% of packets traverse the longer dimension first and the remaining 50% traverse the shorter dimension first. By balancing between distributing the load over both XY and YX paths and reducing the pressure on the channel buffers, LEF can achieve higher throughput than other oblivious routing algorithms.

When LEF is used, deadlock may occur because of the combination of XY DOR and YX DOR. We propose an efficient deadlock avoidance method for LEF in which the use of virtual channels (VCs) is more flexible than in the conventional method used by O1TURN [5]. Using the proposed method, we can expect a more effective utilization of VCs which contributes to improving the overall performance.

Table 2  The utilization of X and Y channels in the 16 × 8 network under the bit reverse traffic pattern when the XY DOR is used in our experiment. The same data are obtained when the routing algorithm is YX DOR

| X channels | Load | #Channels |
|------------|------|-----------|
|            | 0    | 128       |
|            | 1    | 28        |
|            | 2    | 24        |
|            | 3    | 20        |
|            | 4    | 16        |
|            | 5    | 12        |
|            | 6    | 8         |
|            | 7    | 4         |

| Y channels | Load | #Channels |
|------------|------|-----------|
|            | 112  | 28        |
|            | 192  | 24        |
|            | 240  | 20        |
|            | 256  | 16        |

(32 shared by 120 flows and 32 shared by 224 flows) have a higher probability to be less congested than some Y channels.

Since most X channels are much more congested than Y channels, under the XY DOR algorithm, packets tend to traverse the bottleneck (the most heavily congested) channels in their routes first. Thus, packets waiting to enter their bottleneck channels are mostly queued in memory outside the network. On the contrary, under the YX DOR algorithm, packets waiting to enter their bottleneck channels are queued in the channel buffers inside the network which have very limited capacity. Because of this, the pressure on the channel buffers is higher and the network gets saturated more quickly under the YX DOR algorithm as shown in Fig. 2.

The utilization of X and Y channels under the hotspot and shuffle traffics is different between the XY DOR and the YX DOR. However, like under the uniform traffic, packets tend to traverse from heavily congested channels to equally or less congested ones much more in the XY DOR than in the YX DOR. Therefore, the XY DOR also performs better under the hotspot and shuffle traffics as shown in Fig. 2.

Table 2 shows the utilization of X and Y channels in the 16 × 8 network under the bit reverse traffic when the XY DOR is used. Like under the uniform traffic, the same data are obtained when the routing algorithm is YX DOR. However, we can see that the congestion level in the X dimension is not much different from that in the Y dimension. For this reason, as shown in Fig. 2, there is almost no difference in performance of the XY DOR and the YX DOR under this traffic pattern.

1.2 Our Contributions

In this article, we propose LEF (Long Edge First), a new oblivious routing algorithm for 2D meshes, using the wisdom of selecting the right DOR algorithm explained in Sect. 1.1. LEF is designed to give high throughput, especially when the communication is within an asymmetric mesh, while remaining inexpensive to implement in hardware.

In LEF, we combine XY DOR and YX DOR. Decisions of selecting the right DOR algorithm are not fixed to the network shape but instead made on a per-packet basis. The first dimension of traversal of a packet is the one in which the packet needs to traverse more hops. In this way, LEF distributes the load over both XY and YX paths and thus can outperform the DOR algorithm when the traffic pattern is non-uniform. Moreover, in the cases of global communication in an asymmetric network and local communication within an asymmetric region, there are more source-destination pairs of which the distance in the longer dimension is greater than that in the shorter dimension. Thus, in LEF, the majority of packets traverse the longer dimension, which is more heavily loaded, first. Therefore, the pressure on the channel buffers is lower than in other oblivious routing algorithms like O1TURN [5] which also distributes the load over both XY and YX paths but 50% of packets traverse the longer dimension first and the remaining 50% traverse the shorter dimension first. By balancing between distributing the load over both XY and YX paths and reducing the pressure on the channel buffers, LEF can achieve higher throughput than other oblivious routing algorithms.

When LEF is used, deadlock may occur because of the combination of XY DOR and YX DOR. We propose an efficient deadlock avoidance method for LEF in which the use of virtual channels (VCs) is more flexible than in the conventional method used by O1TURN [5]. Using the proposed method, we can expect a more effective utilization of VCs which contributes to improving the overall performance.

Research and development of NoCs have mainly relied on software simulators such as BookSim [9] and Noxim [10]. Unfortunately, since these simulators are slow, most existing routing algorithms have not been evaluated in large-scale NoCs which consist of thousands of nodes. In this article, we use the fast and cycle-accurate FPGA-based NoC emulator proposed in [11] to evaluate LEF and its counterparts in networks of various sizes ranging from 8 × 8 to 128 × 64.

This article is an extension of our previous work [12] and [13]. We optimize the selection scheme of XY DOR and YX DOR in LEF and discuss how VCs should be used in the proposed deadlock avoidance method. We show that LEF can be implemented with a small overhead. We also provide more detailed analysis and evaluation results. We compare LEF against not only the DOR algorithm but also O1TURN [5], another effective oblivious routing algorithms for 2D meshes, and a minimal adaptive routing algorithm based on the odd-even turn model[14]. We show that the proposed deadlock avoidance method is more effective than the conventional method and can be used to improve the performance of O1TURN. We examine the routing algorithms on NoCs of various sizes from 8 × 8 to 128 × 64 and find that, as the network size increases, the performance becomes more strongly influenced by the resource allocation policy and the effect is different for each routing algorithm: at high loads, the adaptive algorithm, LEF, and O1TURN require a fairer allocation policy than the DOR algorithm.
2. Related Work

Since the DOR algorithm offers no path diversity, it does a poor job of load balancing the network under many traffic patterns. To address this problem, some oblivious routing algorithms including Valiant’s [15], ROMM [16], O1TURN [5], and IX/Y [17] have been proposed.

Valiant’s algorithm [15] offers a high path diversity by routing each packet from the source node to the destination node via a randomly chosen intermediate node. With the high path diversity, this routing algorithm achieves optimal worst-case throughput1. However, it suffers from low average-case throughput and high latency because a packet may traverse from source to destination through a non-minimal path.

ROMM [16] is similar to Valiant’s algorithm in which it routes each packet from the source node to the destination node via an intermediate node2. However, the intermediate node in ROMM must be within the minimum rectangle defined by the source node and the destination node whereas the intermediate node in Valiant’s algorithm can be anywhere in the network. Because of this, packets are always routed through minimal paths in ROMM. Thus, ROMM has better average-case throughput and latency than Valiant’s algorithm. However, it has been demonstrated that, in the worst case, ROMM may deliver lower performance than the DOR algorithm in 2D tori and meshes [5], [18].

O1TURN is a simple but effective oblivious routing algorithm proposed by Seo et al. [5]. As shown in Fig. 1 (c), this routing algorithm combines XY DOR and YX DOR. The first dimension of traversal is chosen randomly. It has been shown that O1TURN can deliver higher performance than the DOR algorithm as well as Valiant’s algorithm and ROMM.

IX/Y is another oblivious routing algorithm [17]. In this algorithm, a packet is routed alternately to the X dimension and the Y dimension from the source node to the destination node (the dimension change is stopped when the packet reaches a node in the same row or column as the destination node). At the source node, the packet is routed to the X dimension if the previous packet sent by this source node is routed to the Y dimension, and to the Y dimension otherwise. Since IX/Y distributes packets over both XY and YX paths, we believe that it requires a deadlock avoidance method. However, the authors of IX/Y did not provide any discussions about this issue.

Kinsy et al. [19] propose a framework for statically determining deadlock-free routes that minimize maximum channel load based on analysis of the given applications. Since the global knowledge of the application characteristics is taken into account for determining each route, this approach can deliver better performance than traditional oblivious routing algorithms. However, reconfigurable routing tables are required to store the pre-computed routes, which might be expensive to implement, especially when the network is large. Also, as the network size increases and the target applications become more complicated, determining optimized routes while preserving the deadlock freedom becomes more difficult.

Adaptive routing algorithms can provide better performance than the above-mentioned oblivious routing algorithms by adjusting routing decisions to best fit the condition of the network. Most existing adaptive routing algorithms are based on either Dally’s theory [20] or Duato’s theory [21] for deadlock freedom. The adaptive routing algorithm that we implement in this article is based on Dally’s theory.

3. The LEF Routing Algorithm

3.1 Selection of XY DOR and YX DOR

Figure 1 (d) shows the basic idea of the selection of XY DOR and YX DOR in LEF. When routing a packet, which DOR algorithm is chosen depends on the relative position between the source node and the destination node. The XY DOR will be chosen if the distance of the X coordinates of the source node and the destination node (Δx) is greater than the distance of the Y coordinates (Δy). Otherwise, if Δy is greater than Δx, the YX DOR will be chosen. In the case that Δx is equal to Δy, the first dimension of traversal will be selected randomly like in O1TURN; however, other strategies can also be considered. For example, we can statically choose XY DOR or YX DOR based on the location of the source node: if the source node is in an even column, then XY DOR is chosen; otherwise, YX DOR is chosen. This static strategy may not perform as well as the random strategy but has an advantage of making it easier to preserve the order of packets which is important in many cases. The detailed comparison between the strategies, however, is left as future work.

When Δx is always greater than Δy, LEF is reduced to XY routing. Similarly, when Δx is always smaller than Δy, LEF is reduced to YX routing. However, we believe that such cases do not commonly occur in practice. Figure 4 shows the distribution of packets with Δx > Δy, Δx < Δy, and Δx = Δy in the PARSEC [22] traces collected by Hestness et al. [23] from an 8 × 8 chip multiprocessor. We can see that there is no case in which most of the packets have Δx > Δy or Δx < Δy.

In the example in Fig. 1 (d), because Δy1 is greater than Δx1, packets sent from node S1 to node D1 are routed with YX DOR. On the other hand, packets sent from node S2 to node D2 are routed with XY DOR because Δx2 is greater than Δy2. Finally, packets sent from node S3 to node D3 can be routed with both XY DOR and YX DOR (randomly chosen) because Δx3 is equal to Δy3.

By selecting XY DOR or YX DOR for each packet...

---

1 The minimum throughput over all traffic patterns.

2 Here, we consider the two-phase ROMM which is the most popular version of ROMM. In general, in the n-phase ROMM algorithm, a packet is routed from source to destination via n − 1 intermediate nodes.
as described above, LEF naturally distributes the load over both XY and YX paths. It thus achieves better load balancing than the DOR algorithm which routes every packet in a fixed dimension first. Existing oblivious routing algorithms like O1TURN described in Sect. 2 also distribute the load over both XY and YX paths. However, O1TURN is less effective than LEF because it causes higher pressure on the channel buffers. This occurs especially when the nodes participating in communication lie on an asymmetric mesh, including the global communication in an asymmetric network and the local communication occurring when multiple parallel applications are mapped into a network (may be symmetric or asymmetric) like in Fig. 5. For the convenience of explanation, let us consider an asymmetric mesh with the X dimension longer than the Y dimension. In this case, a channel in the X dimension is shared by more flows and thus potentially more heavily loaded than a channel in the Y dimension as discussed in Fig. 3. Also, there are more source-destination pairs of which the distance in the X dimension is greater than that in the Y dimension. Therefore, in LEF, the majority of packets traverse the X dimension first and thus tend to move from a heavily loaded channel to a lightly loaded one when turning to the Y dimension.

This results in lower pressure on the channel buffers than in O1TURN where 50% of packets tend to move from a lightly loaded channel to a heavily loaded one when turning from the Y dimension to the X dimension.

Besides the advantages discussed above, LEF has another strength of being effective under any application mapping pattern. This is significant since the application mapping pattern often changes over time in practice. To illustrate this strength of LEF, let us look at the following two examples.

Figure 5 (a) shows the first example in which three parallel applications are mapped into three sub-meshes 5×3, 5×5, and 3×8 of an 8×8 mesh. Interestingly, if we use the XY DOR algorithm for this network, the application running in the 5×3 sub-mesh will get benefited but the one running in the 3×8 sub-mesh will be harmed. On the other hand, if the YX DOR algorithm is used, the application running in the 3×8 sub-mesh will get benefited but the one running in the 5×3 sub-mesh will be harmed. By using LEF, the performance of both applications in the two sub-meshes will be boosted.

Figure 5 (b) shows the second example in which three parallel applications are mapped into three sub-meshes 4×2, 4×3, and 4×3 of a 4×8 mesh. According to the conventional wisdom explained in Sect. 1, we should use the YX DOR algorithm for this network because the Y dimension (8) is longer than the X dimension (4). However, for the mapping pattern in Fig. 5 (b), the XY DOR algorithm is more suitable since every sub-mesh has the X dimension longer than the Y dimension. By using LEF, this problem will disappear. This is because LEF is effective regardless of which dimension is longer.

In the above two examples, applications can be mapped into both symmetric and asymmetric sub-meshes. To demonstrate the benefits of mapping parallel applications into asymmetric sub-meshes, let us consider the following example. Assume that we have three applications and want to run them in parallel on an 8×8 system: application A requires 12 processing elements (PEs), application B requires 20 PEs, and application C requires 25 PEs. In the case of not using asymmetric sub-meshes, we will need three square sub-meshes: one 4×4 for application A and two 5×5 for two applications B and C. However, there is no way that an 8×8 mesh can be divided into one 4×4 and two 5×5 sub-meshes. On the other hand, the applications can be mapped into three asymmetric sub-meshes 5×3, 3×8, and 5×5 as shown in Fig. 5 (a).

### 3.2 Deadlock Avoidance

In the explanations below, we call a packet that is routed with the XY DOR an XY packet. Similarly, a YX packet is the one routed with the YX DOR.

Since LEF combines XY DOR with YX DOR, dependency cycles involving XY packets and YX packets may be formed, and thus deadlock may occur. This is the same as in O1TURN.
Fig. 6 O1TURN[5] avoids deadlock by completely separating XY packets and YX packets. In each physical channel, half of the VCs are for XY packets while the other half for YX packets.

As shown in Fig. 6, O1TURN avoids deadlock by completely separating XY packets and YX packets. It requires two or more VCs per physical channel. In each physical channel, half of the VCs are for XY packets while the other half for YX packets. In this way, it is clear that no dependency cycle can be formed, and therefore, deadlock does not occur.

3.2.1 Proposed Deadlock Avoidance Method

We propose a new deadlock avoidance method in which the use of VCs is more flexible than in the O1TURN’s method described above. Our method requires that the atomic VC allocation policy is used. In this policy, a VC can be re-allocated to a new packet only when the tail flit of the last packet occupying this VC has left. Thus, at any given time, a VC can be occupied by only one packet. This is also an essential requirement for making fully adaptive routing algorithms deadlock-free [24].

Like in the O1TURN’s method, the number of VCs per physical channel ($v$) in our method must be also greater than or equal to two ($v \geq 2$). We provide two design options:

1. **Y-restricted**: reserve $k$ VCs ($1 \leq k \leq v - 1$) in each physical channel in the Y dimension exclusively for XY packets (Fig. 7 (a)).
2. **X-restricted**: reserve $k$ VCs ($1 \leq k \leq v - 1$) in each physical channel in the X dimension exclusively for YX packets (Fig. 7 (b)).

The VCs that are reserved exclusively for XY packets and YX packets are called **XY-exclusive VCs** and **YX-exclusive VCs**, respectively. In the Y-restricted approach, XY packets are prohibited from using XY-exclusive VCs in the Y dimension while there is no restriction on VCs that XY packets can use. Similarly, in the X-restricted approach, XY packets are prohibited from using YX-exclusive VCs in the X dimension while YX packets can use any VC in both the X and Y dimensions. The proof of deadlock freedom will be presented in Sect. 3.2.2.

The decision of which design option, Y-restricted or X-restricted, is chosen is taken according to the shape of the network. In the case the X dimension is longer than the Y dimension, we choose the Y-restricted approach. This is because there are fewer Y channels than X channels. Also, there is a high probability that a channel on the Y dimension is shared by fewer flows and thus potentially less heavily utilized than a channel on the X dimension. Thus, using the Y-restricted approach is better. For the similar reasons, we choose the X-restricted approach in the case the Y dimension is longer than the X dimension. For symmetric networks, either approach can be chosen.

As will be described in the proof of deadlock freedom in Sect. 3.2.2, exclusive VCs are the keys that prevent potential deadlock situations from becoming real ones. The number of exclusive VCs per channel $k$ affects how fast a potential deadlock situation can be broken. Using too few exclusive VCs may make it too slow to break potential deadlock situations, especially at high loads, which may hurt the overall network performance. On the other hand, using too many exclusive VCs may make the non-exclusive VCs in

---

†In our evaluation, we adopt the Y-restricted approach for symmetric networks.
the same dimension with the exclusive VCs too congested, which in turn also leads to poor performance. In Sect. 4.2.4, we quantitatively evaluate the impact of $k$ on LEF’s performance.

Using the proposed method, we can expect a more effective utilization of VCs than in O1TURN’s method. This contributes to improving the overall performance, which we will see in Sect. 4.

3.2.2 Proof of Deadlock Freedom

We will prove that deadlock does not occur in LEF when the proposed deadlock avoidance method in Sect. 3.2.1 is used. We provide the proof for the X-restricted approach (Fig. 7 (b)). The proof for the Y-restricted approach (Fig. 7 (a)) is almost the same.

In our proof, we assume an $m \times n$ mesh shown in Fig. 8. As mentioned in Sect. 3.2.1, our deadlock avoidance method requires the atomic VC allocation policy in which a VC can be re-allocated to a new packet only when the tail flit of the last packet occupying this VC has left. Thus, packets are always at the heads of the VCs that they are occupying. A packet can always be eventually transmitted to the next node in the route to the destination if this node has at least one VC $c$ satisfying the following two conditions: (1) $c$ is not blocked forever by any other packet, and (2) the packet is not prohibited from using $c$. In our proof, we use this fact that any VC is not blocked forever by any packet, and thus LEF is deadlock-free.

Our proof consists of three lemmas and one theorem. We first prove the following lemma.

Lemma 3.1. Any YX-exclusive VC is eventually released.

Proof. As shown in Fig. 7 (b), YX-exclusive VCs are located at the input ports +x and -x. They can be occupied by only YX packets. Thus, there are only two cases for a packet that is currently in a YX-exclusive VC.

- Case 1: the packet already arrived at its destination.
- Case 2: the packet will proceed to next channel in the X dimension.

In case 1, the packet will be forwarded to the processing core of the current node. Thus, the YX-exclusive VC that the packet is occupying will be released. Below we focus on case 2.

i) input port -x of node (m,1). First, we consider the YX-exclusive VCs in input port -x of node (m,1). Case 2 never happens since output port +x of node (m,1) is not connected to any other nodes. Thus, the YX-exclusive VCs in input port -x of node (m,1) are eventually released.

ii) input port -x of node (m-1,1). Next, we consider the YX-exclusive VCs in input port -x of node (m-1,1). When case 2 happens, the packet should be transmitted to node (m,1). At least, the YX-exclusive VCs in input port -x of node (m,1) are eventually released by i). Thus, the packet will certainly forwarded to node (m,1) and the YX-exclusive VC in input port -x of node (m-1,1) currently occupied by the packet will be released.

iii) input ports -x of the remaining nodes in row 1 (nodes from (m-2,1) to (1,1)). With similar arguments as in i) and ii), we can prove that the YX-exclusive VCs in the input ports -x of nodes from (m-2,1) to (1,1) are surely released. Input port -x of node (1,1) is not connected to any other nodes and thus can be ignored.

iv) input ports -x of nodes in rows 2, 3, ..., n. The proofs for rows 2, 3, ..., n are similar to that of row 1 (from i) to iii)).

v) input ports +x. By the similar arguments as from i) to iv), we can prove that the YX-exclusive VCs in input ports +x of all nodes are also eventually released.

By i) – v), we can conclude that any YX-exclusive VC is eventually released. □

Next, we prove the following lemma.

Lemma 3.2. Any VC in the Y dimension is eventually released.

Proof. As shown in Fig. 7 (b), all VCs in the Y dimension are non-exclusive VCs. Since both XY packets and YX packets can utilize non-exclusive VCs, there are three cases for a packet that is currently in a VC in an input port +y or -y.

- Case 1: the packet already arrived at its destination.
- Case 2: the packet will proceed to next channel in the Y dimension.
- Case 3: the packet will proceed to next channel in the X dimension.

As same as in the proof of Lemma 3.1, case 1 is trivial. Below we focus on case 2 and case 3.

i) input port -y of node (1,n). First, we consider the VCs in input port -y of node (1,n). Case 2 never happens since output port +y of node (1,n) is not connected to any other nodes. When case 3 happens, the packet should be transmitted to the next channel via output port -x or +x (note that output port -x of node (1,n) is not connected to any other nodes; we add it here for the generality of explanations). In this situation, the packet can be surely sent to the X dimension since at least the YX-exclusive VCs in input port -x or +x of the next node is surely released by Lemma 3.1. Thus, the VCs in input port -y of node (1,n) are eventually released.

ii) input port -y of node (1,n-1). Next, we consider the VCs in input port -y of node (1,n-1). When case 2 happens,
the next node of the packet is (1,n). The VCs in input port -y of node (1,n-1) are eventually released since the VCs in input port -y of node (1,n) are eventually released by i). When case 3 happens, the VCs in input port -y of node (1,n-1) are also eventually released as similarly discussed in i). Therefore, the VCs in input port -y of node (1,n-1) are eventually released in every case.

iii) input ports -y of the remaining nodes in column 1 (nodes from (1,n-2) to (1,1)). As similarly discussed in i) and ii), we can prove that the VCs in the input ports -y of nodes from (1,n-2) to (1,2) are surely released. Input port -y of node (1,1) is not connected to any other nodes and thus can be ignored.

iv) input ports -y of nodes in columns 2, 3, · · · , m. The proofs for columns 2, 3, · · · , m are similar to that of column 1 (from i) to iii)).

v) input ports +y. All VCs in the input ports -y of all nodes are eventually released by i) – iv). In the same way, we can prove that all VCs in the input ports +y of all nodes are also eventually released.

By i) – v), we can conclude that any VC in the Y dimension is eventually released. □

Finally, we prove the following lemma.

Lemma 3.3. Any non-exclusive VC in the X dimension is eventually released.

Proof. We will prove this lemma using Lemma 3.1 and Lemma 3.2. There are three cases for a packet that is currently in a non-exclusive VC in the X dimension.

• Case 1: the packet already arrived at its destination.
• Case 2: the packet will proceed to next channel in the X dimension.
• Case 3: the packet will proceed to next channel in the Y dimension.

As same as in the proof of Lemma 3.1, case 1 is trivial. For case 3, the VC is eventually released because all VCs in input port +x of this node are not connected to any nodes. Therefore, all non-exclusive VCs of this input port are eventually released by case 1 and case 3. Since all YX-exclusive VCs are also released by Lemma 3.1, all VCs of this input port are released.

i) input port -x of node (m,1). Case 2 cannot happen since output port +x of this node is not connected to any nodes. Therefore, all non-exclusive VCs of this input port are eventually released by case 1 and case 3. Since all YX-exclusive VCs are also released by Lemma 3.1, all VCs of this input port are released.

ii) input port -x of node (m-1,1). When case 2 happens, the packet should be transmitted to node (m,1). Since all VCs of input port -x of node (m,1) are released, all VCs of input port -x of node (m-1,1) are also released.

iii) input ports -x of the remaining nodes in row 1 (nodes from (m-2,1) to (1,1)). With similar arguments as in i) and ii), we can prove that all VCs of input ports -x of nodes from (m-2,1) to (1,1) are eventually released.

iv) input ports -x of rows 2, 3, · · · , n. The proofs are similar to that of row 1 (from i) to iii)).

v) input ports +x. The proofs are similar to those of input ports -x.

By i) – v), we can conclude that any non-exclusive VC in the X dimension is eventually released. □

We can easily derive the following theorem using Lemma 3.1, Lemma 3.2, and Lemma 3.3.

Theorem 3.4. All VCs in the network are eventually released.

Since all VCs in the network are eventually released by Theorem 3.4, LEF is deadlock-free. The proposed deadlock avoidance method can also be used for any other routing algorithms that combine XY DOR and YX DOR. In Sect. 4, we will show that it helps to improve the performance of O1TURN considerably.

There is an important issue about our deadlock avoidance method (like in the proof, here we focus on the X-restricted approach; there is a similar discussion for the Y-restricted approach). When a YX packet occupies a YX-exclusive VC at a given node, say node N1, it can freely use any of the available VCs at the next node, say node N2. It is not necessary to restrict that packet to use only YX-exclusive VCs. Deadlocks do not occur because of the following reason. Lemma 3.1 indicates that the packet can always be eventually transmitted to node N2. At node N2, if the packet occupies a YX-exclusive VC, then Lemma 3.1 is applied again. Otherwise, Lemma 3.3 is applied. In this way, the packet will eventually reach its destination.

3.3 Optimization on the Selection of XY and YX DOR

In this section, we discuss how the utilization of buffers (VCs) can be improved by optimizing the selection of XY DOR and YX DOR.

In our implementation of LEF, each packet is tagged with a routing flag at its source node. This flag can be either “XY DOR” or “YX DOR”. It determines (1) how the packet will be routed to the destination node and (2) whether the packet is prohibited from using some specific VCs for deadlock freedom.

Figure 9 shows the straightforward algorithm for determining each packet’s routing flag. The behavior of this algorithm is the same as that described in Sect. 3.1. We will first show that it may cause unnecessary restrictions on

---

**Fig. 9** The straightforward algorithm for determining each packet’s routing flag described in Sect. 3.1.

---

1: Input: source node $S(x_S, y_S)$ and destination node $D(x_D, y_D)$
2: Output: routing flag $R$
3: $\Delta x = |x_D - x_S|$
4: $\Delta y = |y_D - y_S|$
5: if $\Delta x > \Delta y$ then
6: $R = \text{XY DOR}$
7: else if $\Delta x < \Delta y$ then
8: $R = \text{YX DOR}$
9: else
10: $R = \text{Random(XY DOR or YX DOR)}$
11: end if
the VC usage of certain packets. Then we propose a solution to eliminate those restrictions.

When combining XY DOR and YX DOR together, if deadlock arises, it is caused by only packets of which routed paths lie on both the X and Y dimensions. However, the VC usage restriction in the deadlock avoidance method described in Sect. 3.2.1 is applied to all packets. According to the algorithm in Fig. 9, packets of which routed paths lie only on the X dimension ($\Delta x > 0$ and $\Delta y = 0$) are tagged with “XY DOR” flag. If the X-restricted approach (Fig. 7 (a)) is used, these packets will be prevented from using the YX-exclusive VCs which are exclusively reserved for YX packets. Likewise, packets of which routed paths lie only on the Y dimension ($\Delta x = 0$ and $\Delta y > 0$) are tagged with “YX DOR” flag. Thus, if the Y-restricted approach (Fig. 7 (a)) is used, these packets will be prevented from using the XY-exclusive VCs which are exclusively reserved for XY packets. These restrictions are unnecessary and may make the exclusive VCs underutilized while the non-exclusive VCs are congested, especially when the traffic is dominated by packets of which routed paths lie on only one dimension.

We eliminate the unnecessary restrictions described above by slightly modifying the strategy for determining each packet’s routing flag as shown in the algorithm in Fig. 10. Here we assume that the Y-restricted deadlock avoidance method (Fig. 7 (a)) is used. When the path of a packet lies only on the Y dimension ($\Delta x = 0$), it is tagged with “YX DOR” flag and thus can use all VCs in the Y dimension. The algorithm for the case the X-restricted deadlock avoidance method (Fig. 7 (b)) is used is similar to the algorithm in Fig. 10. We only have to modify lines 5 and 6 as follows:

\[
\text{if } \Delta y == 0 \text{ then } R = \text{XY DOR.}
\]

This means that, when the path of a packet lies only on the X dimension ($\Delta y = 0$), it is tagged with “YX DOR” flag and thus can use all VCs in the X dimension. In both cases of the deadlock avoidance method, the selection of “XY DOR” or “YX DOR” routing flag for packets that must traverse both X and Y dimensions to reach their destinations is the same as in the algorithm in Fig. 9.

3.4 LEF Implementation

We adopt the conventional input-queued VC router architecture [25] with wormhole flow control [26]. Each packet is composed of multiple flits (flow control digits) and the routing information is carried by the head flit.

We implement LEF by adding one bit to the flit structure. This bit stores the routing flag described in Sect. 3.3, which is determined at the source node and passed along with the head flit to all nodes of the route. In this way, every packet is always aware of its type (XY or YX), and therefore, it always knows which VCs it can use since the information of which VCs are XY/YX-exclusive and which VCs are non-exclusive is predetermined. In our router implementation, we add some logic to ensure that an XY packet does not send requests for XY-exclusive VCs (and similarly, a YX packet does not send requests for YX-exclusive VCs) to the VC allocator.

4. Evaluation

4.1 Evaluation Methodology

To evaluate the performance of LEF compared to other routing algorithms, we use the FPGA-based NoC emulator proposed in [11] which is about two to three orders of magnitude faster than conventional software-based NoC simulators and can support designs with thousands of nodes. Another advantage of this FPGA-based NoC emulator is that it supports asymmetric networks which are not officially supported by some widely used software-based NoC simulators like BookSim [9]. It is confirmed that the FPGA-based NoC emulator and BookSim report exactly the same results when they use the same pseudo-random number generator. The results in this article are obtained when using the xorshift128+ pseudo-random number generator [27].

Table 3 shows the simulation parameters. We consider both symmetric and asymmetric meshes. We present results

| Table 3 | Emulation parameters |
|---------|---------------------|
| Middle-scale NoCs | 8 x 8, 16 x 8, 16 x 16 |
| Large-scale NoCs | 64 x 64, 128 x 64 |
| Router architecture | Input-queued VC router |
| Router pipeline | 5-stage |
| # of VCs per physical channel | 4 |
| VC size | 4-flit |
| VC/switch allocator | iSLIP [28] |
| Arbiter type | Round-robin |
| Flow control | Wormhole & credit-based |
| Packet length | 16-flit |
| Injection process | Bernoulli process |
| Traffic pattern | Uniform, hotspot, transpose, bit reverse, shuffle |

Fig. 10 The optimized algorithm for determining each packet’s routing flag in LEF for the case the Y-restricted deadlock avoidance method (Fig. 7 (a)) is used. For the case the X-restricted deadlock avoidance method (Fig. 7 (b)) is used, lines 5 and 6 should be modified as follows: if $\Delta y == 0$ then $R = \text{XY DOR.}$
for five network sizes: 8x8, 16x8, 16x16, 64x64, and 128x64. The implemented router has five pipeline stages, namely RC (routing computation), VA (VC allocation), SA (switch allocation), ST (switch traversal), and LT (link traversal). We assign four VCs per each physical channel, each can hold four flits.

We consider five synthetic traffic patterns: uniform, hotspot, transpose, bit reverse, and shuffle. Four patterns, uniform, hotspot, bit reverse, and shuffle, have been described in Sect. 1.1. With regard to the hotspot pattern, we increase the number of hotspot nodes to 256 (a 16 x 16 cluster) in the large-scale NoCs. The transpose traffic pattern is only used with symmetric meshes because it is not applicable to asymmetric meshes. Under this traffic pattern, node \((i, j)\) only sends messages to node \((j, i)\). In an asymmetric mesh, given a node \((i, j)\), node \((j, i)\) may not exist.

For each traffic pattern, each of the NoCs is emulated with different flit injection rates: from a low injection rate at which there is almost no contention in the network to a high injection rate at which the network gets saturated. The emulation at each injection rate is executed in three phases: warm-up (middle-scale NoCs: 100,000 cycles; large-scale NoCs: 200,000 cycles), measurement (middle-scale NoCs: 100,000 cycles; large-scale NoCs: 200,000 cycles), and drain. The average packet latency is calculated based on the latencies of all packets generated during the measurement phase. The network throughput is calculated from the number of packets arriving at their destinations during the measurement phase.

To evaluate the hardware overhead and timing of LEF compared to other routing algorithms, we have developed synthesizable 4 x 4 NoC designs in Verilog HDL and implemented them on a Virtex-7 VX485T 28nm FPGA. Vivado 2017.4 is used as the synthesis and implementation tool. The NoCs employ the conventional input-queued 5-stage pipelined router architecture with four VCs per port as shown in Table 3. In every case, we set the timing constraint to 90MHz, the operating frequency that the designs can easily achieve. This helps to minimize the extra hardware overhead for timing optimizations.

4.2 Performance Analysis

4.2.1 Middle-Scale NoCs

We compare LEF against the DOR algorithm (both XY DOR and YX DOR), O1TURN [5], and a minimal adaptive routing algorithm based on the odd-even turn model [14]. In the adaptive routing algorithm, when there are two available output ports, our selection strategy selects the port with more free VCs. A more complicated selection strategy may provide better performance but can also produce diminishing results if it makes the router critical path longer. In the description below and the result graphs, we denote the adaptive routing algorithm by Odd-Even. We include the results of Odd-Even to show that LEF can provide comparable or even slightly better performance than complex adaptive routing algorithms in certain cases. Our main objective is to compare LEF with O1TURN and the DOR algorithm which are in the same category of oblivious routing algorithms as LEF.

Figures 11, 12, and 13 show the average packet latency and throughput results of three network sizes 8x8, 16x8, and 16x16, respectively. The results vary with both the shape of the network and the offered traffic pattern.
In the symmetric 8 × 8 NoC, the XY DOR and YX DOR algorithms provide almost the same results regardless of the offered traffic pattern. They outperform the other routing algorithms under the uniform traffic. This result is the same as those reported in [29] and [14] which indicate the superiority of the DOR algorithm under the uniform traffic. The main reason for this is that, under the uniform traffic pattern, the DOR algorithm can make the traffic distribution more even than the other routing algorithms [14], [29]. LEF, O1TURN, and Odd-Even show their strengths with the non-uniform traffics. Thanks to the adaptability of routing decisions to the state of the network, Odd-Even performs better than the other routing algorithms in most cases. Under the transpose traffic, LEF, O1TURN, and Odd-Even deliver by far better performance than the DOR algorithm. In this traffic pattern, since all source-destination pairs have the distance in the X dimension equal to the distance in the Y dimension, LEF randomly selects XY DOR and YX DOR for every packet, which is the same as O1TURN. However, LEF is slightly better than O1TURN thanks to its effective deadlock avoidance method described in Sect. 3.2.1. This will be discussed in more detail in Sect. 4.2.3. Under the bit reverse traffic, although LEF results in higher average packet latency than O1TURN at injection rates near to the saturation point of the network, it achieves higher throughput. Un-

Fig. 12 16 × 8 NoC: average packet latency and throughput results with four different traffic patterns.
nder the shuffle traffic, LEF slightly outperforms O1TURN in terms of both latency and throughput. In summary, compared to the DOR algorithm, the throughput provided by LEF is around 2.5% lower under the uniform traffic but around 10.4%, 33.6%, 39.3%, and 63% higher under the hotspot, transpose, bit reverse, and shuffle traffics, respectively. Compared to O1TURN, the throughput provided by LEF is around 4.5%, 10.2%, 3.1%, 5.7%, and 3% higher under the uniform, hotspot, transpose, bit reverse, and shuffle traffics, respectively.

In the former implementations [12], [13], LEF is reduced to XY routing for traffic patterns like transpose where distance in each coordinate is equal for every source-destination pair. The above result shows that the improved LEF can deliver better performance than XY routing (and thus the former implementation of LEF) under transpose traffic pattern. We believe that applications exhibiting the transpose traffic pattern should get benefits from this change.

In the asymmetric 16 × 8 NoC, LEF is the best-performing routing algorithm under four traffic patterns. It is outperformed by Odd-Even and O1TURN (only in terms of latency) only under the bit reverse pattern. We recall from Sect. 1.1 that bit reverse is the only traffic pattern in our experiment under which the XY DOR does not outperform the YX DOR. When the XY DOR performs better than the YX DOR, LEF is superior to the other algorithms. Under the uniform traffic, LEF delivers almost the same throughput as the XY DOR and around 9.9%, 10.2%, and 8.5% higher than the YX DOR, O1TURN, and Odd-Even, respectively. Under the hotspot traffic, the throughput provided by LEF is higher than all other algorithms with the approximate differences of 11%, 39%, 28.5%, and 21% compared to the XY DOR, YX DOR, O1TURN, and Odd-Even, respectively. Interestingly, O1TURN is outperformed by the XY DOR and LEF is slightly better than Odd-Even. Under the bit reverse traffic, the throughput delivered by LEF is around 43.7% and 4% higher than the DOR algorithm and O1TURN, and around 8.5% lower than Odd-Even. Finally, under the shuffle traffic, LEF is slightly better than the XY DOR. These two algorithms outperform the others by large margins, delivering around 86%, 64.5%, and 17% higher throughput than the YX DOR, O1TURN, and Odd-Even, respectively. The performance gain of LEF comes from two sources. First, as explained in Sect. 1, LEF balances between distributing the load over both XY and YX paths and reducing the pressure on the channel buffers by routing a packet along the dimension in which the source-destination distance is longer first. Second, LEF uses a more effective deadlock avoidance method than the conventional one which is used by O1TURN.

The results in the 16 × 16 NoC are basically similar to those in the 8 × 8 NoC. However, we observed a phenomenon that did not occur in the 8 × 8 NoC. When Odd-Even is used, the network becomes unstable at extremely high loads (Fig. 13 (a) and Fig. 13 (b)). This phenomenon is caused by our selection of using round-robin arbiters for VC and physical channel allocation. At high loads, it is likely that most VCs in the network are occupied at most the time. When Odd-Even is used, packets tend to have to compete for VC and physical channel allocation more than in the case the DOR algorithm is used. For instance, with the XY DOR, a packet at input port +y or -y will not go to output ports +x and -x since the packet is routed first in the X dimension and then in the Y dimension to reach its destination. Because of this, when the XY DOR is used, there is less competition for the VCs and physical channels in the X dimension. However, this is not the case for Odd-Even. Because of the higher competition for VCs and physical channels, Odd-Even requires a better allocation strategy, especially for large networks where there exist many long paths. Since we currently use round-robin arbiters, the VC and physical channel arbitration is locally fair but globally unfair. The latency of a long-path packet may be extremely high because it is treated the same as short-path packets in every hop from source to destination.

Like Odd-Even, we observed that LEF also caused the 16 × 16 NoC slightly unstable at high loads. The reason is the same as for Odd-Even. This is also the reason why the throughput provided by LEF becomes slightly lower than that provided by O1TURN under the uniform, bit reverse and shuffle traffics, while being slightly better in the 8 × 8 NoC. Compared to Odd-Even, LEF is generally less affected by the globally unfair resource allocation because some of the VCs in the Y dimension are XY-exclusive. The competition for these VCs is low since only XY packets can occupy them. Despite this, we will see in Sect. 4.2.2 that the effect of the globally unfair resource allocation on LEF is significant when the network is large.

4.2.2 Large-Scale NoCs

This section presents our results of evaluating LEF on large-scale NoCs. Figure 14 and Fig. 15 show the average packet latency and throughput results of two networks 64 × 64 and 128 × 64, respectively. In most cases, the trend of average packet latency is the same as in the middle-scale networks. LEF is still particularly effective when the network is asymmetric. However, the effect of the global unfairness of resource allocation makes LEF perform worse than in the middle-scale networks. For instance, under the uniform traffic, LEF is outperformed by O1TURN in the 64 × 64 NoC while the result in the 8 × 8 NoC is opposite. In terms of throughput, we can see that the large-scale NoCs are unstable at high loads when LEF is used. The reason is the same as that used to explain the behavior of Odd-Even in the 16 × 16 NoC in Sect. 4.2.1.

O1TURN is less affected by the globally unfair resource allocation than LEF because it uses the deadlock avoidance method in which the VCs are completely separated into two layers, one for XY packets and the other for YX packets. In this way, there is less competition for each VC in O1TURN than in LEF and the effect of the globally unfair resource allocation is smaller.
To solve the instability issue, we have implemented a new resource allocation policy which is based on age-based arbiters. Specifically, each packet is assigned an age counter which is initialized to 0 when the packet enters the network. This age counter is incremented as the packet traverses the network. In this way, a long-path packet will likely have a larger age counter compared to a short-path packet. Therefore, by prioritizing packets with larger age counters over those with smaller age counters, a long-path packet will likely have a higher priority than a short-path packet, and thus the problem of the presence of long-path packets with extremely high latencies can be solved.

Figures 16 (a), 16 (b), and 16 (c) show the evaluation results under the hotspot traffic pattern obtained in the case of using the resource allocation policy based on age-based arbiters. Compared with the results obtained in the case of using the resource allocation policy based on round-robin arbiters (Fig. 13 (b), 14 (b), and 15 (b)), we can see that the network is stable even when Odd-Even or LEF is used.

4.2.3 The Effectiveness of the Proposed Deadlock Avoidance Method

Figure 17 illustrates the effectiveness of the proposed deadlock avoidance method over the conventional method which...
Fig. 16  Throughput results under the hotspot traffic pattern when the resource allocation policy based on age-based arbiters is used.

Fig. 17  Comparison of the proposed deadlock avoidance method and the conventional method which is used by O1TURN [5]. In the graphs, O1TURN++ indicates O1TURN with the proposed deadlock avoidance method while LEF-- indicates LEF with the conventional deadlock avoidance method.

is used by O1TURN [5]. The network size and traffic pattern here are $16 \times 8$ and hotspot, respectively. As mentioned earlier, the proposed method can also be applied to O1TURN. In the graphs in Fig. 17, O1TURN++ indicates O1TURN with the proposed deadlock avoidance method while LEF-- indicates LEF with the conventional deadlock avoidance method. Thus, LEF and O1TURN++ use the same deadlock avoidance method. This is also the case for LEF-- and O1TURN.

Figure 17 shows that LEF and O1TURN++ deliver around 18.1% and 12%, respectively, higher throughput than LEF-- and O1TURN. Therefore, the proposed deadlock avoidance method is effective not only for LEF but also for O1TURN.

Another result obtained from Fig. 17 is that, when using the same deadlock avoidance method, LEF outperforms O1TURN. This indicates that the strategy of selecting the XY DOR and YX DOR in LEF is better than that in O1TURN.

4.2.4 Impact of The Number of Exclusive VCs on LEF’s Performance

In all experiments above, we use the Y-restricted deadlock avoidance method proposed in Sect. 3.2.1 for LEF. The number of XY-exclusive VCs in each Y channel ($k$) is set to 2. In this section, we vary the value of $k$ and evaluate how the network’s saturation throughput is affected. We also discuss how to set $k$ to maximize the network’s saturation throughput.

Fig. 18  $8 \times 8$ NoC: impact of the number of XY-exclusive VCs per Y channel ($k$) on LEF’s performance.

Fig. 19  $16 \times 8$ NoC: impact of the number of XY-exclusive VCs per Y channel ($k$) on LEF’s performance.

Fig. 20  $16 \times 16$ NoC: impact of the number of XY-exclusive VCs per Y channel ($k$) on LEF’s performance.

Figures 18, 19, and 20 show the results in three different network sizes: $8 \times 8$, $16 \times 8$, and $16 \times 4$. The parameters for this evaluation are the same as those described in Sect. 4.1.
Four traffic patterns are used: uniform, hotspot, bit reverse, and shuffle. The number of VCs per physical channel \( \nu \) is set to 4. We vary \( k \) from 1 to 3.

We can see that the results differ with respect to the network size and the offered traffic pattern. In the \( 8 \times 8 \) NoC, \( k = 1 \) delivers the highest throughput while \( k = 3 \) is the worst configuration across all traffic patterns. In the \( 16 \times 8 \) NoC, \( k = 3 \) narrows the gap with the other two configurations. The gap between \( k = 2 \) and \( k = 1 \) is also reduced. Under the two traffic patterns uniform and hotspot, \( k = 2 \) slightly outperforms \( k = 1 \). Finally, in the \( 16 \times 4 \) NoC, three configurations provide almost the same throughput under three traffic patterns uniform, hotspot, and shuffle. Only under the bit reverse traffic pattern, \( k = 3 \) performs worse than \( k = 1 \) and \( k = 2 \).

Recall that, in our deadlock avoidance method, YX packets are prohibited from using XY-exclusive VCs. In each Y channel, they can use only \( v - k \) VCs, which are non-exclusive, among \( v \) VCs. On the other hand, XY packets are allowed to use both XY-exclusive and non-exclusive VCs. The XY-exclusive VCs help to break potential deadlock situations that often occur when the network is highly congested. However, if \( k \) is set too large, that is, using too many XY-exclusive VCs, then the non-exclusive VCs may become much more congested than the XY-exclusive VCs in case the proportion of XY packets is significant. This highly uneven use of XY-exclusive and non-exclusive VCs makes the network get saturated at low throughput. Therefore, in order to maximize the network’s saturation throughput, it is important that a balance between using exclusive VCs for the deadlock avoidance purpose and distributing the load evenly among all VCs is achieved. This balance depends on the proportion of XY and YX packets which depends on the network size and the traffic pattern.

Among the three networks in our experiment, the \( 8 \times 8 \) has the highest proportion of YX paths while the \( 16 \times 4 \) has the lowest. The reason is that the larger the difference between the length of the X dimension and the length of the Y dimension, the smaller the proportion of source-destination pairs with the distance in the X dimension smaller than in the Y dimension (LEF selects YX paths for packets of these source-destination pairs). Therefore, \( k \) should be set smaller for the \( 8 \times 8 \) network than for the other two networks to increase the proportion of non-exclusive VCs that YX packets can use. We can see in Fig. 18 that the saturation throughput of the \( 8 \times 8 \) network when \( k = 3 \) is much lower than when \( k = 1 \) and \( k = 2 \) across all four traffic patterns. On the other hand, the saturation throughput of the \( 16 \times 8 \) network when \( k = 3 \) is noticeably lower than when \( k = 1 \) and \( k = 2 \) under only two traffic patterns. For the \( 16 \times 4 \) network, saturation throughput when \( k = 3 \) is lower than when \( k = 1 \) and \( k = 2 \) under only the bit reverse traffic pattern.

In general, if \( k \) is set too small, that is, using too few XY-exclusive VCs, then breaking potential deadlock situations at high loads might become too slow, which leads to lower network’s saturation throughput. However, since we do not restrict XY packets to use only XY-exclusive VCs, this issue might still be not significant in some cases even \( k \) is set to 1, the smallest possible value. We believe how small \( k \) should be set is not straightforward and requires careful simulations to decide. In our experiment, we observe that \( k = 2 \) slightly outperforms \( k = 1 \) in only several cases (Fig. 19(a) and Fig. 19(b)); in most cases, \( k = 1 \) is the best-performing configuration.

4.3 Hardware Overhead and Timing Analysis

Table 4 shows the hardware resource usages of three 4 × 4 NoCs that are based on XY, O1TURN, and LEF on a Virtex-7 28nm FPGA. The detailed parameters of these NoCs are described in Sect. 4.1. The LEF-based NoC requires 1.8% more LUTs and 5.2% more FFs than the XY-based NoC, and 0.4% more LUTs and 1.8% more FFs compared to the O1TURN-based NoC.

Our analysis shows that changing from XY routing to O1TURN and LEF does not change the critical paths of the NoC design which are in the VC allocator.

5. Conclusion

We proposed LEF, a new oblivious routing algorithm for 2D meshes, and an effective deadlock avoidance method for it. By routing a packet along the dimension in which

---

**Table 4** The hardware resource usages of three 4 × 4 NoCs that are based on XY, O1TURN, and LEF on a Virtex-7 28nm FPGA

| NoC   | LUTs | FFs  | BRAMs |
|-------|------|------|-------|
| XY    | 89,553 | 29,730 | 96 |
| O1TURN | 90,743 | 30,725 | 96 |
| LEF   | 91,146 | 31,275 | 96 |
it needs to traverse more hops first, LEF balances between distributing the load over both XY and YX paths and reducing the pressure on the channel buffers, which leads to better load balancing than other oblivious routing algorithms like O1TURN. This, together with the proposed deadlock avoidance method, enables LEF to achieve higher performance than the DOR algorithm and O1TURN and provide comparable performance to a complicated adaptive routing algorithm. Our evaluation results show that, in an 8 × 8 NoC, the throughput provided by LEF is from around 3% to around 10.2% higher than O1TURN under five different traffic patterns. In a 16 × 8 NoC, LEF delivers from around 4% up to around 64.5% higher throughput than O1TURN and is even better than the adaptive routing algorithm in some cases. Our results also show the effectiveness of the proposed deadlock avoidance method over the conventional method. The use of a fast FPGA-based NoC emulator enables us to examine LEF in NoCs with thousands of nodes in a practical time. Our evaluation results show that, to maximize the potential of LEF in such large-scale NoCs, it is necessary to pay more attention to the resource allocation policy.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Numbers JP16H02794 and JP17J09956.

References

[1] A. Sodani, R. Gramunt, J. Corbal, H.-S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y.-C. Liu, “Knights Landing: Second-Generation Intel Xeon Phi Product,” IEEE Micro, vol.36, no.2, pp.34–46, 2016.

[2] B.D. de Dinechin, R. Ayriyanac, P.-E. Beaucamps, P. Couvert, B. Ganne, P.G. de Massas, F. Jacquet, S. Jones, N.M. Chaisemartin, F. Riss, and T. Strude, “A Clustered Manycore Processor Architecture for Embedded and Accelerated Applications,” Proceedings of the 2013 IEEE High Performance Extreme Computing Conference (HPEC), pp.1–6, 2013.

[3] J. Balkind, M. McKeown, Y. Fu, T. Nguyen, Y. Zhou, A. Lavrov, M. Shahrad, A. Fuchs, S. Payne, X. Liang, M. Matl, and D. Wentzlaff, “OpenPiton: An Open Source Manycore Research Framework,” Proceedings of the 21st International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp.217–232, 2016.

[4] B. Bohnenstiehl, A. Stillmaker, J.J. Pimentel, T. Andreas, B. Liu, A.T. Tran, E. Adeagbo, and B.M. Baas, “KiloCore: A 32-nm 1000-Processor Computational Array,” IEEE J. Solid-State Circuits, vol.52, no.4, pp.891–902, 2017.

[5] D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi, “Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks,” Proceedings of the 32nd Annual International Symposium on Computer Architecture (ISCA), pp.432–443, 2005.

[6] S. Kumar, Y. Sabharwal, R. Garg, and P. Heidelberger, “Optimization of All-to-All Communication on the Blue Gene/L Supercomputer,” Proceedings of the 37th International Conference on Parallel Processing (ICPP), pp.320–329, 2008.

[7] D. Chen, N.A. Easley, P. Heidelberger, R.M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D.L. Satterfield, B. Steinmacher-Burrow, and J.J. Parker, “The IBM Blue Gene/Q Interconnection Network and Message Unit,” Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp.26:1–26:10, 2011.

[8] D. Chen, N. Easley, P. Heidelberger, S. Kumar, A. Mamidala, F. Petrini, R. Senger, Y. Sugawara, R. Walkup, B. Steinmacher-Burrow, A. Choudhury, Y. Sabharwal, S. Singhal, and J.J. Parker, “Looking Under the Hood of the IBM Blue Gene/Q Network,” Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC), pp.69:1–69:12, 2012.

[9] N. Jiang, D.U. Becker, G. Michelogiannakis, J. Balfour, B. Towlès, D.E. Shaw, J. Kim, and W.J. Dally, “A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator,” Proceedings of the 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp.86–96, 2013.

[10] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti, “Cycle-Accurate Network on Chip Simulation with Nixom,” ACM Transactions on Modeling and Computer Simulation (TOMACS), vol.27, no.1, pp.4:1–4:25, 2016.

[11] T.V. Chu, S. Sato, and K. Kise, “Fast and Cycle-Accurate Emulation of Large-Scale Networks-on-Chip Using a Single FPGA,” ACM Transactions on Reconfigurable Technology and Systems (TRETS), vol.10, no.4, pp.27:1–27:27, 2017.

[12] R. Sasakawa and K. Kise, “LEF: Long Edge First Routing for Two-dimensional Mesh Network on Chip,” Proceedings of the 6th International Workshop on Network on Chip Architectures (NoCArc), pp.5–10, 2013.

[13] T.V. Chu, M. Kang, S. FA, and K. Kise, “Enhanced Long Edge First Routing Algorithm and Evaluation in Large-Scale Networks-on-Chip,” Proceedings of the 11th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoc), pp.83–90, 2017.

[14] G.-M. Chiu, “The Odd-Even Turn Model for Adaptive Routing,” IEEE Trans. Parallel Distrib. Syst., vol.11, no.7, pp.729–738, 2000.

[15] L.G. Valiant and G.J. Brebner, “Universal Schemes for Parallel Communication,” Proceedings of the 13th Annual ACM Symposium on Theory of Computing (STOC), pp.263–277, 1981.

[16] T. Nesson and S.L. Johnsson, “ROMM Routing on Mesh and Torus Networks,” Proceedings of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp.275–287, 1995.

[17] A.M. Shafiiee, M. Montazeri, and M. Nikdast, “An Innovational Intermittent Algorithm in Networks-On-Chip (NOC),” International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol.2, no.9, pp.2907–2909, 2008.

[18] B. Towlès and W.J. Dally, “Worst-Case Traffic for Oblivious Routing Functions,” IEEE Computer Architecture Letters, vol.1, no.1, p.4, 2002.

[19] M.A. Kinsky, M.H. Cho, T. Wen, E. Suh, M. van Dijk, and S. Devadas, “Application-Aware Deadlock-free Oblivious Routing,” Proceedings of the 36th Annual International Symposium on Computer Architecture (ISCA), pp.208–219, 2009.

[20] W.J. Dally and C.L. Seitz, “Deadlock-Free Message Routing in Multiprocessor Interconnection Networks,” IEEE Transactions on Computers (TC), vol.C-36, no.5, pp.547–553, 1987.

[21] J. Duato, “A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks,” IEEE Transactions on Parallel and Distributed Systems (TPDS), vol.4, no.12, pp.1320–1331, 1993.

[22] C. Bienia, Benchmarking Modern Multiprocessors, Ph.D. thesis, Princeton University, 2011.

[23] J. Hestness, B. Grot, and S.W. Keckler, “Netrace: Dependency-driven Trace-based Network-on-chip Simulation,” Proceedings of the 3rd International Workshop on Network on Chip Architectures (NoCArc), pp.31–36, 2010.

[24] S. Ma, N.E. Jerger, and Z. Wang, “Whole Packet Forwarding: Efficient Design of Fully Adaptive Routing Algorithms for Networks-on-chip,” Proceedings of the 18th IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp.1–12, 2012.

[25] W.J. Dally and B. Towlès, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers, 2003.
[26] W.J. Dally, “Virtual-Channel Flow Control,” IEEE Transactions on Parallel and Distributed Systems (TPDS), vol.3, no.2, pp.194–205, 1992.

[27] S. Vigna, “Further scramblings of Marsaglia’s xorshift generators,” Journal of Computational and Applied Mathematics, vol.315, pp.175–181, 2017.

[28] N. McKeown, “The iSLIP Scheduling Algorithm for Input-queued Switches,” IEEE/ACM Trans. Netw. (TON), vol.7, no.2, pp.188–201, 1999.

[29] J. Hu and R. Marculescu, “DyAD: Smart Routing for Networks-on-Chip,” Proceedings of the 41st Annual Design Automation Conference (DAC), pp.260–263, 2004.

Thiem Van Chu received the Doctoral Degree in Computer Science from Tokyo Institute of Technology in 2018. He is currently an assistant professor at the School of Information Science, Japan Advanced Institute of Science and Technology. He is interested in computer architecture, systems, and FPGA-related research. He is a member of ACM, IEEE, and IPSJ.

Kenji Kise received the B.E. degree from Nagoya University in 1995, the M.E. degree and the Ph.D. degree in information engineering from the University of Tokyo in 1997 and 2000, respectively. He is currently an associate professor of the School of Computing, Tokyo Institute of Technology. His research interests include computer architecture and parallel processing. He is a member of ACM, IEEE, IEICE, and IPSJ.