MUA-Router: Maximizing the Utility-of-Allocation for On-chip Pipelining Routers

CUNLU LI, DEZUN DONG, and XIANGKE LIAO, National University of Defense Technology, China

As an important pipeline stage in the router of Network-on-Chips, switch allocation assigns output ports to input ports and allows flits to transit through the switch without conflicts. Previous work designed efficient switch allocation strategies by maximizing the matching efficiency in time series. However, those works neglected the interaction between different router pipeline stages. In this article, we propose the concept of Utility-of-Allocation (UoA) to indicate the quality of allocation to be practically used in on-chip routers. We demonstrate that router pipelines can interact with each other, and the UoA can be maximized if the interaction between router pipelines is taken into consideration. Based on these observations, a novel class of routers, MUA-Router, is proposed to maximize the UoA through the collaborative design (co-design) between router pipelines. MUA-Router achieves this goal in two ways and accordingly implements two novel instance router architectures. In the first, MUA-Router improves the UoA by mitigating the impact of endpoint congestion in the switch allocation, and thus Eca-Router is proposed. Eca-Router achieves an endpoint-congestion-aware switch allocation through the co-design between routing computation and switch allocation. Based on Eca-Router, CoD-Router is proposed to feed back switch allocation information to routing computation stage to provide switch allocator with more conflict-free requests. Through the co-design between pipelines, MUA-Router significantly improves the efficiency of switch allocation and the performance of the entire network. Evaluation results show that our design can achieve significant performance improvement with moderate overheads.

CCS Concepts: • **Computer systems organization → Interconnection architectures**; • **Hardware →** *Networking hardware;*

Additional Key Words and Phrases: Network on Chip, switch allocation, endpoint congestion, co-design

This is an extension of a conference paper. This article is extended from our previous conference publication [19]. The work in Reference [19] proposes Eca-Router, which is designed to relieve the impact of endpoint congestion through the co-design between SA and RC pipelines. Based on the conference paper, we give an example to present the benefit of this design. We also discuss different design methods of Eca-Router and present the detailed datapath of Eca-Router. Going beyond the conference paper, we propose the concept of UoA and a new class of routers named MUA-Router. We classify Eca-Router as an instance of MUA-Router and propose another instance named CoD-Router. Based on Eca-Router, CoD-Router further feeds back allocation results of SA and VA stages to the RC stage, so that the RC stage can provide more conflict-free requests to improve the matching efficiency of SA.

This work was supported in part by Excellent Youth Foundation of Hunan Province under Grant No. 2021JJ10050, NSFC under Grant No. 62002368 and National Postdoctoral Program for Innovative Talents under Grant No. BX20190091.

Authors' address: C. Li, D. Dong (corresponding author), and X. Liao, National University of Defense Technology, 109 Deya Rd., Changsha, Hunan 410073, China; emails: {cunluli, dong, xkliao}@nudt.edu.cn.

1544-3566/2022/05-ART33 $15.00
https://doi.org/10.1145/3519027

ACM Reference format:
Cunlu Li, Dezun Dong, and Xiangke Liao. 2022. MUA-Router: Maximizing the Utility-of-Allocation for On-chip Pipelining Routers. *ACM Trans. Arch. Code Optim.* 19, 3, Article 33 (May 2022), 23 pages.
https://doi.org/10.1145/3519027

1 INTRODUCTION

As **network-on-chips (NoCs)** technology has become fundamental to interconnect cores in manycore architecture due to abundant on-chip resources [18], the design of efficient NoCs remains a challenge, as it interconnects hundreds and even thousands of cores [30]. Router microarchitecture [6], which largely determines the network latency, has been involved in many prior works to implement efficient NoCs. The **Input-queued (IQ)** router [6, 27] is the most popular router architecture of modern NoCs, which usually contain five pipeline stages: **Routing Computation (RC)**, **Virtual-Channel (VC)** allocation, **Switch Allocation (SA),** switch traversal, and link traversal [6]. Most optimization work just focuses on one pipeline stage of the IQ router, such as RC and SA. Among all the pipeline stages, SA is important, because it determines the throughput of a router by assigning output ports to input ports and ensures conflict-free[1] transmission of flits during switch traversal. Therefore, many optimization work on IQ routers has been carried out by improving the matching efficiency of SA [1, 3, 22, 23, 25, 29]. Current proposed SA strategies mostly strive to maximize the matching efficiency by considering the requests as time series [3, 25] or adding dedicated circuit to bypass the packets [1, 29]. However, most prior works do not consider to maximize the **Utility-of-Allocation (UoA)** while designing switch allocation strategies.

We define UoA as the ability of a router to achieve or enhance additional functions during the allocation. UoA can be improved by utilizing the results of the allocation into other pipeline stages to enhance router functions such as adaptive routing selection and congestion control. Previous work aiming to maximize matching is for the switch allocation stage and only focuses on the efficiency of the allocation. Most existing work makes allocation decisions only based on SA request information and rarely considers the information from other pipeline stages or to improve the performance of other pipelines. If considered, then UoA can be improved to deliver higher matching efficiency in SA to further improve the performance of NoCs. UoA is associated with the interaction between router pipelines. Therefore, *collaborative design* (co-design) between router pipelines should be performed to improve the UoA in SA. In particular, UoA can be improved through the co-design between router pipelines by the following methods.

First, congestion information in the RC stage can be utilized in SA to mitigate the impact of endpoint congestion. On-chip congestions include network congestion and endpoint congestion. Network congestion occurs if the network or a network channel cannot transmit the packet in time. Adaptive routing, aiming to increase routing adaptiveness, can help to alleviate the impact of network congestion by routing packets around the congested network channel [5, 8, 9, 21, 33]. Recent work [10] strives to alleviate the influence of endpoint congestion by minimizing the resource used by **Endpoint-Congestion-Causing (ECC)** packets[2] in the RC stage. However, resource limitation can be weakened by traditional switch allocators, since they treat ECC requests and normal requests indiscriminately. Moreover, matching efficiency can be degraded by endpoint congestion

[1]The conflict refers to the situation where there are multiple requests for the same port of the switch during the switch allocation, including input conflicts and output conflicts.
[2]ECC packets refer to packets that can cause endpoint congestion, that is, multiple packets with the same destination. Other packets, including randomly issued packets and packets that can cause network congestion but with different destinations, are not ECC packets.

due to the resulting tree saturation [6] and the congestion in output ports, which can reduce the number of requests provided to the switch allocator. Therefore, the impact of endpoint congestion should be taken into consideration in the SA.

Second, SA results can be utilized to optimize other pipeline stages to provide more conflict-free requests to the switch allocator. Specifically, this goal can be achieved through the co-design of RC and SA. Traditional routing algorithms are usually designed with the purpose of alleviating congestions in the network, without considering to improve the performance of other pipelines such as SA. In fact, the SA matching efficiency can be directly affected by the result of RC. In detail, if RC process tends to choose an output port competed by other SA requests, then conflicts will occur in that output port during the SA, reducing SA performance as a result. Therefore, SA (and other pipeline stages) result should be considered during the RC process, so that the RC result does not introduce unnecessary conflicts in SA.

In this article, we propose a new class of routers aims to maximize the UoA, and we call such routers MUA-Routers. We use the two methods mentioned above to improve the UoA and propose two instance routers for MUA-Routers accordingly. We first propose Eca-Router, within which an endpoint-congestion-aware SA strategy is implemented. In Eca-Router, we propose to collect the information of requests that contribute to the endpoint congestion during the RC stage. Then, these requests will be given lower priority during SA. Eca-Router predicts the endpoint congestion and attempts to minimize the impact of it in the SA, which in turn achieves higher SA performance in long periods of time. Based on Eca-Router, CoD-Router is proposed to make full use of the co-design between pipeline stages to improve the UoA. CoD-Router moves one step forward by adding contention information collected in **VC Allocation (VA)** and SA to the RC stage. CoD-Router records the number of requests for each port in SA and passes this information to the RC unit. When selecting output ports in the RC stage, CoD-Router gives lower priority to the port with more SA conflicts. In other words, CoD-Router tends to select an output port that is free from or has less contention in the SA. Such a design can provide more conflict-free requests to the switch allocator and greatly improve the matching efficiency of the SA. We carefully design the SA processes of our design to relieve the impact of unfair allocation where requests with low priority would have no chance to be allocated for a long period of time. It should be noted that our design is based on a two-dimensional (2D) mesh topology where the size of a NoC flit is often very wide and is not necessarily efficient for 3D ones. The implementation of our design is light-weighted, and efficient latency reduction can be achieved with moderate hardware overhead.

The rest of this article is organized as follows. In Section 2, related work is introduced, including the background of SA and endpoint congestion in NoCs. We give a scheduling instance in Section 3 to demonstrate the benefits of co-design between SA and RC in alleviating endpoint congestion. An overview of different SA strategies is also presented to show the design purpose of our design. In Section 4, a detailed architecture of our design is presented. An SA example is also provided to show that SA processes in our design outperform previous allocation strategies. Evaluation methodology and the results are presented in Section 5. Finally, we conclude our article in Section 6.

2 RELATED WORK

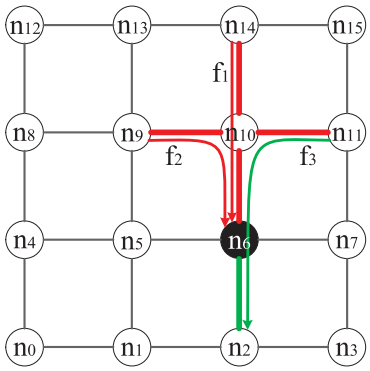
For a given routing algorithm, SA is an important pipeline stage in on-chip routers, because it effects packet latency in NoCs. SA allows packets to transit through the switch by assigning output ports to input ports. The purpose of SA is to move packets from input ports to output ports as soon as possible without introducing any conflicts during this process. The SA is done per flit by allocating a time slot for each flit to pass through the switch. To achieve efficient SA, most traditional strategies, such as iSLIP, wavefront, and augmenting paths [23, 27], are proposed to

maximize the matching number in a single SA cycle. However, most of those works are too complex for NoCs or need multiple cycles to achieve desirable performance. To achieve efficient SA in NoCs, previous work tends to take advantage of time-series allocation. For example, Pseudo-circuit [1] and Packet Chaining [25] introduce the previous allocation information into the current SA decision. TS-Router [3] further adds the future request information to the current allocation. Some other works also attempt to improve the matching efficiency of SA by adding additional hardware. Virtual input crossbar [29] makes it possible to allow more than one input VC in an input port to transit flits through the switch in a single cycle. RoB-Router [20] organizes the input buffer as reorder buffer to enable packet-scheduling during the SA. References [13, 26] explore dividing the SA stage into some substages and pipelining these substages to avoid damage to router frequency.

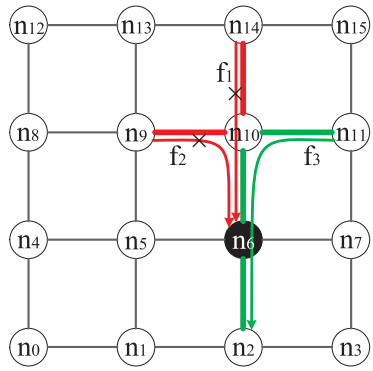
However, existing work rarely improve the performance of SA by improving the UoA or taking into the consideration of the interaction between different pipelines. A significant benefit of the co-design is to alleviate the endpoint congestion. Plenty of short packets in NoCs can stress the impact of endpoint congestion. In NoCs, most packets only carry a memory address or control instruction, and these packets can be encoded as short or even single-flit packets. Most of these short packets share the same destination to form the endpoint congestion. Endpoint congestion can negatively influence the matching efficiency of SA due to contention in output ports and the consequently reduction in the number of SA requests over a long period of time. To the best of our knowledge, so far there has no previous work to apply endpoint congestion information to the SA process.

Early work has been proposed to design efficient traffic management strategies to mitigate the impact of endpoint congestion. Congestion notification [14, 16, 24, 31] is a notable hardware approach utilized to relieve the impact of endpoint congestion. **Explicit Congestion Notification (ECN)** protocol [31], as an example, reduces the traffic injection rate based on the congestion signal detected in the network. However, the response time of ECN is slow due to the time needed for detecting and throttling the congestion-causing traffic. Moreover, the throttling parameters of ECN highly impact its performance. **Speculative Reservation Protocol (SRP)** [14] uses a reservation handshake between the source node and the destination node to avoid overloading in network endpoints. Some SRP-based researches also use multiple resource reservation protocols in oversubscribed network [24]. Reference [16] proposes a short-message-oriented reservation protocol to addresses possible endpoint congestion caused by short messages. Reference [17] proposes to utilize the contention information within the intermediate routers to identify endpoint congestion at the endpoint node. Reference [34] attempts to achieve efficient reservation for short messages by chaining packets as a flexible reservation granularity. However, those congestion-management-based strategies are too complex to be adopted in NoCs.

Current work tends to alleviate the impact of endpoint congestion through the optimization of adaptive routing [10]. Adaptive routing effectively alleviate the performance loss caused by network congestion [5, 8, 9, 21, 33] by providing more selections on the path/port to bypass the congested link. While adaptive routing can address network congestion necessarily and efficiently improve the network performance under adversarial traffic pattern conditions, it cannot address endpoint congestion and can be even worse when combined with congestion management strategies [17]. CBCM [17] first addresses that adaptive routing can be problematic with congestion management when faced with endpoint congestion, and it is necessary to differentiate between endpoint congestion and network congestion. Footprint [10] is proposed to minimize the saturation tree [6] caused by endpoint congestion by limiting hot-spot packets waiting on the footprint VC, which alleviates the **Head-of-Line (HoL)** blocking [6] as a result. However, the mitigation effect of routing algorithms on the impact of endpoint congestion would be weakened in SA process. This is because the switch allocator can give higher priorities to ECC requests rather than normal



(a) In traditional SA, ECC requests and other requests have the same probability to be allocated, which makes the links more easily to be congested by hotspot traffic.



(b) In the SA of Eca-Router, allocation of ECC requests can be limited to enable the allocation of other requests, reducing the congestion probability of network links.

Fig. 1. The benefits of Eca-Router. By avoiding the allocation of ECC requests in the SA, the number of congested links (red links) can be minimized, and uniform packets can be transferred through free links (green links).

requests during the SA process. Therefore, the impact of endpoint congestion should be relieved in the SA process, and fortunately, this goal can be achieved through the co-design between router pipelines.

3 MOTIVATION

The purpose of Eca-Router is to limit the allocation of ECC requests during the SA. In theory, endpoint congestion occurs when packets with the same destination reach the hotspot node. Therefore, if multiple packets traveling in the network share the same destination, then these packets can be regarded as ECC packets. Consequently, in Eca-Router, if a router detects that multiple arriving packets have the same destination, then these packets will be regarded as ECC packets.

In SA, limiting the allocation of ECC requests can alleviate the congestion of the output port, resulting in more free links in the network in long time series. Figure 1 presents the benefit of adopting such strategy in a 4×4 2D mesh network. In Figure 1, the red link denotes that this link is congested and the green indicates that there is no congestion in this link. As shown in the figure, permutation traffic is formed with the following three traffic flows:

$$\{f_1, f_2, f_3\} = \{n_{14} \rightarrow n_6,\ n_9 \rightarrow n_6,\ n_{11} \rightarrow n_2\}.$$

With **Dimension-Order Routing (DOR)**, if a traditional SA strategy is adopted in the network, then n_6 will be oversubscribed by f_1 and f_2 to cause endpoint congestion. Moreover, the endpoint congestion in n_6 also propagates backpressure all the way to the source nodes (n_9, n_{11} and n_{14}), congesting four links to form a congestion tree. As can be seen in Figure 1(a), DOR cannot necessarily address the endpoint congestion in this situation. However, as shown in Figure 1(b), if the ECC request can be restricted during the SA process, then the number of congested links will be reduced. As has been presented in Figure 1(b), the RC process in the router within node n_{10} chooses the same output port (the south port) for f_1, f_2, and f_3, providing three requests to the switch allocator, including two ECC requests and one uniform request. In the router within node n_{10}, by limiting the allocation of ECC requests during the SA process, the congestion state of the

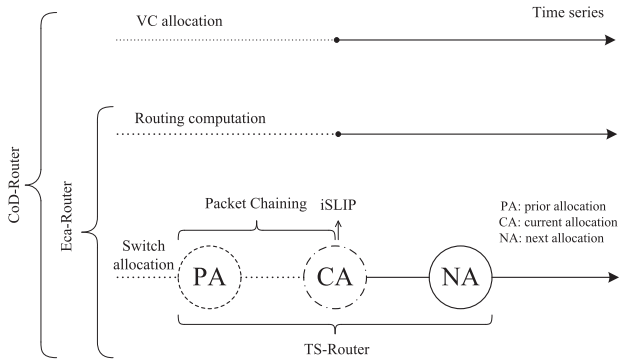


Fig. 2. An overview of different kinds of MUA-Routers and the time-series-based SA strategies. Eca-Router achieves more performance improvement in SA by introducing the endpoint congestion information from the RC pipeline stage. CoD-Router further improves the performance of SA by taking into account the interaction among RC, SA, and VA.

south output port will be alleviated. Therefore, links $n_{11} \rightarrow n_{10}$ and $n_{10} \rightarrow n_6$ will not be congested, and f_3 can be transferred to node n_2 without congestion.

Although mitigating the impact of endpoint congestion can be done in the RC stage, it is also necessary to address this problem in the SA process. Eca-Router achieves this goal by detecting endpoint congestion at intermediate routers rather than the endpoint node. With this strategy, endpoint congestion can be detected earlier than adopting RC-based optimizations. This strategy is effective, because endpoint congestion always causes congestion in the intermediate router. As shown in Figure 1(a), although endpoint congestion occurs on node n_6, the contention can be detected in the SA process of the router within node n_{10}. Therefore, we can identify endpoint congestion and improve NoCs performance by optimizing the SA process in intermediate routers. In detail, contention requests can be detected in the SA stage while the corresponding destinations of these requests can be detected in the RC stage of the intermediate router. This information can be used in SA to identify endpoint congestion and relieve its impact on network performance. Unlike Eca-Router, other metrics (such as congestion management and adaptive routing) can estimate endpoint congestion dynamically but only begin to work when the congestion has begun to damage network performance. Differently, Eca-Router detects endpoint congestion at intermediate routers instead of endpoint nodes, providing a new solution for endpoint congestion with a faster response. It should be noted that our design only performs local observation based on the information of the router in which SA is done. However, remote congestion cannot be detected by our design. For example, in Figure 1(a), if the router within node n_2 is congested, then the router within node n_{10} is unable to detect the congestion and thus cannot make the right decision of giving the priority to f_1 or f_2. This problem can be solved by storing congestion status of the remote router in the current router. However, since the benefits of obtaining the congestion status of the remote router cannot offset the hardware overhead, our design does not consider introducing the congestion status of the remote router to assist the current router's decision-making.

We classify our designs and the existing SA strategies in time and space dimensions, and the spectrum of these strategies has been presented in Figure 2. Existing SA strategies strive to improve SA performance by scaling in time series. As shown in Figure 2, TS-Router not only adds the information of current arrival requests but also considers the future request information during the SA process, achieving higher performance in long time series. It should be noted that the TS-Router only uses the request information of the next cycle, and information of more future requests

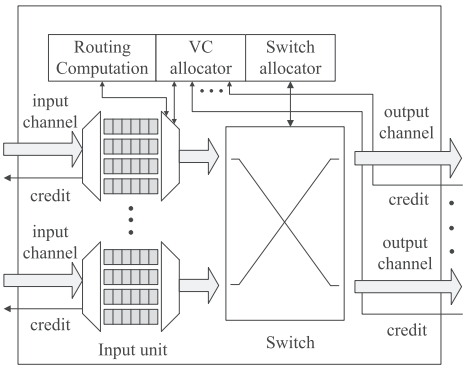


Fig. 3. Baseline router architecture of Eca-Router and CoD-Router.

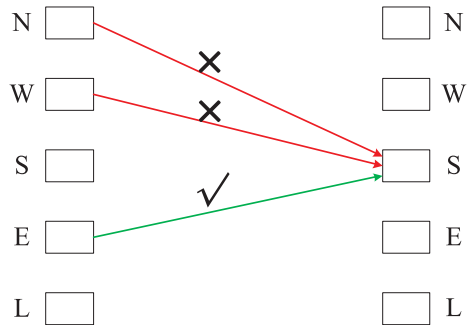


Fig. 4. An alternative switch allocation strategy for Eca-Router.

is hard to obtain due to hardware overheads. Therefore, our design turns to expand SA in the spatial dimension through the co-design of different pipeline stages. Eca-Router adds the endpoint congestion information from the RC stage to the SA stage to improve the matching efficiency through the co-design of RC and SA. In fact, Eca-Router also expends time-series-based SA [3]. Specifically, Eca-Router delays the congestion of the output port by limiting the allocation of ECC requests. As a result, more SA requests can be provided to the switch allocator in long time series, thereby achieving more matchings during SA. Compared with Eca-Router, CoD-Router strives to improve the matching efficiency through the co-design between more pipelines. CoD-Router takes more pipeline stages into account to provide more requests to the switch allocator, further improving the SA performance.

4 DETAILED ROUTER ARCHITECTURE

4.1 Baseline Router Architecture

The baseline router of our design is a state-of-the-art router [28], as shown in Figure 3. The baseline router contains five pipeline stages: RC, VA, SA, Xbar/Switch Traversal, and Link Traversal. Traditional VC flow control [4] is adopted, and each input port has multiple VCs. Each VC in the input port is associated with a private buffer, and packets can be held in this buffer. The total buffer of all the VCs in an input port constitutes the whole buffer of this input port, and the bandwidth of the input port can be shared by all the VCs in the input port. A VC is actually a FIFO queue to buffer flits. By observing how a packet gets through the router, we can have a better understanding of the baseline router architecture. When a packet arrives at the input port, it will be written into the input buffer (the VC to be cached in has been assigned in the last-hop router) first, and then RC will be performed once the packet reaches at the head of the VC. Then, the VC allocator will allocate an available VC in the input buffer of the next-hop router. After that, switch allocator will allocate a switch path from the input port to the output port for this packet. The packet is then driven onto the output channel link and destined to the next-hop router after traversal through the switch. In our design, each pipeline stage only takes one cycle. This architecture is the baseline of Eca-Router and CoD-Router as well as Packet Chaining and TS-Router.

4.2 SA Strategy in Eca-Router

4.2.1 Alternative SA Strategy. Eca-Router relieves the impact of endpoint congestion by limiting the allocation of ECC requests during SA. To implement Eca-Router, a trivial and alternative

strategy is giving the lowest priority to the ECC request and allowing ECC requests to be allocated only when there is no other requests. Figure 4 shows an example of the SA process using this strategy. With this strategy, ECC requests (marked in red) can only be allocated when there is no uniform requests (marked in green) in the current cycle. As shown in Figure 4, there are three requests competing for output S, among which requests $N \rightarrow S$ and $W \rightarrow S$ are ECC requests and the request $E \rightarrow S$ is a uniform random request. When adopting this strategy in the SA, switch allocator will prioritize the uniform request ($E \rightarrow S$), and thus ECC requests cannot be allocated.

This strategy can relieve the impact of endpoint congestion but introduces new problems. First, fairness cannot be guaranteed during the SA process. With this strategy, if there always exists a uniform request, then the ECC request will not be successfully allocated for a long time, causing ECC requests to be starved. This phenomenon can lead to a degradation in matching efficiency and performance degradation can be more serious in routers near the destination of ECC requests. This is because excessive restrictions on ECC requests can reduce the number of packets arriving at the endpoint node, thereby reducing the number of requests arriving at the router connected to the endpoint. Moreover, such strategy can improperly identify some temporary (or instantaneous) congestion as endpoint congestion to limit the allocation of the corresponding requests, resulting in low matchings efficiency in SA.

4.2.2 Fairness-guaranteed SA Strategy. To ensure fairness and improve the matching efficiency of SA, a new SA strategy is proposed in Eca-Router. With this strategy, ECC requests can be allocated even if there exist other requests. In our design, a new process, **Request Selection (RS)** is added before the SA to deal with the ECC request.[3] The RS process detects all the ECC requests and randomly selects one or several of these requests. The number of selected ECC requests does not depend on the network configuration or the traffic. In fact, the number of selected ECC requests is equal to the number of different destinations of all ECC requests. To identify an ECC request, the RS process collects and compares the destination of each request; if multiple requests have the same destination, then they will be judged as ECC requests. It should be noted that the RS process compares the destination of each request rather than the destination of each request with same outport. This is because Eca-Router strives to identify all the ECC requests to relieve the impact of endpoint congestion in time series. After finding all the ECC requests, the RS process will randomly select some from all the ECC requests, and the selected ECC requests can be passed to the switch allocator to be regarded as normal requests, while other unselected ECC requests will be discarded. In addition, after the RS process, all non-ECC requests will also be added as normal requests to the switch allocator. Note that at least one ECC request (if exists) will be selected in the RS process. In addition, RS process could select two or more ECC requests in a single cycle. For example, if there exist different destinations (endpoints) in ECC requests, then the RS process will select more than one ECC request, and the selected ECC requests number is equal to the number of destinations. After RS process, normal SA process will be executed.

Figure 5 presents an instance to detail the RS and the SA processes of Eca-Router. In this instance, an output-first separable allocator [6] is adopted for the SA process. As shown in Figure 5(a), the RS process is performed before SA. There are three ECC requests, namely $N \rightarrow S$, $W \rightarrow S$, and $W \rightarrow E$, and other requests are uniform random requests. In the figure, there two requests from inport E at the same time. This is because each VC is one input candidate for switch, and there can be multiple VCs in an input port simultaneously applying for traversal the switch. RS process randomly select the ECC request $N \rightarrow S$ and adds it as a normal request to the switch allocator, along with all other

[3]In fact, the RS process can be executed in parallel with router pipelines. But for a better description, here we take RC as an independent pipeline stage.

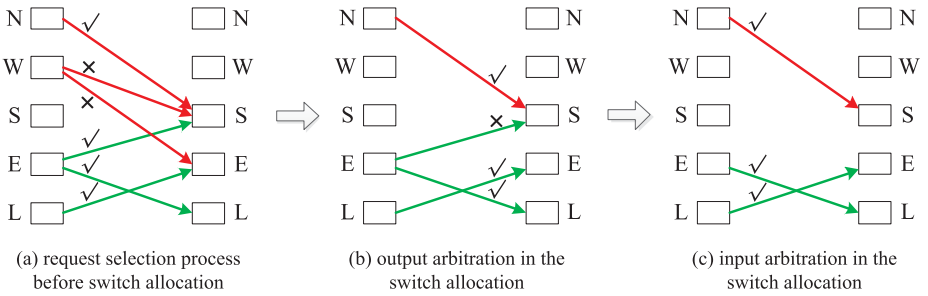


Fig. 5. An instance of the SA process of Eca-Router. Before SA, the RS process will be performed to select a request from all ECC requests (marked in red), and this selected ECC request and all uniform requests (marked in green) will be used as input requests of the switch allocator. The switch allocator of Eca-Router is a traditional output-first separable allocator, and the selected ECC request and other uniform requests have the equal probability to be allocated during the SA. Such design restricts the allocation of ECC requests while ensuring the fairness by giving ECC requests a certain probability to be allocated.

uniform requests. During the output arbitration of SA, as shown in Figure 5(b), the ECC request $N \rightarrow S$ is selected, and the uniform request $E \rightarrow S$ is removed, because this request is in conflict with the ECC request $N \rightarrow S$ on output port S. Finally, after the input arbitration of SA, three requests are selected by the switch allocator, including an ECC request. As can be observed from the figure, by limiting allocation of ECC requests, Eca-Router can relieve the impact of endpoint congestion while ensuring the fairness in SA.

4.3 SA Strategy in CoD-Router

Eca-Router relieves the impact of endpoint congestion by adding information collected in the RC stage to the SA stage. To further improve the performance of SA, CoD-Router optimizes the SA process through the co-design of more router pipelines. The purpose of CoD-Router is to improve the matching number in SA by providing more conflict-free requests. This purpose can be achieved in the RC process of CoD-Router, using the SA request information obtained from the SA stage and the VA stage.

Figure 6 shows an example of the RC and SA processes in CoD-Router, where the corresponding processes of the baseline router are also presented for comparison. The baseline router also adopts a traditional routing strategy but cannot use information from other pipelines to produce routing results more suitable for SA. In the RC process of CoD-Router, SA request information can be used to select an output port that will not be competed by future SA requests in the next cycle. As shown in Figure 6, in the RC stage, the packet in input port S has two candidate output ports N and E, and these two output ports have the same probability to be selected as the final output port. If an output port, say, N, is selected, then a new SA request $S \rightarrow N$ will be provided to the switch allocator in the next cycle (in this case, we adopt speculative SA [28] where SA is performed in parallel with VA, so that there is only one-cycle delay between RC and SA). To make the RC stage generate fewer conflicts in the next-cycle SA, some SA information needs to be collected and utilized. This information includes the SA request not allocated in the current cycle ($W \rightarrow S$, $W \rightarrow E$, and $E \rightarrow S$ in the RC stage in Figure 6) and the new generated SA request that will arrive in the next cycle ($L \rightarrow E$ in the RC stage in Figure 6). These two kinds of requests will reach the SA stage in the next cycle. Traditional adaptive routing algorithms do not take advantage of this information when making routing decisions. Without this information, as shown in Figure 6(a), a traditional routing strategy usually randomly select a candidate output port (say, $S \rightarrow E$). If the

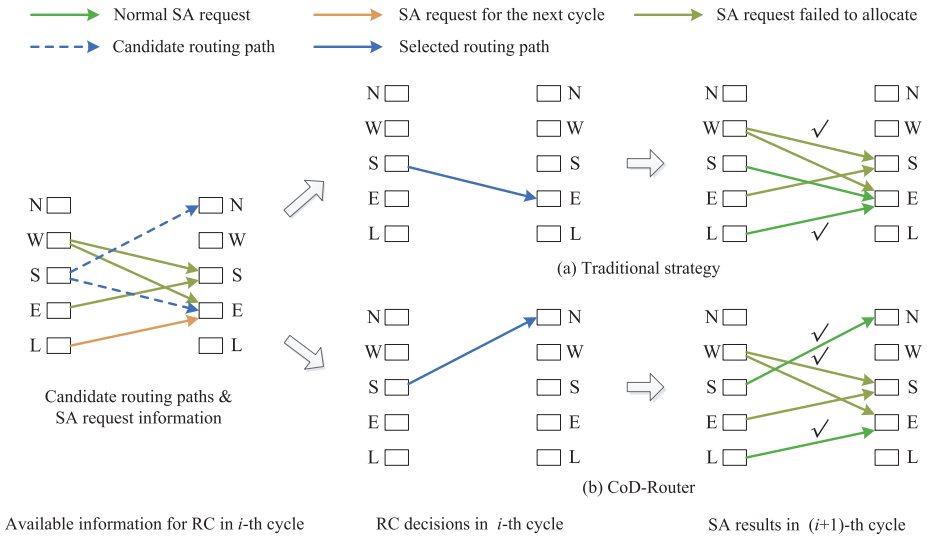


Fig. 6. Pipeline comparisons of (a) the baseline router that adopts traditional RC strategy and (b) CoD-Router. Before RC, some information will be collected, including unallocated SA requests in current cycle and the SA requests to arrive in the next cycle (these requests will be used as requests for the next SA cycle). Based on this information, CoD-Router can select a routing path with the least competition so that the SA process can have more non-conflicting requests to achieve more matchings in the next cycle. However, traditional RC strategy do not use this information, thus the selected routing path can compete with future SA requests to reduce matching efficiency.

selected output port is in conflict with future SA requests, then fewer conflict-free requests will be provided to the switch allocator in next cycle, and thus fewer matchings can be achieved in the next-cycle SA. However, if considering SA request information during the RC process, then more matchings can be achieved in the SA due to more conflict-free requests provided to the switch allocator. As shown in Figure 6(b), by adopting SA request information, CoD-Router selects the routing path $S \rightarrow N$ rather than $S \rightarrow E$ to not introduce new conflicts into the SA in the next cycle. With such a routing decision, CoD-Router achieves more matchings in the SA in the next cycle.

Figure 7 shows a comparison of the RC and SA processes for TS-Router, Eca-Router, and CoD-Router. In Figure 7, a request matrix [3] is used to indicate the relationship between input ports and output ports. A thin circle in the figure refers to a uniform request, and the thick circle represents the ECC request. In the ith request matrix, there are four requests, namely request (2,3), (3,3), (4,3), and (3,2); among them, (2,3) and (3,3) are ECC requests, and others are uniform requests. We assume sing-flit packets in this case, and requests provided to the switch allocator are consistent across each cycle. TS-Router achieves better performance by adding future SA request information into the current SA process, but it can be problematic when faced with endpoint congestion. TS-Router prefers to choose requests that are in conflict with next-cycle requests. However, if the selected request is an ECC request, then conflicts can be introduced in the next cycle as well. As shown in Figure 7(a), TS-Router chooses ECC request (3,3), because it has the highest priority (this request is in conflict with the most requests arriving in the next cycle). However, since this request is an ECC request and it is in conflict with all other requests, only this ECC request can be allocated in the ith allocation. Moreover, TS-Router can cause congestion on the output port competed by ECC requests (outport 3 in this case). As a result, after a period of time, outport 3 will be congested, and

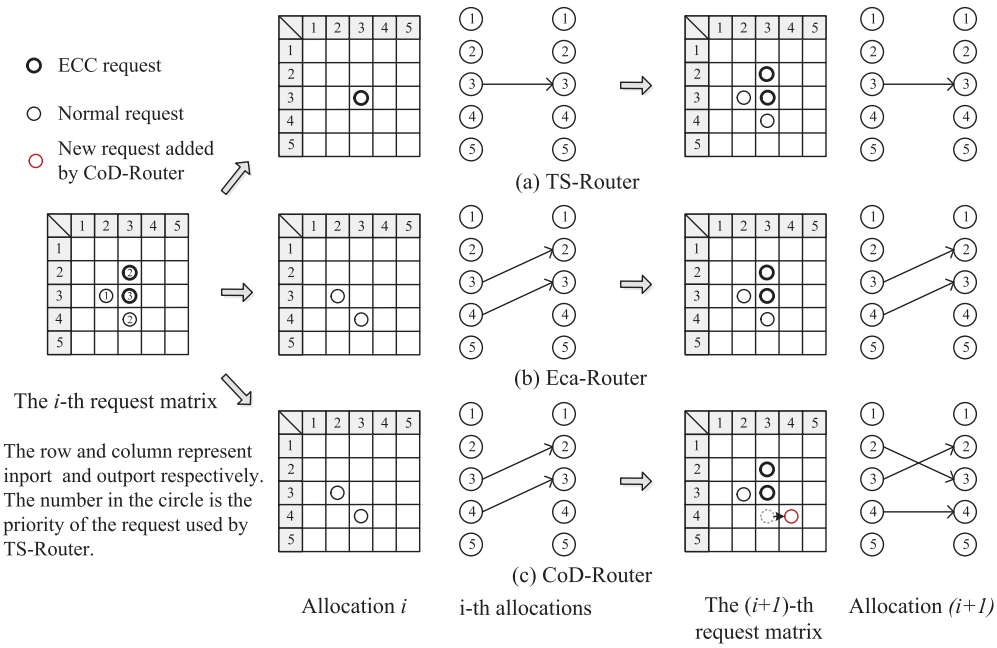


Fig. 7. Switch allocation comparisons. The thin circles are the uniform requests and the thick circles are the ECC requests used by Eca-Router. We assume that there are continuous request arrivals, and the arriving requests are the same as the ith request matrix in each cycle. TS-Router chooses the ECC request (3,3) in each cycle, because it has the highest priority but achieves the least matchings. Eca-Router lowers the priority of ECC requests and thus achieves more allocations. Request in inport 4 has an alternate routing outport 4 in addition to outport 3. CoD-Router selects outport 4 in the RC stage in the ith cycle and thus adds a new conflict-free request (4,4) in the SA in the $(i+1)$-th cycle. Therefore, CoD-Router achieves more matchings.

thus requests to this outport will not be allocated by the switch allocator. Eca-Router outperforms TS-Router, because Eca-Router relieves the impact of endpoint congestion in SA and thus achieves higher matching efficiency on long time series. In the SA of Eca-Router, as shown in Figure 7(b), the switch allocator prefers to choose uniform requests rather than the ECC requests and thus relieves the impact of endpoint congestion and provides more requests to the switch allocator on time series. To further improve the matching efficiency of SA, CoD-Router actually changes the SA requests with optimizations in the RC stage. As shown in Figure 7(c), in ith cycle, RC process detects that routing path $4 \rightarrow 3$ (resulting in request (4,3) in SA in the $(i+1)$-th cycle) can be conflict with other SA requests in the $(i+1)$-th cycle. Therefore, the candidate routing path $4 \rightarrow 4$ is chosen to avoid such conflict. Because CoD-Router introduces a new conflict-free request in SA, it can achieve more matchings in the $(i+1)$-th allocation compared to the other two SA strategies.

4.4 Pipeline Design in Eca-Router and CoD-Router

Eca-Router adopts the information from the RC stage to optimize the SA process. A simple design is to add a new pipeline stage (RS) before the SA stage for processing RC information and removing ECC requests from SA requests. However, such a design can increase the router latency by one cycle and damage the low latency characteristics of the on-chip router as a result. To mitigate performance loss, we divide the RS process into two stages. The first stage detects the ECC

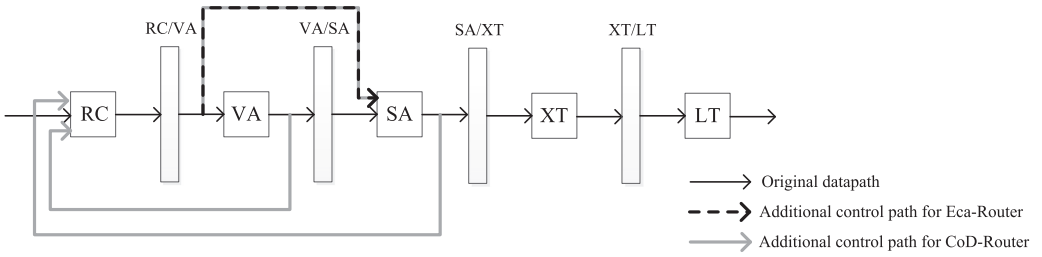


Fig. 8. Datapath of Eca-Router and CoD-Router. In Eca-Router, one forwarding link is added for prefetching the ECC request information to the SA stage. This link is from the output of the RC stage to the input of the SA stage. CoD-Router further adds two additional control paths to feed back the SA request information to the RC stage. Among these two paths, one transits unallocated SA request information from the SA stage, and the other transits the new generated SA request information from the VA stage, to the RC stage.

packets by analyzing the destination information of packets in input buffers, and the second stage determines which ECC requests should be discarded before the SA process. The first stage can be executed in parallel with the RC stage, because the input data (destinations of packets) of these two stages are the same. The second stage requires a random selection and removal of ECC requests. Selection of ECC requests can be implemented with simple selection logic, and it can be performed in parallel with the router critical pipeline stages. Removal of the selected ECC requests can be implemented by marking the ECC request before the start of the SA process. The marked ECC requests can be directly removed during the SA. Therefore, although the second phase of the RS cannot be performed in parallel with router pipelines, it can be integrated in the SA phase without introducing significant delay in the SA. To investigate the overheads in router frequency, we perform estimations on the critical path latency of our router architecture using Synopsys Design Compiler. Behavioral RTL is performed using 65-nm design libraries with 1.20 V and 1 GHz frequency. The result shows that our design achieves a 13.6% slower router, which runs up to 864 MHz.

Figure 8 presents the datapath of Eca-Router and CoD-Router in a five-stage pipeline router. As shown in the figure, a single forwarding link from RC stage to SA stage is added for transmitting the ECC request information in Eca-Router. It should be noted that the process of identifying the ECC request in the first stage of RS has been merged into the RC process. CoD-Router is designed based on Eca-Router. In CoD-Router, two additional links are added to further feed back the SA request information to the RC stage. These two links, one from the SA stage to the RC stage, which is designed for the purpose to transit the information of unallocated SA requests to the RC process. The other from the VA stage to the RC stage to transit the new generated SA request information to the RC process.

MUA-Router can be extended to other IQ routers as well, since the design of MUA-Router is not dependent on a particular router architecture or routing algorithms. The design of Eca-Router is orthogonal to most previous SA strategies, because the RS process and the SA process are independent. Therefore, previous advanced SA strategies can be adopted in the SA process of Eca-Router to further improve the performance. CoD-Router can be integrated with other adaptive routing algorithms as well. This is because CoD-Router makes sense when there exist multiple candidate routing paths in the RC stage, and these paths have the same probability to be selected. Although different routing algorithms select routing paths differently, the probability that two paths share the same probability to be selected is still very high. Therefore, CoD-Router can be extended to most IQ routers combined with efficient adaptive routing algorithms. It should be noted that CoD-Router cannot be integrated with DOR. This is because the routing decision of each hop in the

Table 1. Network Simulation Configuration

Parameters	Values
Network Topology	4×4, **8×8**, 16×16 2D meshes
Routing Algorithms	**Footprint** [10]
VC	2, **4**, 8 VCs per physical channel, VC buffer size is 8
Traffic Patterns	**Uniform Random**, Transpose, Tornado, Hotspot, PARSEC traces
Packet Size	**single-flit**, {2,3,4 $\cdots$ 16}-flit packets
Flow Control Technique	**credit-based wormhole**
VC Allocator	**Round-Robin**
Switch Allocator	**iSLIP**, Packet Chaining [25], TS-Router [3], Eca-Router, CoD-Router
Speedup	internal_speedup = 1.0

The default values are shown in bold.

routing path is determined for DOR, and there is no optional output port provided to CoD-Router to select an output port that can reduce SA conflicts.

5 EXPERIMENTAL STUDIES

5.1 Evaluation Methodology

The evaluations are performed using Booksim [15], a cycle-accurate interconnection network simulator, with synthetic traffic patterns as well as traces from real-workload. The detailed configuration has been listed in Table 1. We adopt an 8×8 2D mesh with 64 nodes as the baseline network topology. The 4×4 and 16×16 2D meshes are also evaluated to study the scalability of our design. Our evaluations focus on 2D mesh topologies due to the benefit of mapping well to the 2D layout and the fact that this topology has been implemented in commercial and experimental manycore systems. All channels have one cycle delay and each router connects one terminal node. Traditional VC flow control [4] is used and there are 4 VCs in each input port. Different number of VCs are also evaluated and each VC has a buffer size of eight flits in all the evaluations. We choose Footprint [10] routing for the mesh, because it is the most efficient adaptive routing in NoCs. Duato's theory [8] is utilized in Footprint to avoid routing deadlock. Router architecture has been described in Section IV, and the credits can be transmitted upstream in two cycles. We use one-iteration iSLIP as the baseline allocator for Packet Chaining, TS-Router and our design. In the evaluation, we implement the second type of Packet Chaining (same port, different VCs). Unless indicated otherwise, our evaluation is performed with single-flit packets.

Synthetic traffic patterns, such as uniform, transpose, tornado, and bitrev are used in the evaluation. In addition, a hotspot traffic is designed to evaluate the performance of our design under endpoint congestion conditions. For hotspot traffic pattern, 10% nodes are randomly selected as the hotspot receivers to accept traffic from other nodes. Other nodes have a 20% probability of sending packets to hotspot endpoint nodes, and an 80% probability of sending packets to a randomly selected node.

To evaluate the performance of our design with real-workload, trace-driven simulations [12] are performed under PARSEC benchmarks [2]. The traces are gathered from full-system simulations of 64 in-order-issue 2-way SMT cores. Dsent [32] is used to evaluate area and power consumption. Area and power evaluations are performed at 22-nm technology scaling with 1.0-V operating voltage. Operating frequency is set to 1.0 GHz while data width of links and flit width are all set to 128 bits.

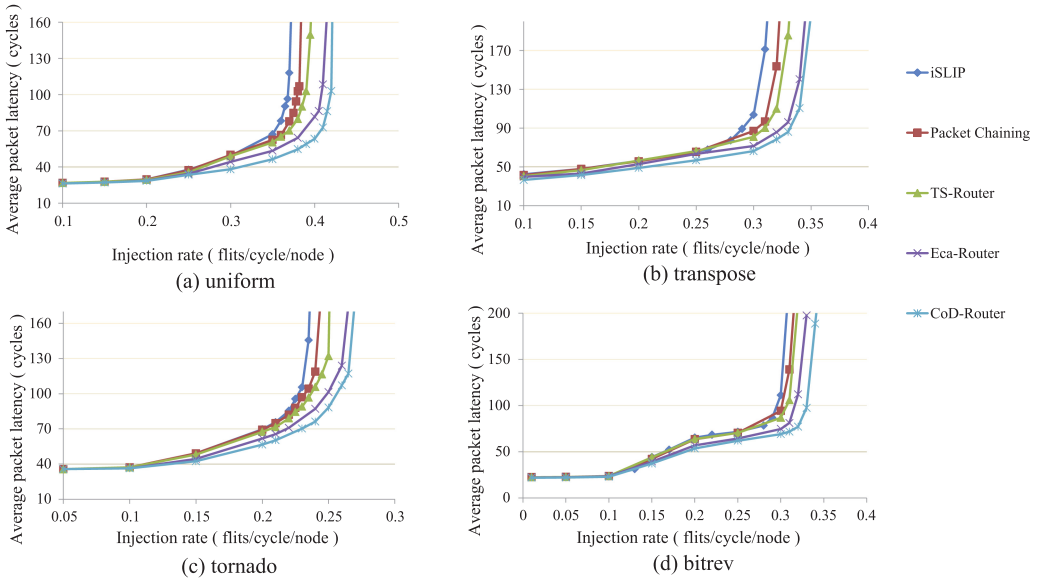


Fig. 9. Latency comparison of CoD-Router and Eca-Router with other switch allocators in the mesh.

5.2 Network Performance of Synthetic Traffic

We first evaluate the performance of Eca-Router and CoD-Router in the mesh under three different traffic patterns. We also compare our design with other advanced SA strategies. Average packet latency is measured and the result has been presented in Figure 9. As can be seen from the figure, Eca-Router outperforms most other allocators and CoD-Router achieves even better performance than Eca-Router. Our design not only achieves higher saturation throughput but also achieves lower latency when the injection rate is less than the saturation injection rate. Compared with TS-Router, Eca-Router achieves 6.7%, 6.2%, 8.9%, and 5.8% performance improvement in saturation throughput under uniform, transpose, and tornado traffic patterns, respectively. The throughput improvements of CoD-Router under these three traffic patterns are 7.2%, 7.8%, 12.8%, and 11.5% respectively. Under each traffic pattern, Eca-Router achieves much lower packet latency; this is because the impact of congestion is relieved during the SA process, which reduces the packet latency as a result. CoD-Router is designed based on Eca-Router and further introduces the SA information to the RC stage, achieving better performance than Eca-Router and other SA strategies due to more conflict-free requests provided to the switch allocator. It should noted that the overall performance improvement cannot compensate for the decreased frequency (13.6%) in this evaluation. This is because the complicated SA process will bring down the router frequency, which is a flaw in our design. However, our design gives a novel method to improve the performance of SA, and we can achieve higher performance through further optimization to offset the router frequency overhead. However, our design can achieve high performance under certain traffic patterns or larger NoCs, which is enough to offset the overhead of router frequency. Detailed experimental results will be presented in the following paragraphs.

We further evaluate the impact of the number of VCs on performance and the result is shown in Figure 10. In this evaluation, we vary the number of VCs and compare our design with TS-Router. To avoid deadlock in RC, at least two VCs are required in the input buffer [8]. Therefore, in this evaluation, we adopt two VCs, four VCs, and eight VCs in the input buffer. More VCs can

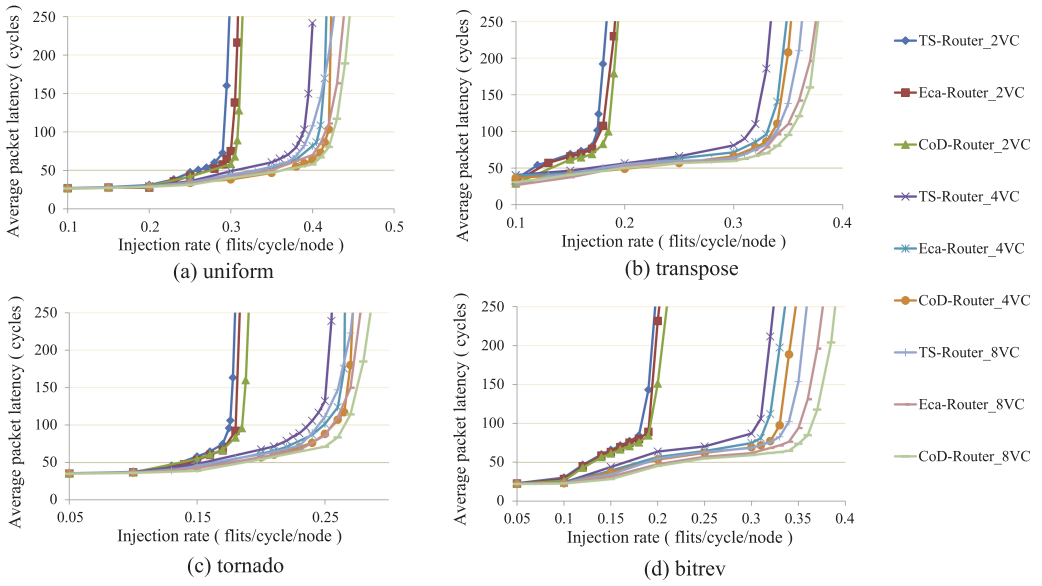


Fig. 10. Latency comparison of Eca-Router, CoD-Router, and TS-Router as the number of VCs is varied for different traffic patterns.

provide more requests to the switch allocator, resulting in more matchings in the SA process. As shown in Figure 10, for different SA strategies, adopting more VCs can achieve better performance. As the number of VCs increases, the performance of SA strategy increases accordingly. However, when the number of VCs continues to increase, the number of requests is no longer a performance bottleneck for SA, the allocation efficiency of the switch allocator is. Therefore, the performance difference between different SA strategies is more obvious when a larger number of VCs is used. As shown in Figure 10, given the same number of VCs, CoD-Router and Eca-Router outperform TS-Router under different traffic patterns. Eca-Router improves the saturation throughput by 5.2% with 2 VCs and increases to 7.3% with 8 VCs under uniform traffic pattern, and the corresponding improvements achieved by CoD-Router are 6.7% and 8.1%, respectively.

One reason why MUQ-ROUTER is efficient lays in its ability to relieve the impact of endpoint congestion. Thus we evaluate our design and compare it with other SA strategies under a hotspot and uniform mixed traffic pattern, described earlier in Section 5.1. The hotspot traffic will create a congestion tree and congest other uniform traffic due to the HoL blocking, which can reduce network throughput. We present the latency-throughput curve in Figure 11 to demonstrate the benefit of our design in relieving the impact of endpoint congestion. As shown in Figure 11, TS-Router saturates when the injection rate reaches approximately 34%. Compared with TS-Router, the saturation injection rate of Eca-Router and CoD-Router can be increased by 4.6% and 8.8%, respectively. In addition to increasing network throughput, our design also reduces the transmission latency of packets under low load. As shown in Figure 11, when the injection rate is lower than the saturation point, CoD-Router can achieve much lower packet latency compared with other SA strategies. When the injection rate is 0.2 flits/cycle, CoD-Router can reduce average packet latency by 37.5% and 30.2% compared with TS-Router and Eca-Router.

We also compare our design with TS-Router using different network scale and the result is shown in Figure 12. The throughput of Eca-Router and CoD-Router has been normalized to TS-Router under each traffic pattern. As shown in Figure 12, the performance improvement of our

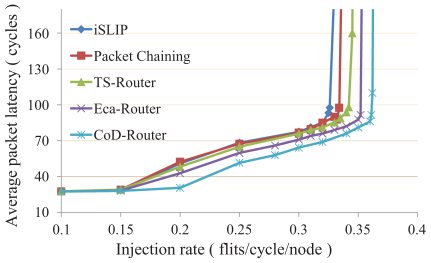


Fig. 11. Performance comparison of TS-Router, Eca-Router, and CoD-Router with hotspot traffic.

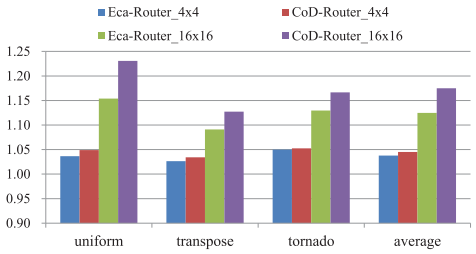


Fig. 12. Throughput comparison of TS-Router, Eca-Router, and CoD-Router for different network size. The throughput of Eca-Router and CoD-Router is normalized to that of TS-Router.

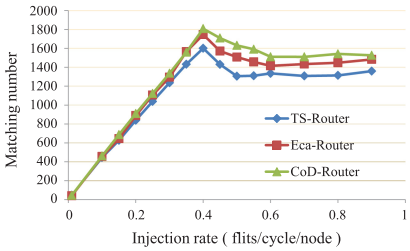


Fig. 13. Comparison of matching numbers in the SA process of TS-Router, Eca-Router, and CoD-Router in a single router of the mesh network under uniform traffic.

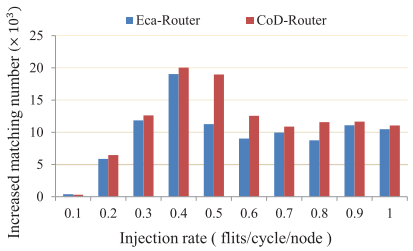


Fig. 14. Increased matching number of SA for Eca-Router and CoD-Router (compared to TS-Router) under uniform traffic in the mesh.

design is larger in the 16 × 16 mesh than in the 4 × 4 mesh, since larger network can stress the congestion. For uniform traffic pattern, the throughput gained by Eca-Router over TS-Router in 4×4 and 16×16 meshes is 3.6% and 15.4%, respectively. The corresponding improvement achieved by CoD-Router in these two meshes is 4.9% and 23.1%, respectively. Compared with TS-Router, the improvement in throughput of Eca-Router in 4 × 4 and 16 × 16 meshes is 3.8% and 12.5%, respectively, while the corresponding improvement of CoD-Router in these two meshes is 4.5% and 17.5%, respectively. Considering the overhead of router frequency, the performance improvement of our design is more pronounced for larger NoCs, which can offset the overhead of router frequency. For this evaluation, the number of VCs is not increased as the network size increases, and we do not increase the depth of VCs as well. Although adopting more VCs or deeper VCs implies better performance in NoCs, the increased buffer size and the complex scheduling logic are difficult to implement in the on-chip router. Note that input buffer consumes a considerable amount of area and power in a NoC router, it is difficult to adopt a large buffer in on-chip routers [20].

The performance of switch allocator is associated with the matching number, and the larger matching number achieved, the better performance of the switch allocator. We compare the number of matchings in a single router for CoD-Router, Eca-Router, and TS-Router by increasing the injection rate from 0 to 0.9 flits/cycle/node. The result is presented in Figure 13, and it is collected from 1,000 consecutive stable cycles in the mesh network under uniform traffic pattern. As Figure 13 shows, when the injection rate exceeds 0.38 flits/cycle/node, the network will be saturated. Moreover, when the injection rate exceeds the saturation point, as the injection rate increases, the matching number does not continue to increase but decreases. This is because when

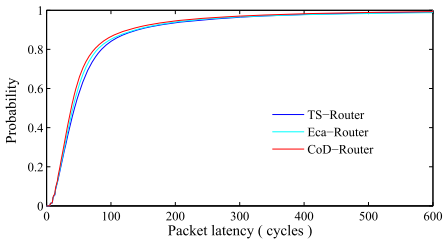


Fig. 15. The CDF of packet latency for TS-Router, Eca-Router, and CoD-Router.

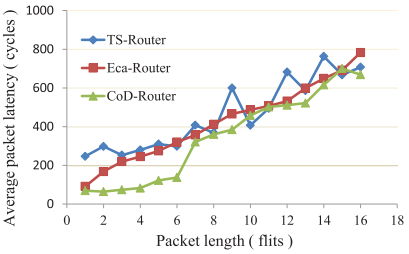


Fig. 16. Average packet latency by packet length for TS-Router, Eca-Router, and CoD-Router under uniform traffic pattern in the mesh.

the output port is congested, requests provided to the switch allocator will be reduced, thereby reducing the number of SA requests. However, due to the advantage in SA process, CoD-Router and Eca-Router can achieve more matchings when the injection rate is near or exceeds the saturation injection rate.

We further demonstrate the advantage of our design in increasing the number of matchings and present the result in Figure 14. Figure 14 presents the improved matching number of the SA process for CoD-Router and Eca-Router compared with TS-Router. We collect the matching number of the switch allocator for CoD-Router, Eca-Router, and TS-Router in 40,000 cycles in an 8×8 mesh network with the injection rate changing from 0.1 to 1.0 flits/cycle/node. As presented in Figure 14, CoD-Router and Eca-Router achieve more matchings than TS-Router at each injection rate, and CoD-Router can deliver more matchings than Eca-Router. When the injection rate is less than 0.5, the increased matching numbers of both methods increase as the injection rate increases. This is because the matching number relies on the number of requests, and the more requests provided to the switch allocator, the more matchings can be achieved. However, when the injection rate exceeds 0.5 flits/cycle/node, the number of increased matchings begins to decrease. This is because the excessive load can block the output port of the router, which leads to a decrease in the number of valid requests and in turn damages the allocation efficiency of the switch allocator.

Eca-Router achieves better performance in average packet latency by relieving the impact of endpoint congestion. In this way, the long waiting latency of packets caused by endpoint congestion can be reduced, resulting in lower average packet latency. Although CoD-Router changes the execution process of routing algorithm, it does not negatively affect the performance. We evaluate the latency distributions of TS-Router, Eca-Router, and CoD-Router and present the result of **cumulative distribution function (CDF)** in Figure 15. The evaluation is performed in an 8×8 mesh network and packets are injected in saturation injection rate. As can be seen in the figure, the proportion of packets with low latency is larger in our design than in TS-Router. Moreover, our design does not introduce significant tail latency. That is, our design can avoid the long packet latency caused by unnecessary waiting when faced with endpoint congestion.

We next evaluate the impact of packet length on the performance of our design. We evaluate the average packet latency of CoD-Router, Eca-Router, and TS-Router with the packet length changing from 1 flit to 16 flits. Results are presented in Figure 16. We set the injection rate to saturation injection rate for each test. As can be observed from the figure, Eca-Router outperforms TS-Router when the packet length is smaller than 8 flits and becomes worse when the packet length beyond 8 flits. The performance improvement of Eca-Router degrades as the packet length keeps increasing. This is because long packets reduce the number of requests provided to the switch allocator, which limits the optimization of the SA process. For TS-Router, there is a mutation after packet length

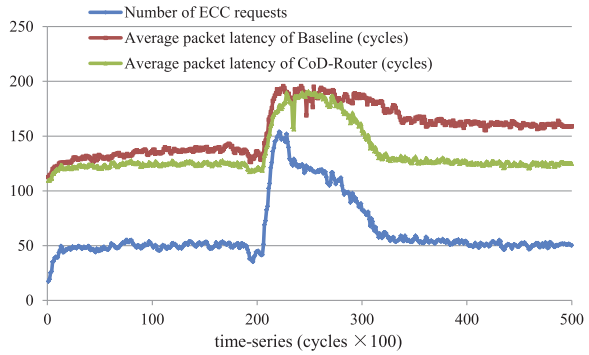


Fig. 17. Average packet of CoD-Router and the baseline router when endpoint congestion occurs in the 20,000th cycle for 500 cycles.

beyond 8 flits. This is because we set the capacity of a VC to 8 flits in our evaluation. When the packet length beyond 8 flits, the probability of two packets in the same VC will be very small. Consequently, the opportunity to get the information of next requests will also be reduced and thus reduces the performance of TS-Router. In most cases, CoD-Router can achieve better performance than the other two methods, especially when the packet length is less than 7. This indicates that CoD-Router is more suitable for NoCs where a large proportion of packets are short packets [25].

Our design strives to find ECC requests and limit the allocation of these ECC requests to relieve the impact of endpoint congestion. To prove the effectiveness of our design, we have counted the changes in ECC requests when endpoint congestion occurs, as well as the changes in the average packet latency of our design and the baseline router. The result has been presented in Figure 17. In this evaluation, hotspot traffic, which can introduce endpoint congestion, occurs in the 20,000th cycle. Background load is a uniform random traffic pattern with an injection rate of 40%, and the hotspot traffic continues to inject 500 cycles from the 20,000th cycle with an injection rate of 50%. As shown in the figure, when endpoint congestion occurs, the average packet latency of both the CoD-Router and the baseline router increase sharply, but the CoD-Router can recover to normal levels more quickly, while the baseline router takes a long time to eliminate the impact of endpoint congestion on network performance. This is because CoD-Router can use the collected ECC request information to reduce the impact of endpoint congestion by limiting the allocation of requests that cause endpoint congestion. It can be seen from the figure that when congestion occurs in the network, the ECC requests detected by the CoD-Router also increase rapidly, which provides the required information for the performance optimization of the CoD-Router, and also proves that our design can effectively use ECC request information to optimize network performance.

5.3 Application-Level Performance

In addition to synthetic traffic patterns, we also compare our design with TS-Router using network traces from PARSEC 2.0 workloads [2] in the mesh network. We select eight PARSEC benchmarks to evaluate the performance of our design in some representative application scenarios. For example, the application domains of bodytrack, canneal, ferret, and fluidanimate are computer vision, engineering, similarity search, and animation, respectively. The application domain represented by blackscholes and swaptions is financial analysis, but blackscholes is with a small working set while swaptions is with a large working set to evaluate the impact of working set size. Both vips and x264 are applications from media processing area, but the parallelization model of vips is data-parallel while x264 adopts pipeline parallelization model, which can be used to evaluate the impact of different parallelization models. All the traces are from the whole execution of applications, and

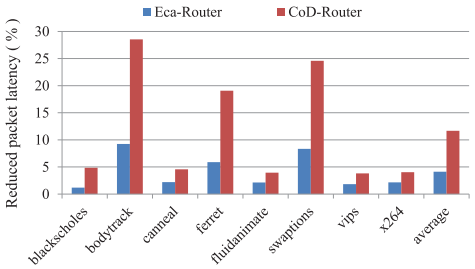


Fig. 18. The reduced packet latency of Eca-Router and CoD-Router (compared to TS-Router) with PARSEC benchmarks in the mesh network.

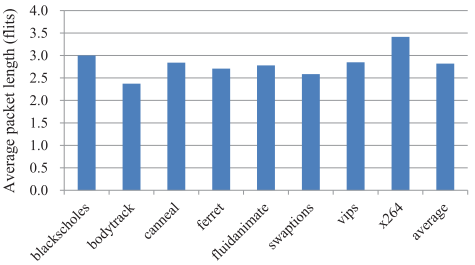


Fig. 19. Average packet length of PARSEC benchmarks.

both the parallel regions and the serial phases of the benchmarks will be taken into account. For each benchmark, results are collected after 1,000,000 cycles of continuous trace injection.

Application-level results are presented in Figure 18 and the latency reduction of CoD-Router and Eca-Router has been normalized to TS-Router. As can be seen from the figure, CoD-Router and Eca-Router outperform TS-Router in all benchmarks, and the performance of CoD-Router is much better than Eca-Router. For Eca-Router, the average improvement over all the benchmarks is 4.1% while the maximum improvement is 9.2% using bodytrack benchmark. The average improvement of CoD-Router over all the benchmarks is 11.7% while the maximal improvement is 28.5% using bodytrack benchmark. For most benchmarks in Figure 18, the performance improvement of Eca-Router is not significant; this is because the average injection rate of PARSEC benchmarks is relatively low, which results in fewer requests to the switch allocator and thus reduces the performance of Eca-Router in relieving the impact of endpoint congestion. However, CoD-Router can achieve much more performance improvements than Eca-Router. This is because most packets in PARSEC benchmarks are short packets, and CoD-Router can achieve much better performance with short packets (as shown in Figure 16).

We also calculate the average length of the packets in PARSEC benchmarks and present the result in Figure 19. As shown in Figure 19, the average packet length varies from 2 to 4 flits for different benchmarks, and the overall average packet length is 2.8 flits. It can be seen from Figure 16 that CoD-Router can best reduce the packet latency when the packet length changes from 1 to 6 flits. This is the reason why CoD-Router can achieve a significant reduction in packet latency under PARSEC benchmarks whose average packet length changes from 2 to 4 flits. For the same reason, the performance of Eca-Router and CoD-Router under different benchmarks has a great relationship with the packet length. Eca-Router adopts fine-granularity endpoint congestion control strategy, and short packet dominated traffic is more suitable for this strategy. For CoD-Router, short packets can provide more precise feedback from the SA stage to the RC stage, which achieves better performance by providing more conflict-free requests to the switch allocator. The three benchmarks that improve performance most are bodytrack, ferret, and swaptions. Accordingly, as shown in Figure 19, the average packet length of these three benchmarks is also the shortest. The performance of Eca-Router and CoD-Router also relates with other factors, such as the number of synchronization operations.

Synchronous operation contributes most to endpoint congestion. To achieve better performance, Eca-Router can alleviate the impact of endpoint congestion directly while CoD-Router can mitigate the competing for the same output port during the SA. Therefore, Eca-Router and CoD-Router get better performance in benchmarks that contain more synchronization operations. For this reason, Eca-Router and CoD-Router achieve the maximum performance improvement under the

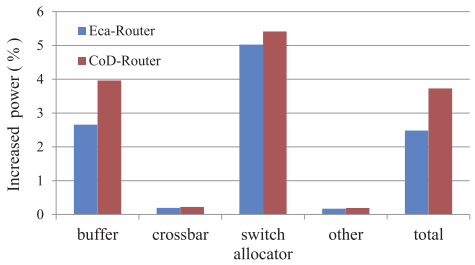


Fig. 20. The increased power consumption in a router of the Eca-Router and CoD-Router compared with the baseline router.

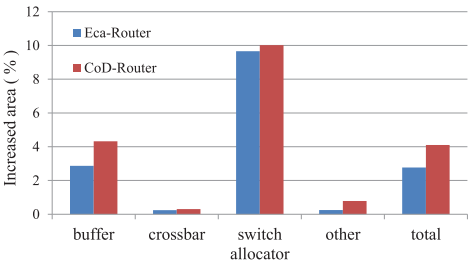


Fig. 21. The increased area in a router of the Eca-Router and CoD-Router compared with the baseline router.

bodytrack benchmark, due to its large amount of barriers synchronization operations. By the way, the performance improvement of Eca-Router is not obvious; this is because the average injection rate of PARSEC is relatively low (less than 0.005 flits per cycle) and the congestion in the network is not serious. The degree of performance improvement and the injection rate also have a certain correlation and the greater the injection rate, the more obvious the performance improvement is. This is because the higher the injection rate, the greater the probability of congestion, resulting in a more significant performance improvement in our design. For PARSEC benchmarks, bodytrack and swaptions have the highest average injection rate thus achieve the highest performance improvement.

5.4 Power and Area

We adopt a light-weighted implementation in our design to minimum the overhead. For Eca-Router, only a simple RS process is added before SA. The main process of RS can be executed in parallel with RC stage, and the other process of RS is preformed by marking the ECC request that has been selected to be discarded. The main overhead of Eca-Router comes from the added registers. In Eca-Router, to track the destination of the request in each VC, a $\log_2(N)$ bits of register is needed for each VC, where N is the network size. For an 8×8 2D mesh with four VCs per physical channel, this results in only 24 bits storage per port, and 120 bits in total for a 5-port router. Given that the size of a NoC flit is often very wide (e.g., 128 [11] or 256 [7] bits), the additional storage overhead is approximately equal to another flit buffer entry at the router. Compared with Eca-Router, CoD-Router adds two new datapaths to prefetch SA information from the SA and the VA stages. However, there is no need to set up new registers to store this information, as this information can be obtained directly from the SA and the VA processes. This information can be used to calculate the SA contention number for each output port, which is the basis of CoD-Router's RC process to select an output port that do not compete with future SA requests. Therefore, only a few registers need to be added to record the SA contention number of each output port. For an n-port router, a $\log_2(n)$ bits of register is needed for each output port, and thus the total capacity in a router is $n\times \log_2(n)$ bits. For a five-port router, compared with Eca-Router, CoD-Router's storage overhead is only 15 bits. CoD-Router has a small storage overhead, and there is only a little impact on the critical datapath of CoD-Router's pipelines. As our design only adds some registers and the corresponding logic circuits, the overhead of resource consumption is marginal.

Dsent is utilized to estimate power and area of our design. Our design is modeled based on the router model in the simulator. The baseline router model has five pipelines stages and five input/output ports. In the baseline router, there are four VCs in each input port and each VC has a buffer size of eight flits. We modify the configuration parameters and the implementation details

in the router model to simulate our design. The buffer space of registers is added to emulate the storage overhead. Figure 20 presents the increased power consumption of a router for CoD-Router and Eca-Router compared with the baseline router, and the power result includes the corresponding leakage power and the dynamic power. It should be noted that we put the cost of registers into the buffer. Therefore, the main increase in power consumption comes from the switch allocator and the buffer. In total, power consumption is increased by 2.6% and 3.8% for Eca-Router and CoD-Router, respectively. Because we have added the cost of RS process to that of switch allocator, the switch allocator contributes the most in power consumption. We also present the area overhead of our design in Figure 21. In general, the area cost of our design is acceptable and similar to that of power consumption. In total, the increase in area of the Eca-Router and CoD-Router is 2.4% and 4.1%, respectively. The increase in area of the switch allocator of Eca-Router and CoD-Router is 9.7% and 10.0%, respectively.

6 CONCLUSION

Communication latency has become the performance bottleneck in on-chip systems. To overcome the bottleneck, designing efficient SA strategy is important for low-latency router architecture, because SA seriously impacts the overall network latency. Existing SA strategies only focus on the SA process itself to improve the matching efficiency, which is difficult to further improve the performance of SA. This paper revisits designing low-latency router architecture through the co-design between different pipeline stages to maximize the UoA for SA. Accordingly, we propose MUA-Router with two instance routers. We first introduce Eca-Router to relieve the impact of endpoint congestion by lowing the priority of requests that cause endpoint congestion in the SA process, collaboratively designed with RC stage. We carefully design the SA strategy in Eca-Router to guarantee the fairness in the SA process. Based on Eca-Router, an improved router architecture, CoD-Router is proposed to explore the co-design between RC, VA, and SA process to further improve the SA performance. CoD-Router strives to provide more conflict-free requests to the switch allocator through the optimization of RC process, using the request information collected from the VA and SA processes. We implement our design with light-weight property to achieve desirable performance with moderate overheads. Evaluations demonstrate that CoD-Router enhances the overall performance of NoCs by 12.8% with synthetic traffic compared with TS-Router, and the average performance improvement under application-level traffic is about 11.7%. As for overhead, CoD-Router improves about 3.7% leakage power consumption and 4.1% area compared with the baseline router.

ACKNOWLEDGMENTS

The authors thank the anonymous reviewers for their valuable feedback. We gratefully acknowledge members of Tianhe interconnect group at NUDT for many inspiring conversations.

REFERENCES

[1] Minseon Ahn and Eun Jung Kim. 2010. Pseudo-circuit: Accelerating communication for on-chip interconnection networks. In *Proceedings of the 43rd Annual IEEE/ACM International Symposium on Microarchitecture*. 399–408.

[2] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh, and Kai Li. 2008. The PARSEC benchmark suite: Characterization and architectural implications. In *Proceedings of the 17th ACM International Conference on Parallel Architectures and Compilation Techniques*. 72–81.

[3] Yuan-Ying Chang, Yoshi Shih-Chieh Huang, Matthew Poremba, Vijaykrishnan Narayanan, Yuan Xie, and Candice King. 2013. TS-Router: On maximizing the quality-of-allocation in the on-chip network. In *Proceedings of the 19th IEEE International Symposium on High Performance Computer Architecture (HPCA'13)*. 390–399.

[4] William J. Dally. 1992. Virtual-channel flow control. *IEEE Trans. Parallel Distrib. Syst.* 3, 2 (1992), 194–205.

[5] William J. Dally and Hiromichi Aoki. 1993. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Trans. Parallel Distrib. Syst.* 4, 4 (1993), 466–475.
[6] William James Dally and Brian Patrick Towles. 2004. *Principles and Practices of Interconnection Networks*. Elsevier.
[7] Reetuparna Das, Soumya Eachempati, Asit K. Mishra, Vijaykrishnan Narayanan, and Chita R. Das. 2009. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *Proceedings of the 15th IEEE International Symposium on High Performance Computer Architecture (HPCA'09)*. 175–186.
[8] JosÍę Duato. 1993. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parallel Distrib. Syst.* 4, 12 (1993), 1320–1331.
[9] Binzhang Fu, Yinhe Han, Jun Ma, Huawei Li, and Xiaowei Li. 2011. An abacus turn model for time/space-efficient reconfigurable routing. In *Proceedings of the 38th ACM/IEEE Annual International Symposium on Computer Architecture (ISCA'11)*. 259–270.
[10] Binzhang Fu and John Kim. 2017. Footprint: Regulating routing adaptiveness in networks-on-chip. In *Proceedings of the 44th ACM/IEEE Annual International Symposium on Computer Architecture (ISCA'17)*. 691–702.
[11] Paul Gratz, Changkyu Kim, Karthikeyan Sankaralingam, Heather Hanson, Premkishore Shivakumar, Stephen W. Keckler, and Doug Burger. 2007. On-chip interconnection networks of the TRIPS chip. *IEEE Micro* 27, 5 (2007), 41–50.
[12] Joel Hestness and Stephen W. Keckler. 2011. Netrace: Dependency-tracking traces for efficient network-on-chip experimentation. The University of Texas at Austin, Department of Computer Science, Technical Report (2011).
[13] Syed Ali Raza Jafri, Hamza Bin Sohail, Mithuna Thottethodi, and T. N. Vijaykumar. 2013. apSLIP: A high-performance adaptive-effort pipelined switch allocator. ECE Technical Reports Paper 451 (2013), 1–14.
[14] N. Jiang, D. Becker, G. Michelogiannakis, and W. Dally. 2012. Network congestion avoidance through speculative reservation. In *Proceedings of the 18th IEEE International Symposium on High Performance Computer Architecture (HPCA'12)*. 1–12.
[15] Nan Jiang, Daniel U. Becker, George Michelogiannakis, James Balfour, Brian Towles, David E. Shaw, John Kim, and William J. Dally. 2013. A detailed and flexible cycle-accurate network-on-chip simulator. In *Proceedings of the 14th IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'13)*. 86–96.
[16] N. Jiang, L. Dennison, and W. J. Dally. 2015. Network endpoint congestion control for fine-grained communication. In *Proceedings of the 29th ACM Annual International Conference on Supercomputing*. 1–12.
[17] Gwangsun Kim, Changhyun Kim, Jiyun Jeong, Mike Parker, and John Kim. 2016. Contention-based congestion management in large-scale networks. In *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture*. 1–13.
[18] Rakesh Kumar, Victor Zyuban, and Dean M. Tullsen. 2005. Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In *Proceedings of the 32nd ACM/IEEE International Symposium on Computer Architecture (ISCA'05)*. 408–419.
[19] Cunlu Li, Dezun Dong, and Xiangke Liao. 2018. Eca-Router: On achieving endpoint congestion aware switch allocation in the on-chip network. In *Proceedings of the 36th IEEE International Conference on Computer Design*. 506–509.
[20] Cunlu Li, Dezun Dong, Xiangke Liao, Ji Wu, and Fei Lei. 2016. RoB-Router: Low latency network-on-chip router microarchitecture using reorder buffer. In *Proceedings of the 24th IEEE Annual Symposium on High Performance Interconnects (HOTI'16)*. 68–75.
[21] Sheng Ma, Natalie Enright Jerger, and Zhiying Wang. 2011. DBAR: An efficient routing algorithm to support multiple concurrent applications in networks-onchip. In *Proceedings of the 38th ACM/IEEE Annual International Symposium on Computer Architecture (ISCA'11)*. 413–424.
[22] Sheng Ma, Natalie Enright Jerger, and Zhiying Wang. 2012. Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip. In *Proceedings of the 18th IEEE International Symposium on High Performance Computer Architecture (HPCA'12)*. 1–12.
[23] Nick McKeown. 1999. The iSLIP scheduling algorithm for input-queued switches. *IEEE/ACM Trans. Netw.* 7, 2 (1999), 188–201.
[24] G. Michelogiannakis, N. Jiang, D. Becker, and W. Dally. 2013. Channel reservation protocol for over-subscribed channels and destinations. In *Proceedings of the 27th ACM Annual International Conference on Supercomputing*. 1–12.
[25] George Michelogiannakis, Nan Jiang, Daniel Becker, and William J. Dally. 2011. Packet chaining: Efficient single-cycle allocation for on-chip networks. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*. 83–94.
[26] Shubhendu S. Mukherjee, Federico Silla, Peter Bannon, Joel Emer, Steve Lang, and David Webb. 2002. A comparative study of arbitration algorithms for the Alpha 21364 pipelined router. In *ACM SIGARCH Computer Architecture News*, Vol. 30. 223–234.
[27] Robert Mullins, Andrew West, and Simon Moore. 2004. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st IEEE Annual International Symposium on Computer Architecture (ISCA'04)*. 188–197.

[28] Li-Shiuan Peh and William J. Dally. 2001. A delay model and speculative architecture for pipelined routers. In *Proceedings of the 7th IEEE International Symposium on High Performance Computer Architecture (HPCA'01)*. 255–266.
[29] Supriya Rao, Supreet Jeloka, Reetuparna Das, David Blaauw, Ronald Dreslinski, and Trevor Mudge. 2014. Vix: Virtual input crossbar for efficient switch allocation. In *Proceedings of the 51st ACM Annual Design Automation Conference*. 1–6.
[30] Daniel Sanchez, George Michelogiannakis, and Christos Kozyrakis. 2010. An analysis of on-chip interconnection networks for large-scale chip multiprocessors. *ACM Trans. Arch. Code Optim.* 7, 1 (2010), 4.
[31] J. Santos, Y. Turner, and G. Janakiraman. 2003. End-to-end congestion control for infiniband. In *Proceedings of the 11th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03)*. 1123–1133.
[32] Chen Sun, Chia-Hsin Owen Chen, George Kurian, Lan Wei, Jason Miller, Anant Agarwal, Li-Shiuan Peh, and Vladimir Stojanovic. 2012. DSENT-a tool connecting emerging photonics with electronics for opto-electronic networks-on-chip modeling. In *Proceedings of the 6th IEEE/ACM International Symposium on Networks on Chip*. 201–210.
[33] Paul V. Gratz, Boris Grot, and Stephen W. Keckler. 2008. Regional congestion awareness for load balance in networks-on-chip. In *Proceedings of the 14th IEEE International Symposium on High Performance Computer Architecture (HPCA'08)*.
[34] Ke Wu, Dezun Dong, Cunlu Li, Shan Huang, and Yi Dai. 2019. Network congestion avoidance through packet-chaining reservation. In *Proceedings of the 48th ACM International Conference on Parallel Processing (ICPP'19)*. 58:1–58:10.

Received May 2021; revised January 2022; accepted February 2022