Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Traffic-oriented reconfigurable NoC with augmented inter-port buffer sharing*

Chenglong SUN†1, Yiming OUYANG†‡2, Huaguo LIANG3

1*School of Computer and Information Engineering, Fuyang Normal University, Fuyang 236041, China*
2*School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China*
3*School of Electronic Science and Applied Physics, Hefei University of Technology, Hefei 230601, China*
†E-mail: chenglson@outlook.com; oyym@hfut.edu.cn

Received July 8, 2023; Revision accepted Dec. 19, 2023; Crosschecked June 5, 2024; Published online Aug. 16, 2024

Abstract: As the number of cores in a multicore system increases, the communication pressure on the interconnection network also increases. The network-on-chip (NoC) architecture is expected to take on the ever-expanding communication demands triggered by the ever-increasing number of cores. The communication behavior of the NoC architecture exhibits significant spatial–temporal variation, posing a considerable challenge for NoC reconfiguration. In this paper, we propose a traffic-oriented reconfigurable NoC with augmented inter-port buffer sharing to adapt to the varying traffic flows with a high flexibility. First, a modified input port is introduced to support buffer sharing between adjacent ports. Specifically, the modified input port can be dynamically reconfigured to react to on-demand traffic. Second, it is ascertained that a centralized output-oriented buffer management works well with the reconfigurable input ports. Finally, this reconfiguration method can be implemented with a low overhead hardware design without imposing a great burden on the system implementation. The experimental results show that compared to other proposals, the proposed NoC architecture can greatly reduce the packet latency and improve the saturation throughput, without incurring significant area and power overhead.

Key words: Network-on-chip; Reconfigurable; Traffic-oriented; Buffer sharing
https://doi.org/10.1631/FITEE.2300458 **CLC number:** TP302

1 Introduction

With the increasing number of cores integrated into one chip, the communication between cores becomes the bottleneck for the performance of the system. To support high-speed communication among cores, network-on-chip (NoC), which functions as a parallel and scalable communication backbone for multicore systems, has been refined and developed in recent research (Jerger et al., 2017; Ouyang et al., 2021). Conventional router-based NoCs, on the other hand, are statically deployed without regard for on-demand traffic, resulting in poor communication performance and low utilization of buffer. The performance of the conventional NoC architecture cannot meet the requirements of growing communication complexity and intensive communication. The traffic within NoC varies across time as well as spatial regions, making the traffic exhibit spatial–temporal variation. Therefore, the design of future NoC is expected to be dynamically reconfigured to satisfy time-varying traffic loads (Krishna et al., 2013b).

Prior reconfigurable NoCs are either infrastructure-based reconfigured or protocol-based reconfigured. Infrastructure-based reconfiguration approaches, such as those applied to routers or switches (Krishna et al., 2013a; Qian et al., 2015;

‡ Corresponding author
* Project supported by the National Natural Science Foundation of China (No. 62374049)
ORCID: Chenglong SUN, https://orcid.org/0000-0002-7492-0542

Baharloo and Khonsari, 2018) and topology (Stensgaard and Sparsø, 2008; Stuart et al., 2011; Zheng et al., 2021), perform reconfigurations at the microarchitecture level of the NoC. Router designs such as SMART (Krishna et al., 2013a) set up multihop bypassing paths with cycle-by-cycle reconfiguration, and Multi-NoC (Baharloo and Khonsari, 2018) supports reconfigurable transmission by dividing NoC into small sub-networks. These alternatives, on the other hand, necessitate a lot of architectural changes and an arbitration procedure, increasing a lot of hardware complexity. As examples of topology designs, we may mention ReNoC (Stensgaard and Sparsø, 2008; Stuart et al., 2011) and Adapt-NoC (Zheng et al., 2021). Due to the lack of additional advances in the router microarchitecture, these topology-changing approaches increase only network performance to a limited extent. Protocol-based reconfiguration concentrates on protocols by dynamically reconfiguring routing (Castillo et al., 2014; Jain et al., 2020) or switching (Nguyen and Tran, 2019; Das et al., 2021). These reconfigurable approaches rely on algorithms for reconfiguration, and they are not flexible enough to provide fast global reconfiguration for adaptivity.

Regarding the NoC components such as crossbars, arbiters, buffers, and links, buffers were the major consumer of leakage dissipation in the experiments conducted by Chen and Peh (2003), consuming around 64% of the entire power budget. The buffer occupies the largest area overhead in the whole router, and optimization of the buffer during the design stage will affect the overall behavior of the system, which in turn can affect the performance of the entire chip (Nicopoulos et al., 2006; Wu et al., 2020). Buffers are considered candidates for performance optimization in this approach since there are still a lot of free buffers even under high traffic loads (Chen and Peh, 2003). Time-varying traffic characteristics make the buffer resources more underutilized, and it is thus necessary to offer an effective buffer management mechanism to alleviate the underutilized buffer, to improve network performance.

To this end, this paper presents a novel sharing-based reconfigurable NoC that augments the router-based network with a modified input port to support buffer sharing between adjacent ports. A reconfiguration mechanism is proposed that dynamically stores newly arriving packets on current or adjacent ports. At the same time, using centralized output-oriented buffer management, the performance of the reconfigurable NoC proposed in this paper is further enhanced. Therefore, our reconfigurable NoC provides the needed flexibility while balancing hardware cost and communication. To summarize, our main contributions are as follows:

First, we propose a traffic-oriented reconfigurable NoC (TOR-NoC) that modifies the input port to support buffer sharing between adjacent ports. Specifically, the modified input port can be reconfigured dynamically to handle on-demand traffic. Consequently, the reconfiguration of the transmission path enhances the diversity of the route.

Then, we present the centralized output-oriented buffer management that works well with the reconfigurable input ports. Specifically, each individual virtual channel identifier (VC_ID) stores only packets for a specific output port, and a dedicated VC_ID is reserved for storing packets from adjacent ports.

Finally, together with the modified input port and enhanced buffer management, we present a method of reconfigurable routing under which packets shared in an adjacent port compete for the output of that adjacent port, thus reconfiguring the transmission path of data. The reconfigured routing reduces congestion on the original port and equalizes traffic without wasting the free buffer resources of the adjacent port.

In this paper, the ports with sufficient or idle buffer resources are shared with other ports in urgent need of buffer, which not only facilitates efficient utilization of the buffer resources but also effectively relieves the load of heavy traffic ports. Therefore, the proposed reconfigurable NoC improves the data transmission efficiency and enhances the performance of the network.

2 Background and related works

2.1 Conventional router architecture

In a conventional NoC, the network consists of routers and communication links connecting these routers to establish the NoC topology. First, we examine the conventional design of router in NoC. The virtual channel (VC) based router architecture is depicted in Fig. 1 with five ports corresponding to four directions (N, S, W, and E) and the local process element (L). The buffer in the router is generally divided into two or more physical VCs as required. A packet is made up of one or more flits (a flit is the smallest format of data that flow across routers). The routing protocol is usually wormhole routing, in which a packet's header flit goes through a path to reserve a route for the packet. The route is reserved until all of the packet's flits have passed through it and arrived at their destination router (Oveis-Gharan and Khan, 2016). The above-mentioned conventional NoC architecture is not necessarily sufficient for traffic-oriented high-performance systems. The deficiency is caused by issues with the structure and routing organization of routers. As highlighted in the literature (Oveis-Gharan and Khan, 2020), there is an urgent need for a router architecture with adequate structural flexibility.

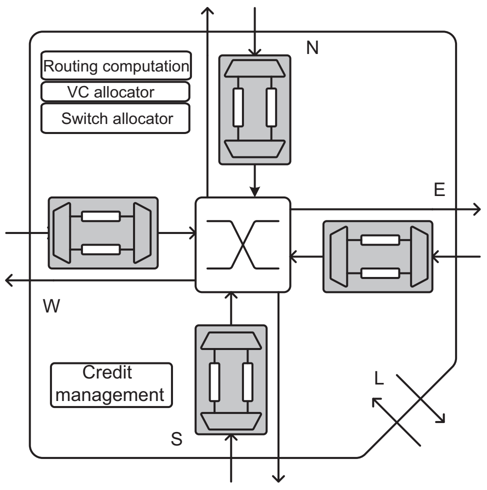


Fig. 1 Conventional router architecture (VC: virtual channel)

Fig. 2 briefly shows a simple example of unbalanced traffic load, where the traffic load of the west input port is much heavier than those of the north and south input ports. Red traffic indicates that the traffic on the link is heavy, whereas green traffic indicates that the traffic on the link is light. Within the router, the heavy load of the west input port is prone to congestion, which affects the transmission of subsequent packets. In contrast, the lighter traffic load on the south input port of the router indicates that fewer packets occupy the buffer of this port, which results in a significant amount of buffer remaining underutilized.

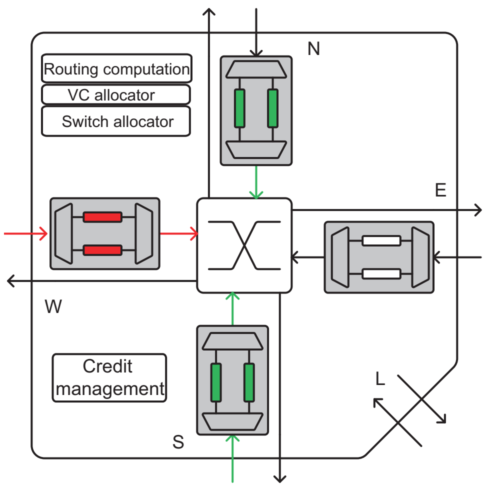


Fig. 2 Schematic of the unbalanced traffic load (References to color refer to the online version of this figure)

The congestion status of each port within the router varies greatly with the variation of traffic load, resulting in part of the buffer resources remaining underutilized. In a circumstance wherein some packets remain congested in the network for a long time, the transmission of the packets cannot be said to have attained completion even though all other packets might have reached the destination; it is thus well-recognized that the latency prevails until the last packet reaches the destination. The overall network performance will be improved pursuant to making the overall packet more balanced and facilitating it to reach the destination faster. Therefore, a proposition for a reconfigurable method within the router is urgently needed to enable sharing of the inter-port buffer resources. The ports with sufficient or idle buffer resources are shared with other ports in urgent need of buffer, which enables not only efficient utilization of the buffer resources but also effective relieving of the load of heavy traffic ports.

2.2 Related works

Many studies have explored various microarchitecture approaches for reconfigurable NoCs. Previous reconfigurable on-chip designs such as express virtual channel (EVC) (Kumar A et al., 2007) and single-cycle multihop asynchronous repeated traversal (SMART) (Krishna et al., 2013a) can reduce the hop count between source and destination routers by reconfiguring multihop transmission paths of packets, thereby strengthening the network performance.

The proposed SMART architecture (Krishna et al., 2013a) reconfigures the bypassing paths according to the current requests at runtime. The packets can travel the multihop path in one cycle after the multihop bypassing path is set up. Although the reconfigured paths can effectively reduce the latency of transmission paths, the improvement of the bypassing paths is limited at high traffic load. Wang L et al. (2020) proposed a hybrid NoC that adds a set of reconfigurable rings to a router-based mesh network. To achieve excellent communication performance, the rings can be reconfigured to accommodate traffic variations. Lan et al. (2011) presented a unique bidirectional NoC (BiNoC) architecture based on dynamic self-reconfigurable bidirectional channels. Each communication channel in BiNoC can dynamically self-reconfigure to send packets in either direction. This adaptability provides higher bandwidth utilization and faster packet delivery. Matos et al. (2011) proposed a reconfigurable router, where the buffer slots were dynamically allocated to increase router efficiency even under rather different communication loads. In this architecture, each buffer depth used in the router's input channel can be reconfigured at runtime, improving buffer utilization while avoiding packet congestion that might occur in the input port. Adapt-NoC, introduced in Zheng et al. (2021), is an application-aware flexible NoC architecture featuring a reinforcement learning (RL) based control strategy. This innovative design is capable of accommodating the diverse communication demands arising from concurrently executing applications. As a result, Adapt-NoC not only offers various topology options for concurrently running applications but also optimizes the optimal topology option for each application to increase performance and energy efficiency. Stuart et al. (2011) designed the ReNoC reconfiguration architecture by wrapping the NoC routers with reconfigurable switch (RS) logic. ReNoC allows for the configuration of application-specific logical NoC topologies, resulting in increased efficiency and flexibility. The above studies do not take into account the load imbalance between the ports within the router. The load gap between the ports will reduce the utilization of buffer resources.

The buffer of router can be structured as a single queue or a series of independent queues known as VCs. Furthermore, these buffers have a considerable impact on network throughput, particularly when the network is overloaded. The probability of a packet being dropped is reduced as the buffer size is increased (Kumar S et al., 2002). However, buffers dominate the NoC router area (Nicopoulos et al., 2006) and power consumption (Ye et al., 2002; Wang HS et al., 2003). Last but still important, the buffer organization has a considerable impact on the architecture of high-performance on-chip networks. The performance is considerably improved when the buffer depth is increased. Increased buffer depth, on the other hand, may result in lower buffer utilization (Zoni et al., 2016). Said et al. (2021) proposed two novel buffering mechanisms called Minimum-First buffering and Inverse-Priority buffering. The first mechanism prioritizes the minimum occupied buffers for storing new flits, aiming to minimize the queuing time of packets in the NoC routers. The second mechanism ensures a balanced load between router buffers to increase throughput. However, it is noteworthy that the input port inside the router was designed with only one VC. Zoni et al. (2016) proposed CUTBUF, a unique router architecture that allows VCs to be dynamically allocated based on their actual demands. The same buffer can be used to hold different kinds of messages at varying periods, allowing for resource reuse. Jindal et al. (2020) proposed a scheme augmented virtual channel (AugVC) to reuse trace buffers to augment router buffers, with the objective of improving the overall network performance. A novel lightweight elastic buffer architecture named ElastiStore was proposed in Seitanidis et al. (2015) that minimizes buffering requirements without sacrificing performance. A simplified router microarchitecture called unified buffer power-efficient router for network-on-chip (UBERNoC) was proposed in Farrokhbakht et al. (2019) that reduces underutilized buffer space. To save power and space, a unified buffer with multiple VCs shared among the input ports was used. The above studies for buffer are optimized within ports, lacking buffer optimization between ports, and the path diversity of packet transmission remains unchanged.

3 Traffic-oriented reconfigurable router architecture

The main goal of the router architecture we propose in this study is to provide a reconfigurable NoC with an elaborated design in terms of high flexibility in routing. We propose a new sharing-based reconfigurable NoC that augments the router-based network with a modified input port to support buffer sharing between adjacent ports. The architecture of the traffic-oriented reconfigurable router is depicted in Fig. 3. Each change to the proposed microarchitecture is obvious. Our TOR-NoC consists mainly of two components: the modified input ports to support reconfiguration and the centralized output-oriented buffer management to handle heavy traffic. An additional demultiplexer (DEMUX) is added in the input port to transfer packets to a centralized buffer of adjacent ports, enabling a flexible reconfiguration of the routing path. If the current port is heavily loaded, packets will transfer to the free buffer resource of the adjacent port through DEMUX to reconfigure the transmission path in such a way to achieve load balancing. The shaded part of Fig. 3 shows the centralized buffer, which receives packets from the current port and adjacent ports. We will describe the structure of the centralized buffer in detail in Section 3.2.

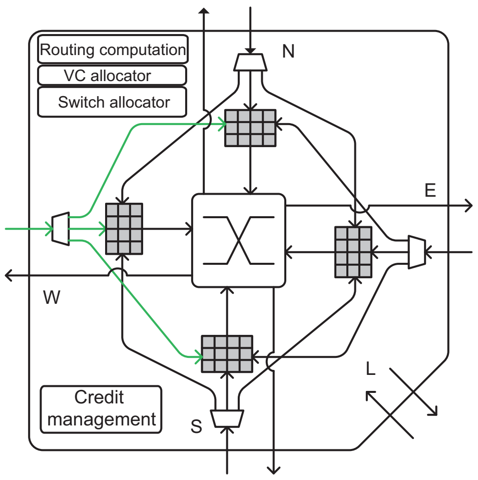


Fig. 3 Microarchitecture of the traffic-oriented reconfigurable router (References to color refer to the online version of this figure)

For a reconfigurable NoC, an important feature is that the transmission path should have the capability to be adjusted to time-varying traffic. To support this feature, the proposed router can perform reconfiguration using DEMUX according to current communication on-demand traffic. As depicted by the solid green lines in Fig. 3, the three green lines are candidates for reconfigurable paths. A 1:3 DEMUX multiplexer selects the DEMUX for transmitting incoming packets to the shared buffer of the port or the neighboring port according to the decision of the corresponding controller. When the buffer of the port is full (one free buffer left), the corresponding logical control unit selects the DEMUX to transfer the packet to the shared buffer of the neighboring port.

1. Look-ahead routing computation (LRC). Since the output-oriented VC allocation mechanism is adopted, when the output port of the flit is determined, the VC_ID of the flit is also confirmed. Therefore, LRC is used to derive the output port. The output port of the downstream router is calculated in the current router to derive its VC_ID, so that when the flit is transmitted to the downstream router it can enter the VC corresponding to the predetermined VC_ID.

2. Credit management. Unlike physically separate VC-based routers, credit management needs to transmit the number of available buffers per VC. Since we use a unified buffer, credit management needs only to transmit the number of free buffers in the unified buffer. Furthermore, credit management collects the number of free flits corresponding to the VC_ID in the shared buffer for subsequent reconfigurable path selection.

A credit-based flow control mechanism is adopted. Flits are forwarded only if the next-hop router has available slots at its corresponding input port. Each router knows the number of available slots in each of its downstream routers. While allocating a VC for a flit, the router looks for an unassigned slot in the downstream router. Whenever a flit leaves a router, the upstream router is requested to update its credit information accordingly.

3.1 Deadlock analysis

Operating the XY routing algorithm with interport buffer sharing makes the network suffer from deadlock. Building on the findings of Li et al. (2023), the XY routing algorithm inherits certain directional restrictions for each buffer to prevent deadlock. These restrictions, known as forbidden turns, impose limitations on the movement of packets within the router buffers. For instance, a south to west turn prohibits packets from moving south to make a west turn. Consequently, the north input port cannot store packets heading west. Therefore, each buffer must adhere to a set of restrictions specifying its allowed or forbidden next-hop directions. A summary of allowed and forbidden packet directions for each buffer in a TOR-NoC is presented in Table 1.

Table 1 Direction restriction in TOR-NoC

Input port	Allowed output	Forbidden output
East (E)	N, S, W, L	E
West (W)	N, S, E, L	W
South (S)	N, L	S, E, W
North (N)	S, L	N, E, W

L: local direction

A noteworthy observation gleaned from Table 1 is the discernible variability in the flexibility of buffers. As illustrated, buffers situated at the east and west input ports demonstrate the highest degrees of flexibility. Specifically, flits originating from the east input port are permitted to store packets heading in the north, south, west, and local directions. However, these flits are prohibited from storing packets that are headed east. On the other hand, flits originating from the south input port are exclusively allowed to store packets heading north or destined for the local direction.

3.2 Inter-port buffer sharing

As can be seen from Fig. 3, the input port has three paths to choose from according to the remaining free buffer of the local port and the adjacent port. The choice is to buffer the current port or transfer to the adjacent port according to the condition of network congestion. The arriving flit is buffered to the current or adjacent port depending on the number of available buffers on the current port and the number of available buffers on two adjacent ports. If there is an available slot in this port, the arriving flit is directly added to the linked list corresponding to the VC_ID of the buffer. If the number of available buffers in this port is zero, the numbers of available buffers in the two neighboring ports will be compared and the neighboring port with more available buffers will be selected for transmission, and thus the arriving flit is entered into the linked list corresponding to VC_Share. Algorithm 1 describes the pseudo-code of the arriving flit transmission.

When an arriving flit is received in the input port, its VC_ID is first extracted. If the destination of the arriving flit is the current router, the arriving flit will eject directly from the local port

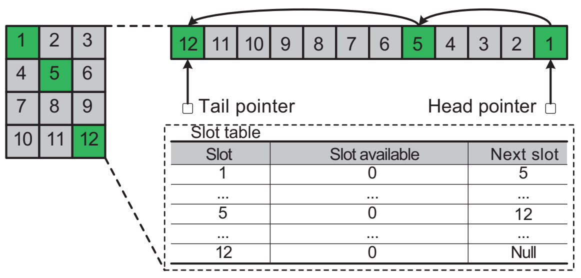
Algorithm 1 Routing algorithm for the arriving flit transmission

```
Require: the arriving flit and numbers of available slots on
    the current and adjacent ports
Ensure: the path of the arriving flit
 1: NumC←Number of available slots on the current port
 2: NumL←Number of available slots on the left adjacent
    port
 3: NumR←Number of available slots on the right
    adjacent port
 4: Extract the VC_ID field of the arriving flit
 5: if the destination of the arriving flit is the current port
    then
 6:     The arriving flit is ejected to the local port
 7: else
 8:     if (NumC ≥ NumL) ∧ (NumC ≥ NumR) then
 9:         Buffer the arriving flit within the current port cor-
            responding to VC_ID
10:     else
11:         if (NumL ≥ NumR) ∧ (NumL > 0) ∧ (NumR > 0)
            then
12:             Route and buffer the arriving flit into the left
                neighboring port corresponding to the shared
                VC_ID
13:         end if
14:         if (NumR > NumL) ∧ (NumL > 0) ∧ (NumR > 0)
            then
15:             Route and buffer the arriving flit into the right
                neighboring port corresponding to the shared
                VC_ID
16:         end if
17:     end if
18: end if
```

(lines 5 and 6). If the arriving flit has not yet reached the destination router, it continues forward routing. If NumL≤NumC and NumR≤NumC, indicating that the current port has idle slots, the arriving flit will be buffered in the corresponding VC directly (lines 8 and 9). If there is no idle slot in the current port, the arriving flit will be transferred to an idle slot in an adjacent port (lines 11–16). When selecting the neighboring port to buffer the arriving flit, the numbers of idle slots of the two neighboring ports are compared. If NumL≥NumR>NumC, it means that the left neighboring port has more idle slots, and accordingly the arriving flit will be routed into the left neighboring port (lines 11–13). If NumR>NumL>NumC, it means that the right neighboring port has more idle slots, and accordingly the arriving flit will be routed into the right neighboring port (lines 14–16).

3.3 Table-based shared buffer management approach

The advantage of inter-port buffer sharing is that it can share resources with neighboring ports, expanding the number of buffers available for data transmission. When credit management detects that the buffer of the current port is unavailable, it moves the arriving flit to the shared buffer of an adjacent port via DEMUX. Here we adopt a unified buffer, logically divided into different table-based VCs according to the linked list, and set a dedicated VC_ID for storing packets of neighboring ports. In this case, the arriving flits are transferred to the nearby idle buffer, so that there are more opportunities to obtain corresponding output ports. This alleviates the congestion at the current port and prevents more serious congestion in the network. The unified buffer design is divided into multiple VCs logically, ensuring that no deadlocks occur in the network. We may take the buffer of the west port as an example. According to the number of output ports, we fix the number of VCs in the buffer at four, which are the south, east, north, and shared VC_ID. Fig. 4 illustrates the table-based shared buffer management within the west port. The flits are stored randomly in the buffer, and the next flit is found based on the information recorded in the table.



Slot	Slot available	Next slot
1	0	5
...	...	...
5	0	12
...	...	...
12	0	Null

Fig. 4 Schematic of the table-based shared buffer management

A set of pointers are maintained for each VC_ID. Head Pointer_E is the head pointer corresponding to VC in the east direction, and tail Pointer_E is the tail pointer corresponding to VC in the east direction. The two pointers point only to packets whose output port is in the east direction. When an arriving flit is entered in the buffer of the current port, the tail pointer corresponding to VC_ID will shift to the arriving flit, and the relevant information in the table will be updated. When an arriving flit is removed from the buffer, the header pointer corresponding to the VC moves to the next data element and updates information in the slot table. In this way, the sequence of flits will not be out of order, ensuring that the data are transmitted in sequence and facilitating a more efficient use of buffer resources.

Fig. 4 illustrates the schematic of the table-based shared buffer management; the example is the case of east VC recorded in the buffer and table. The main fields in the table include slot, slot available, and next slot. The slot field indicates the identification of a single buffer, the slot available field indicates whether the buffer slot is available (0 means that the buffer slot is unavailable, and 1 means that the buffer slot is available), and the next slot field records the identification of the next buffer slot.

When an arriving flit enters the current port, it goes to the VC pointed by the pointer according to its VC_ID; the tail pointer is shifted to point to the newly arriving flit, and slot available is set to 0. When a flit on the current port wins the transmission by switch allocator (SA) arbitration, the flit is transmitted and the head pointer is shifted to point to the next flit in the linked list. The sequence of VC transmission in the port is ensured by such sequential transmission. Flit-based routing is adopted here; that is, each flit carries routing information and is transmitted separately. Otherwise, each transmission needs to check whether it is a header flit or not and to record whether the header flit is stored on the current port or adjacent port. This simplifies the corresponding logical operations and avoids the delay caused by recording and search (80% of packets are single flit (Ouyang et al., 2023)).

As shown in Fig. 5, we briefly illustrate the reconfigurable paths that exist within the router. There is a great diversity of routing paths.

We assume that the arriving flits are transmitted from the west port of the router, and all of its possible reconfigurable output paths are illustrated in Fig. 5. Since the input port is in the east, the ports with shared resources are in the north and south. Figs. 5a–5c show the reconfigurable output paths under the shared use of the north port's buffer, and Figs. 5d–5f show the reconfigurable output paths under the shared use of the south port's buffer. The flexible reconfigurable transmission path can be realized by sharing the buffer of neighboring ports, so

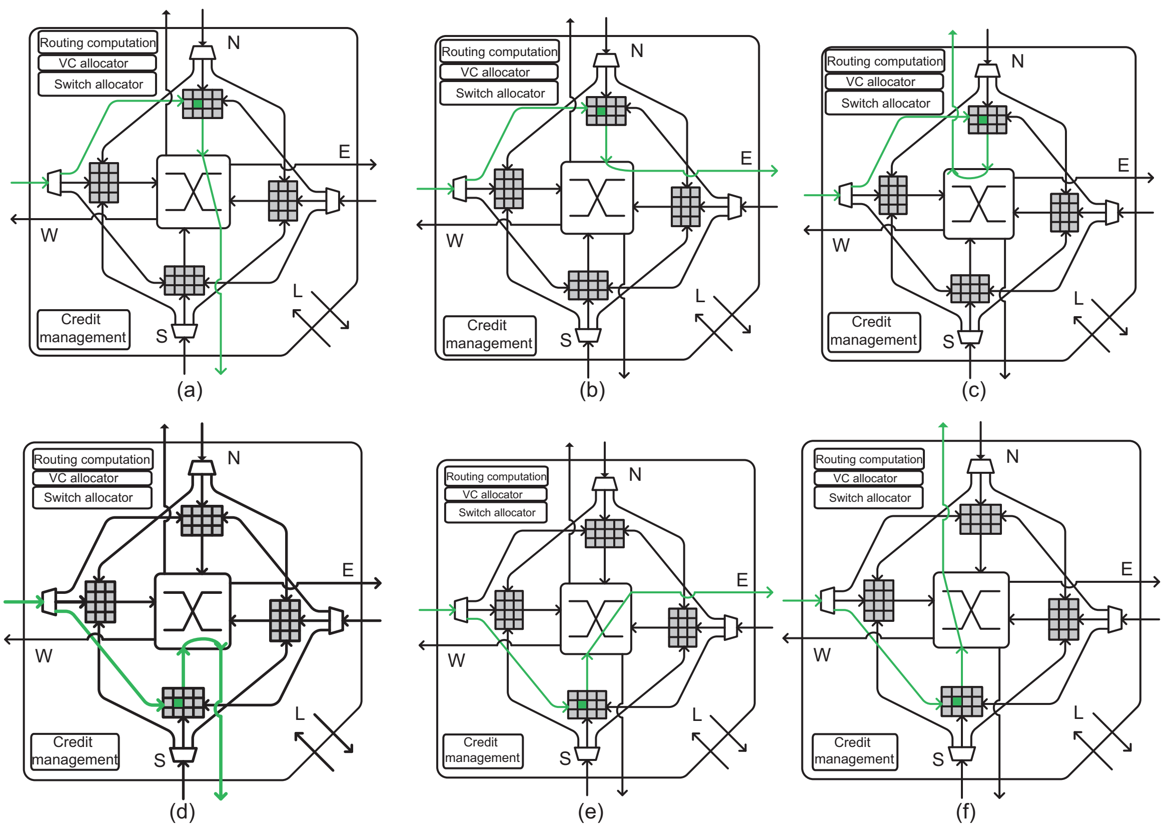


Fig. 5 Walk-through example of the reconfigurable transmission in the west port: (a)–(c) are the transmission of the flits shared within the north port, and (d)–(f) are the transmission of the flits shared within the south port

that the reconfigurable path of packet transmission can be adjusted in real time according to the on-demand network traffic.

When the arriving flit is buffered in an adjacent port, this indicates that the buffer of the current port is insufficient. In cases wherein an unbalanced traffic load results in a sufficient buffer of adjacent ports, an arriving flit that is still buffered in the current port would require multiple clock cycles before its transfer can be facilitated to a next-hop router. The timelines of the reconfigurable route 1 and solid route 2 are illustrated in Fig. 6. If there are many flits in the west port at this time, the solid routing 2 in yellow is the transmission route without reconfiguration, and the reconfigurable routing 1 in green is the reconfigured transmission route. A heavier load within the west port requires untimed waiting for arbitration (m cycles). Consider rephrasing for better clarity; the following is a provisional suggestion: Assuming that at this point, pursuant to a comparison of free buffer slots between two neighboring ports of the west port, it is ascertained that the north port has more idle buffers compared to the south port and that the arriving flit at this time is buffered to the north port, the required number of cycles of arbitration would be given by n, and the pipelines of the reconfigurable route 1 and solid route 2 would be as shown in Fig. 6b. Most likely, n is less than m, and ideally, n is equal to 0. If the reconfigurable route is not adopted, it may lead to a waiting time that is lengthier than normal as well as surpasses the overall runtime of the reconfigurable route, which would result in an adverse impact in terms of the overall network performance. This reflects the advantages of the TOR-NoC proposed in this paper, and we will subsequently confirm, through experimental results, that the reconfigurable router proposed in the present study facilitates an effective improvement of the network performance, over and above those obtained with the use of other comparative models. Our proposed reconfigurable NoC architecture can be easily scaled to any NoC.

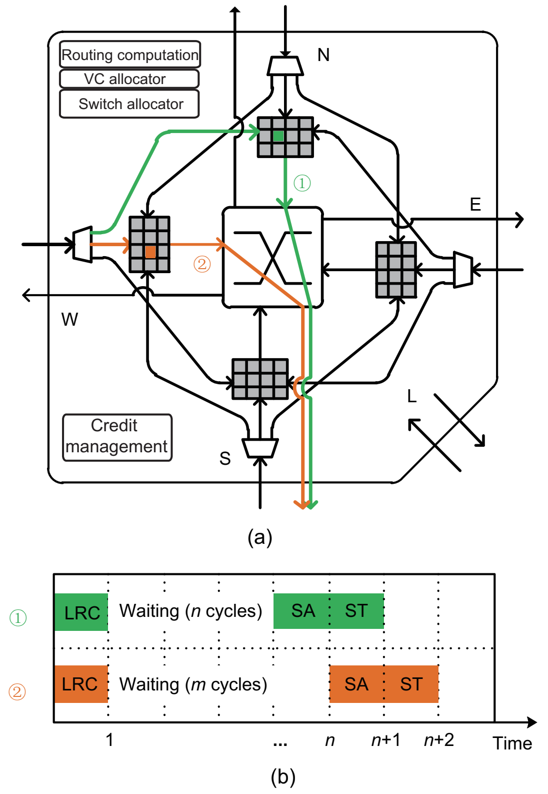


Fig. 6 Routing diagram (a) and pipeline diagram (b) of the reconfigurable route 1 and solid route 2 (References to color refer to the online version of this figure)

4 Experimental results

In this section, we evaluate our proposed design in terms of the overall network performance, area overhead, and power. We use a flit-level cycle-accurate NoC simulator, Noxim (Catania et al., 2017), to conduct a detail evaluation of different designs under discussion. The proposed NoC architecture is evaluated against the following two architectures: baseline and reconfigurable router in Said et al. (2021).

Baseline is the basic 2D mesh NoC without any reconfiguration.

The reconfigurable router in Said et al. (2021) is a modified router microarchitecture, in which the buffer slots are dynamically allocated for increasing router efficiency.

The basic configurations (without reconfiguration) for the routers in Said et al. (2021) and TOR-NoC are the same as those for the baseline router. Dimension-ordered XY routing is implemented in the experiment to guarantee deadlock-free routing, although our proposal is independent of the specific routing algorithm and the topology employed in the system. The configuration parameters for the proposed and baseline networks are listed in Table 2.

Table 2 The experimental parameters

Parameter	Value/Description
Topology	4×4 and 8×8 2D mesh
Channel width	64
Number of VCs	4 or dynamic
Input buffer	4-flit depth or dynamic
Routing	XY routing algorithm
Traffic pattern	Uniform, transpose, bit-reversal, and shuffle

4.1 Performance

To address the congestion problem caused by dynamically changing traffic characteristics, a load-balancing method based on inter-port buffer sharing is proposed. By reconfiguring the physical interconnection line logic in the input ports, the buffers between neighboring ports can be shared to improve the load carrying capacity of the ports and flexibly cope with the time-varying traffic. At the same time, the arbitration logic of cross-switches is optimized to ensure that packets can be transmitted to downstream routers faster. On this basis, all buffers are stored in a unified manner, and the data of each VC are distinguished through the chain table identification to maximize the utilization of buffer resources. This guarantees that the proposed reconfigurable approach balances hardware overhead while providing flexibility. The experimental results under comprehensive traffic load and real application load communication show that the reconfigurable method of load balancing with buffer sharing among ports is significantly superior in avoiding network congestion and balancing network load.

Various traffic patterns are used in the experiment, including synthetic traffic workloads. Accordingly, we provide the worst-case analysis for the TOR-NoC proposal in consideration of the performance obtained with the use of synthetic traffic. The more uneven the traffic loads are, the more performance assurance the TOR-NoC can provide against potential congestion. Moreover, the use of different injection loads, namely from low traffic up to network saturation, demonstrates the TOR-NoC effectiveness with different network loads.

The several synthetic traffic patterns include:

(1) uniform, where each node sends an equal amount of traffic to each destination; (2) permutation traffic patterns (transpose, bit-reversal, and shuffle), where each node selects a fixed destination based on the permutations. The results, presented in Fig. 7, show that the proposed reconfigurable architecture TOR-NoC can have obvious advantages in terms of average latency for all the four synthetic traffic patterns. Specifically, for bit-reversal and transpose traffic patterns, at the injection load of 0.4 flit/(node·cycle) under a 4×4 2D mesh, the decrements of network average latency over the corresponding value for baseline are about 36.6% and 38.3%, respectively. In bit-reversal and transpose traffic patterns, the traffic distribution across the entire network is relatively unbalanced, which can easily lead to congestion in the central area of the network. The flexible buffer-sharing mechanism in TOR-NoC can alleviate the load within the routers and accelerate subsequent packet transmission. The buffer storage is optimized in TOR-NoC compared to the model provided in Said et al. (2021), so the logical depth of each VC can extend to the entire buffer size. The results show that TOR-NoC reduces latency by 22.8% compared to Said et al. (2021)'s model at an injection load of 0.4 flit/(node·cycle) in transpose traffic pattern. Our proposed TOR-NoC adds reconfigurable routing paths for on-demand traffic and improves the flexibility of data transmission, thereby reducing the average network delay.

Furthermore, the throughput results presented in Fig. 8 show that the proposed reconfigurable architecture of TOR-NoC possesses obvious advantages in terms of saturation throughput for all the four synthetic traffic patterns. Specifically, for bit-reversal and transpose traffic patterns, at the injection load of 0.4 flit/(node·cycle) under a 4×4 2D mesh, the saturation throughput improvements over the corresponding value for baseline are about 32.6% and 25.1%, respectively. Note that the circumstance necessitating the dynamic allocation of the additional available slots to the most loaded port guarantees

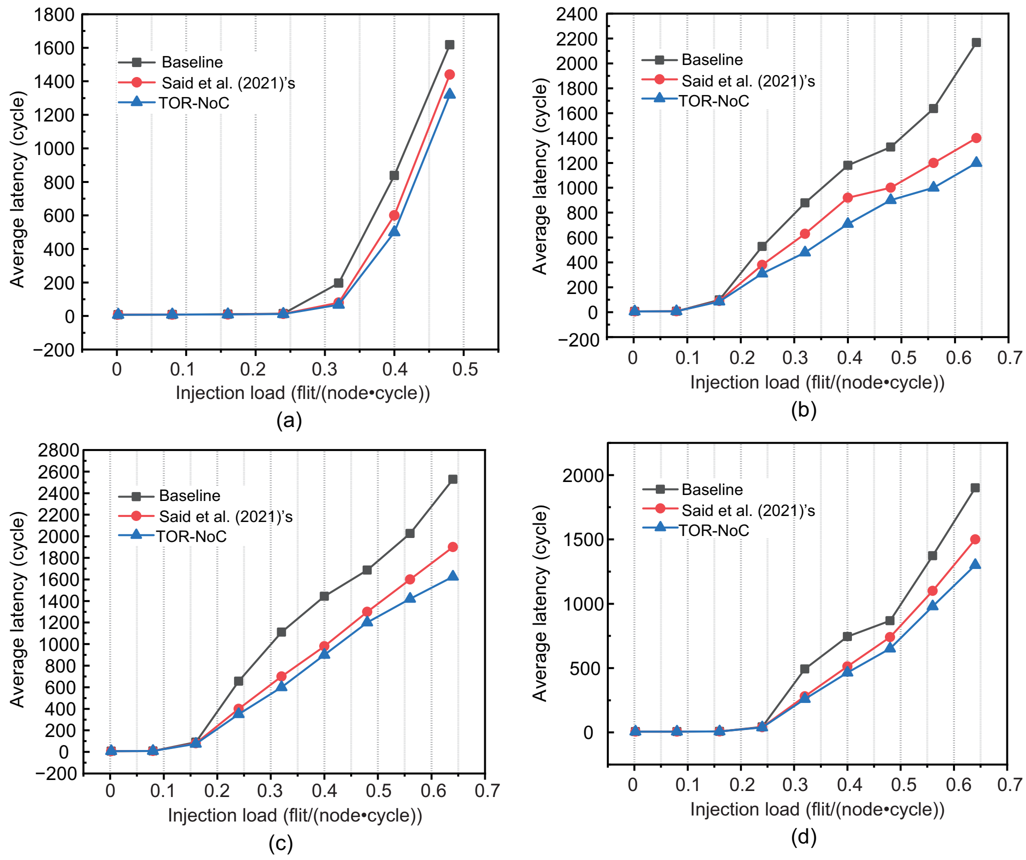


Fig. 7 Network average latency with different injection loads under different traffic workloads in a 4×4 2D mesh: (a) uniform; (b) transpose; (c) bit-reversal; (d) shuffle

congestion avoidance under unbalanced traffic load. With non-uniform traffic and increased traffic on certain ports, the schemes forming part of TOR and Said et al. (2021)'s model allow dynamic allocation of free buffers, alleviating potential congestion and thus bringing latency reduction and throughput improvement.

Our proposed TOR-NoC optimizes the virtual channel management mechanism compared to Said et al. (2021)'s model, reducing latency by 19.8% and increasing throughput by 10.2% at an injection load of 0.4 flit/(node·cycle) of the bit-reversal traffic pattern under a 4×4 2D mesh.

Similar results can be observed in Fig. 9, which depicts the latency comparison in an 8×8 2D mesh. The proposed TOR-NoC, vis-à-vis baseline, results in an average latency reduction of around 26.6% and 39.8% for transpose and bit-reversal traffic patterns respectively, with the injection load under 0.32 flit/(node·cycle). For other traffic patterns, TOR-NoC also leads to obvious reduction over other reconfigurable architectures in terms of average latency. In the 8×8 2D mesh topology, TOR-NoC reduces the latency compared to Said et al. (2021)'s model by 12.5% with transpose at the injection load of 0.4 flit/(node·cycle). With the network scale increased to 8×8, under conditions of uneven network traffic loads, congestion within routers becomes more pronounced. TOR-NoC demonstrates a faster transfer of congested packets to their destination nodes, resulting in a more noticeable reduction in latency compared to Said et al. (2021)'s model. From 4×4 to 8×8 topology extension, TOR-NoC can reduce network latency and show good scalability.

Fig. 10 shows that TOR-NoC is still effective with throughput for a larger NoC size. The proposed TOR-NoC results in around 32.5% and 44.7% saturation throughput improvements over baseline for transpose and bit-reverse traffic patterns, respectively, with the injection load under

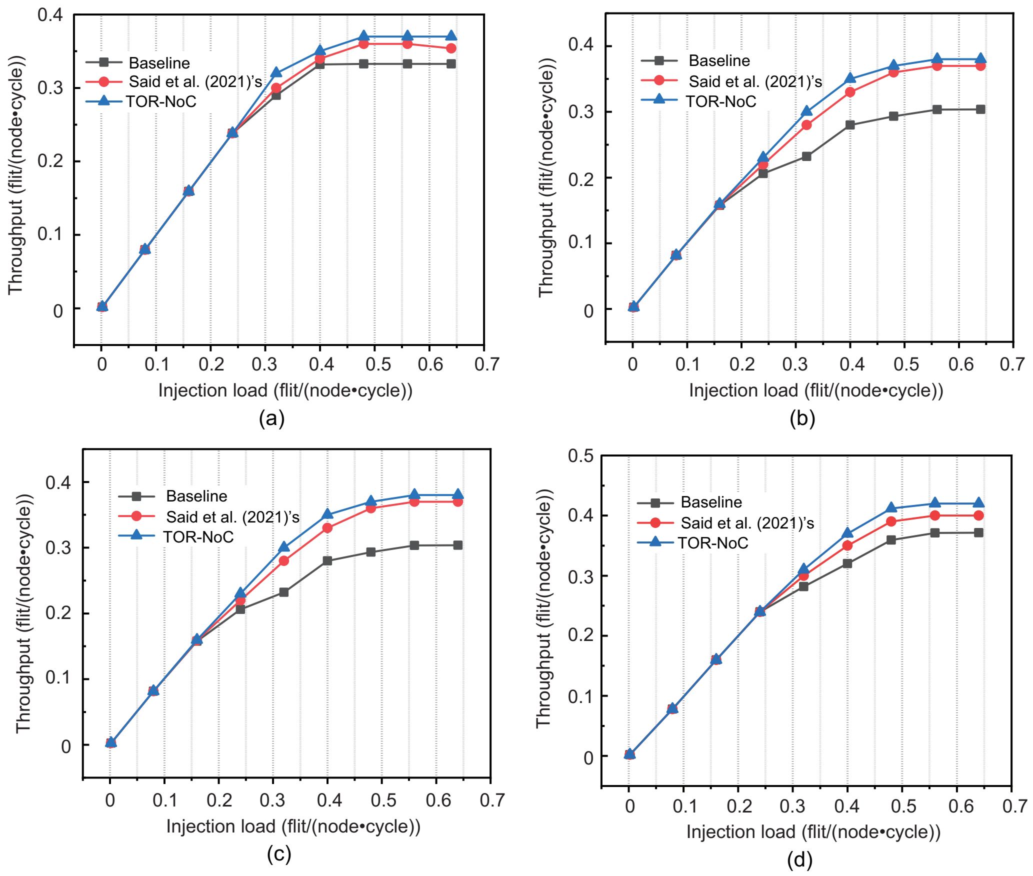


Fig. 8 Throughput with different injection loads under different traffic workloads in a 4×4 2D mesh: (a) uniform; (b) transpose; (c) bit-reversal; (d) shuffle

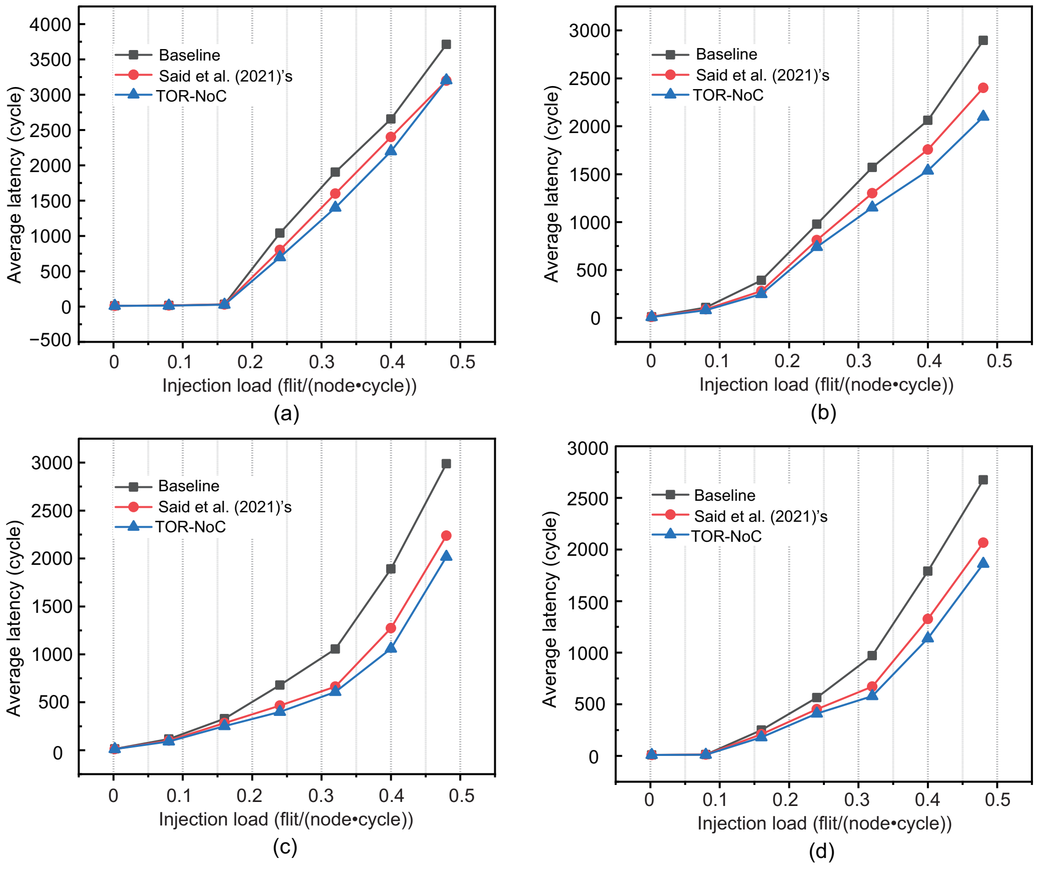


Fig. 9 Network average latency with different injection loads under different traffic workloads in an 8×8 mesh: (a) uniform; (b) transpose; (c) bit-reversal; (d) shuffle

0.4 flit/(node·cycle). For other traffic patterns, TOR-NoC also leads to obvious reduction over other reconfigurable architectures in terms of average latency. Our proposed TOR-NoC optimizes the VC management mechanism compared to Said et al. (2021)'s model, increasing throughput by 10.7% at the injection load of 0.4 flit/(node·cycle) under the 4×4 2D mesh. Under conditions of near-saturation injection load, TOR-NoC demonstrates a more pronounced improvement with an 18.7% increase in throughput compared to Said et al. (2021)'s model. Therefore, TOR-NoC has a high scalability. Our proposed reconfigurable NoC architecture can be easily scaled to any-scale NoC.

4.2 Overhead

Buffer utilization is the ratio of the number of non-free buffers to the number of total buffers (which include free buffers and occupied buffers) in the router. The higher the buffer utilization within the router, the more fully utilized the buffers. When the network injection load is low, the load pressure of the buffer is low, and the buffer utilization is low at this time. As the network injection load gradually increases to saturation, more and more packets are injected into the network and buffer utilization reaches its peak.

The microarchitecture of the router is implemented with Verilog Hardware Description Language (VHDL) and synthesized using the Synopsys Design Compiler, with a 45 nm Taiwan Semiconductor Manufacturing Company (TSMC) library. Area results take into account the entire router, which is composed of five input/output channels (local, east, west, south, and north), a crossbar, and a corresponding logic control unit. The proposed router is modified only in the implementation of the input channels, but after implementing or introducing all the changes in our scheme, a combined result including the entire router is obtained. The area of our proposed router is 12 362.5 μm^2 and the

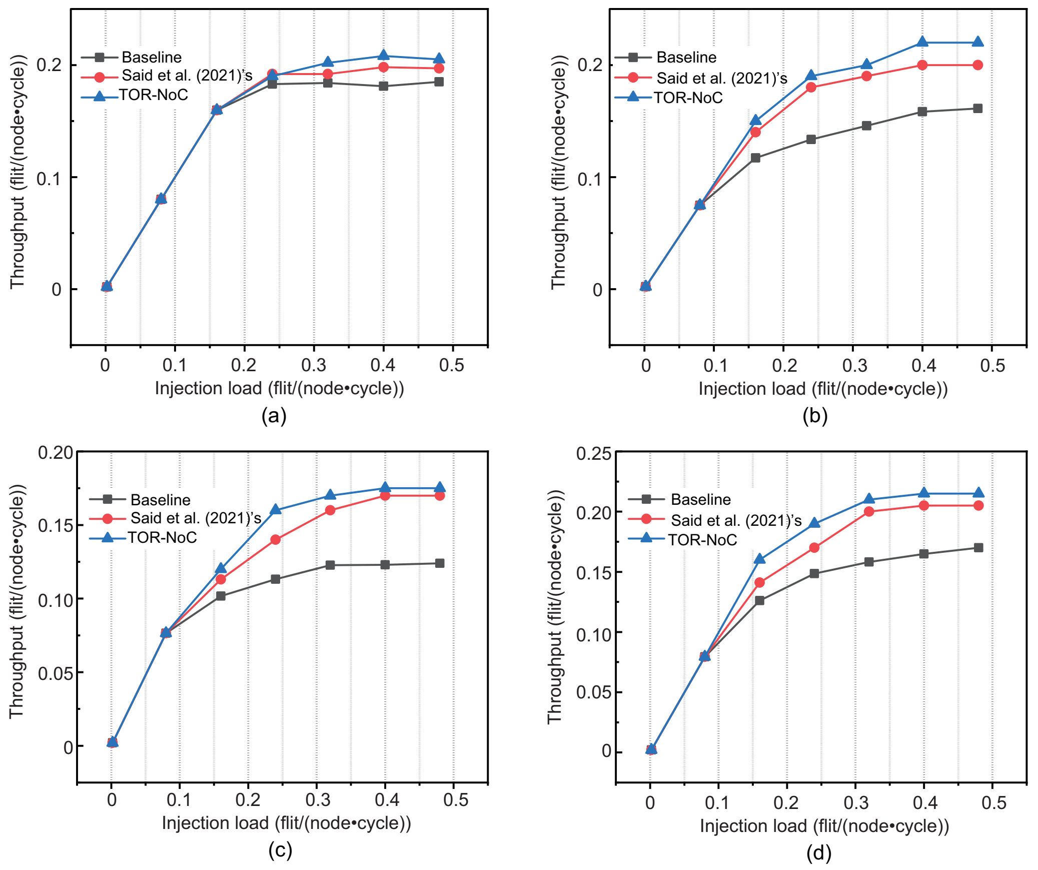


Fig. 10 Throughput with different injection loads under different traffic workloads in an 8×8 mesh: (a) uniform; (b) transpose; (c) bit-reversal; (d) shuffle

corresponding power dissipation is 19.6 mW. The synthesis results are shown in Table 3. Compared to Said et al. (2021)'s model, TOR-NoC has much less overhead, it does not have any VC allocator or channel controller, and its crossbar and switch allocator have lower overhead. TOR-NoC and Said et al. (2021)'s model have higher overhead than baseline, by 5.8% and 6.2% respectively. Therefore, the extra area overhead of TOR-NoC can be considered negligible without imposing a great burden on the system implementation.

Fig. 11a shows the normalized buffer utilization in uniform. The experimental results show that the buffer utilization ratio of TOR-NoC is higher than those of baseline and Said et al. (2021)'s model; that is, the buffer utilization ratio of TOR-NoC is increased by 27.06% compared with that of baseline and by 7.3% compared with that of Said et al. (2021)'s model. The buffer utilizations observed in TOR-NoC and Said et al. (2021)'s model are much higher than that of baseline, indicating that the buffer-sharing strategy can effectively balance the load within the router and disperse traffic on heavily loaded ports. The buffer utilization ratio of TOR-NoC is higher than that of Said et al. (2021)'s model, which proves the effectiveness of the reconfigurable transmission path in TOR-NoC, improves the diversity of the packet transmission path, and realizes a more efficient utilization of buffer resources. In the case of load imbalance within the router, the flexible reconfigurable path within the port can distribute the traffic from the heavily loaded port to other ports, balancing the load between ports within the router.

Fig. 11b shows the normalized buffer utilization in transpose. The experimental results show that the buffer utilization ratio of TOR-NoC is higher than those of baseline and Said et al. (2021)'s model; that is, the buffer utilization ratio of TOR-NoC is increased by 35.7% compared with that of baseline and by 12.1% compared with that of Said et al. (2021)'s model. A comparison of the buffer utilization ratio

Table 3 Area of the three router architectures

Module	Area (μm^2)		
	Baseline	Said et al. (2021)'s	TOR-NoC
Input port	10 147.8	10 832.6	10 755.4
Crossbar	301.3	306.5	309.1
Logic unit	1237.3	1273.8	1298.0
Total	11 686.4	12 412.9	12 362.5
Normalized	1.000	1.062	1.058

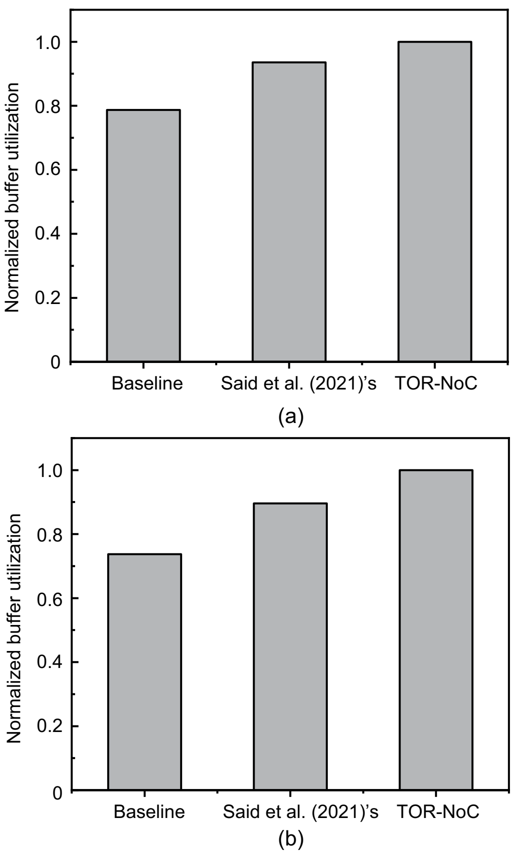


Fig. 11 Normalized buffer utilization: (a) uniform; (b) transpose

between transpose and uniform flow modes reveals that the buffer utilization of TOR-NoC in transpose flow is higher than that in uniform flow. This is due to the fact that the traffic of transpose is more concentrated in the network, the traffic of uniform is more distributed, and TOR-NoC can play the role of load balancing more effectively.

5 Conclusions

In this paper, traffic-oriented reconfigurable routers have been presented as an alternative for baseline routers. A reconfiguration mechanism is proposed that dynamically stores newly arriving packets on current or adjacent ports. At the same time, the use of centralized output-oriented buffer management facilitates a further enhancement in the performance of the reconfigurable NoC proposed in the present research. Therefore, our reconfigurable NoC provides the needed flexibility along with balancing hardware cost and communication. Future research on reconfigurable NoCs can target the means for designing more extensive reconfigurable NoCs for large multicore systems that are expected to arise in future, as well as the means for the prediction of real traffic behaviors using algorithms.

Contributors

Chenglong SUN designed the research. Yiming OUYANG and Huaguo LIANG processed the data and drafted the paper. Chenglong SUN helped organize, revised, and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Baharloo M, Khonsari A, 2018. A low-power wireless-assisted multiple network-on-chip. *Microprocess Microsyst*, 63:104-115.
https://doi.org/10.1016/j.micpro.2018.09.001

Castillo EV, Miorandi G, Chau WJ, 2014. DyAFNoC: characterization and analysis of a dynamically reconfigurable NoC using a DOR-based deadlock-free routing algorithm. Proc 8th IEEE/ACM Int Symp on Networks-on-Chip, p.190-191.
https://doi.org/10.1109/NOCS.2014.7008788

Catania V, Mineo A, Monteleone S, et al., 2017. Cycle-accurate network on chip simulation with Noxim. *ACM Trans Model Comput Simul*, 27(1):4.
https://doi.org/10.1145/2953878

Chen XN, Peh LS, 2003. Leakage power modeling and optimization in interconnection networks. Proc Int Symp on Low Power Electronics and Design, p.90-95.
https://doi.org/10.1145/871506.871531

Das TS, Ghosal P, Chatterjee N, 2021. VCS: a method of in-order packet delivery for adaptive NoC routing. *Nano Commun Netw*, 28:100333.
https://doi.org/10.1016/j.nancom.2020.100333

Farrokhbakht H, Kao H, Jerger NE, 2019. UBERNoC: unified buffer power-efficient router for network-on-chip. Proc 13th IEEE/ACM Int Symp on Networks-on-Chip, p.1-8.
https://doi.org/10.1145/3313231.3352362

Jain A, Laxmi V, Tripathi M, et al., 2020. TRACK: an algorithm for fault-tolerant, dynamic and scalable 2D mesh network-on-chip routing reconfiguration. *Integration*, 72:92-110.
https://doi.org/10.1016/j.vlsi.2020.01.005

Jerger NE, Krishna T, Peh LS, 2017. On-Chip Networks (2nd Ed.). Springer, Cham, Germany.
https://doi.org/10.1007/978-3-031-01755-1

Jindal N, Gupta S, Ravipati DP, et al., 2020. Enhancing network-on-chip performance by reusing trace buffers. *IEEE Trans Comput-Aided Des Integr Circ Syst*, 39(4):922-935.
https://doi.org/10.1109/TCAD.2019.2907909

Krishna T, Chen CHO, Kwon WC, et al., 2013a. Breaking the on-chip latency barrier using SMART. Proc IEEE 19th Int Symp on High Performance Computer Architecture, p.378-389.
https://doi.org/10.1109/HPCA.2013.6522334

Krishna T, Chen CHO, Park S, et al., 2013b. Single-cycle multihop asynchronous repeated traversal: a smart future for reconfigurable on-chip networks. *Computer*, 46(10):48-55.
https://doi.org/10.1109/MC.2013.260

Kumar A, Peh LS, Kundu P, et al., 2007. Express virtual channels: towards the ideal interconnection fabric. *ACM SIGARCH Comput Archit News*, 35(2):150-161.
https://doi.org/10.1145/1273440.1250681

Kumar S, Jantsch A, Soininen JP, et al., 2002. A network on chip architecture and design methodology. Proc IEEE Computer Society Annual Symp on VLSI. New Paradigms for VLSI Systems Design, p.117-124.
https://doi.org/10.1109/ISVLSI.2002.1016885

Lan YC, Lin HA, Lo SH, et al., 2011. A bidirectional NoC (BiNoC) architecture with dynamic self-reconfigurable channel. *IEEE Trans Comput-Aided Des Integr Circ Syst*, 30(3):427-440.
https://doi.org/10.1109/TCAD.2010.2086930

Li J, Qin CQ, Sun XC, 2023. An efficient adaptive routing algorithm for the co-optimization of fault tolerance and congestion awareness based on 3D NoC. *Microelectron J*, 142:105989.
https://doi.org/10.1016/j.mejo.2023.105989

Matos D, Concatto C, Kreutz M, et al., 2011. Reconfigurable routers for low power and high performance. *IEEE Trans Very Large Scale Integr (VLSI) Syst*, 19(11):2045-2057. https://doi.org/10.1109/TVLSI.2010.2068064

Nguyen HK, Tran XT, 2019. A novel reconfigurable router for QoS guarantees in real-time NoC-based MPSoCs. *J Syst Archit*, 100:101664.
https://doi.org/10.1016/j.sysarc.2019.101664

Nicopoulos CA, Park D, Kim J, et al., 2006. ViChaR: a dynamic virtual channel regulator for network-on-chip routers. Proc 39th Annual IEEE/ACM Int Symp on Microarchitecture, p.333-346.
https://doi.org/10.1109/MICRO.2006.50

Ouyang YM, Sun CL, Jia BY, et al., 2021. Architecting a priority-based dynamic media access control mechanism in wireless network-on-chip. *Microelectron J*, 116:105218.
https://doi.org/10.1016/j.mejo.2021.105218

Ouyang YM, Sun CL, Li RF, et al., 2023. Transit ring: bubble flow control for eliminating inter-ring communication congestion. *J Supercomput*, 79(2):1161-1181.
https://doi.org/10.1007/s11227-022-04712-z

Oveis-Gharan M, Khan GN, 2016. Efficient dynamic virtual channel organization and architecture for NoC systems. *IEEE Trans Very Large Scale Integr (VLSI) Syst*, 24(2):465-478.
https://doi.org/10.1109/TVLSI.2015.2405933

Oveis-Gharan M, Khan GN, 2020. Reconfigurable on-chip interconnection networks for high performance embedded SoC design. *J Syst Archit*, 106:101711.
https://doi.org/10.1016/j.sysarc.2020.101711

Qian ZL, Abbas SM, Tsui CY, 2015. FSNoC: a flit-level speedup scheme for network on-chips using self-reconfigurable bidirectional channels. *IEEE Trans Very Large Scale Integr (VLSI) Syst*, 23(9):1854-1867.
https://doi.org/10.1109/TVLSI.2014.2351833

Said M, Sarihi A, Patooghy A, et al., 2021. Novel flexible buffering architectures for 3D-NoCs. *Sustain Comput Inform Syst*, 29:100472.
https://doi.org/10.1016/j.suscom.2020.100472

Seitanidis I, Psarras A, Chrysanthou K, et al., 2015. ElastiStore: flexible elastic buffering for virtual-channel-based networks on chip. *IEEE Trans Very Large Scale Integr (VLSI) Syst*, 23(12):3015-3028.
https://doi.org/10.1109/TVLSI.2014.2383442

Stensgaard MB, Sparsø J, 2008. ReNoC: a network-on-chip architecture with reconfigurable topology. Proc 2nd ACM/IEEE Int Symp on Networks-on-Chip, p.55-64.
https://doi.org/10.1109/NOCS.2008.4492725

Stuart MB, Stensgaard MB, Sparsø J, 2011. The ReNoC reconfigurable network-on-chip: architecture, configuration algorithms, and evaluation. *ACM Trans Embed Comput Syst*, 10(4):45.
https://doi.org/10.1145/2043662.2043669

Wang HS, Peh LS, Malik S, 2003. Power-driven design of router microarchitectures in on-chip networks. Proc 36th Annual IEEE/ACM Int Symp on Microarchitecture, p.105-116.
https://doi.org/10.1109/MICRO.2003.1253187

Wang L, Liu LB, Han J, et al., 2020. Achieving flexible global reconfiguration in NoCs using reconfigurable rings. *IEEE Trans Parall Distrib Syst*, 31(3):611-622.
https://doi.org/10.1109/TPDS.2019.2940190

Wu YB, Liu LB, Wang L, et al., 2020. Aggressive fine-grained power gating of NoC buffers. *IEEE Trans Comput-Aided Des Integr Circ Syst*, 39(11):3177-3189.
https://doi.org/10.1109/TCAD.2020.3012170

Ye TT, De Micheli G, Benini L, 2002. Analysis of power consumption on switch fabrics in network routers. Proc 39th Annual Design Automation Conf, p.524-529.
https://doi.org/10.1145/513918.514051

Zheng H, Wang K, Louri A, 2021. Adapt-NoC: a flexible network-on-chip design for heterogeneous manycore architectures. Proc IEEE Int Symp on High-Performance Computer Architecture, p.723-735.
https://doi.org/10.1109/HPCA51647.2021.00066

Zoni D, Flich J, Fornaciari W, 2016. CUTBUF: buffer management and router design for traffic mixing in VNET-based NoCs. *IEEE Trans Parall Distrib Syst*, 27(6):1603-1616.
https://doi.org/10.1109/TPDS.2015.2468716