Microprocessors and Microsystems 85 (2021) 104242

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

MMNNN: A tree-based Multicast Mechanism for NoC-based deep Neural Network accelerators

Yiming Ouyang, Feiyang Tang *, Chunlei Hu, Wu Zhou, Qi Wang

School of Computer Science and Information Engineering, Hefei University of Technology, 485 Danxia Road, Hefei 230601, China

ARTICLE INFO

Keywords:
Network-on-Chip
Deep Neural Network (DNN) accelerator
Multicast routing algorithm
Router architecture

ABSTRACT

Network-on-Chip (NoC) devices have been widely used in multiprocessor systems. In recent years, NoC-based Deep Neural Network (DNN) accelerators have been proposed to connect neural computing devices using NoCs. Such designs dramatically reduce off-chip memory accesses of these platforms. However, the large number of one-to-many packet transfers significantly degrade performance with traditional unicast channels. We propose a multicast mechanism for a NoC-based DNN accelerator called Multicast Mechanism for NoC-based Neural Network accelerator (MMNNN). To do so, we propose a tree-based multicast routing algorithm with excellent scalability and the ability to minimize the number of packets in the network. We also propose a router architecture for single-flit packets. Our proposed router transfers flits to multiple destinations in a single process and has no head-of-line blocking issue, offering higher throughput and lower latency than traditional wormhole router architectures. Simulation results show that our proposed multicast mechanism offers excellent performance in classification latency, average packet latency, and energy consumption.

1. Introduction

The Network-on-Chip (NoC) architecture provides high scalability, high throughput, and low-latency interconnection networks in multiple core devices [1]. Traditional shared bus structures can connect only a small number of components, and they do not accommodate a large number of processors [2]. Such traditional buses have become the bottlenecks in recent multiprocessor systems.

A NoC is a type of packet-switched network that is composed of routers and links. Significant research has been made into both wired [3–5] and wireless NoCs [6–9]. In a typical NoC, each router is connected to a processor element (PE) or group of PEs, with routers connected to each other by links [10]. Different PEs communicate with each other through the NoC, with topology determining the placement of routers and links. The 2D mesh topology is most common because it works well with planar silicon geometry and provides better scalability and higher throughput [2]. Routing algorithms select packet paths between source and destination routers in the topology. The routers are the most significant part of a NoC: receiving packets in its input buffers, selecting output ports for packets based on the routing algorithm, and forwarding them to the next router [11].

A new NoC-based DNN accelerator design paradigm has been proposed more recently [10]. There are also many spiking neural network accelerators (such as SpiNNaker [12,13]), DNN accelerators (such as Eyriss-v2 [14] and Neu-NoC [15]), and DNN accelerator simulators (such as NN-Noxim [16], CNN-Noxim [17], and DNNoC-Sim [18]) developed using NoCs. NoCs can dramatically reduce off-chip memory accesses and enhance design flexibility [10]. The multiple core designs mentioned previously fully utilize the high throughput and low communication latency characteristics of NoCs.

The traffic in NoC-based DNN accelerators is very different from traditional traffic models in NoCs. Large volumes of one-to-many traffic in this kind of accelerator [19] significantly degrades performance using traditional unicast methods. Although we can implement multicast using multiple unicast packets, this injects too many identical packets into the network, significantly increasing the queuing latency of packets and degrading performance. Similar to the dataflow architecture [20], NoC-based DNN accelerators also face the problem of a high packet injection rate and require low transfer latency to achieve better performance.

The best way to solve this problem is to design an effective multicast routing algorithm and a low-latency, high-throughput multicast router architecture. In this paper, we propose a multicast mechanism for a NoC-based DNN accelerator. This approach minimizes the number of packets in the network and decreases classification latency. Our proposed mechanism includes a multicast routing algorithm and router design.

The most commonly used multicast routing methods are tree-based [21,22] and path-based [23,24]. In a tree-based multicast routing

* Corresponding author.
E-mail address: tangfeiyang@mail.hfut.edu.cn (F. Tang).

https://doi.org/10.1016/j.micpro.2021.104242
Received 2 November 2020; Received in revised form 20 February 2021; Accepted 25 March 2021
Available online 14 June 2021

algorithm, the fundamental problem is to determine when and where to replicate multicast packets [2]. A poor replication decision uses a large number of links to transmit multicast packets, which increases network latency, exhausts network bandwidth quickly, and increases network power consumption. In our method, we first modify the traditional neuron clustering strategy proposed by Chen et al. [16,17]. We then propose a tree-based multicast routing algorithm suited to the neuron clustering and mapping strategy. Our proposed routing algorithm transmits a packet to a large number of destination nodes using a small number of bits in the packet and ensuring that a minimum number of links are used to transmit the packet to multiple destination nodes.

Also, using multi-flit packets to transmit data increases packet latency. Thus, in our method, we use single-flit packets to transmit data in the network. To support single-flit packets in a NoC, we design a router architecture that also supports the multicast routing algorithm mentioned above. This router architecture makes fuller use of the buffers compared to the router architecture proposed by Shen et al. [20] to yield better performance.

Our main contributions are summarized as follows:

- We modify the neuron clustering strategy proposed by Chen et al. [16,17] to develop a tree-based multicast routing algorithm for supporting multicast traffic in a NoC.
- We propose a single-flit packet format to carry data in a NoC. We also propose a router architecture that abandons the traditional wormhole architecture to support multicast traffic. This method has no head-of-line blocking issue [25] and improves buffer utilization, to achieve better NoC performance.
- We evaluate different multicast mechanisms by mapping deep neural networks of different sizes onto NoCs. Detailed simulation results show that our proposed mechanism significantly reduces the classification latency, average packet latency, and the number of routed packets.

This paper is organized as follows. Section 2 reviews related works. In Section 3, we present the modified neuron clustering strategy of our NoC-based DNN accelerator and the multicast routing algorithm based on it. Section 4 presents the single-flit packet format and a router architecture to support multicast traffic in NoCs. In Section 5, we present our evaluation methodology and experiment results. In Section 6, we present our conclusions.

2. Related work

2.1. NoC-based DNN accelerator

Currently, the most common DNN accelerator platforms are based on CPUs, GPUs, ASICs, and FPGAs. Multi-core CPUs (such as the 48-core Qualcomm Centriq 2400 [26] and the 72-core Intel Xeon Phi [27]) have excellent computing power. GPUs are also very popular for DNNs because of GPUs' intrinsic compatibility with parallel computing. However, CPUs and GPUs are designed for general-purpose computations and consume more power as a result [10]. FPGAs are widely used in DNN accelerators because of their re-programmability [28], which allows for a more flexible design than ASICs. In ASIC-based designs, on-chip computing units are optimized for particular applications [29–32], achieving better performance and power efficiency than their CPU and GPU counterparts. ASIC-based DNN accelerators are designed for a specific DNN model, with better performance and lower power consumption [10]. In contrast to CPUs and GPUs, ASIC- and FPGA-based DNN accelerators are less reconfigurable at runtime.

Currently, popular DNN models such as AlexNet [33] and VGG-16 [34] consist of a large number of neurons and parameters, which result in high computational complexity. In order to accelerate the DNN computations, it is necessary to employ computational parallelism. Since the DNNs have a multi-layer structure, accelerators can make use of multi-core architectures. Hence, NoC-based DNN accelerator architectures have been proposed due to NoCs' intrinsic ability to provide high-bandwidth and low-latency communications for multi-core systems.

A NoC-based DNN accelerator design has been proposed by Chen et al. [10]. The authors analyze the advantages and disadvantages of traditional neural network accelerators and propose a NoC-based DNN accelerator design. Their design decouples the DNN operations, performing computations in a PE and data transmission in a NoC. This NoC supports different DNN data flows, increasing design flexibility and runtime reconfigurability. The experimental results show that their approach dramatically reduces off-chip memory accesses, which reduces power consumption and enhances performance enhancement.

A DNN accelerator called Neu-NoC has also been proposed [15]. Neu-NoC is a hybrid ring-mesh NoC. In Neu-NoC, neurons in the same layer are connected by a ring, and neurons in the same ring share the same data to improve packet transmission efficiency. The authors also analyze the characteristics of data traffic in the accelerator and propose a sophisticated mapping algorithm that is neural network-aware and a multicast transmission scheme. This scheme balances traffic and reduces redundant network traffic. However, this multicast scheme requires many address bits in the packet, which increases hardware overhead. The ring topology in Neu-NoC also suffers from lower throughput and higher latency.

Other researchers [16–18] have proposed a series of neural network accelerator simulators based on Noxim [35]. NN-Noxim [16] is a cycle-accurate NoC simulator that can support the artificial neural network (ANN) model. This simulator maps neuron clusters onto a NoC and executes computational tasks. CNN-Noxim [17] is designed for the CNN model. This simulator first flattens the CNN model to an ANN model and then maps the neuron clusters onto the NoC. DNNoC-sim [18] dynamically maps neurons onto a NoC and is suitable for large-scale DNN models on a resource-constrained NoC platform. These papers have proposed many clustering algorithms, mapping algorithms, and flow control methods in PE that inspire our work. However, none of these papers present an effective multicast scheme. That means the transmission of massive numbers of identical packets degrades the performance of the accelerator. This is the problem we are discussing in this paper.

2.2. Multicast mechanism

Broadcast (one-to-all) and multicast (one-to-many) are traffic patterns that send the same message from a source node to multiple destination nodes. A growing number of parallel applications make multicast services necessary [2]. Multicast communication has been widely applied in distributed systems [36], local-area networks [37], and computer networks [38].

In ANN or DNN models, each neuron in the next layer usually receives output data from all of the neurons in the previous layer before neuron in the next layer can be computed. Therefore, a suitable multicast mechanism must be designed for NoC-based DNN accelerators. The most commonly used multicast routing methods are tree-based [21,22] and path-based [23,24]. In tree-based methods, a multicast packet is first transmitted along a common path as far as possible. Then, this packet is replicated into multiple copies based on the location of destination nodes and the multicast routing algorithm. Routers transmit these copies to different destination nodes. In path-based methods, a multicast packet selects a path that allows all of the destination nodes to be concatenated into the packet and transmits the packet on this path. In path-based methods, the packet is replicated only when it goes to a destination node. In tree-based methods, blocking of one branch affects the transmission of another branch. Tree-based methods are prone to congestion, but the average number of packet hops is shorter than with path-based methods. Path-based methods are not prone to blocking, but the average number of packet hops is longer.

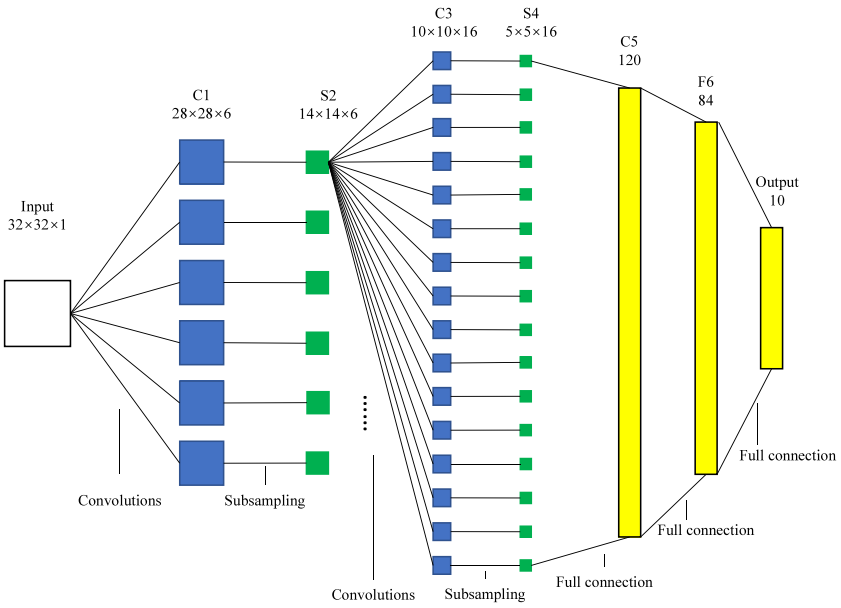


Fig. 1. Lenet-5.

Researchers have proposed many multicast mechanisms for NoC environments. One multicast mechanism called Virtual Circuit Tree Multicasting (VCTM) [39] sends a setup packet to build a tree before transmitting the multicast packets. Multicast packets are transmitted through the established tree. VCTM needs extra storage to maintain the multicast tree, increasing the chip area and energy consumption. VCTM also uses extra cycles to send setup packets for the multicast tree, increasing network transmission latency. VCTM also needs to build a large number of multicast trees to serve different sets of nodes, limiting VCTM's scalability.

Another multicast mechanism called Recursive Partitioning Multicast (RPM) [2] divides the entire network into eight sections based on the location of the packet. The output ports of a packet are determined by the distribution of the packet's destination nodes among the eight sections. Compared to VCTM, RPM eliminates the need to set up VCT and CAM tables in routers, reducing the area and power consumption of routers. RPM packet headers contain a bit string of the same length as the number of network nodes. These header bits record the location of the destination nodes of the packet. For large networks, this scheme increases hardware requirements. RPM uses two virtual networks to avoid deadlock, and these networks may create a traffic imbalance and degrade performance.

Others have analyzed the traffic characteristics of the dataflows of multi-core architectures and proposed an effective multicast routing algorithm called XY and a corresponding router architecture [20]. Their proposed routing algorithm is based on XY and supports four-address multicasting. Their router architecture uses single-flit (or non-flit) packets. When a packet comes in from a port, the router first calculates the output ports of the packet based on its routing information and then writes the packet to the corresponding buffers based on the output ports. This approach eliminates the head-of-line blocking issue and supports only single-flit packets, leading to lower latency and higher throughput than the traditional wormhole router architecture. However, this architecture does not have efficient and straightforward dynamic buffer management, which reduces buffer utilization.

In this paper, we propose a multicast routing algorithm and router architecture specifically for DNN accelerators based on the router architecture proposed by Shen et al. [20]. Our proposed multicast mechanism supports more nodes and offers smarter dynamic buffer management than existing approaches.

3. Multicast routing algorithm

In this section, we introduce our clustering and mapping strategies for a DNN accelerator and incorporate them into our proposed multicast routing algorithm.

3.1. Neural network reshape

In a NoC, data are packetized and transmitted. If each PE only computes one neuron process, the traffic load increases dramatically in a large scale DNN. Heavy traffic loads significantly degrade the performance of the platform. Thus, we design better clustering and mapping strategies to map multiple neurons to an appropriate PE.

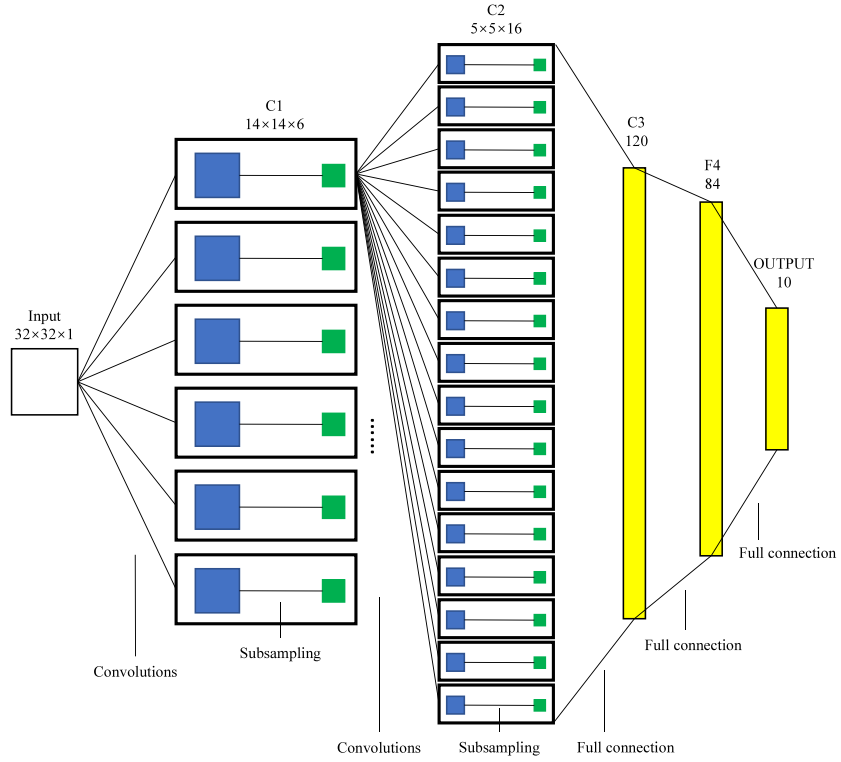
We mainly focus on two types of neural networks: fully connected artificial neural networks (ANNs) and convolutional neural networks (CNNs).

For fully connected ANNs, we use a clustering strategy and the Lyr_X mapping strategy [16]: neurons in the same layer are clustered according to specific group size, and clusters are mapped into the PEs along the positive X-axis in the mesh network. In this strategy, each row in the mesh network maps only neurons in the same layer.

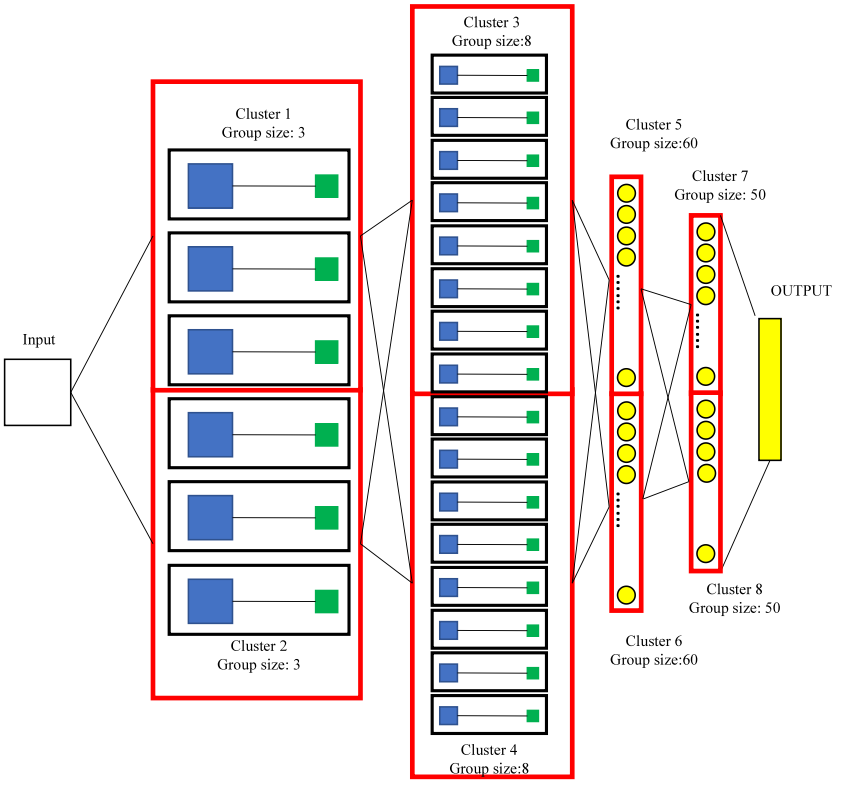
For CNNs, we use our own proposed clustering strategy. We use Lenet-5 [40], a simple CNN shown in Fig. 1, to describe our clustering strategy.

As shown in Fig. 1, there is often a pooling layer (such as S2 and S4) after a convolutional layer (such as C1 and C3). A pooling layer reduces the size of the feature map output from the convolutional layer while preserving dominant features of the feature maps. There is a one-to-one match between each neuron in a convolutional layer and its counterpart in pooling layer. If we map all of the convolutional and pooling layers into the network, the network will transmit a large number of feature maps, increasing the number of packets in the network and degrading performance. Thus, mapping the pooling layer neurons and their convolutional layer counterparts to the same PE reduces the number of transmitted packets. The pooling layer can also use the convolutional layer's feature maps in the PE to calculate results.

In our clustering strategy, we treat each neuron in a convolutional layer and its counterpart in a pooling layer as a single neuron, as shown in Fig. 2. In other words, we merge the convolutional and pooling layers of the CNN, referring to these merged layers as convolutional

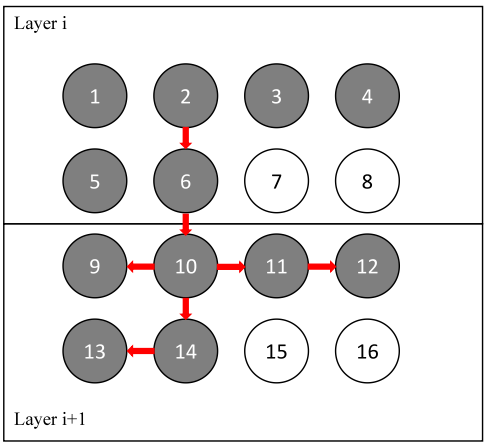


(a)

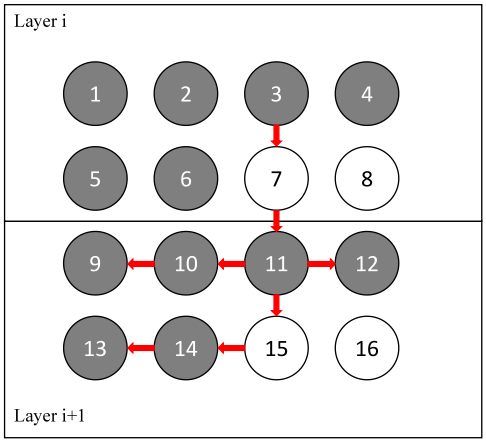


(b)

Fig. 2. Clustering strategies: (a) merging strategy, and (b) clustering strategy.



(a)



(b)

Fig. 3. Routing algorithm Neural Network Multicast Routing Algorithm (NNMRA) examples: (a) node 2 in layer i sends packets to the nodes in layer $i+1$; (b) node 3 in layer i sends packets to the nodes in layer $i+1$.

layers. Doing so transforms the CNN into a neural network similar to a fully connected ANN. In this case, the number of neurons in different convolutional layers varies considerably, making it inappropriate to specify a fixed group size for each convolutional layer. Therefore, different group sizes should be specified for different convolutional layers. We specify the maximum number of PEs that can be occupied for each convolutional layer. We refer to this maximum number as *MPC* and the number of neurons in each convolutional layer as *NM*, with the group size of each convolutional layer being $\lceil NM/MPC \rceil$. In actual use, the minimum value of the group size is 1.

Fig. 2(b) shows the MPC2F50 clustering strategy in Lenet-5. MPC2 means that each convolutional layer can be mapped to at most two PEs, and F50 means that the group size of the fully connected layer is 50. Hence, the group size is 3 in C1, 8 in C2, 60 in C3, and 50 in F4. Similar to the clustering strategy proposed by Chen et al. [16], our cluster strategy creates clusters from neurons in the same layer.

After clustering, we map each cluster to an appropriate PE in the mesh network according to the Lyr_X mapping strategy [16].

After we cluster and map a neural network onto a NoC according to these strategies (Figs. 2 and 9), we find that each neuron in layer i needs to send its computational results to each cluster in layer $i+1$. For example, all of the neurons in Clusters 1 and 2 of layer C1 need to send their computation results to Cluster 3 and Cluster 4 of layer C2. This is clearly a one-to-many traffic pattern. Multicasting using multiple unicast packets degrades the network performance of the network, increases communication delays, and increases power consumption. Therefore, a suitable multicast routing algorithm needs to be designed to solve this problem.

3.2. Multicast routing algorithm

After mapping neuron clusters to a mesh network, we find that proposed clustering and mapping strategies have high spatial locality. Suppose there are 6 clusters in layer i and layer $i+1$ of a neural network. After mapping the clusters to network, nodes 1 to 6 are neuron clusters from layer i, and nodes 9 to 14 are neuron clusters from layer $i+1$ as shown in Fig. 3. Nodes 7, 8, 15, and 16 have no clusters. In this case, each row in the mesh network has only clusters from the same layer, and these clusters are arranged in left-to-right order.

Based on the properties in the previous paragraph, we propose a tree-based multicast routing algorithm called NNMRA, as shown in Algorithm 1. To guide the packet branch, we define several flag bits in this routing algorithm. *row_layer* denotes the layer number of the neuron clusters mapped in the row, and *packet_layer* denotes *row_layer* of the destination nodes set. For example, in Fig. 3(a), if node 2 in layer i needs to send packets to nodes 9 to 14 in layer $i+1$, then node 2's *row_layer* is i, and *packet_layer* of the packets it sends is $i+1$. Although there are no clusters on nodes 7 and 8, their *row_layer* is i.

In NNMRA, *input_port* refers to the input port receiving the packet. For a given node, its *valid_in_PE* indicates whether the node has clusters or not. For this node, *valid_in_west* indicates whether its neighboring western node has clusters, and *valid_in_east* indicates whether its neighboring eastern node has clusters. Further, *valid_in_south* of this node indicates whether the *row_layer* in its neighboring south node is equal to its *row_layer*. For example, in Fig. 3(a), the *valid_in_PE*, *valid_in_west*, *valid_in_east*, and *valid_in_south* values of node 2 are 1, 1, 1, 1. At node 6, these flag bits are 1, 1, 0, 0. At node 7, these flag bits are 0, 1, 0, 0.

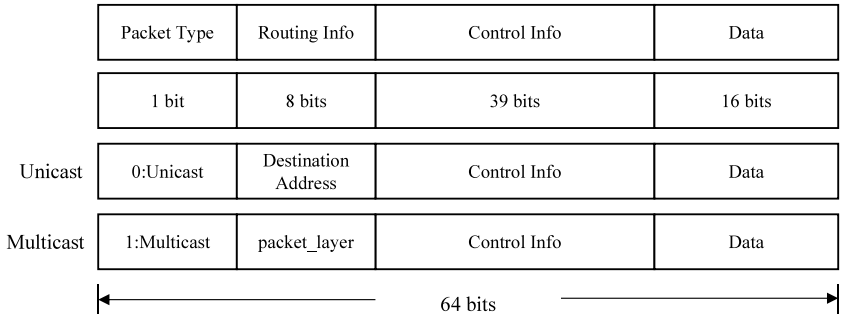


Fig. 4. Supported packet formats.

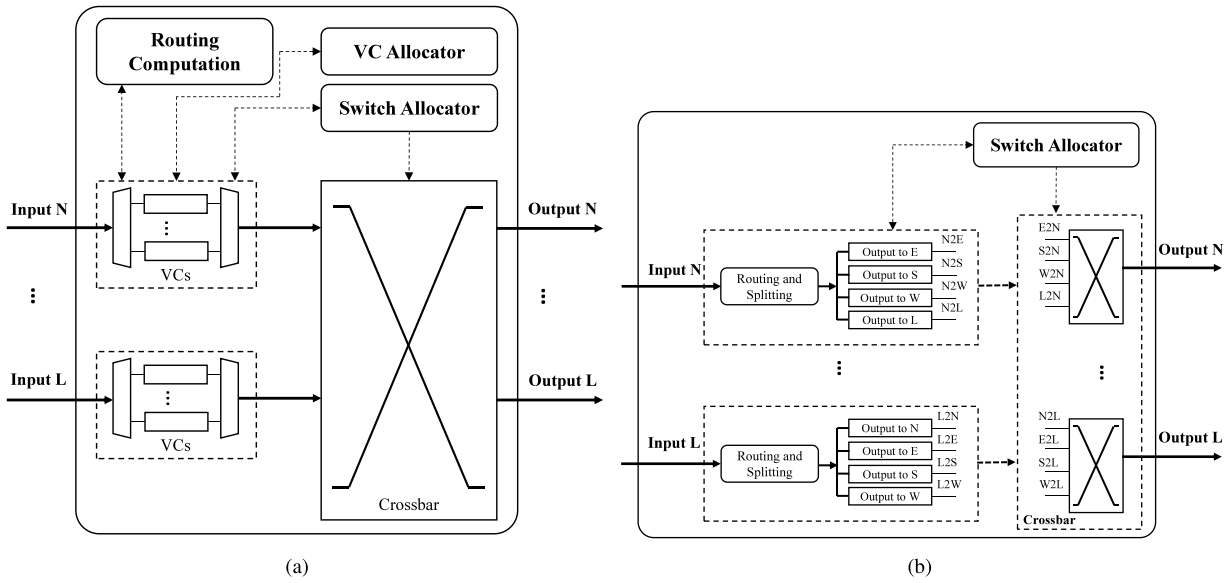


Fig. 5. Traditional NoC router architecture: (a) traditional VC router architecture; (b) the router from the data flow architecture [20].

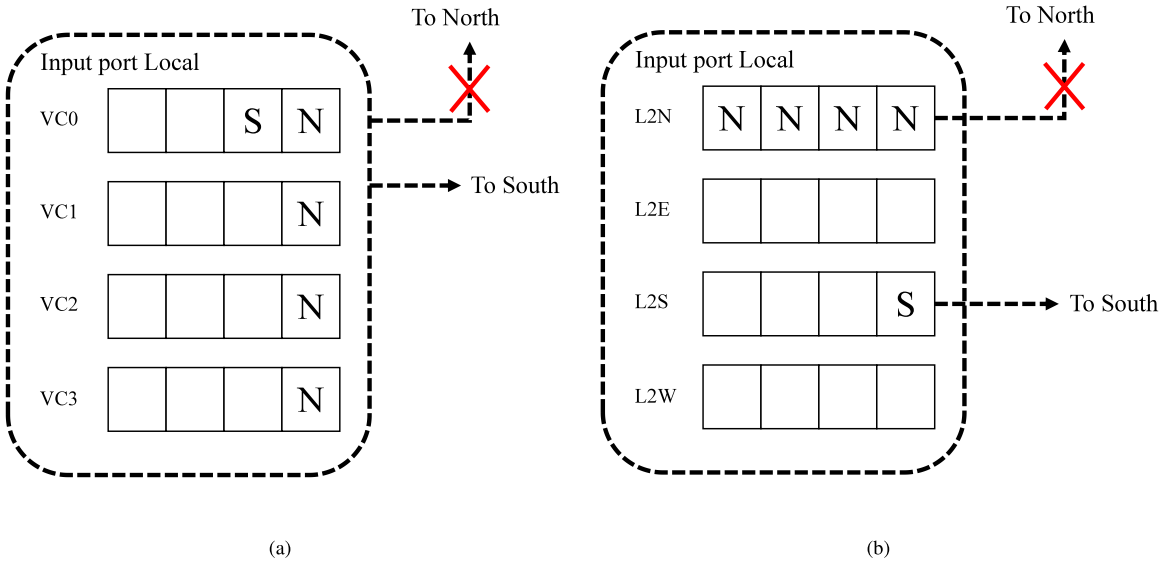


Fig. 6. Analyses of head-of-line blocking issues in different router architectures: (a) traditional VC router architecture; (b) the router from the data flow architecture [20].

Examples of NNMRA execution are shown in Fig. 3. In Fig. 3(a), one packet is sent from source node 2 in layer i to 6 destination nodes: nodes 9 to 14 in layer $i+1$. As mentioned above, *packet_layer* of the packet sent by node 2 is $i+1$. In nodes 2 and 6, the packet is transmitted to the south because the packet's *packet_layer* is larger than the *row_layer* in routers. In node 10, the packet reaches layer $i+1$ and begins to branch to the destination nodes. We use the flag bits mentioned above to limit unnecessary branches of the packet. For example, in node 14, flag bits are 1, 1, 0, 0, so the packet will be transmitted to local and west.

The transmission process in Fig. 3(b) is similar to that in Fig. 3(a). However, we note that node 15's *valid_in_PE* is 0, and the destination nodes are concentrated on its left side so that the packet will be transmitted to west.

The NNMRA algorithm is a multicast routing algorithm based on the YX routing algorithm, making it deadlock-free. NNMRA combined with the mapping and clustering strategies mentioned above reduce the number of links used in transmission. The four flag bits limit the branching of the data packet. Compared with traditional multicast routing algorithms, NNMRA uses layer numbers for multicasting, which reduces the number of address bits in the packet and the hardware demands.

4. Router architecture

4.1. Packet format

Routers commonly used in NoCs adopt multi-flit mechanisms to reduce the buffer and data bus sizes. A multi-flit mechanism reduces the size of the needed data buffers in routers. With the multi-flit mechanism, big packets are divided into several flits, with the flit being the smallest transmission unit in the network. Thus, the multi-flit mechanism reduces the area and power consumption of the routers.

However, in a NoC-based DNN accelerator, the router injection rate is very high, with performance sensitive to communication latency. Each packet transferred from one router to another will consume several cycles if the multi-flit mechanism is adopted. Furthermore, a multi-flit mechanism increases the packet transmission delay. In the

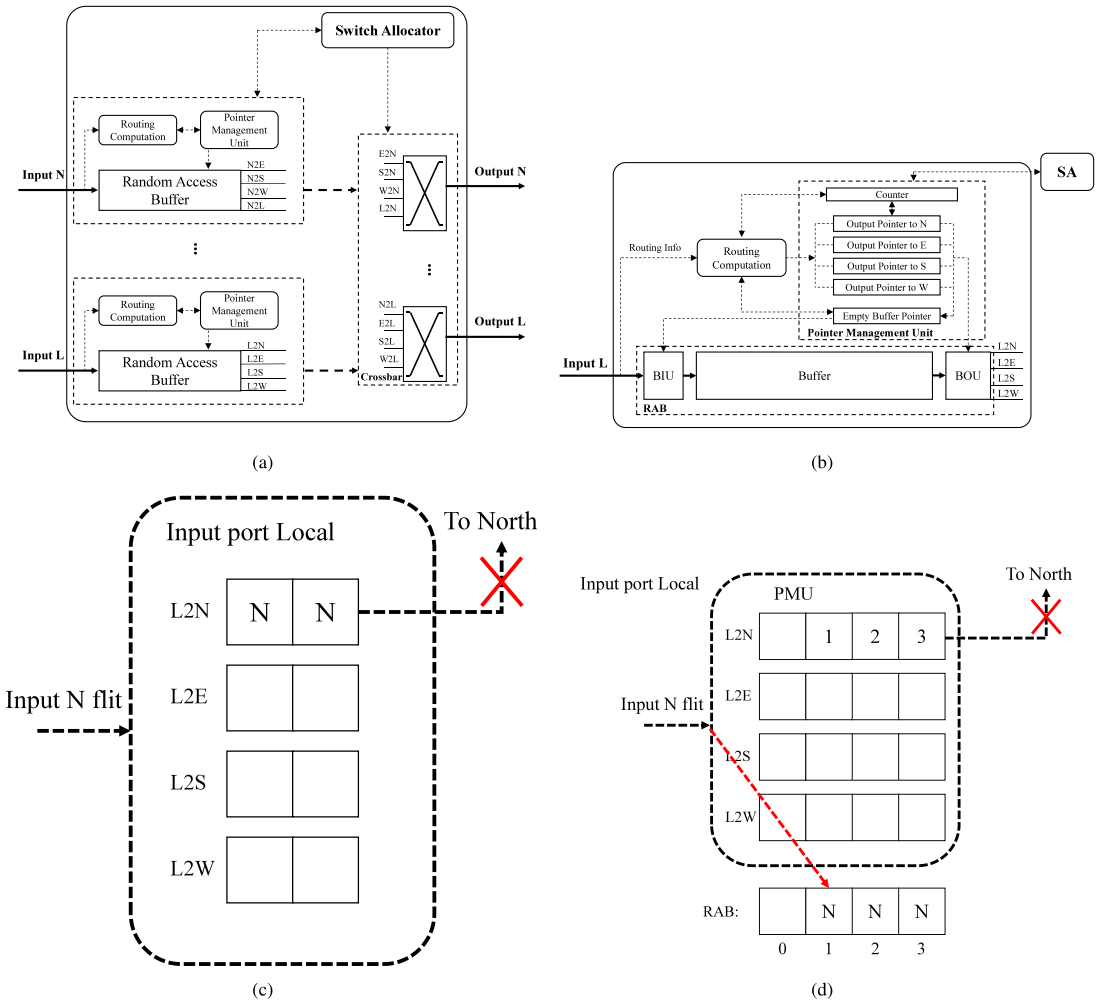


Fig. 7. Proposed router architecture: (a) overall architecture; (b) essential components of the Local input port; (c) and (d) analyses of buffer utilization in the data flow architecture router compared to this router.

multi-flit mechanism, different flits of the same packet may be distributed in multiple routers, causing severe congestion and significantly decreasing the network performance. Moreover, the payload in each packet in the NoC-based DNN accelerator is not very big. Thus, dividing a packet into multiple flits is not a suitable method.

In view of the preceding paragraph, we use a single-flit mechanism rather than the traditional multi-flit mechanism. Fig. 4 shows the formats of all of the single-flit packets that our router supports.

Our propose router supports unicast and multicast packets. For unicast packets, *Routing Info* is the address of the destination node. For multicast packets, *Routing Info* is the *packet_layer* defined in Section 3. *Control Info* records the packet sequence, and stores check codes. Further, other designers can use *Control Info* to implement their own network protocols. We use 16-bit data precision, so the packet *Data* is 16 bits.

4.2. Micro-architecture

The conventional virtual-channel (VC) router is illustrated in Fig. 5(a). The router is composed of input buffers, routing computation logic, a VC allocator, a switch allocator, and a crossbar. Fig. 5(b) shows a NoC router for the data flow architecture [20]. This router has no head-of-line blocking issue, yielding higher throughput and lower latency than the traditional wormhole router architecture.

For example, if we assume that the north output port of the router is congested while the other output ports are idle, there are four *N* flits that need to transmit to the north port and one *S* flit that needs to transmit to the south port, as shown in Figs. 6(a)–(b). In the conventional VC router architecture shown in Fig. 5(a) and Fig. 6(a), although the south output port is idle, the *S* flit in VC0 cannot transmit to the south output port because the *S* flit is blocked by the *N* flit in the buffer header. In contrast, this blocking problem does not occur in the routers shown in Fig. 5(b) and Fig. 6(b), because all *N* flits are queued in the L2N buffer, and the *S* flit can be transmitted directly.

The router proposed by Shen et al. [20] also supports multicast and single-flit packets in the data flow architecture. However, this kind of router only supports four-address multicast and carries a large number of address bits in the packet. Furthermore, this router lacks a simple and effective dynamic buffer management mechanism, which reduces the router's buffer utilization.

Based on Fig. 5(b), we propose a router architecture that supports single-flit packets and multicasting. Fig. 7(a) shows the overall architecture of the router proposed in this paper. Each input port of the router has three parts: routing computation (RC), pointer management unit (PMU), and random access buffer (RAB). Fig. 7(b) shows the essential components of the local input port. The PMU is composed of 5 FIFO pointer buffers, and its buffer depth is consistent with that of the RAB. In the PMU, the *Empty Buffer Pointer* stores the pointers of the empty buffer entries in the RAB. Other FIFO buffers store pointers to buffer entries that have stored packets. Different FIFO buffers indicate different output ports of the packet. There is also a *Counter* in the PMU, which records how many output ports required by the packet in the

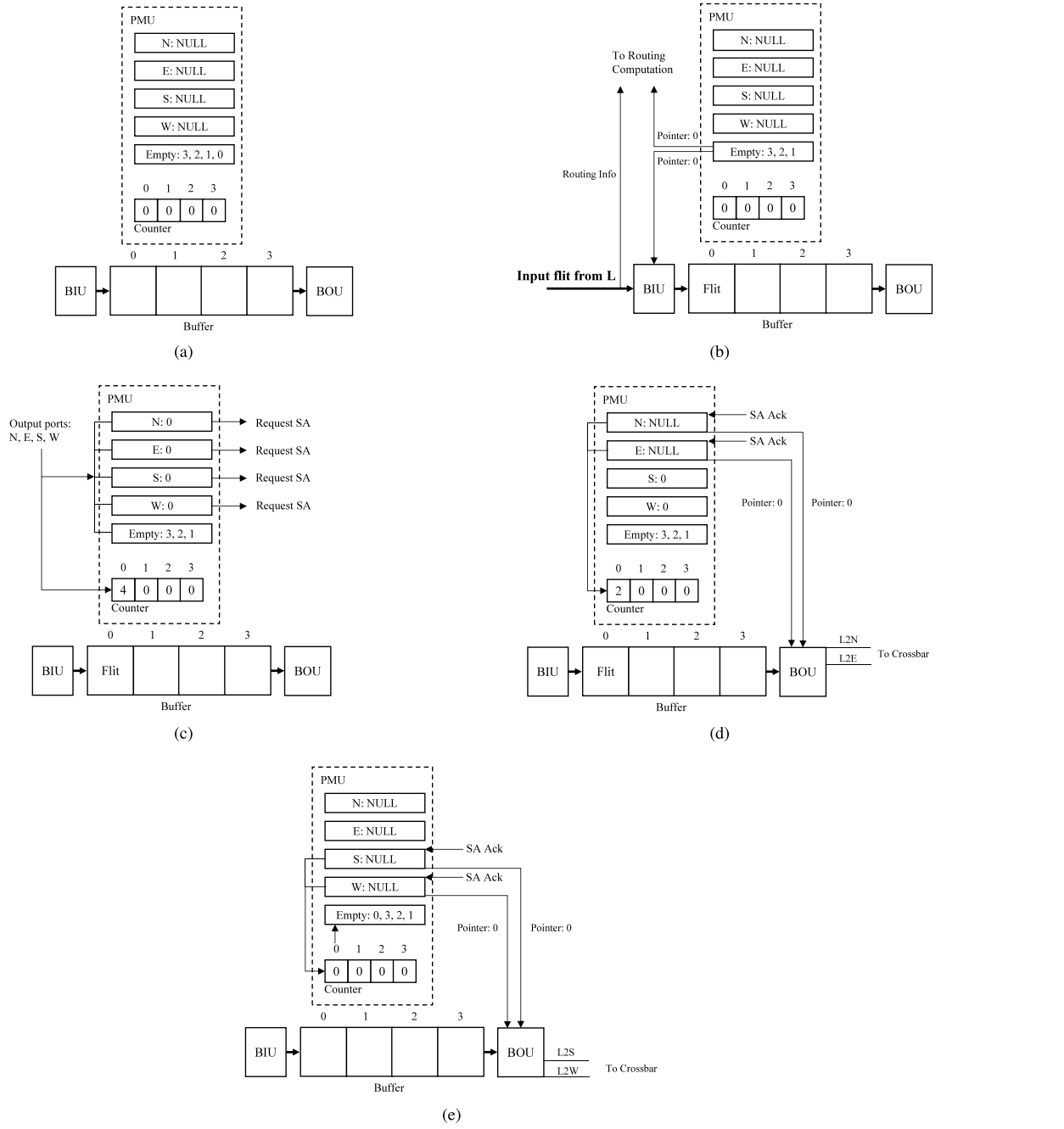


Fig. 8. Router transmission example. (a) Initial state. (b) Packet injected into the router. (c) The behavior of the PMU after routing calculation. (d) SA allocates N and E output ports to this input port. (e) SA allocates S and W output ports to this input port.

RAB. Within the RAB, the primary function of the *BIU* stores the input packet into the corresponding position in the RAB buffer according to the pointer. This pointer is provided by *Empty Buffer Pointer*. The primary function of the *BOU* is to output the packets in the RAB buffer according to the input pointer. The *BOU* has four output ports that are connected to four corresponding crossbars.

To help with understanding, we use the example in Fig. 8 to walk through the process of sending a multicast packet from the local input port to multiple output ports.

Fig. 8(a) shows the initial state of the PMU and RAB without packet input. In this case, there is no packet in the RAB. *Empty Buffer Pointer* in the PMU stores the pointers of all the buffer entries of the RAB. Each entry in *Counter* is 0.

In Fig. 8(b), a packet is injected into the router. In this case, *Empty Buffer Pointer* outputs the pointer of an empty RAB buffer (in this example the pointer is 0) to the *BIU* and RC. The *BIU* puts the packet into the corresponding position in the RAB buffer according to the input pointer.

In Fig. 8(c), the output ports of this packet are N, E, S, and W. After the routing calculation, the RC unit will inject the input pointer mentioned above into corresponding pointer buffers in the PMU. At the same time, RC sets entry 0 of *Counter* to 4, because this flit has four destination ports. With the injection of pointers, the N, E, S, and W pointer buffers are nonempty, which means that there are packets in the RAB buffer that need to be transmitted to the four output ports. Nonempty pointer buffers in PMU will request the SA unit to allocate crossbars for them.

In Fig. 8(d), if the SA allocates N and E output ports for the flit, then the N and E pointer buffers will output the stored pointers (0 in this

Algorithm 1: *NNMRA*

```
Input: row_layer, packet_layer, input_port, valid_in_PE,
valid_in_west, valid_in_east, valid_in_south
Output: Destination set: Dst
if row_layer <packet_layer then
    Add S to Dst.
end
else if row_layer == packet_layer then
    if input_port is N then
        if valid_in_PE is true then
            Add L to Dst.
            if valid_in_east is true then
                Add E to Dst.
            end
            if valid_in_west is true then
                Add W to Dst.
            end
            if valid_in_south is true then
                Add S to Dst.
            end
        end
        else
            Add W to Dst.
        end
    end
    else if input_port is E then
        if valid_in_PE is true and valid_in_west is true then
            Add L to Dst.
            Add W to Dst.
        end
        else if valid_in_PE is true and valid_in_west is false then
            Add L to Dst.
        end
        else
            Add W to Dst.
        end
    end
    else if input_port is W then
        if valid_in_PE is true then
            Add L to Dst.
        end
        if valid_in_east is true then
            Add E to Dst.
        end
    end
end
```

example) to the *BOU*. The *BOU* outputs packets to the corresponding crossbars. At the same time, entry 0 of *Counter* needs to be decremented by 2, because the transmission of two ports has been completed.

In Fig. 8(e), the SA allocates S and W output ports for the packet. After the transmission of the S and W ports is completed, entry 0 of *Counter* is 0, which means all branches of entry 0 of the RAB buffer have been transmitted. Thus, pointer 0 will be input to the *Empty Buffer Pointer*.

Our proposed router architecture improves buffer utilization, reduces communication latency, and increases its throughput. Figs. 7(c) and (d) compare the buffer utilization of the data flow architecture router and our proposed router. We assume that the north output port of the router is blocked, the other ports are idle, and two *N* flits are already stored in the router. In the router architecture of Fig. 7(c), if there is an *N* flit that needs to be input from the Local port, then this flit cannot enter the L2N buffer because the L2N buffer is full. Although all

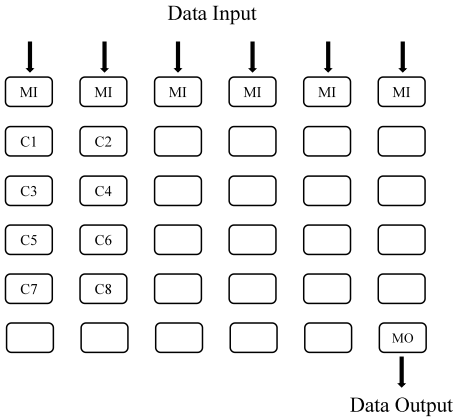


Fig. 9. Distribution of computational and memory nodes within a NoC.

of the other buffers of this input port are free, this input port remains blocked. In contrast, considering our router shown in Fig. 7(d), the *N* flit that needs to be input can still enter the buffer and the input port is not blocked. If a flit that needs to be output to another port arrives, then that flit can be successfully routed out. This not only increases buffer utilization, but it also reduces the possibility of blocking. Since both router architectures support multicasting algorithms, our proposed scheme copies the pointers of the flit in the PMU, which requires less power than copying the flits directly.

Replication schemes often affect the performance of multicast routers. Replication schemes are fall into categories of synchronous and asynchronous. In synchronous replication, a flit is transmitted when all of its target output ports are available. In asynchronous replication, a flit is transmitted asynchronously as long as some target output ports are available [2]. In the previous example, our multicast router uses asynchronous replication, as synchronous replication is susceptible to deadlock.

5. Evaluation

5.1. Simulation setup

To evaluate the performance of our proposed methods, we used a simulator with accurate cycles based on System-C to develop our platform [16–18,35]. To describe our experiment clearly, as shown in Fig. 9, we give an example of mapping Lenet-5 onto a 6 × 6 Mesh network. Lenet-5 is clustered using the MPC2F50 clustering strategy shown in Fig. 2(b).

In Fig. 9, MI refers to a node that reads data from memory. The MO node is responsible for receiving data from the last hidden layer of the DNN. In our experiment, all of the nodes in the first row of the network are MI nodes, and the last node of the network is the MO node. The MO node communicates with clusters in the last hidden layer via unicast. We assume that the memory access bandwidth of all of the MI and MO nodes is 2 GB/s, which means that MI and MO nodes with a frequency of 1 GHz can read or store a 16-bit unit of data from memory each cycle. The MO node calculates results based on the received data and stores these results in memory. In Fig. 9, C1 to C8 are the clusters in Lenet-5.

We simulated different sizes of ANN and DNN models as shown in Table 1. We implement multiple unicast in the baseline virtual channel router (BASE), as well as the four-address multicast mechanism in the output buffer router of the dataflow architecture (CMPA) [20] for comparison. In a 16 × 16 mesh network, the packet size of CMPA is only 91 bits, making the area and power consumption of CMPA router basically the same as the router in this paper. If traditional multicast mechanisms (e.g., RPM [2] and Neu-NoC [15]) are used, the flit size

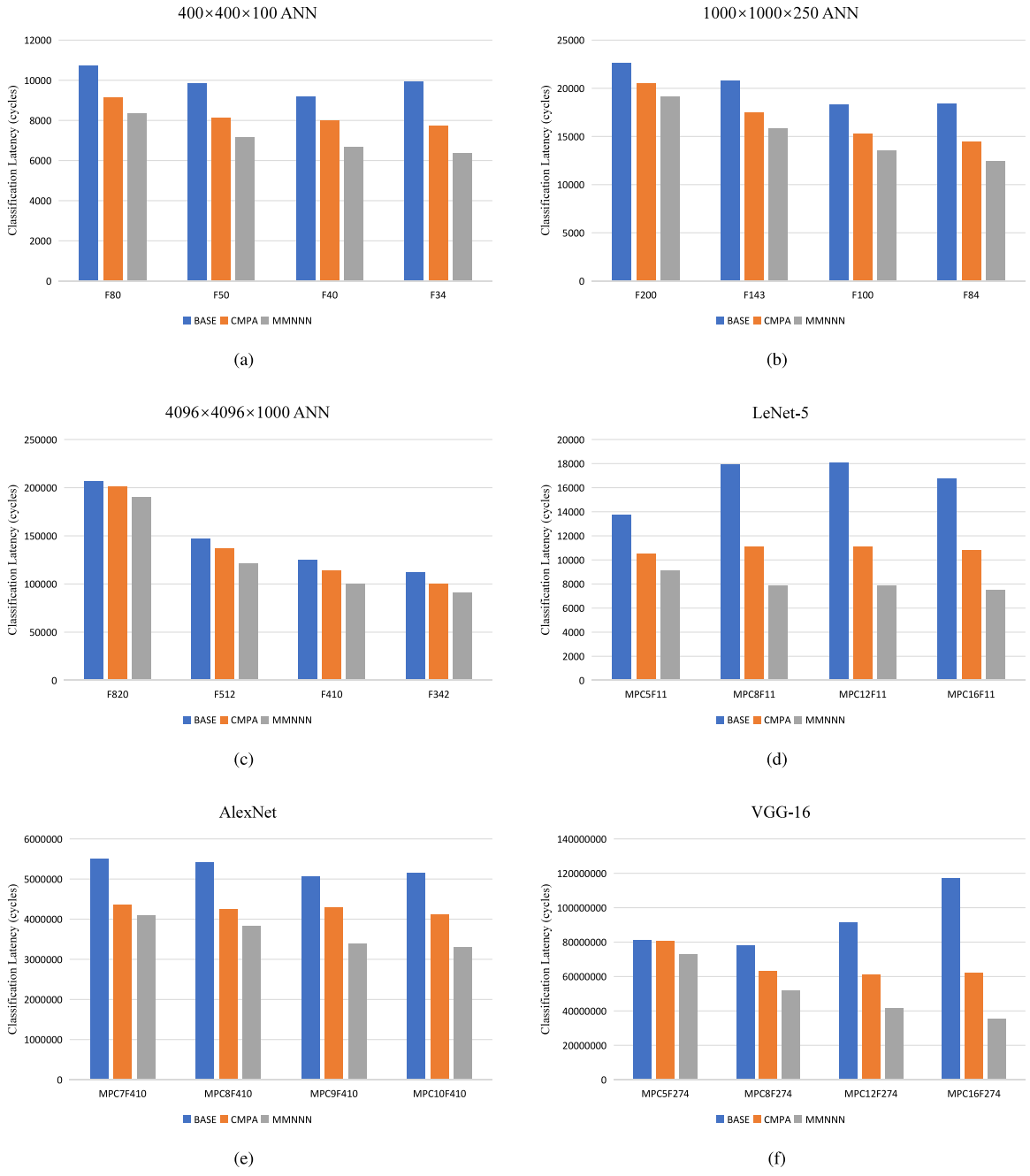


Fig. 10. Classification latency.

Table 1
The DNN to be simulated.

DNN	Input
400 × 400 × 100 ANN	28 × 28
1000 × 1000 × 250 ANN	32 × 32
4096 × 4096 × 1000 ANN	32 × 32
Lenet-5	32 × 32
AlexNet [33]	227 × 227 × 3
VGG-16[34]	224 × 224 × 3

will exceed 256 bits, making the hardware overhead too large. CMPA also uses the same single-flit mechanism as we do. Therefore, we use CMPA for comparison. Network parameters are given in Table 2. It is worth noting that in Table 2, we mapped all of the neurons in the last fully connected layer of VGG-16 to the MO nodes because our mechanism can maximally support 16 × 16 mesh network, but VGG-16's layer number (including input layer) is larger than 16.

Considering the proposed clustering and mapping strategies, we chose different network topologies for DNNs of different depths. Since the multicast routing algorithm proposed in this paper is based on the YX routing algorithm, we use the YX unicast routing algorithm to avoid deadlocks. Regarding existing DNN accelerator designs, in this paper, the peak performance per PE is 86.4 GOPS, i.e., similar to the UNPU [41] design.

For the sake of a fair comparison, we use 16 buffers per port. The baseline router in the BASE is designed with four virtual channels. We also note that each buffer entry's size in CMPA is 91 bits, rather than 64 bits in BASE and MMNNN, as each packet in CMPA needs to carry four destination addresses.

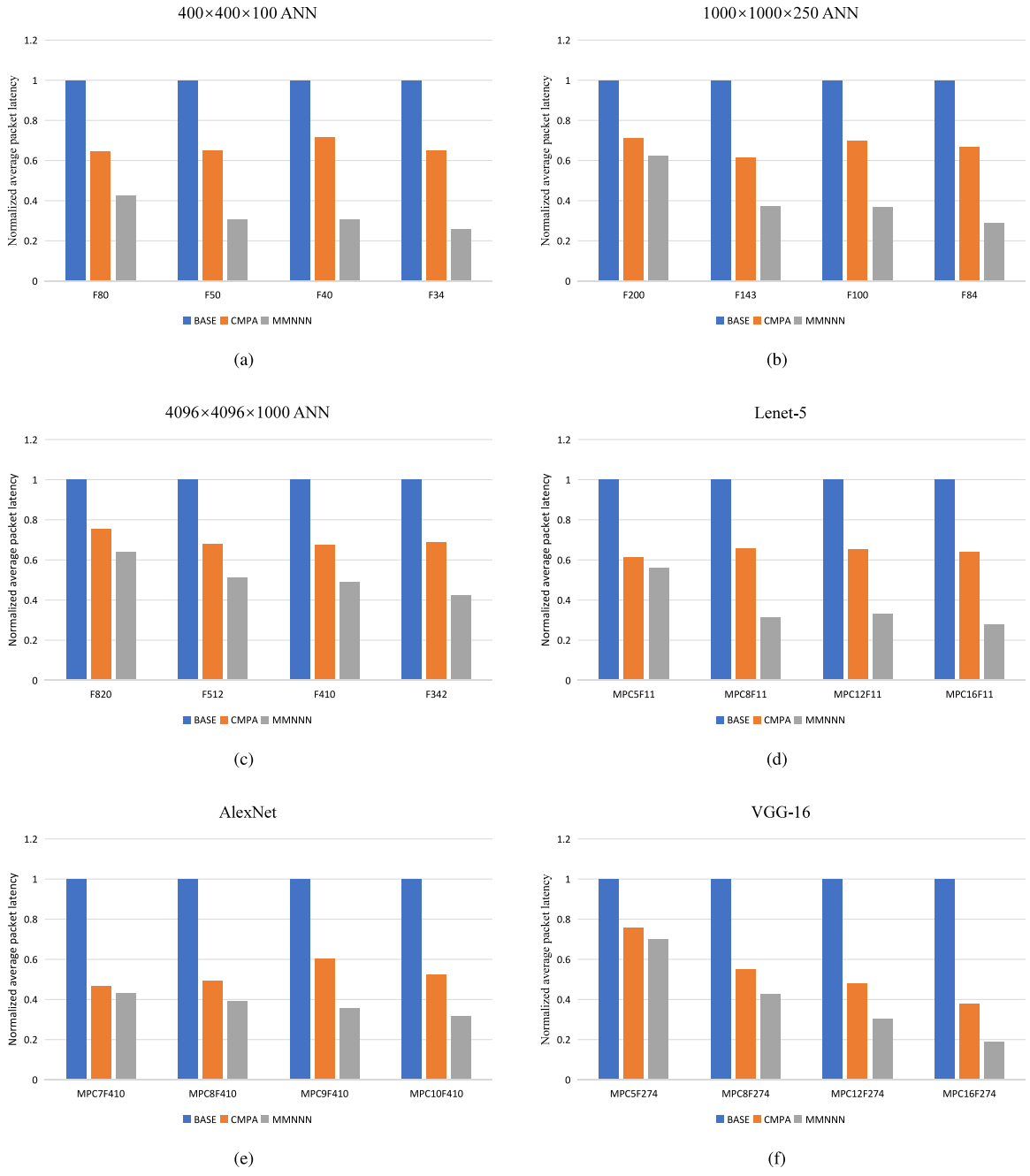


Fig. 11. Normalized average packet latency.

5.2. Performance evaluation

To measure the effect of different multicast mechanisms on the NoC-based DNN accelerators' performance, we considered three metrics: classification latency, average packet latency, and the number of routed packets. In DNN accelerator, an image will be first input to NoC by MI nodes. MI nodes will then transfer the data to the computation nodes of the first layer in DNN. The PEs in each layer receive data from the PEs in its previous layer via the NoC. After the PEs have completed the computation, they will transfer the results to the computation nodes of the next layer via the NoC. Finally, results will be output from MO node. The classification latency refers to the time it takes for an image to complete all of these above stages. Average packet latency refers to the average latency of a packet from the source node to the destination nodes. The number of routed packets refers to the sum of the number of packets output by all of the routers in the network.

Fig. 10 shows the classification latency for different DNNs with different network design parameters. For ANNs, as shown in Fig. 10(a)–(c), MMNNN's classification latencies were reduced by an average of 28%, 24%, and 15% compared to BASE, and 14%, 10%, and 9% to CMPA, respectively. For CNNs, as shown in Fig. 10(d)–(f), the MMNNN's classification latencies were reduced by an average of 51%, 31%, 45% compared to BASE, and 26%, 14%, and 25% to CMPA. MMNNN in the CNN accelerator achieved better performance.

While using more PEs in a NoC-based DNN accelerator reduces computational latency, communication latency rises as the number of packets rises. MMNNN reduces the number of packets in the network, thereby reducing communication latency and breaking communication bottlenecks.

As shown in Fig. 10, the DNN accelerator with MMNNN did not increase the classification latency due to the increase in number of PEs. Fig. 11 shows the normalized average packet latencies for the

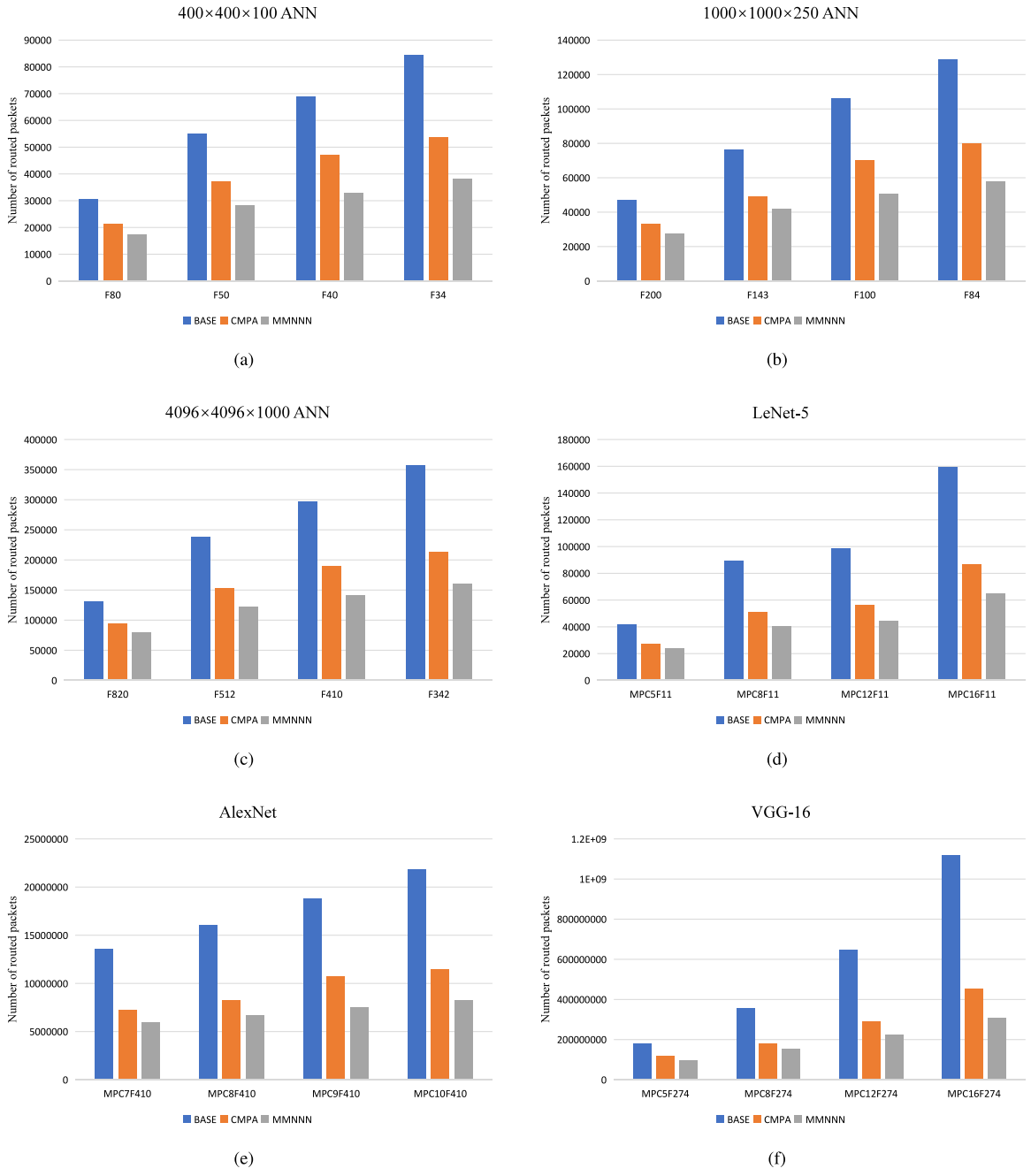


Fig. 12. Number of routed packets.

different DNNs with different network design parameters. As shown in Fig. 11, the MMNNN's average packet latencies decreased by 35%–81% compared to BASE and by 7%–60% compared to CMPA. Furthermore, increasing the number of PEs makes MMNNN more effective (e.g., MPC5F11 and MPC16F11 in Fig. 11(d)). Thus, the results of Fig. 11 show that MMNNN effectively decreased communicate latency. Thus, the DNN accelerator using MMNNN effectively uses more resources to accelerate computation.

Fig. 12 shows the numbers of routed packets for each DNN and parameter combination. As shown in Fig. 12(a)–(f), MMNNN reduced the number of routed packets by an average of 51%, 50%, 51%, 55%, 59%, and 62% compared to the BASE, and 27%, 23%, 22%, 22%, 25%, and 25% to CMPA, respectively. Further, the number of routed packets with MMNNN increased at a much lower rate than with BASE as the number of PEs increased. For example, in Fig. 12(d), the number of routed packets in MPC16F11-BASE was 3.8 times higher than MPC5F11-BASE, while the number of routed packets for MPC16F11-MMNNN was 2.7 times higher than MPC5F11-MMNNN.

Reducing the number of routed packets has many advantages, relieving traffic congestion and reducing communication latency. However, this reduction also reduces the load on routers and saves power. The results in both Figs. 10 and 12 show that MMNNN is more suitable for NoC-based DNN accelerators than BASE and CMPA.

5.3. Power and area analysis

We used the Synopsis Design Compiler experimental tool, described the relevant hardware logic with the Verilog HDL language, and combined this with ModelSim-SE 2019.2 for functional and structural simulation. We simulated the area and power consumption of the router using a 65 nm CMOS standard cell library from TSMC, operating at

Table 2
Simulator configuration.

Parameters	Settings
Topology	6 × 6 Mesh (400 × 400 × 100 ANN) 6 × 6 Mesh (1000 × 1000 × 250 ANN) 6 × 6 Mesh (4096 × 4096 × 1000 ANN) 8 × 8 Mesh (Lenet-5) 10 × 10 Mesh (AlexNet) 16 × 16 Mesh (VGG-16)
Routing	Y–X Routing (BASE, CMPA)
Buffers per port	16 (4 virtual channels for baseline router)
PE performance	86.4 GOPS
Frequency of router	1 GHz
Memory bandwidth	2 GB/s
Channel width	64 bits (BASE, MMNNN) 91 bits (CMPA)

Table 3
DNN to be simulated.

Different schemes	Area (μm^2)	Total power (mW)
BASE	107005.32	72.2888
CMPA	116935.92	76.6649
MMNNN	142396.56	72.7406

1 GHz. The simulation results for the area and power consumption are shown in Table 3.

Table 3 shows that MMNNN's router has the largest area due to the PMU in each port. Although the PMUs increase the area of the router, their presence avoids copying a flit to multiple buffers. However, MMNNN's packet buffers are smaller than CMPA. As a result, MMNNN's router consumes less power than CMPA.

The results from Section 5.2 show that MMNNN has lower communication latency and fewer numbers of routed packets, but with power consumption similar to that of BASE. We conclude that MMNNN is a power efficient mechanism.

6. Conclusion

In recent years, NoC-based DNN accelerators have been proposed. However, the presence of one-to-many packet traffic means that a suitable multicast mechanism in NoC will significantly improve the performance of the accelerator. We present a multicast mechanism for a NoC-based DNN accelerator, called MMNNN, along with a tree-based multicast routing algorithm and a router architecture. This multicast routing algorithm minimizes the number of transmitted packets within the network, and the proposed router architecture reduces overhead of packet branching and improves buffer utilization. Experimental results show that the proposed mechanism reduces classification latency, average packet latency, and the number of routed packets. These contributions enable NoC-based DNN accelerators to break the communication latency bottleneck and utilize more on-chip resources. Although MMNNN's router area has increased by 33% compare to BASE, its power consumption does not increase significantly. Therefore, MMNNN is a more power-efficient and quicker mechanism for NoC-based DNN accelerators.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research is supported in part by the National Natural Science Foundation of China (NSFC) research project no. 61874157 and the Natural Science Foundation of Department of Education of Anhui Province China under grant no. KJ2017A477.

References

[1] K. Goossens, J. Dielissen, A. Radulescu, Æthereal network on chip: Concepts, architectures, and implementations, IEEE Des. Test Comput. 22 (5) (2005) 414–421, http://dx.doi.org/10.1109/MDT.2005.99.

[2] L. Wang, Y. Jin, H. Kim, E.J. Kim, Recursive partitioning multicast: A bandwidth-efficient routing for networks-on-chip, in: Third International Symposium on Networks-on-Chips, NOCS 2009, May 10–13 2009, la Jolla, CA, USA. Proceedings, IEEE Computer Society, 2009, pp. 64–73, http://dx.doi.org/10.1109/NOCS.2009.5071446.

[3] L. Peh, W.J. Dally, A delay model and speculative architecture for pipelined routers, in: Proceedings of the Seventh International Symposium on High-Performance Computer Architecture (HPCA'01), Nuevo Leone, Mexico, January 20–24, 2001, IEEE Computer Society, 2001, pp. 255–266, http://dx.doi.org/10.1109/HPCA.2001.903268.

[4] A. Kumar, L. Peh, P. Kundu, N.K. Jha, Express virtual channels: towards the ideal interconnection fabric, in: D.M. Tullsen, B. Calder (Eds.), 34th International Symposium on Computer Architecture (ISCA 2007), June 9–13, 2007, San Diego, California, USA, ACM, 2007, pp. 150–161, http://dx.doi.org/10.1145/1250662.1250681.

[5] H. Matsutani, M. Koibuchi, H. Amano, T. Yoshinaga, Prediction router: Yet another low latency on-chip router architecture, in: 15th International Conference on High-Performance Computer Architecture (HPCA-15 2009), 14–18 February 2009, Raleigh, North Carolina, USA, IEEE Computer Society, 2009, pp. 367–378, http://dx.doi.org/10.1109/HPCA.2009.4798274.

[6] S. Deb, A. Ganguly, P.P. Pande, B. Belzer, D.H. Heo, Wireless NoC as interconnection backbone for multicore chips: Promises and challenges, IEEE J. Emerg. Sel. Topics Circuits Syst. 2 (2) (2012) 228–239, http://dx.doi.org/10.1109/JETCAS.2012.2193835.

[7] Y. Ouyang, J. Yang, K. Xing, Z. Huang, H. Liang, An improved communication scheme for non-HOL-blocking wireless NoC, Integration 60 (2018) 240–247, http://dx.doi.org/10.1016/j.vlsi.2017.10.005.

[8] Y. Ouyang, Z. Li, J. Li, C. Sun, H. Liang, G. Du, CPCA: an efficient wireless routing algorithm in winoc for cross path congestion awareness, Integration 69 (2019) 75–84, http://dx.doi.org/10.1016/j.vlsi.2019.03.008.

[9] Y. Ouyang, Q. Wang, L. Hu, H. Liang, DVFS based error avoidance strategy in wireless network-on-chip, J. Electron. Test. 35 (6) (2019) 767–777, http://dx.doi.org/10.1007/s10836-019-05841-9.

[10] K.J. Chen, M. Ebrahimi, T. Wang, Y. Yang, Noc-based DNN accelerator: a future design paradigm, in: P. Bogdan, C. Silvano (Eds.), Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip, NOCS 2019, New York, NY, USA, October 17–18, 2019, ACM, 2019, pp. 11:1–11:8, http://dx.doi.org/10.1145/3313231.3352376.

[11] W.J. Dally, B.P. Towles, Principles and Practices of Interconnection Networks, Elsevier, 2004.

[12] E. Painkras, L.A. Plana, J.D. Garside, S. Temple, F. Galluppi, C. Patterson, D.R. Lester, A.D. Brown, S.B. Furber, SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation, IEEE J. Solid State Circuits 48 (8) (2013) 1943–1953, http://dx.doi.org/10.1109/JSSC.2013.2259038.

[13] S. Carrillo, J. Harkin, L. McDaid, F. Morgan, S. Pande, S. Cawley, B. McGinley, Scalable hierarchical network-on-chip architecture for spiking neural network hardware implementations, IEEE Trans. Parallel Distributed Syst. 24 (12) (2013) 2451–2461, http://dx.doi.org/10.1109/TPDS.2012.289.

[14] Y. Chen, T. Yang, J.S. Emer, V. Sze, Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices, IEEE J. Emerg. Sel. Topics Circuits Syst. 9 (2) (2019) 292–308, http://dx.doi.org/10.1109/JETCAS.2019.2910232.

[15] X. Liu, W. Wen, X. Qian, H. Li, Y. Chen, Neu-NoC: A high-efficient interconnection network for accelerated neuromorphic systems, in: Y. Shin (Ed.), 23rd Asia and South Pacific Design Automation Conference, ASP-DAC 2018, Jeju, Korea (South), January 22–25, 2018, IEEE, 2018, pp. 141–146, http://dx.doi.org/10.1109/ASPDAC.2018.8297296.

[16] K.J. Chen, T. Wang, NN-Noxim: High-level cycle-accurate noc-based neural networks simulator, in: 11th International Workshop on Network on Chip Architectures, NoCArc@MICRO 2018, Fukuoka, Japan, October 20, 2018, IEEE Computer Society, 2018, pp. 1–5, http://dx.doi.org/10.1109/NOCARC.2018.8541173.

[17] K.J. Chen, T.G. Wang, Y.A. Yang, Cycle-accurate noc-based convolutional neural network simulator, in: F. Firouzi, K. Chakrabarty, B. Farahani, F. Ye, V.F. Pavlidis (Eds.), Proceedings of the International Conference on Omni-Layer Intelligent Systems, COINS 2019, Crete, Greece, May 5–7, 2019, ACM, 2019, pp. 199–204, http://dx.doi.org/10.1145/3312614.3312655.

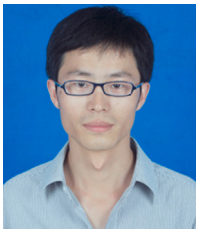
[18] K.J. Chen, M. Ebrahimi, T. Wang, Y. Yang, Y. Liao, A noc-based simulator for design and evaluation of deep neural networks, Microprocess. Microsystems 77 (2020) 103145, http://dx.doi.org/10.1016/j.micpro.2020.103145.

[19] S. Xiao, Y. Guo, W. Liao, H. Deng, Y. Luo, H. Zheng, J. Wang, C. Li, G. Li, Z. Yu, Neuronlink: An efficient chip-to-chip interconnect for large-scale neural network accelerators, IEEE Trans. Very Large Scale Integr. Syst. 28 (9) (2020) 1966–1978, http://dx.doi.org/10.1109/TVLSI.2020.3008185.

[20] X. Shen, X. Ye, X. Tan, D. Wang, L. Zhang, W. Li, Z. Zhang, D. Fan, N. Sun, An efficient network-on-chip router for dataflow architecture, J. Comput. Sci. Technol. 32 (1) (2017) 11–25, http://dx.doi.org/10.1007/s11390-017-1703-5.
[21] D.R. Kumar, W.A. Najjar, P.K. Srimani, A new adaptive hardware tree-based multicast routing in K-ary N-cubes, IEEE Trans. Computers 50 (7) (2001) 647–659, http://dx.doi.org/10.1109/12.936232.
[22] W. Hu, Z. Lu, A. Jantsch, H. Liu, Power-efficient tree-based multicast support for networks-on-chip, in: Proceedings of the 16th Asia South Pacific Design Automation Conference, ASP-DAC 2011, Yokohama, Japan, January 25–27, 2011, IEEE, 2011, pp. 363–368, http://dx.doi.org/10.1109/ASPDAC.2011.5722214.
[23] X. Lin, P.K. McKinley, L.M. Ni, Deadlock-free multicast wormhole routing in 2-D mesh multicomputers, IEEE Trans. Parallel Distributed Syst. 5 (8) (1994) 793–804, http://dx.doi.org/10.1109/71.298203.
[24] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, H. Tenhunen, HAMUM - A novel routing protocol for unicast and multicast traffic in mpsocs, in: M. Danelutto, J. Bourgeois, T. Gross (Eds.), Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-Based Processing, PDP 2010, Pisa, Italy, February 17–19, 2010, IEEE Computer Society, 2010, pp. 525–532, http://dx.doi.org/10.1109/PDP.2010.81.
[25] S.T. Nguyen, S. Oyanagi, A low cost single-cycle router based on virtual output queuing for on-chip networks, in: S. López (Ed.), 13th Euromicro Conference on Digital System Design, Architectures, Methods and Tools, DSD 2010, 1–3 September 2010, Lille, France, IEEE Computer Society, 2010, pp. 60–67, http://dx.doi.org/10.1109/DSD.2010.15.
[26] T. Speier, B. Wolford, Qualcomm centriq 2400 processor, in: Hot Chips: A Symposium on High Performance Chips, HC29 (2017), 2017.
[27] J. Jeffers, J. Reinders, A. Sodani, Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition, Morgan Kaufmann, 2016.
[28] X. Lian, Z. Liu, Z. Song, J. Dai, W. Zhou, X. Ji, High-performance FPGA-based CNN accelerator with block-floating-point arithmetic, IEEE Trans. Very Large Scale Integr. Syst. 27 (8) (2019) 1874–1885, http://dx.doi.org/10.1109/TVLSI.2019.2913958.
[29] C. Farabet, C. Poulet, Y. LeCun, An fpga-based stream processor for embedded real-time vision with convolutional networks, in: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, IEEE, 2009, pp. 878–885.
[30] S. Gupta, A. Agrawal, K. Gopalakrishnan, P. Narayanan, Deep learning with limited numerical precision, 2015, CoRR abs/1502.02551, arXiv:1502.02551.
[31] B. Moons, M. Verhelst, A 0.3-2.6 TOPS/W precision-scalable processor for real-time large-scale convnets, in: 2016 IEEE Symposium on VLSI Circuits, VLSIC 2016, Honolulu, HI, USA, June 15–17, 2016, IEEE, 2016, pp. 1–2, http://dx.doi.org/10.1109/VLSIC.2016.7573525.
[32] S. Yin, P. Ouyang, S. Tang, F. Tu, X. Li, L. Liu, S. Wei, A 1.06-to-5.09 TOPS/W reconfigurable hybrid-neural-network processor for deep learning applications, in: 2017 Symposium on VLSI Circuits, IEEE, 2017, pp. C26–C27.
[33] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: P.L. Bartlett, F.C.N. Pereira and C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a Meeting Held December 3–6, 2012, Lake Tahoe, Nevada, United States, 2012, pp. 1106–1114.
[34] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015.
[35] V. Catania, A. Mineo, S. Monteleone, M. Palesi, D. Patti, Cycle-accurate network on chip simulation with noxim, ACM Trans. Model. Comput. Simul. 27 (1) (2016) 4:1–4:25.
[36] H. Garcia-Molina, A. Spauster, Message ordering in a multicast environment, in: 9th International Conference on Distributed Computing Systems, ICDCS 1989, Newport Beach, CA, USA, June 5–9, 1989, IEEE Computer Society, 1989, pp. 354–361, http://dx.doi.org/10.1109/ICDCS.1989.37965.
[37] J. Chang, N.F. Maxemchuk, Reliable broadcast protocols, ACM Trans. Comput. Syst. 2 (3) (1984) 251–273, http://dx.doi.org/10.1145/989.357400.
[38] X. Lin, L.M. Ni, Multicast communication in multicomputer networks, IEEE Trans. Parallel Distrib. Syst. 4 (10) (1993) 1105–1117, http://dx.doi.org/10.1109/71.246072.
[39] N.D.E. Jerger, L. Peh, M.H. Lipasti, Virtual circuit tree multicasting: A case for on-chip hardware multicast support, in: 35th International Symposium on Computer Architecture (ISCA 2008), June 21–25, 2008, Beijing, China, IEEE Computer Society, 2008, pp. 229–240, http://dx.doi.org/10.1109/ISCA.2008.12.
[40] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324, http://dx.doi.org/10.1109/5.726791.
[41] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, H. Yoo, UNPU: an energy-efficient deep neural network accelerator with fully variable weight bit precision, IEEE J. Solid State Circuits 54 (1) (2019) 173–185, http://dx.doi.org/10.1109/JSSC.2018.2865489.

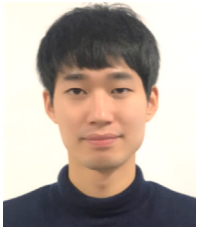
Yiming Ouyang received his bachelor's degree, master's degree, and doctoral degree from Hefei University of Technology, China in 1984, 1991, and 2013 respectively. He was a fellow researcher in Pennsylvania State University, USA during 2010–2011. He is currently a professor at Hefei University of Technology and a leading expert in research. The main research directions are Network-on-Chip (NoC) and on-chip systems (SoC), integration and testing of embedded systems, fault tolerant computing, NoC-based neural network accelerator design.

Feiyang Tang received his B.S. degree in Anhui Jianzhu University in 2019. He is currently pursuing M.S. degree at the School of Computer Science and Information Engineering, Hefei University of Technology, focusing on network-on-chip (NoC), system-on-chip (SoC) and NoC-based neural network accelerator design.



Chunlei Hu received his B.S. degree in Anhui Agricultural University in 2009 and his M.S. degree in Hefei University of Technology in 2012. He is a visiting scholar at the School of Computer Science and Information Engineering, Hefei University of Technology, focusing on network-on-chip (NoC) and internet security.

Wu Zhou received his B.S. degree in Information Engineering from Hefei University of Technology in 2018. He is currently pursuing a joint M.S.-Ph.D. program at the School of Computer Science and Information Engineering, Hefei University of Technology, focusing on network-on-chip (NoC) and system-on-chip (SoC), fault-tolerant systems and NoC-based neural network accelerator design.



Qi Wang graduated from Jilin University in 2016, spent one year in Stockholm University as exchange student majoring in Computer and System Science in 2014. He obtained his bachelor's degree from Stevens Institute of Technology in 2018. Now he is a PHD student in Hefei University of Technology. His main research field is Network on Chip, Machine learning and Fault tolerance.