IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 15, 2019, accepted May 2, 2019, date of publication May 17, 2019, date of current version June 10, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2916338

Customized High Performance and Energy Efficient Communication Networks for AI Chips

WEI GAO[1,2,3], (Student Member, IEEE), AND PINGQIANG ZHOU[2], (Member, IEEE)
[1]Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China
[2]School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China
[3]University of Chinese Academy of Sciences, Beijing, China

Corresponding author: Wei Gao (gaowei@shanghaitech.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant.61401276.

ABSTRACT The convolutional and deep neural networks are prevalent machine learning algorithms for real-world applications. As the neural network needs large computations, many artificial intelligence (AI) chips are designed to accelerate the computation. AI chips have achieved better energy efficiency and high computational capacity in the neural network implementation. The communication network in AI chips influences the data transformation and hardware efficiency. The network-on-chip (NoC) is one feasible solution to meet the data communication requirements in AI chips. This paper introduces the communication network in AI chips and the strategy of mapping neural network to chips with the extensible hierarchical architecture. We also conclude the opportunities for communication optimization in the design of AI chips. In this paper, we propose our processor architecture and optimize the performance and energy of intra-communication in chips from three aspects: data reuse, topology, and router architecture. The experimental results show that our optimization can totally achieve 25.31× latency reduction and 79.92× energy less than the baseline. The results show that our design can reduce the latency by 5.47× and save communication energy by 7.5× when compared with the state-of-the-art design DaDianNao. When compared with another design Eyeriss, our design can reduce latency by 7.57× and save communication energy by 3.03×.

INDEX TERMS AI accelerators, NoC, communication network.

I. INTRODUCTION

Deep learning algorithms such as Convolutional and Deep Neural Networks (CNNs and DNNs) have been widely used in real-world application domains such as autonomous driving, robotics, and intelligent security [1], [2]. To achieve better inference accuracy, the depth of the neural network is extremely large in general [4]. Take CNN for image recognition (shown in Fig. 1) as an example, the modern deep CNN typically consists of 5−1,000 convolution layers and 1−3 fully-connected layers. Each layer consists of neurons (functional nodes) and the adjacent layers are connected by synapses (weights). In the training process, to train the synapses, the figures in the database pass through this neural network in a sequential manner – entering from the first convolution layer and exiting from the last fully-connected layer.

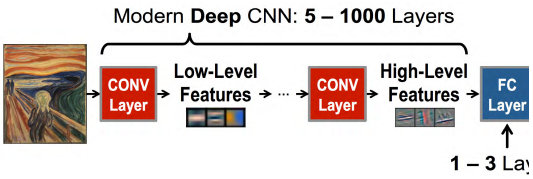


FIGURE 1. The CNN for image classification [5].

To achieve better training performance, the database is increasingly large. For example, the AlphaGo is trained with a database of about 30 million moves [12]. In addition to the database, the scale of the neural networks also increases significantly over the past a few years. Table 1 presents the evolution of the neural networks for image classification applications. The classification accuracy increases over the

The associate editor coordinating the review of this manuscript and approving it for publication was Mingjun Dai.

TABLE 1. Summary of several popular neural networks [5].

Metrics	AlexNet	VGGNet	GoogLeNet	ResNet 50
Year released	2012	2014	2014	2015
Input Size	227*227	224*224	224*224	224*224
Depth	8	16	22	50
Top-5 error	19.8	8.8	10.7	7
Total MACs (billion)	0.724	15.5	1.43	3.9
Total Weights (million)	61	138	7	25.5

years and it exhibits a strong correlation with the depth and the scale of the neural network. For instance, even the smallest neural network architecture, AlexNet, which has a top-5 accuracy of 80.2%, requires 61 million floating point weights and 724 million MAC (multiply-accumulate) operations to classify a single image of $227 \times 227 \times 3$. The large size of such networks poses big challenges to the underlying computing devices in terms of both computational complexity and memory requirements.

In the beginning, the neural networks are implemented on the CPU [6]. However, given the fact that most of the computations in deep learning algorithms are multiplications and additions that can be processed in parallel, in recent years, accelerators including FPGA, GPU and ASIC have been proposed for neural networks to achieve better performance and energy efficiency [8]–[10], [17]–[19]. Generally speaking, FPGA provides the most flexible reconfigurability; GPU can achieve high throughput at the cost of large energy consumption; ASIC offers the highest computation efficiency. For instance, in [9], it is reported that compared with a CPU (SIMD), the GPU can achieve one to two orders of speedup; while the fully-customized ASIC accelerator DaDiannao can further improve the processing speed by one to two orders but with much lower (up to two-order) energy consumption.

For the data communication within the ASIC accelerators, both Eyeriss [18] and Thinker [10] use the bus to transmit the weights of synapses and neuron values between buffers and processing elements (PEs). Eyeriss uses 12 horizontal X-buses connected to 168 PEs and one Y-bus to determine which X-bus the data transmitted to. The bus is one feasible method for transmissions in AI chips, but it lacks the scalability and flexibility when the amounts of PEs increases. To meet the stringent requirements on data communication among the processing nodes in a large-scale AI chip, Network-on-Chip (NoC) has been proposed as the communication infrastructure in [8], [9], [17], [19]. Dadiannao [8] arranges inter-chip communication in a regular ring pattern and the H-tree topology for intra-chip communication in the NoC design. However, when the scale of the neural networks increases, the NoC-based data communication within the AI accelerators will still become the performance bottleneck. For instance, more than half of the runtime is spent on data communication in DaDiannao [9] and its ratio grows up fast as the processor scale increases. In addition to performance, the energy consumption of the NoC in a deep learning accelerator is also a big concern. For example, the reported power consumption for NoC in DaDiannao can account for about 50% of the total chip power. Therefore, it is essential to optimize both the performance and energy efficiency of the NoC architecture in an AI chip.

There has been very limited prior work on NoC optimization for AI chips. Compared with Eyeriss, Eyeriss v2 [17] proposes one hierarchical mesh NoC network for DNNs by using routers to allocate the data transmission. The input data and weights need to be transmitted from global buffer to PE array with the broadcast pattern. This pattern increases the data reuse which means the number of MACs that each data is used for. However, the Eyeriss v2 lacks optimization techniques for the component of the NoC network, such as router optimization. Dadiannao [8] uses the mesh and torus topology to replace the ring arrangement with inter-chip communication. The data reuse and router architecture optimization are ignored in the design of chips. Neu-NoC [19] designs a hybrid NoC for neuromorphic acceleration systems aiming at accelerating for Multilayer Perceptron. All the PEs in the Neu-NoC are flat and the multi-cast is considered to share the same path in the data transmission. The data is reused through the ring topology and the PEs in the same ring share the same data in the network. However, the latency and throughput of ring topology is still a distinct disadvantage.

In recent years, the router architecture plays an important role in the design of NoC with achieving high throughput and low latency. In the NoC, router buffers are used to store flits that can't be forwarded immediately to the output which consumes lots of dynamic and static power of the router [15]. Thus, increasing the utilization of buffers reduces the area and power consumption. Further, one method of decreasing the latency is to design the router architecture which also achieving high throughput [24].

In this work, we propose the methodology to optimize the performance and energy of the communication in NN accelerators:

- We use a processor-chip-tile hierarchical architecture and demonstrate the necessity of redesigning this architecture. We propose the 64-tile distribution in one chip to improve the data reuse.
- We compare the performance and energy between different topologies in chips based on the traffic characteristic.
- Based on the characteristic of the traffic, we redesign the router architecture which incorporates the broadcast, buffer reducing, and additional path in the router that achieves better performance, higher throughput, and less energy.
- The results show that the broadcast router can improve the latency and energy by 8% and 9% respectively. The tree topology can achieve 37% lower energy consumption than the mesh topology. Finally, the optimized router architecture achieves 48% latency reduction and 61% energy reduction than traditional router architecture. When compared with the state-of-art AI chips, our optimized communication network can reduce the communication latency by 5.47× and save the energy

by 7.5× over DaDianNao, and by 7.57× for latency and 3.03× for energy over Eyriss.

The rest of the paper is organized as follows. First, we introduce the background of this paper in Section II. In Section III, we describe the motivation of our work. Then we propose our processor architecture in Section IV and describe our optimization methods of communication in Section V. Finally, we state our experimental results in Section VI and we conclude in Section VII.

II. BACKGROUND

In this section, we first introduce the basic computations of convolutional layers in CNNs. Then we briefly review the architecture of one typical AI processor DaDianNao and the strategy of mapping neural network to the AI processor. Finally, we describe the communication requirements of the communication network in AI chips.

A. THE STRUCTURE OF CNNS

A CNN uses a set of filters to extract features of the input feature maps (ifmaps), and each convolutional layer executes high-dimensional convolutions. As shown in Figure 2, the blue cube represents a set of two-dimensional (2D) ifmaps of the neural network; each of the M green cubes represents a three-dimensional (3D) filter which is composed of weights; the yellow cube is output feature maps (ofmaps) which consist of M yellow rectangles. Each yellow rectangle represents one channel of ofmaps.

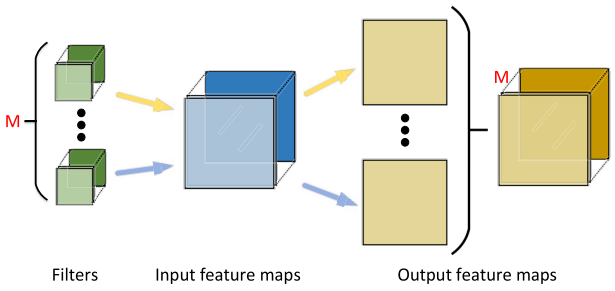


FIGURE 2. High dimensional convolutions in CNNs.

There are two types of data reuse in the convolution computation. First, as shown in Figure 2, the process of high dimensional convolutions is that M 3D filters are calculated with the same ifmaps to obtain ofmaps in each convolutional layer of the neural network. The same ifmaps are reused in the calculation of different channels (M) of 3D ofmaps. Second, as shown in Figure 3, the green 3D filter is convoluted with the data of ifmaps labeled by the red and orange cubes to obtain the data points in the channel of ofmaps. In the process of obtaining one channel of ofmaps, the same 3D filter performs a sliding convolution calculation on the ifmaps which means this filter is reused in these convolutions.

We use one convolutional layer in VGGNet [23] as an example of Figure 2. There are 256 filters ($M = 256$) in this layer, so the final ofmaps has 256 channels. These filters are

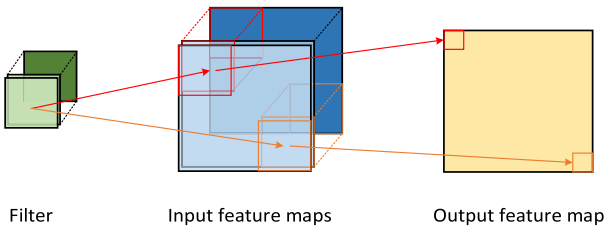


FIGURE 3. The computation in convolutional layers.

convoluted with the same ifmaps in the convolutional layer. If the data of 3D ifmaps cannot be reused, to calculate the 256 ofmaps, we need to read the ifmaps from the memory for 256 times, which brings large burdens to the memory and the communication network. Therefore, it is desired to reuse the ifmaps data as much as possible in the computing of CNNs.

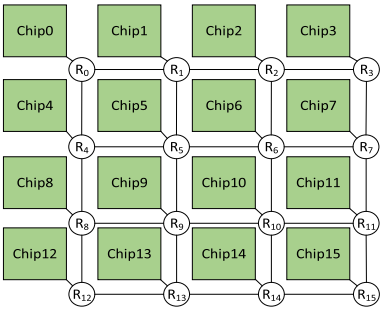


FIGURE 4. The mesh topology between chips.

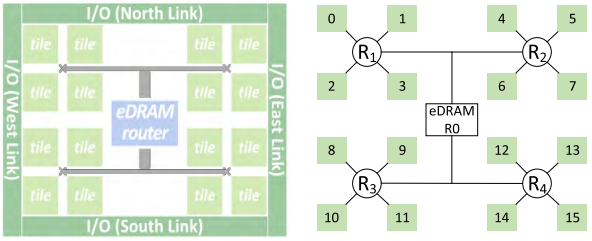


FIGURE 5. The Dadiannao chip. (a) The architecture of one DaDianNao chip [9]. (b) The tree topology in one DaDianNao chip with 16 tiles.

B. THE ARCHITECTURE OF AI CHIP

The hierarchical architecture has been applied to the architectural design of large-scale systems, such as the DaDianNao supercomputer with multiple chips [9]. As an example, Figure 4 shows one DaDianNao system of 16 chips connected by a mesh communication network. Each chip of this system is composed of 16 tiles, central eDRAMs, communication network, and I/O links shown in Figure 5. These 16 tiles are connected by five routers and links which constitute the H-tree topology (see Figure 5(b)). Each tile consists of the computation units (NFUs) and eDRAM banks as shown in Figure 6. The central memory is used to store the inputs

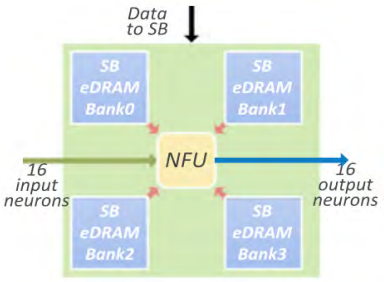


FIGURE 6. The architecture of one tile in the Dadiannao chip [9].

and outputs of each layer in CNNs, and the tile memory is used to store the weights of synapses between adjacent layers in CNNs and the intermediate results of computation when needed.

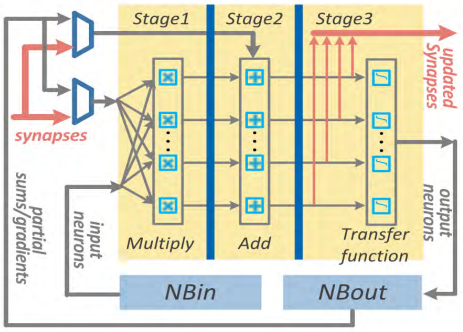


FIGURE 7. The architecture of computation units in the Dadiannao tile [9].

The computation process of NFUs is shown in Figure 7. The MACs in CNNs are implemented by the multipliers and adders in NFUs. The inputs of the multipliers include input neurons and the weights of synapses. Then the results of the multiplication, together with the previously calculated partial sum, are sent to the adders for addition. Then after the processing of transfer function, the final results are stored in the central memory.

C. CNNS MAPPED TO THE DADIANNAO SUPERCOMPUTER

Each layer of CNNs can be mapped to one or more chips to complete the MAC calculation, depending on the size of the CNN layer and the chip configuration. For instance, in an AI chip, each tile has limited tile memory, and this puts a constraint on the number of neurons that can be mapped to each tile. Therefore, given the architecture of an AI chip (such as the DaDianNao processor shown in Figure 4), we can figure out how to map a CNN to the AI chips.

As an example, Figure 8 shows how we can map three fully-connected layers in VGGNet [23] to the DaDianNao chips. Neurons in layer 1 are mapped to eight chips (chip0 - chip7) in the mesh topology. Meanwhile, neurons in layer 2 and layer 3 are mapped to another four chips (chip8, chip9, chip12 and chip13). The rest four chips

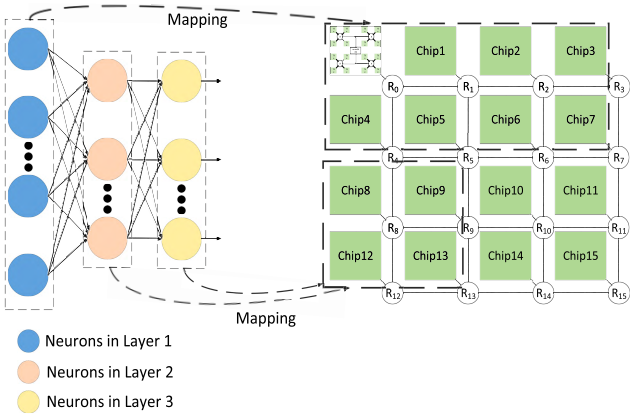


FIGURE 8. The neurons are mapped to different chips.

(chip10, chip11, chip14 and chip15) are used by the convolutional layers in VGGNet.

According to the hierarchical architecture, the computational tasks mapped to each chip are divided and assigned to smaller computing units in the chip. For example, one convolutional layer in VGGNet which has 512 filters are mapped to four chips (chip10, chip11, chip14 and chip15). Then eight filters and one 3D ifmaps of this convolutional layer are mapped to each tile of these chips shown in Figure 9.

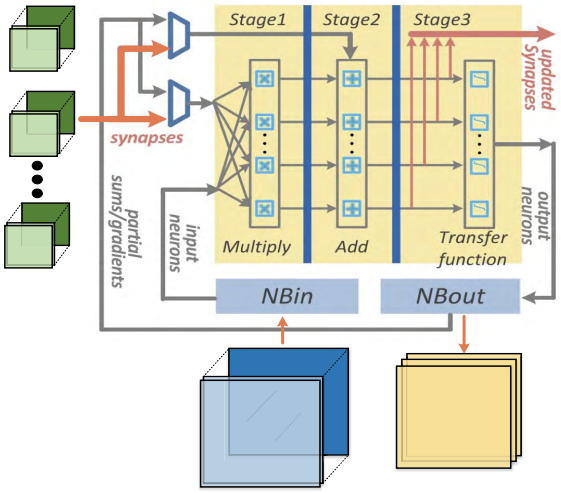


FIGURE 9. The computational tasks are mapped to each tile in DaDianNao chip.

The filters mapped to the tiles in these chips are different but tiles in these chips share the same ifmaps during the transmission. For these tiles, ifmaps are transmitted from the central memory to tiles and the flows are shown in the column ***process one*** of Table 2 based on the topology shown in Figure 5(b). When inputs from the central memory and weights from the tile memory are ready, these data enter the NFUs to complete the calculation. After the calculation, the results are transmitted from the tiles to central memory and flows are shown in the column ***process two*** of Table 2.

TABLE 2. The transmission flows between the central memory and tiles in the communication network of one DaDianNao chip shown in Figure 5(b).

Process one	Process two
$R_0 \rightarrow R_1 \rightarrow Tile_0$	$Tile_0 \rightarrow R_1 \rightarrow R_0$
$R_0 \rightarrow R_1 \rightarrow Tile_1$	$Tile_1 \rightarrow R_1 \rightarrow R_0$
$R_0 \rightarrow R_1 \rightarrow Tile_2$	$Tile_2 \rightarrow R_1 \rightarrow R_0$
$R_0 \rightarrow R_1 \rightarrow Tile_3$	$Tile_3 \rightarrow R_1 \rightarrow R_0$
$R_0 \rightarrow R_2 \rightarrow Tile_4$	$Tile_4 \rightarrow R_2 \rightarrow R_0$
$R_0 \rightarrow R_2 \rightarrow Tile_5$	$Tile_5 \rightarrow R_2 \rightarrow R_0$
$R_0 \rightarrow R_2 \rightarrow Tile_6$	$Tile_6 \rightarrow R_2 \rightarrow R_0$
$R_0 \rightarrow R_2 \rightarrow Tile_7$	$Tile_7 \rightarrow R_2 \rightarrow R_0$
$R_0 \rightarrow R_3 \rightarrow Tile_8$	$Tile_8 \rightarrow R_3 \rightarrow R_0$
$R_0 \rightarrow R_3 \rightarrow Tile_9$	$Tile_9 \rightarrow R_3 \rightarrow R_0$
$R_0 \rightarrow R_3 \rightarrow Tile_{10}$	$Tile_{10} \rightarrow R_3 \rightarrow R_0$
$R_0 \rightarrow R_3 \rightarrow Tile_{11}$	$Tile_{11} \rightarrow R_3 \rightarrow R_0$
$R_0 \rightarrow R_4 \rightarrow Tile_{12}$	$Tile_{12} \rightarrow R_4 \rightarrow R_0$
$R_0 \rightarrow R_4 \rightarrow Tile_{13}$	$Tile_{13} \rightarrow R_4 \rightarrow R_0$
$R_0 \rightarrow R_4 \rightarrow Tile_{14}$	$Tile_{14} \rightarrow R_4 \rightarrow R_0$
$R_0 \rightarrow R_4 \rightarrow Tile_{15}$	$Tile_{15} \rightarrow R_4 \rightarrow R_0$

Without special design, each tile needs to load ifmaps 8 times from the central memory. This process consumes a large amount of energy and incurs significant latency in the communication network. Therefore, we should reuse the ifmaps whenever possible in the design of the AI communication network.

D. TRANSMISSION OF DIFFERENT DATA IN A COMMUNICATION NETWORK

In the chip, modules can be divided into computation units and memory units by functional classification. Before the calculation of neurons of each layer in CNNs, the weights of synapses and the inputs (ifmaps) of neurons need to be transmitted from the memory to the computation tiles as shown in Figure 10(a). After calculation, the results of the computation (ofmaps) are transmitted to memory and stored in the memory as shown in Figure 10(b).

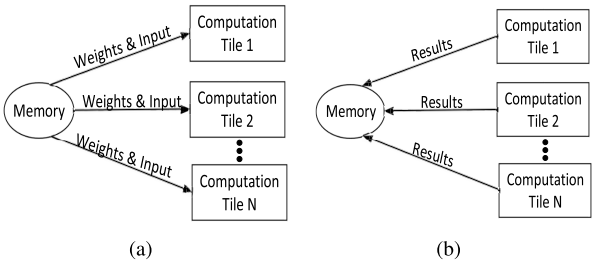


FIGURE 10. Communication requirements between memory and computation tiles.

From the aspect of data transmission in CNNs, the communication requirements are that computation results of neurons of layer1 and weights of synapses are transmitted to the neurons of the next layer 2 as shown in Figure 11. Therefore, a normal CNN has only communication requirements between the adjacent layers.

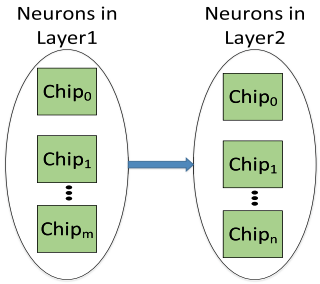


FIGURE 11. The communication requirements between adjacent layers.

III. MOTIVATION OF OUR WORK

The hierarchical architecture as processor-chip-tile presented in Section II-B is an effective hardware architecture design for neural network implementation. In this section, we study the potential problems of this architecture and then propose some opportunities for improvements in terms of communication network design and optimization.

A. THE PROBLEMS OF THE ARCHITECTURE

1) INTER-CHIP COMMUNICATION

If each CNN layer is mapped to several chips, the neurons of one layer are distributed in different chips. The communication requirements between different chips increase significantly, correspondingly, inter-chip communication becomes a bottleneck.

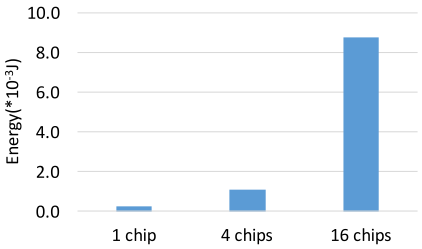


FIGURE 12. The energy cost of inter-chip communication. Note that the x-bar represents the number of chips that each CNN layer is mapped to.

The Alexnet [20] is composed of 5 convolutional layers and 3 fully-connected layers. In our experiments, we map each layer of Alexnet to 1, 4 or 16 chips respectively. In this way, the CNN is mapped to a multi-chip system whose chips are connected with a mesh topology. As shown in Figure 12, the energy of inter-chip communication increases obviously when we map the same CNN layer to more chips. Meanwhile, it is known that the energy consumption of inter-chip transmission is higher than the energy of intra-chip transmissions [21]. Thus, to reduce communication energy, we can decrease the communication between chips.

2) MEMORY READ AND WRITE

According to the architecture of Dadiannao, the data needs to be transmitted between the memory and tiles. Among the

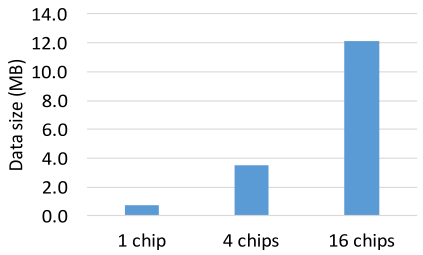


FIGURE 13. The amounts of data read from the memory in chips. Note that the x-bar represents the number of chips that each layer is mapped to.

transmission, we read the input neurons from the memory and write the output results to the memory. When one CNN layer is mapped to several chips, each chip has to store its separate copy of the inputs of neurons in its memory. This is because each neuron needs the outputs of all the neurons in the previous CNN layer as the inputs. As shown in Figure 13, the size of data which read from the memory varies directly with the number of chips that each layer is mapped to. The operations that read and write from the memory cost significant energy and latency. When the size of each tile (i.e., the number of neurons it can hold) is fixed, the more tiles we place in one chip, the less number of chips will be used by one CNN layer. Thus the number of tiles in one chip has a close correlation with the energy consumption and latency of the communication network in an AI chip.

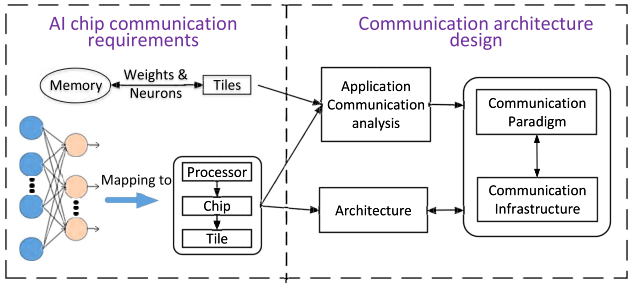


FIGURE 14. The NoC design and optimization of AI chips.

B. OPPORTUNITIES FOR COMMUNICATION OPTIMIZATION

As shown in Figure 14, the weights of synapses and the inputs to the neurons in a CNN need to be transmitted between memory and computation tiles following the hierarchical structure of an AI chip. This imposes the communication requirements on the AI hardware. According to the characteristics of transmission and the architecture, we can design the communication infrastructure and communication paradigm of the NoC, whose communication infrastructure includes the topology, router and channel, and communication paradigm includes power management, packet routing and switching techniques.

As discussed in Section III-A, the architecture design of an AI chip influences the energy consumption of inter-chip communications and the burden of loading data from memory, thus the architecture design of AI accelerators is one opportunity for optimizing the communication network. In addition, to optimize the communication network in the neural network, there are several methods to decrease the latency and energy of NoC:

- Data reuse. The data read from the memory can be reused using the broadcast technique [28]. In our work, we design a new type of routers with the broadcast function (see Section V-A) to reuse the data transferred from the memory to all the tiles in one chip. This method can increase the utilization of data read from memory and decrease the latency of the intra-chip communication network.
- Topology. The intra-chip NoC topology can be optimized to boost the performance and reduce the energy consumption of the communication network in AI chips. We analyze the characteristic of data transformation and compare the performance and energy of different intra-chip topologies in Section V-B and Section VI-A.
- Router architecture. In the NoC network, the important components are routers. The router architecture can be redesigned to decrease the latency and energy of NoCs in Section V-C.

IV. OUR PROPOSED PROCESSOR ARCHITECTURE

As aforementioned, neurons of one CNN layer may be mapped to several chips that induces more inter-chip communication and memory cost. One feasible method is to place more tiles in one chip. In our work, we assume P tiles are placed in one chip. Our experiments show that $P = 64$ can achieve a good trade-off between chip complexity and performance, in contrast to $P = 16$ in DaDiannao. As we place 64 tiles in one chip, the complexity of communication between tiles grows rapidly compared with 16 tiles. With the growing number of tiles, one common and efficient NoC topology is mesh [22]. In order to decrease the dimension of topology, 64 tiles are organized as four similar minor topologies (MT) in one chip. In the MT, each tile equips with a router (denoted by R), and 16 tiles represented as green squares are arranged into a 4x4 mesh shown in Figure 15. We use one router R_0 in the center of the chip to assign the on-chip memory resources.

In the transformation process of CNNs, the input neurons are fetched from the central memory to each tile and the weights of synapses are fetched from the memory of tiles. After that, these input neurons and the weights are calculated in each tile. Finally, the calculation results of neurons are transmitted from tiles to the central memory.

The above process is implemented based on the topology in Figure 15. From the network point of view, the above transformations can be divided into two processes:

- Data-in process: The input data is transmitted from R_0 to other 64 routers. As some packets including the same data are transmitted from the same source to different destinations, these packets are duplicated.

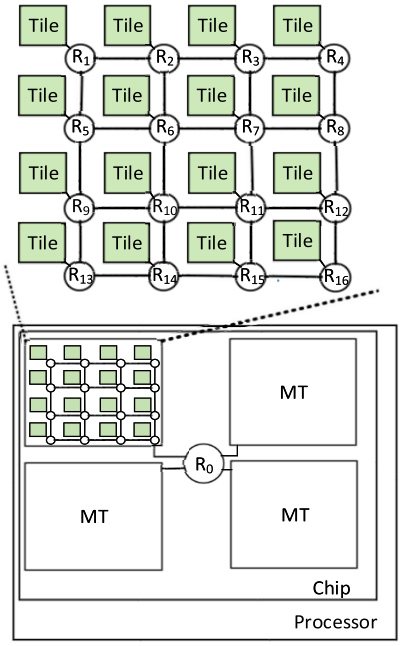


FIGURE 15. The novel architecture of AI processor.

- Data-out process: The results of all output neurons are transmitted from 64 routers to R_0.

V. NOC OPTIMIZATION

In this section, we propose optimizations of the communication network from three aspects: data reuse, topology optimization and router optimization.

A. DATA REUSE

As described in Section IV, the traffic is transmitted from router R_0 to routers $R_1 \sim R_{16}$ in one MT in the data-in process. And for routers, most packets travel from the same input to different outputs in the data-in process. This characteristic leads to large latency because of the long waiting time before entering routers. In order to decrease the latency, we apply the broadcast to the router.

1) ORIGINAL TRANSMISSION SCHEME OF DATA-IN PROCESS

The original scheme uses the regular routers in the communication network, which results in large transmission latency between the memory and tiles. For example, in Figure 15, the data is first transmitted from R_0 to R_{16}. Their next step is from the R_{16} to R_{12}, R_{15} and the tile connected to R_{16}. In this process, the data are transmitted in the form of packets as shown in Figure 16. Each packet consists of flits which are divided into head flit, body flits, and tail flit. The head flit contains the source and destination of the packet. The body flits is the main carrier of data transmission. In R_{16}, three packets (A, B and C) which contain the same data are transmitted from the input East to outputs (West, North and Eject) with arriving different destinations shown

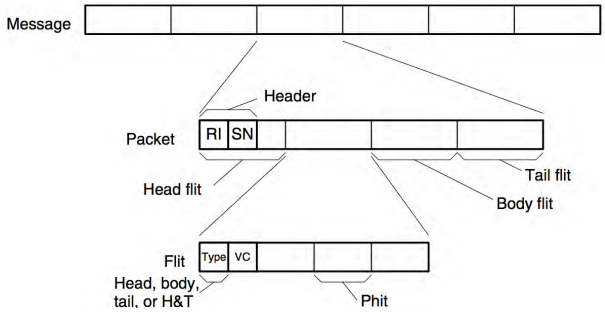


FIGURE 16. Message structure [13].

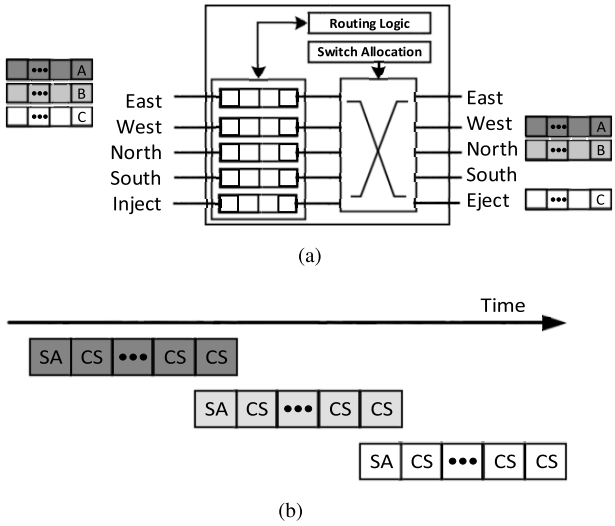


FIGURE 17. Transmission of three packets include the same data in the regular router. (a) Regular router architecture. (b) Transmission process of three packets in the crossbar of regular router.

in Figure 17(a). The router in Figure 17(a) is one regular router in the mesh topology. The regular router has four pipelines: buffer write (BW), routing logic (RL), switch allocation (SA), crossbar switch (CS). In switch allocation, the crossbar allocates only one input for one output at a time. When the traffic comes from the same input to different outputs or traffic from different inputs to the same output, the switch allocation can only handle these traffic one by one.

When these packets (A, B and C) go through the crossbar, the switch allocation allocates one packet (assume the packet A) going through the crossbar first. The packets B and C need to wait until whole flits of packet A go through the crossbar, then switch allocation allocates the next packet going through crossbar, shown in Figure 17(b). If most packets go through the router from the same input to different outputs, the latency cost in the router will be large because of the long waiting time before transmitting the crossbar.

2) BROADCAST TRANSMISSION SCHEME OF DATA-IN PROCESS

For one convolutional layer in CNNs, in the data-in process, the inputs of neurons are ifmaps, which are transmitted from

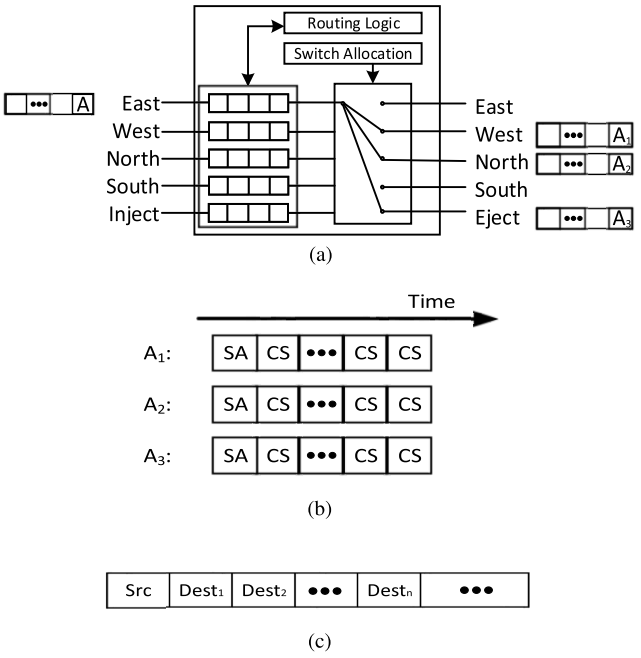


FIGURE 18. Broadcast. (a) Broadcast router architecture. (b) Transmission process of packets in the crossbar of broadcast router. (c) Head flit of broadcast packets.

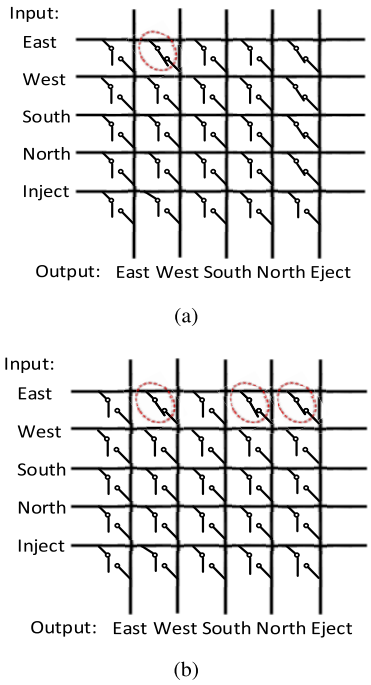


FIGURE 19. The architecture of crossbar. (a) Original crossbar architecture. (b) Optimized crossbar architecture.

the central memory to each tile in the chip. If the convolutional layer is mapped to all the tiles in the chip, ifmaps for these tiles are identical and can be reused in the transmission. In the original transmission scheme, the packets from R_0 to routers $R_1 \sim R_{16}$ are 16 copies of ifmaps for a convolutional layer. For example, three packets (A, B and C) stores the same data of ifmaps and their destinations are respectively R_{16}, R_{15} and R_{12}. If we use the broadcast technique, these duplicate packets can be reduced to one broadcast packet (A). The broadcast packet stores the same source and different destinations in the head flit of packets as shown in Figure 18(c). In the broadcast router shown in Figure 18(a), one packet (A) is transmitted across the router and packet A is copied to three packets (A_1, A_2 and A_3) which have the same data and different destinations. To implement the broadcast in routers, we add the parallel transformation parts in the new router architecture and redesign the crossbar linking method. In the regular crossbar linking, each row line (East) only connects one column line (West) at the same time shown in Figure 19(a). This method guarantees orderly transmissions of packets from same input to different outputs in routers. In the broadcast router, the crossbar is redesigned to transmit the same data to different outputs so that each row line (East) can connect several column lines (West, North and Eject) at the same time shown in Figure 19(b). After buffer write and routing computation, we establish paths between the input East and outputs (West, North and Eject) in the crossbar. By setting the parallel paths, each flit of the packet is transmitted three copies to different outputs. The latency of the data transmission is decreased in the crossbar because flits are transmitted through it in a parallel way. As shown in Figure 18(b), after SA pipeline, the flits of packet A are transmitted to the outputs as packets A_1, A_2 and A_3 in parallel.

B. TOPOLOGY OPTIMIZATION

The performance and energy are two important metrics in the design of the communication network. The topology of the network defines the connection between routers and tiles, and it plays an important role in the NoC network. Although the mesh topology is used in Figure 15, more topologies, such as torus and tree, can be chosen for specific applications. The torus topology which adds some direct links between routers decreases the hop-counts of two routers in the bordering edges of the mesh topology. Meanwhile, the tree topology has a different connection between routers to further decrease the number of routers.

In the Section IV, the analysis of the traffic pattern shows that most of the packets are transmitted between the central memory and tiles while rarely transmitted between the tiles. Thus we want to find a topology that decreases the latency and energy between the central memory and tiles. Meanwhile, the freedom of transmission between tiles can be sacrificed.

Three topologies can be applied in the intra-chip communication network: mesh, torus and tree. After using the broadcast routers in the communication network, the traffic arrives at different tiles almost at the same time in the data-in process. If the numbers of neurons mapped to different tiles are similar, the end times of these calculations within different tiles are also close to each other. That means the flows begin to transmit at a similar time in the data-out process. The similar

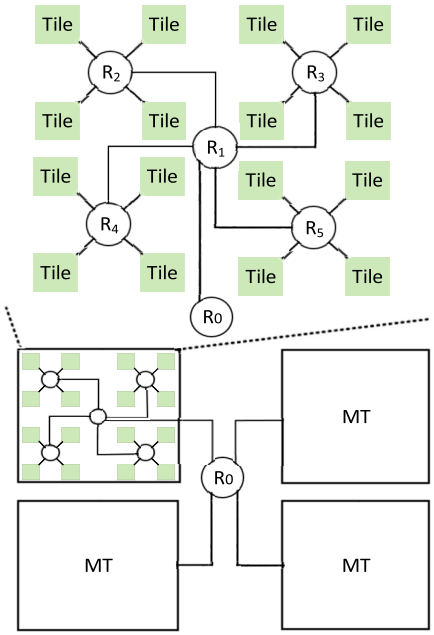


FIGURE 20. Tree topology of one chip.

start time of the traffic flows leads to the crowd in the communication network. In order to achieve short latency and less communication energy, the tree topology shown in Figure 20 is preferred over mesh or torus due to fewer used routers and links in this topology. The detailed comparisons between different topologies will be discussed in Section VI.

C. ROUTER OPTIMIZATION

In this section, we design one novel router architecture according to the traffic patterns in the data-in and data-out process. We analyze the traffic paths between routers in tree topology and find out that many routine patterns are fixed. Based on this observation, we redesign the router architecture to speed up the transmission and improve the throughput of routers.

1) DATA-IN PROCESS

Before the traffic starts, the results of the former layer is stored in the central memory. The data is transmitted from the central memory to tiles, and these tiles begin to calculate the MACs of the next layer.

As shown in Figure 20, the traffic flow broadcasts from the router R_0 to the 16 tiles through the $R_1 \sim R_5$. The traffic is transmitted from R_0 to R_1, then R_1 broadcasts to $R_2 \sim R_5$. For example, in the transformation of router R_1, the traffic is transmitted from one stationary input (Inject) to different outputs (East, West, South and North). The buffers in regular router architecture except the buffer of input Inject are idle in the data-in process. Thus, we optimize the number of buffers of routers and only use one buffer in simplified router shown in Figure 21. We link this buffer with five inputs (East, West, North, South and Inject). At a time, only one input enters the traffic flow which means there is no contention in the buffer in the data-in process.

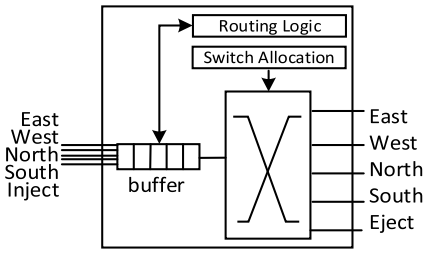


FIGURE 21. Router architecture with simplified buffer.

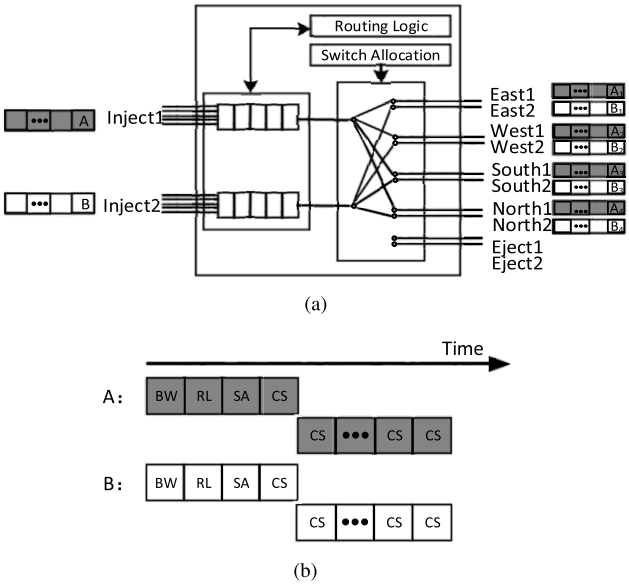


FIGURE 22. Optimized router architecture in the data-in process. (a) Two packets are transmitted through the router in the data-in process. (b) Transmission process of two packets in the router.

Although we simplify the architecture of routers, the throughput of routers has not increased. We add an additional path to the router as shown in Figure 22(a). Each output has two paths connecting to the two inputs of next routers. For example, the inputs Inject1 and Inject2 are connected with the two paths of the same output of the previous router. Two packets (A, B) enter inputs Inject1 and Inject2 separately. Both packets A and B are transmitted to the outputs East, West, South and North. In the regular router, the results of routing computation are stored in the look up table (LUT). The routing request of one packet which has known the source and destination from the routing computation is represented as 'Yes' in the LUT in Figure 23(a). The LUT configures the connections of the crossbar. As we redesign the router architecture, the LUT is transformed as shown in Figure 23(b). The input rows are reduced to Input 1 and Input 2 as the optimized router only has two buffers. Meanwhile, because of the broadcast, each input can connect several outputs in the crossbar. Each row has four routing requests in the LUT, and flits of packet A and packet B can be transmitted in parallel. The packets A, B are transmitted to four different destinations, and the transmission process of these two packets in the router is shown in Figure 22(b).

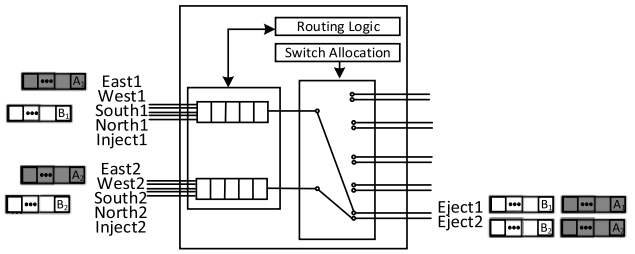
Routing request					
Input/Output	East	West	South	North	Eject
Inject	Yes				
East					
West					
South					
North					

(a)

Routing request										
Input/Output	East1	East2	West1	West2	South1	South2	North1	North2	Eject1	Eject2
Input1	Yes		Yes		Yes		Yes			
Input2		Yes		Yes		Yes		Yes		

(b)

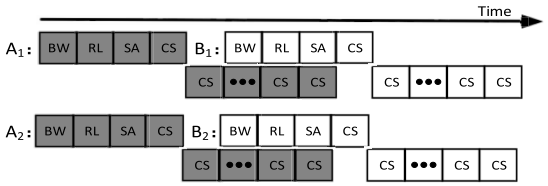
FIGURE 23. Look up table optimization. (a) The general LUT. (b) Optimized LUT.



(a)

Routing request										
Input/Output	East1	East2	West1	West2	South1	South2	North1	North2	Eject1	Eject2
Input1									Yes	
Input2										Yes

(b)



(c)

FIGURE 24. Router architecture in the data-out process. (a) Four packets are transmitted through the router in the data-out process. (b) Optimized LUT. (c) Transmission process of four packets in the router.

2) DATA-OUT PROCESS

In the Data-out process, the outputs of neurons are transmitted from tiles to the central memory. The traffic is transmitted from 16 tiles to the router R_0 as shown in Figure 20. The traffic is transmitted from $R_2 \sim R_5$ to R_1, then R_1 to R_0. In the transformation of routers, the traffic enters different inputs and merge to the stationary output. However, these packets that merge to the same output have the same routing request and connections of the crossbar because of the novel router architecture shown in Figure 24(a). For example, in router R_1, the traffic which comes from inputs East1 and South1 to the output Eject1 has the same crossbar connections from Input1 to Eject1. The configure in Input1 row and Eject1 column of the LUT is "Yes" shown in Figure 24(b). The priority of packets entering buffer and switch allocation is first come first serve (FCFS). In the crossbar, two paths of output Eject can export two flits at the same time. The transmission process in the router is shown in Figure 24(c).

VI. EXPERIMENTAL RESULTS

In this section, we first present the performance and energy results of our methods from three aspects: broadcast, topology and router architecture. Then we compare the performance and energy of communication network between our design and DaDianNao [9], Eyeriss [18]. We simulate the design in C++ and all the experiments are conducted on a computer with a 3.4 GHz Intel i7 processor and 32 GB memory running Linux. The benchmarks are Alexnet [20], VGGNet [23], Drive [1], Neuflow [26]. The parameters for the NoCs are set as 64-bit flit and 16-flit packet.

A. THE RESULTS OF OUR OPTIMIZATION METHODS

In this section, we compare our work with a baseline design, which uses 1) mesh topology for each MT in the chip (see Figure 15), 2) 64 tiles for each chip, and 3) regular 4-pipeline wormhole pipeline in each router.

1) IMPACT OF BROADCAST

In this section, we present the latency and energy comparison between the baseline and the broadcast router to mesh topology. Table 3 shows that the benchmarks we have tested between baseline and the broadcast router. The latency results are intra-chip communication latency. Table 3 shows the latency results of adding broadcast routers are 8% of the baseline and the energy results of adding broadcast routers are 9% of the baseline.

TABLE 3. Comparisons of latency and energy results of benchmarks between baseline and broadcast router.

Layer	Latency (cycles)		Energy ($*10^{-3}$J)	
	Mesh	Mesh+Broadcast	Mesh	Mesh+Broadcast
Drive	730458	61626	5.023	0.425
AlexNet	1353918	105126	9.309	0.725
Neuflow	32554018	362923	22.377	2.5
VGGNet_A	11674447	913486	60.936	6.295
VGGNet_E	40617447	3107898	259.982	21.42
Average	1	0.08	1	0.09

We use three convolutional layers in Alexnet benchmark to explain the reasons for the improvements. The broadcast technique is used in the data-in process while it influences both the data-in and the data-out process. In the data-in process, the former layers' results are transmitted to the tiles to complete the computation. For example, the outputs of the 1st convolutional layer which are transmitted from the central memory to tiles are the input of the 2nd convolutional layer. The input size is 27*27*96 and the size of 256 3D convolution filters is 5*5*96. We map filters to 64 different tiles in one chip and each tile has four 3D filters with the size of 5*5*96. Also, the calculation amounts in tiles are roughly the same, the amounts and value of the input data are the same which are

TABLE 4. Comparisons of latency results between baseline and broadcast router.

Layer	Mesh Latency (cycles)		Broadcast Mesh Latency (cycles)	
	Data-in	Data-out	Data-in	Data-out
CONV1	321030	321	24272	705
CONV2	275310	209	20424	259
CONV3	171570	289	12728	2829
Average	1	1	0.07	4.63

transmitted to different tiles. Then each tile has 27*27*4 size of outputs after MACs of convolution and these output data converges to the central memory.

With applying the broadcast, the latency of Data-in decreases to 7% latency of the baseline in Alexnet. That is because the broadcast router can transfer the traffic from the same input to different outputs in parallel and the number of packets is reduced. In the data-out process, because the traffic arrived at different tiles at a similar time, the ending time of computation is also similar. When the packets are transmitted back to the central memory, the network is more crowd than it in the baseline. The latency of the data-out process is 4.63X larger of the baseline.

However, the amount of data of the data-in process is larger than the data-out process in general. The broadcast technique is worth adding. After using the broadcast technique, the number of packets can be merged into one if packets have the same data and from the same input. Because of the shorter latency and fewer packets, the energy of the process has been decreased to 9% in Table 3.

TABLE 5. Comparisons of latency results of benchmarks among different topologies.

Layer	Latency ($*10^4$cycles)			Energy ($*10^{-3}$J)		
	Mesh	Torus	Tree	Mesh	Torus	Tree
Drive	6.16	6.09	5.99	0.425	0.515	0.284
Neuflow	56.82	36.08	37.90	2.500	3.057	1.004
AlexNet	10.51	10.75	10.70	0.725	0.911	0.284
VGGNet_A	91.35	90.49	89.51	6.295	7.670	2.320
VGGNet_E	310.79	304.74	294.10	21.420	25.833	7.694
Average	1.0	0.942	0.921	1.00	1.21	0.37

2) TOPOLOGY COMPARISON

In this section, we present the results of three different topologies: mesh, tree, torus. Table 5 shows that the latency of tree topologies is 7.9% reduction than Mesh and the energy is 37% of Mesh topology. Thus we choose the tree topology as our topology.

In general, the latency of torus has better performance than the mesh and tree topology. However, in the result of Alexnet, the latency of the torus is larger than tree and mesh topology. As shown in Figure 25, we can find that the benefits of torus topology in the data-in process are counteracted in the data-out process. Although the torus decreases the hop-count from the central memory to the tiles, the contention degree of traffic in Data-out process is increased after applying the broadcast method.

Also, in the torus and mesh topology, the amounts of routers are 3X of the routers in torus topology. In torus topology, the routers in the edge have one or two more inputs and outputs than routers in the mesh. The static power of the input buffers in the torus topology costs significant energy of NoC.

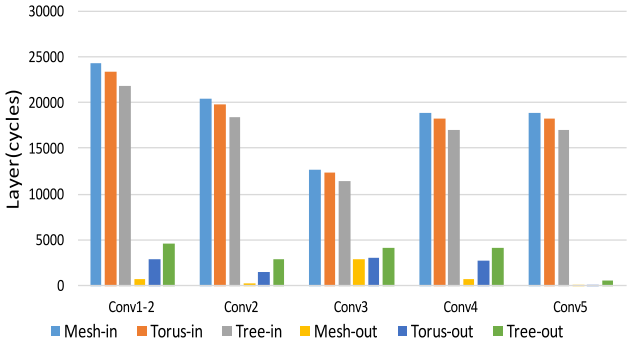


FIGURE 25. Latency of different topology in Alexnet.

TABLE 6. Comparisons of latency and energy results of benchmarks between TB and OTB.

Layer	Latency (cycles)		Energy ($*10^{-3}$J)	
	TB	OTB	TB	OTB
Drive	59916	31110	0.284	0.090
AlexNet	106082	55561	0.284	0.161
Neuflow	379005	193179	1.004	0.559
VGGNet_A	89511	465149	2.320	1.350
VGGNet_E	2940893	1531633	7.694	2.439
Average	1	0.52	1	0.39

3) ROUTER ARCHITECTURE OPTIMIZATION

In this section, we present the results of the architecture optimization of the broadcast router. Table 6 shows the latency and energy results comparison of TB (broadcast router in tree topology) and OTB (the optimized broadcast router in tree topology). The results show that OTB is 52% of TB and the energy results of OTB are 39% of TB.

Intuitively speaking, adding one path in routers should decrease 50% latency than before. For example, in the data-in process, the input of R_2 all comes from R_1 and the packets transformation is shown in Figure 22(a). When the packets come, two different packets A and B come to enter the input from Inject1 and Inject2, then the two packets go through the routing computation and switch allocation. Thus each packet needs these additional cycles to allocate the paths in routing. The improvements in OTB's latency is slightly less than 50% of TB. The optimized router architecture of OTB has less buffer, the static and dynamic power of the router are decreased. Thus, the energy of OTB decreases more than 50% of TB.

B. COMPARISONS BETWEEN DADIANNAO, EYERISS AND OUR DESIGN

In this section, we present the comparison results among DaDianNao [9], Eyeriss [18] and our design on five benchmarks, in terms of the latency and energy consumption of the communication network.

TABLE 7. Comparisons of latency and energy results of five benchmarks between DaDianNao and our method.

Layer	Latency (cycles)			Energy ($*10^{-3}$J)		
	Eyeriss	DaDianNao	Our method	Eyeriss	DaDianNao	Our method
Drive	282686	368166	55561	0.114	0.665	0.046
AlexNet	568988	200678	31374	0.310	0.938	0.082
Neuflow	2379415	1833671	193179	1.614	4.087	0.559
VGGNet_A	4444794	3145969	467010	2.585	10.822	1.374
VGGNet_E	15583866	11246197	2324411	9.068	17.413	2.463
Average	7.57	5.47	1	3.03	7.5	1

1) COMPARISONS BETWEEN DADIANNAO AND OUR DESIGN

For a fair comparison, we use mesh topology for inter-chip communication and assign the same amount of computation tasks to each computation units in both DaDianNao and our design. The routers in the mesh topology are regular 4-pipelined routers. As shown in Figure 26, the left is a multi-chip system composed of 16 DaDianNao chips which are connected with mesh, and the right is a multi-chip system formed with our design chips with mesh topology. We place 64 tiles in one chip ($P = 64$) in our design chip, while each DaDianNao chip has 16 tiles.

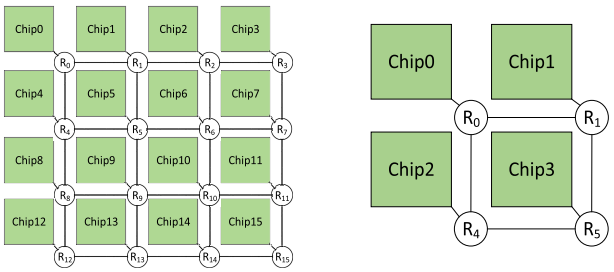


FIGURE 26. The comparison of DaDianNao and our design system.

Table 7 shows that the latency of DaDianNao is 5.47× of our design, and the energy is 7.5× of our design communication network. In the DaDianNao system (left in Figure 26), each convolutional layer is mapped to four Dadiannao chips. In our design system (right in Figure 26), each convolutional layer is mapped to one chip correspondingly. In the Dadiannao system, as calculations of one convolutional layer are completed, the results from each chip need to be transmitted to the other three chips for the next layer calculations, so each layer needs to spend some inter-chip transmission time to transfer the results to the next layer. However, in our design, the calculation results of each convolutional layer only need to be transmitted from the central memory to each computation unit in one chip to complete the data transmission, thus our design saves a large amount of inter-chip transmission time. Meanwhile, the DaDianNao system uses more routers because of larger mesh topology for inter-chip communication shown in Figure 26. The inter-chip communication of DaDianNao consumes more energy because of more routers in the communication network.

The intra-chip communication network of DaDianNao chip and our design chip are tree topology shown in Figure 5(b) and Figure 20. Our design uses 21 routers in one chip while DaDinaoNao system uses 20 routers in four chips. However, the optimization of router architecture in our design reduces the transmission time inside the chip and decrease the energy of routers in the chip. So our design reduces the latency and energy of both inter-chip and intra-chip transmission compared to DaDianNao.

2) COMPARISONS BETWEEN EYERISS AND OUR DESIGN

As shown in Figure 27, Eyeriss [18] uses 12 horizontal X-buses connected to 168 processing elements (PEs) and one Y-bus to determine which X-bus the data transmitted to. Eyeriss has two large memory blocks. One is the global buffer and another one is the off-chip memory. The data in Eyeriss is transmitted from off-chip DRAM to PEs with traveling across the global buffer.

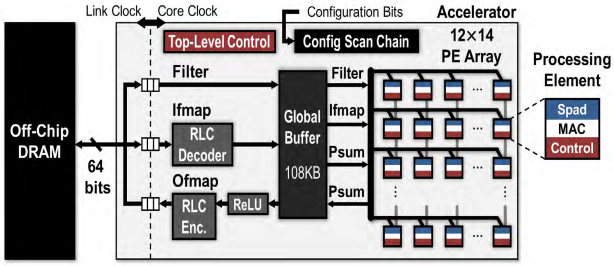


FIGURE 27. The architecture of Eyeriss [18].

Table 7 shows that the latency of Eyeriss is 7.57× of our design, and the energy is 3.03× of our design communication network. From the aspect of transmission distance, the data in Eyeriss need to be transmitted from off-chip DRAM to PEs with travel across global buffer, while the data is transmitted from the central memory to PEs in our design. Also, the size of the global buffer is not large enough to store inputs and weights of layers in CNNs, most data is stored in off-chip memory. The bandwidth of data transmission is limited (64 bits/cycle) from off-chip memory to the global buffer in Eyeriss. In contrast, our design broadcasts the data from the memory to all the tiles in one chip through optimized routers simultaneously. Thus the latency of Eyeriss is much higher than our design.

The energy of bus in Eyeriss is calculated based on parameters from [27]. The Eyeriss uses more control units of communication than our design, but the power of bus control units in Eyeriss is approximate 33% of the power of control

units (routers) in our design. Thus the energy of Eyeriss is only 3.03× of our design.

VII. CONCLUSION

In this work, we propose our processor architecture based on the hierarchical architecture. We also optimize the topology and router architecture to improve the performance and energy of intra-communication network in chips. In the results, we totally achieve better latency and energy results than the baseline and other architecture design.

REFERENCES

[1] M. Bojarski *et al.* (Apr. 2016). "End to end learning for self-driving cars." [Online]. Available: https://arxiv.org/abs/1604.07316
[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015. [Online]. Available: https://www.nature.com/articles/nature14539
[3] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," presented at the ICML, Jun. 2007. [Online]. Available: https://dl.acm.org/citation.cfm?id=1273556
[4] K. He, X. Zhang, S. Ren, and J. Sun. (Dec. 2015). "Deep residual learning for image recognition." [Online]. Available: https://arxiv.org/abs/1512.03385
[5] J. Emer, V. Sze, Y. Chen, and T. Yang, "Hardware architectures for deep neural networks," presented at the ISCA Tutorial, Jun. 2017. [Online]. Available: http://www.jics.tennessee.edu/files/images/csure-reu/2015/Tutorial/DNN/DNN-Intro-2.pdf
[6] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," presented at the NIPS, Dec. 2011. [Online]. Available: https://ai.google/research/pubs/pub37631
[7] Q. V. Le *et al.* (Dec. 2011). "Building high-level features using large scale unsupervised learning." [Online]. Available: https://arxiv.org/abs/1112.6209
[8] Y. Chen *et al.*, "DaDianNao: A machine-learning supercomputer," presented at the MICRO, Dec. 2014. [Online]. Available: https://dl.acm.org/citation.cfm?id=2742217
[9] T. Luo *et al.*, "Dadiannao: A neural network supercomputer," *IEEE Trans. Comput.*, vol. 66, no. 1, pp. 73–88, Jan. 2017. doi: 10.1109/TC.2016.2574353.
[10] S. Yin *et al.*, "A high energy efficient reconfigurable hybrid neural network processor for deep learning applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 968–982, Apr. 2018. doi: 10.1109/JSSC.2017.2778281.
[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," presented at the CVPR, Jun. 2009. [Online]. Available: https://www.computer.org/csdl/proceedings/cvpr/2009/3992/00/05206848-abs.html
[12] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.
[13] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Amsterdam, The Netherlands: Elsevier, 2004.
[14] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler, "vDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design," presented at the MICRO, Oct. 2016. [Online]. Available: https://dl.acm.org/citation.cfm?id=3195660
[15] C. Van Thiem and S. Oyanagi, "An input buffer architecture for on-chip routers," presented at the ICNC, Dec. 2011. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6131842
[16] X. Liu *et al.*, "Harmonica: A framework of heterogeneous computing systems with memristor-based neuromorphic computing accelerators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 5, pp. 617–628, May 2016. doi: 10.1109/TCSI.2016.2529279.
[17] Y.-H. Chen, J. Emer, and V. Sze. (Jul. 2018). "Eyeriss v2: A flexible and high-performance accelerator for emerging deep neural networks." [Online]. Available: https://arxiv.org/abs/1807.07928
[18] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–128, Jan. 2017. doi: 10.1109/JSSC.2016.2616357.
[19] X. Liu, W. Wen, X. Qian, H. Li, and Y. Chen, "Neu-NoC: A high-efficient interconnection network for accelerated neuromorphic systems," presented at the ASPDAC, Jan. 2018. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8297296.
[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," presented at the NIPS, 2012. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks
[21] F. Akopyan *et al.*, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015. doi: 10.1109/TCAD.2015.2474396.
[22] E. Cota, A. Amory, and M. S. Lubaszewski, *Reliability, Availability and Serviceability of Networks-on-chip*. Springer, 2011.
[23] K. Simonyan and A. Zisserman. (Sep. 2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[24] R. S. Ramanujam, V. Soteriou, B. Lin, and L.-S. Peh, "Design of a high-throughput distributed shared-buffer NoC router," presented at the NOCS, May 2010. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5507561.
[25] Q. V. Le *et al.* (Dec. 2011). "Building high-level features using large scale unsupervised learning." [Online]. Available: https://arxiv.org/abs/1112.6209
[26] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "Neuflow: A runtime reconfigurable dataflow processor for vision," presented at the CVPR Workshops, Jun. 2011. [Online]. Available: https://www.computer.org/csdl/proceedings/cvprw/2011/0529/00/05981829-abs.html
[27] H. G. Lee, N. Chang, U. Y. Ogras, and R. Marculescu, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," *ACM Trans. Design Autom. Electron. Syst.*, vol. 12, no. 3, p. 23, Aug. 2007. doi: 10.1145/1255456.1255460.
[28] M. Hosseinabady, "A concurrent testing method for NoC switches," presented at the DATE, Mar. 2006. [Online]. Available: https://dl.acm.org/citation.cfm?id=1131804

WEI GAO (S'13) received the B.E. degree from Shanghai University, Shanghai, China, in 2013. She is currently pursuing the Ph.D. degree with the Chinese Academy of Sciences, Shanghai. Her current research interests include network-on-chip and artificial intelligence accelerators.

PINGQIANG ZHOU (M'14) received the B.E. degree from the Nanjing University of Posts and Telecommunications, China, in 2005, the M.E. degree from Tsinghua University, Beijing, China, in 2007, and the Ph.D. degree from the University of Minnesota, in 2012.

In 2011, he was a Research Intern with the IBM T. J. Watson Research Center. From 2012 to 2013, he was a Postdoctoral Researcher with the University of Minnesota. Since 2013, he has been an Assistant Professor with the School of Information Science and Technology, ShanghaiTech University, Shanghai, China. In 2015, he was a Visiting Scholar with the University of California at Berkeley. His current research interests include computer-aided design of VLSI circuits, computer architecture, multicore processors, and 3D integration circuits.

Dr. Zhou received the best paper nominations at ASP-DAC 2010 and CSTIC 2016. He has been serving on the technical program committees of many international conferences, such as DAC, ICCAD, and ASP-DAC. He is an Associate Editor of the ACM SIGDA Newsletter.