Shen XW, Ye XC, Tan X *et al.* An efficient network-on-chip router for dataflow architecture. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 32(1): 11–25 Jan. 2017. DOI 10.1007/s11390-017-1703-5

An Efficient Network-on-Chip Router for Dataflow Architecture

Xiao-Wei Shen [1,2], *Student Member, CCF*, Xiao-Chun Ye [1,*], *Member, CCF*, Xu Tan [1,2], *Student Member, CCF*
Da Wang [1], *Member, CCF*, Lunkai Zhang [3], Wen-Ming Li [1], *Member, CCF*
Zhi-Min Zhang [1], *Senior Member, CCF*, Dong-Rui Fan [1], *Senior Member, CCF*, and
Ning-Hui Sun [1], *Fellow, CCF*

[1] *State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences Beijing 100190, China*

[2] *School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China*

[3] *Department of Computer Science, The University of Chicago, IL 60637, U.S.A.*

E-mail: {shenxiaowei, yexiaochun, tanxu01, wangda}@ict.ac.cn; lunkai@uchicago.edu
{liwenming, zzm, fandr}@ict.ac.cn; snh@ncic.ac.cn

Received September 2, 2016; revised December 13, 2016.

Abstract Dataflow architecture has shown its advantages in many high-performance computing cases. In dataflow computing, a large amount of data are frequently transferred among processing elements through the network-on-chip (NoC). Thus the router design has a significant impact on the performance of dataflow architecture. Common routers are designed for control-flow multi-core architecture and we find they are not suitable for dataflow architecture. In this work, we analyze and extract the features of data transfers in NoCs of dataflow architecture: multiple destinations, high injection rate, and performance sensitive to delay. Based on the three features, we propose a novel and efficient NoC router for dataflow architecture. The proposed router supports multi-destination; thus it can transfer data with multiple destinations in a single transfer. Moreover, the router adopts output buffer to maximize throughput and adopts non-flit packets to minimize transfer delay. Experimental results show that the proposed router can improve the performance of dataflow architecture by 3.6x over a state-of-the-art router.

Keywords multi-destination, router, network-on-chip, dataflow architecture, high-performance computing

1 Introduction

Domain-specific architecture is one of the most important trends in the development of computer architecture, such as DianNao[1] for deep learning, PuDianNao[2] for machine learning and SGMF[3] for parallel applications. Modern processors are based on control-flow architecture. As an alternative different from control-flow architecture, dataflow architecture has shown its advantages on specific applications[4-6]. In dataflow computing, data are transferred from instructions to instructions directly and instructions can be fired (executed) as soon as the operands are prepared[7]. Dataflow computing can efficiently exploit both ILP and DLP at the same time[8]. In dataflow processors, each processing element (PE) is much simpler than a general RISC core. Normally there are no branch predictor, out-of-order control and other complex logic in PEs of dataflow processors. The proportion of chip area occupied by function units in dataflow cores is much larger than that in general cones. Dataflow computing can achieve higher performance than general cores on specific parallel applications[8].

In dataflow processors, there are usually many PEs and each PE executes several instructions of a whole dataflow program. Data are exchanged among PEs

Regular Paper

Special Section on Dataflow Architecture

This work was supported by the National High Technology Research and Development 863 Program of China under Grant No. 2015AA01A301, the National Natural Science Foundation of China under Grant No. 61332009, the National HeGaoJi Project of China under Grant No. 2013ZX0102-8001-001-001, and the Beijing Municipal Science and Technology Commission under Grant Nos. Z15010101009 and Z151100003615006.

*Corresponding Author

through the network-on-chip (NoC)[9]. In dataflow instructions, the destinations (consumers) are specified in the fields of instructions and PEs send results of instructions to their consumers according to their destinations. In NoCs of dataflow architecture such as TRIPS[10] and WaveScalar[11], routers are used to route data in dataflow processors and the efficiency of routers is very important to the performance of dataflow architecture[9].

There are three main features of data transfers in dataflow architecture. Firstly, each instruction can have multiple consumers; thus the result should be transferred to multiple destinations[10]. Secondly, in control-flow many-core architecture, transfers in NoCs occur usually only when cache misses happen. In dataflow processors, each instruction produces a result and the result will be transferred through NoCs except that their destinations are all in the same PE[9]. Therefore the injection rate of NoCs in dataflow architecture is usually much higher than that in control-flow architecture. Thirdly, delays of data transfers have a significant impact on performance in dataflow architecture. In each PE, there are a limited number of instructions that can be executed concurrently. If every instruction is waiting for their operands because of large data transfer delays, function units in the PE will be idle and the performance will be decreased. However, common routers[12-22] do not take advantage of these features and cannot achieve high performance in dataflow architecture.

In this work, we propose an efficient NoC router for dataflow architecture. The proposed router supports multi-destination; thus it can transfer data with multiple destinations in a single transfer. Moreover, the router adopts output buffer to maximize throughput and adopts non-flit packets to minimize transfer delay. Our contributions are summarized as follows.

• We analyze and extract the main features of data transfers in NoCs of dataflow architecture: multiple destinations, high injection rate, and performance sensitive to delay.

• We propose an efficient router for dataflow architecture. The proposed router adopts multi-destination mechanism, output buffer mechanism and non-flit packet mechanism to improve the performance of dataflow architecture.

• We compare our proposal with commonly used routers and state-of-the-art routers in domain-specific areas. The proposed router can improve the performance of dataflow architecture by 3.6x over a state-of-the-art router.

• We evaluate different routers with a number of different destinations and observe that routers supporting two destinations can achieve higher performance per area than those supporting three or four destinations.

The remainder of the work is organized as follows. Section 2 analyzes the features of data transfers in dataflow architecture. Section 3 describes the proposed router for dataflow architecture. Section 4 describes the evaluation platform and presents the performance results. Section 5 reviews related work, and finally, Section 6 concludes the paper.

2 Data Transfers in Dataflow Architecture

Fig.1 shows the architecture of the dataflow accelerator used as the experimental platform in this work. For completeness, we add a RISC core and a DMA con-

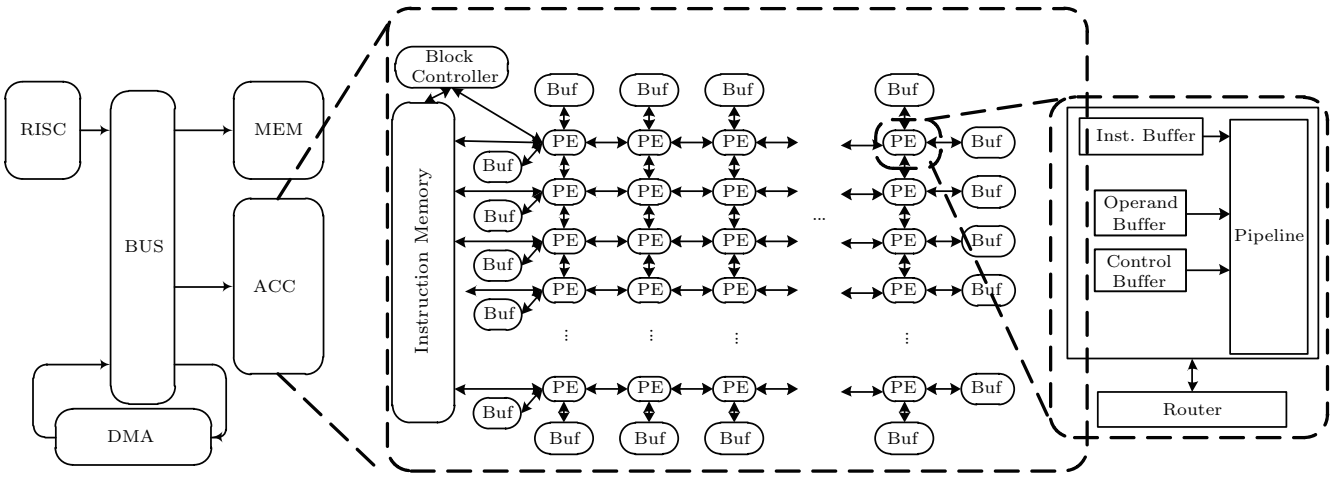


Fig.1. Architecture of the dataflow processor used as the experimental platform. Buf: buffer.

troller to form the whole simulation system. The RISC core executes the work-flow program that transfers data and instructions between off-chip memory and on-chip memory through configuring DMA. The dataflow accelerator is very similar to SGMF[3] and consists of $N \times N$ PEs which are connected by 2D mesh networks. Each PE contains several function units that execute dataflow instructions. There are an instruction buffer and an operand buffer in each PE. In the left of the array, an instruction memory is used to store the instructions of PEs. Data buffers surround the PE array to provide large data bandwidth for PEs. The latency of data transfers from off-chip memory to data buffers is covered by the computation time of the accelerator through double buffering technique.

2.1 Multiple Destinations

In dataflow processors, programs are represented by dataflow graph. A producer can have multiple consumers. Fig.2 shows an example of dataflow graph, the mapping result of the dataflow graph, and the corresponding dataflow program. When compiling, each instruction is mapped to a PE and the position of the instruction is also determined. The encoding of each instruction will be completed according to their consumers' positions. Both LD0 and LD1 have two consumers. After the execution of LD0, PE (0, 0) will send two packets to ADD0 [0, 1, 0] and SUB0 [0, 1, 1] respectively. The two packets have the same data and different destinations.

Fig.3 shows the proportion of packets of different types in total packets. The statistics are collected from three typical application kernels: fast fourier transform (FFT)[23], stencil[24] and general matrix-multiplication (GEMM)[25]. The adopted dataflow instruction set supports up to four destinations in instructions. Therefore packets in routers can be divided into four types according to the number of destinations of instructions that produce the packet. For example, the two packets produced by LD0 in Fig.2 belong to type 2 because LD0 has two destinations and the packet produced by MUL0 belongs to type 1 because MUL0 has only one destination.

From Fig.3, we can find that packets of type 1 occupy less than 54.3% in the total packets and reach 25.8% in FFT. On average, only 43% of total packets do not share any data with other packets. It is because more than half of dataflow instructions have more than one destination. Data produced by instructions will be transferred to multiple destinations in different packets. It will waste network bandwidth if packets with same data share the same path.

2.2 High Injection Rate

In the control-flow many-core architecture, transfers in NoCs occur only when cache misses happen. The injection rate of general cores is relatively low. However, in dataflow processors, each instruction produces a result and the result will be transferred through NoCs except that their destinations are all in the same PE with the producer. If a dataflow processor wants to maintain a relatively high performance, it has to execute instructions at every cycle. Then it will produce results and send packets to the connected router at almost every cycle. Therefore the injection rate of NoCs in dataflow architecture is usually much higher than that in control-flow architecture.

Considering the high injection rate of dataflow architecture, we directly adopt four mesh on-chip networks to evaluate the dataflow accelerator in the experiment. Table 1 shows the injection rate of three typical kernels on dataflow processor with four mesh networks. The injection rate is collected in an ideal situation where the transfer delay of routers is set to zero. Only when dataflow architecture achieves high performance can the injection rate be meaningful.

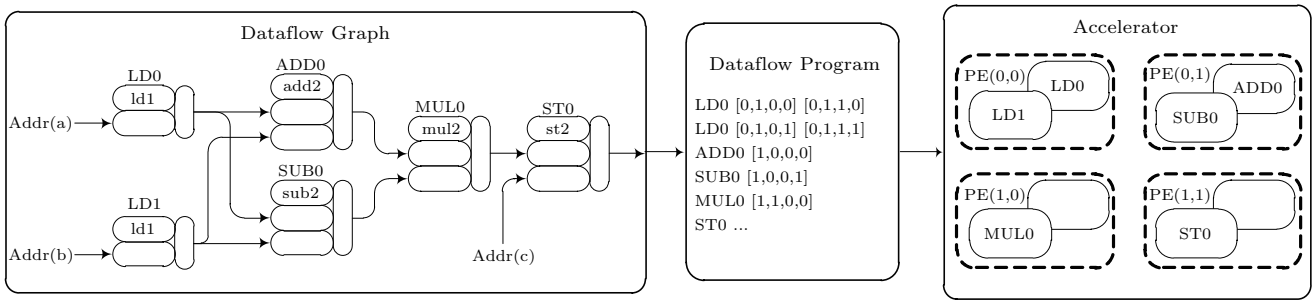


Fig.2. Dataflow graph and dataflow program.

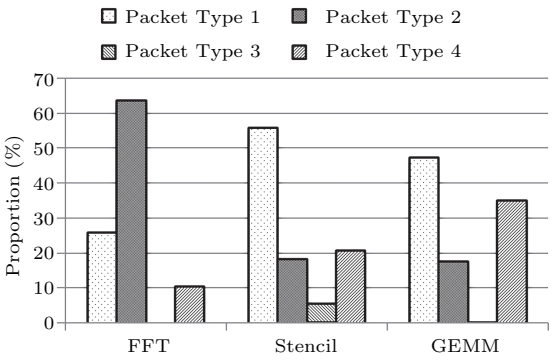


Fig.3. Proportion of packets of different types in total packets.

Table 1. Injection Rate of Routers in Dataflow Architecture

	Injection Rate (%)
FFT	47.5
Stencil	44.7
GEMM	62.3
Avg.	51.5

The NoC networks consist of four 8×8 mesh networks, thereby the total number of routers is 256. Even with so many routers, the average injection rate of routers still reaches 51.5%. However, in many-core architectures the injection rate is generally under 5%[26]. The injection rate of routers in dataflow architecture is much higher than that in control-flow many-core architecture.

2.3 Delay Sensitive

In control-flow many-core architecture, the cache miss rate is usually very low so that caches filter most of memory accesses. Therefore, even though the delay of data transfers in NoCs is large, the performance of control-flow cores will not be decreased significantly. However, in dataflow computing, instructions can be fired only after the operands are prepared. The number of instructions in each PE is limited by the area and logic complexity. Therefore there are normally dozens of instructions that can be executed concurrently in each PE. For example, each PE supports up to 64 instructions in TRIPS. If every instruction is waiting for their operands because of large data transfer delays, function units in the PE will be idle and the performance will be decreased.

Delays of data transfers have a significant impact on performance in dataflow architecture. Fig.4 shows the effect of router delays on performance. The delay means the cycles for a router to transfer a packet from the input port to the output port without congestion. When the delay of routers is increased from 1 to 4, the average performance of three typical kernels is decreased from 72.2 GFLOPS to 33.5 GFLOPS. Performance of dataflow architecture is sensitive to router delays; thus packet congestion should occur as less likely as possible in NoC routers.

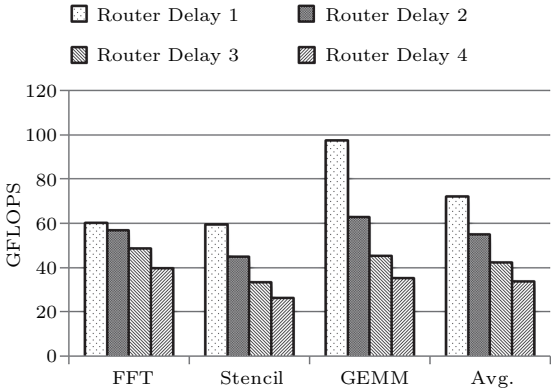


Fig.4. Effect of router delays on performance.

In conclusion, data transfers in NoCs of dataflow architecture have three features: multiple destinations, high injection rate, and performance sensitive to delay. Routers for dataflow architecture should take advantage of these features to improve the performance.

3 Proposed Router for Dataflow Architecture

Based on the three features of data transfers in dataflow architecture, an efficient router is proposed to improve the performance of dataflow architecture. The proposed router supports multiple destinations so that it can transfer data with multiple destinations in a single transfer. Moreover, the router adopts output buffer to maximize throughput and adopts non-flit packets to minimize transfer delay.

Fig.5 shows the architecture of the proposed NoC router. Packets from input ports are routed and split into output queues. For each input port, $P-1$ output queues are set for P output ports except for the input port direction. L, N, E, S and W in the queues represent for local, north, east, south and west direction. For each output port, the corresponding output arbiter selects a packet from corresponding output queues to the output port at every cycle.

3.1 Multiple Destinations

In the dataflow architecture, after the execution of an instruction, it will produce a result that will be directed to multiple destinations. In common routers, the

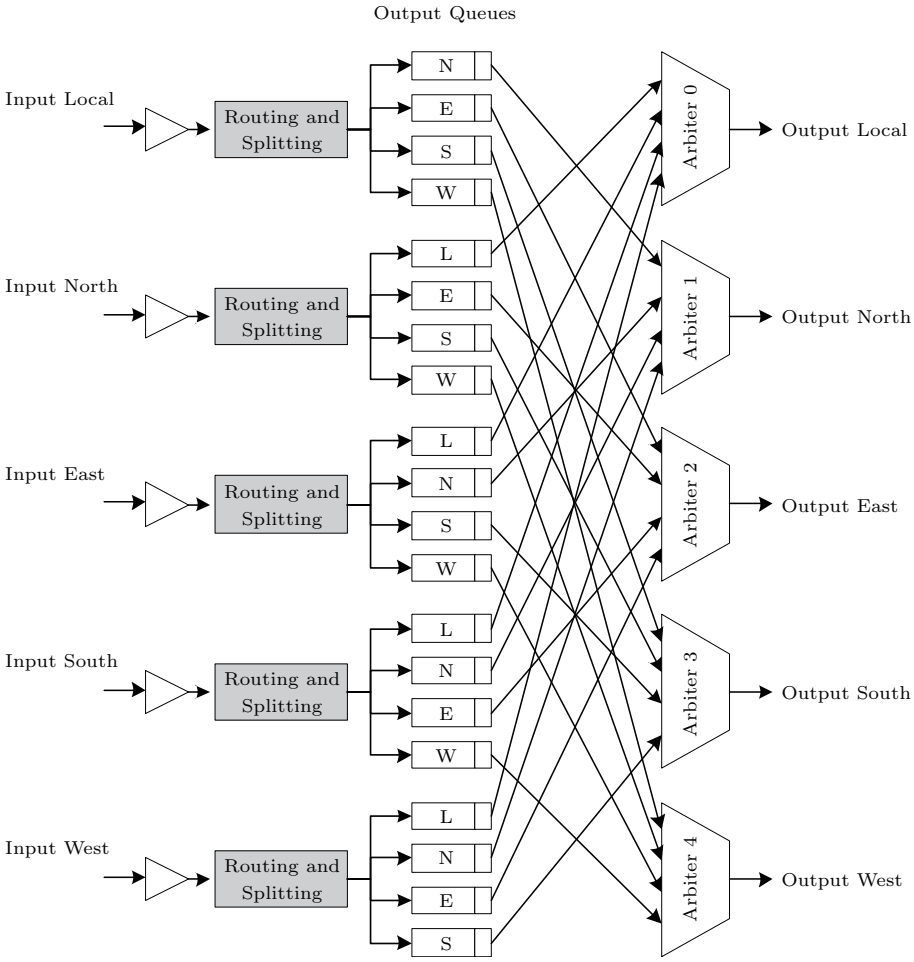


Fig.5. Architecture of the proposed NoC router.

result will be sent to different destinations in different packets. It can result in multiple transfers of the same data along the same path. Then the on-chip network bandwidth will be wasted. For example, instruction LD0 in Fig.2 will produce two packets that will be both sent to PE(0, 1) and the data of the two packets are the same.

The proposed router supports that each packet in the NoC has multiple destinations. Fig.6 shows the packet format of the proposed router. The packet consists of four destination addresses and data. Each destination address has a valid flag to show the validation of the address. It implies that the data will be sent to all of the valid addresses.

v	Dest3	v	Dest2	v	Dest1	v	Dest0	Data

Fig.6. Packet format of the proposed router with 4 destinations.

In the routers, a packet will be divided into multiple packets if the destinations (addresses in the packet) point to different directions (output ports). Addresses corresponding to the same output port will still be within a same packet. Fig.7 shows an example to explain the routing of a packet. Router (0, 1) receives a packet from the local PE and the packet has four destinations: (1, 1), (1, 2), (1, 3) and (2, 1). In router (0, 1), the four destinations will all point to the south direction; thus router (0, 1) will not split the packet and queue the packet to the south output queue. In the next cycle, router (0, 1) will send the packet with all of the four destinations to the south output port. Then router (1, 1) will receive the packet with all of the four destinations. In router (1, 1), the four destinations will be divided into three sets according to their output ports. Set 0 contains the address (1, 1) that points to the local output port. Set 1 contains two addresses: (1, 2) and (1, 3) that point to east output port. Set 2 contains the address (2, 1) that points to south output port. Therefore the packet will be split into three packets with the same data but with different

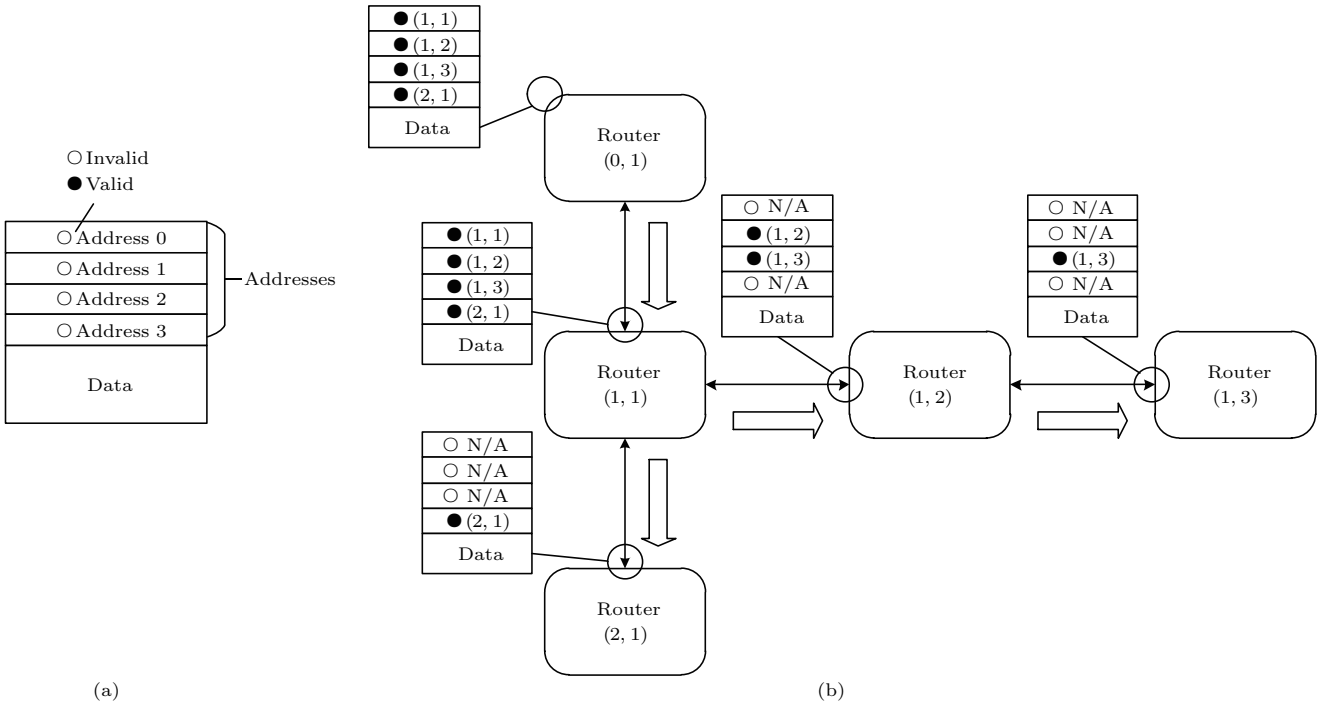


Fig.7. Example of routing in the proposed routers. (a) Organization of packets. (b) Routing example of multi-destination packets.

addresses that correspond to the three sets of addresses respectively. The packet with set 0 will be sent to the local PE and the two packets with set 1 and set 2 will be sent to router (1, 2) and router (2, 1) respectively. According to the splitting strategy, the data will be sent to their destinations efficiently.

The multi-destination router can improve the efficiency of data transfers among routers. For packets that contain multiple destinations, if more than one destination points to the same output port, the router can transfer the packet of multiple destinations in one cycle. Even though the destinations may point to different output ports in routers, the multi-destination packet at least can send data with multiple addresses from local PE to the connected router in one cycle. The multi-destination packet can both reduce the requirement of NoC bandwidth and reduce the transfer delay of packets. In routers, the routing of each destination can be processed in parallel.

The overhead of multiple destinations is just the extra address bits in router buffers and links between routers. The width of routing address is usually much smaller than the width of data. For example in an 8×8 mesh network, the width of address is 6 bits and that of data can usually be 32 bits or 64 bits. The overhead is very small especially when compared with the whole chip area.

3.2 Output Buffer

In dataflow architecture, the injection rate of NoC is relatively high as shown in Subsection 2.2. It implies that the requirement of NoC bandwidth is relatively high. If routers cannot achieve high throughput, the packets will be blocked in the routers and transfer delays will be much larger. Dataflow architecture is sensitive to transfer delay. If packets are blocked in routers, the performance will be reduced significantly. Therefore, the throughput of routers for dataflow architecture must be relatively high.

Routers usually can be divided by buffer design into two types: input buffer router and output buffer router. Fig.8 shows the architecture of routers with input buffer. In input buffer routers, there is an input queue for each input port and packets coming from each input port are queued in corresponding input queue. For each output port, an arbiter selects a packet from the head of all input queues to route to corresponding output port. In the input buffer router, the head packets of different input queues may ask for the same output port but each output port can only transfer one packet per cycle. Then only one of these packets can go through the router. It results in that some output ports will not be utilized and the throughput of the router will be decreased. Virtual channels can alleviate

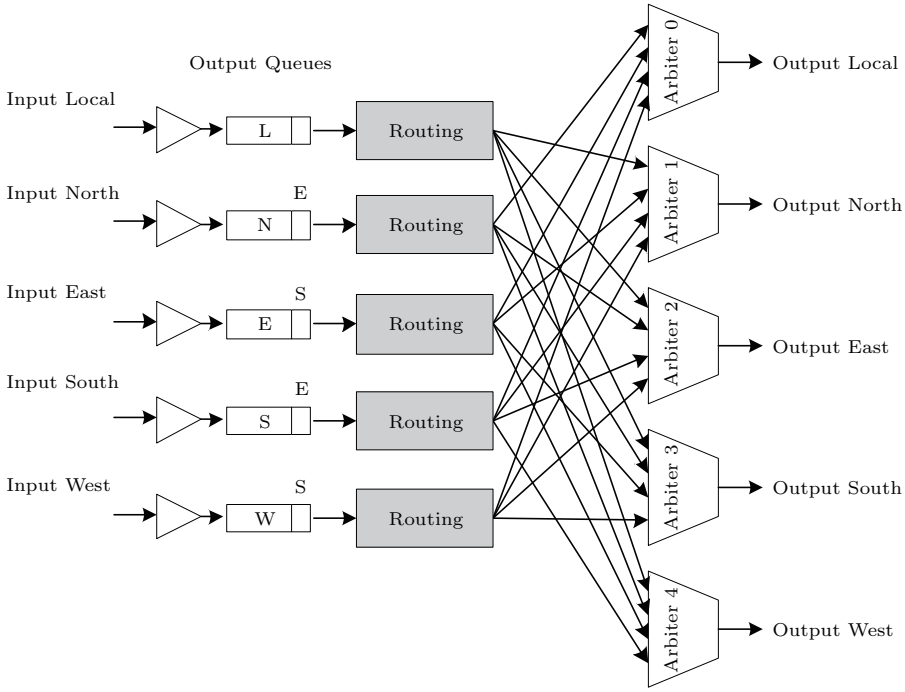


Fig.8. Architecture of routers with input buffer.

the situation with multiple input queues for each input port; however, it still cannot achieve the maximal throughput.

For example, in Fig.8, L, N, E, S and W in the queues represent the input directions of the input queues. E and S at the head of input queues represent the output ports of the head packets of the queues. The heads of east queue and west queue are both routed to the south output port and the heads of north queue and south queue are both routed to the east output port. Then only two of the four head packets can go through the router and the utilization of the router bandwidth is only 50%.

In output buffer router as Fig.5 shows, there are $P-1$ (P is the number of output ports of the router) output queues for each input port. Packets from each input port will be routed and queued into the corresponding output queues according to their output ports after routing. The arbiter of each output port then selects one packet from the $P-1$ corresponding output queues and sends it to the output port. In the output buffer router, packets are queued in the output queues and arbiters of each output port directly select packets from the corresponding output queues. Therefore, an output port will be utilized as long as there is one packet that is routed to the output port in the router. It implies that the output buffer router can achieve the maximal throughput. With multi-destination mechanism, the output buffer router routes all valid addresses of each packet and splits the packet to different output queues.

In dataflow architecture, the injection rate of routers is relatively high and the performance is sensitive to the transfer delay. Therefore, the throughput of routers is one of the most significant factors to the performance. In output buffer routers, buffers in different directions may not be utilized fully because different packets may be routed to the same output port. Through dynamic buffer allocation mechanism, buffers can be fully utilized and the overhead is just a few control logics. Though the output buffer router can result in either wasting of buffers or complex control logic, output buffer routers can achieve maximal throughput so that they are still adopted by the router for dataflow architecture. Compared with input buffer routers, output buffer routers can achieve high throughput and can improve the performance of dataflow architecture.

3.3 Non-Flit Packet

Routers commonly used in control-flow many-core architecture adopt the flit mechanism to reduce the buffer size and data bus size. In these architectures, NoC routers are used to transfer cache lines which are usually relatively large. To reduce the size of data

buffers in routers, a flit mechanism is usually adopted by routers. With the flit mechanism, big packets are usually divided into several flits and flits are transferred one by one between routers. The size of each flit can be very small, thereby the buffers in routers are very small. The flit mechanism can reduce the area and power of routers.

However in dataflow architecture, the injection rate of routers is relatively high and the performance is sensitive to the transfer delay. If the flit mechanism is adopted, each operand will have to consume several cycles to be transferred from one router to another. The flit mechanism can both increase the transfer delay and increase traffic congestion when the injection rate is relatively high. It can significantly decrease the performance of dataflow architecture. Moreover in dataflow architecture, routers are used to transfer operands between instructions. For a specified instruction set, the width of operands is specified to the same and is usually not very large. For example in TRIPS, the operands are 64 bits.

As Fig.9 shows, the flit and the non-flit mechanism are described in packet format. In the flit mechanism, 64 bits data are divided into four flits and each flit contains 6 bits data. Besides, the head flit and the tail flit have additional control information to indicate the head and the tail of the packet. However in the non-flit mechanism, the 64 bits data are transferred in a single packet and there are neither head nor tail packets. The packet is very simple and easy to encode and decode. The injection rate of routers with the non-flit mechanism will be much lower than that with the flit mechanism and the transfer delay will also be much smaller because traffic congestion will be much less.

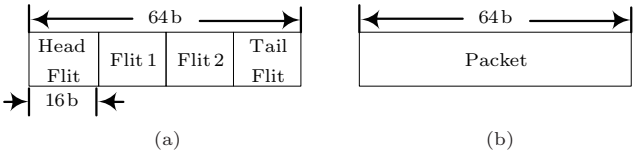


Fig.9. (a) Flit and (b) non-flit mechanism.

The non-flit mechanism is not always suitable for any situation. For example, when the data size of packets is relatively large, non-flit packets may suffer from long latency sending packets to wire. The long latency will decrease the performance and efficiency of NoCs. For dataflow architectures with relatively small packets, the non-flit mechanism is more suitable than the flit mechanism.

The chip area of non-flit packets is only increased a little and is still acceptable. However, the performance of non-flit packets can be much better than that of packets with flit. Therefore, in routers for dataflow architecture, non-flit packets can achieve higher performance per area than those with flit mechanism.

4 Evaluation and Results

We evaluate the proposed router architecture in a dataflow processor simulator implemented based on a simulator framework SimICT[27]. Fig.1 shows the architecture of the simulation system which consists of a dataflow accelerator and a RISC core. The specifications of the baseline system are listed in Table 2. The RISC core acts as the master core. It transfers the data and dataflow instructions to the accelerator and transfers results to the memory through configuring the DMA controller. As dataflow architecture demands a large amount of on-chip network bandwidth, NoCs in the experiment consist of four 8×8 mesh networks. Each PE is connected to four routers belonging to different mesh networks. To achieve high frequency, different networks are independent so that packets cannot cross different networks.

Table 2. Configurations of the Simulation System

Component	Configuration
RISC core	1 GHz, ARM
Memory	64 GB/s
PE	8×8, 1 GHz, 1 FMAC, 2 ALUs, 32 instructions, FRFO, 8 blocks in flight
SPM	2 MB, 32 banks, 64 b/bank
2D-Mesh	1 GHz, 1 cycle/hop, X-Y routing, 4 8×8 mesh networks, 64 bits packet data, 5 directions, 4 flits/packet, input buffer, 1 destination
Peak performance (DP)	128 GFLOPS

We adopt a load-balance based instruction mapping algorithm in the experimental platform. The algorithm focuses on the load balance of PEs and function units and is also augmented with network contention optimization. The adopted instruction mapping algorithm is the best algorithm in the adopted experimental platform and is also optimized with the proposed multi-destination mechanism to achieve higher performance.

4.1 Evaluation

We evaluate the proposed router with three typical parallel applications suitable for dataflow architecture

in double-precision floating point: FFT, stencil, and GEMM. The three testcases are examples to show the performance. They are very typical and popular in high performance computing especially in exa-scale scientific computing. Besides, most parallel applications can be accelerated by dataflow architecture and the proposed mechanisms can also improve the performance of parallel applications. The testcases are listed in Table 3. The block data size represents the size of data processed by the dataflow graph once in double-precision floating point and the tile data size is the data size of a tile in double-precision floating point. A tile consists of many blocks which are pipelined in the dataflow graph. The tile data size is limited by the size of buffers. It is worth noting that any data scale of a kernel can be processed by the processor. Data of any scale are firstly divided by the tile data size into many tiles. Data of each tile are divided by the block data size into many blocks which are processed by the PE array in the streaming model.

Table 3. Configurations of Testcases

Testcase	Block Data Size	Computation Data Size
1D FFT	1D 32p FFT	256 1D 32p FFTs
3D stencil	8×8×32 3d7p stencil	32×32×32 3d7p stencil
GEMM	8×64 and 64×8 GEMM	64×64 and 64×64 GEMM

The testcases are three examples to show the performance of the proposal. Actually the proposal can be applied in most parallel applications because parallel applications usually process large amount of data with a same program. The program is represented with a dataflow graph and data are processed in many tiles. Dataflow architecture is usually used to run parallel applications[3] and the proposal is applied in the architecture. Therefore, the proposal can improve most parallel applications in dataflow architecture.

We evaluate the proposed router with six different configurations to show the performance of each component of the router. Six different routers are tested in the experiment and are listed in Table 4. The routers differ from three aspects: with or without flit, input buffer or output buffer and with or without multi-destination. The F_IB_1 (4 flits/packet, input buffer and 1 destination) is a router that is commonly used in general many-core architecture[12,14-15] and is the state-of-the-art router. NF_IB_1 (non-flit, input buffer and 1 destination)[18-20] and NF_OB_1 (non-flit, output buffer and 1 destination)[21-22] are used in some application-specific architecture and are more suitable for dataflow architecture than the baseline. NF_OB_2, NF_OB_3 and NF_OB_4 are the proposed routers supporting different destinations for dataflow architecture.

Table 4. Configurations of Different Routers

Router	Flit	Buffer	Multiple Destination
F_IB_1	Yes, 4 flits	Input	No, 1 destination
NF_IB_1	No	Input	No, 1 destination
NF_OB_1	No	Output	No, 1 destination
NF_OB_2	No	Output	Yes, 2 destinations
NF_OB_3	No	Output	Yes, 3 destinations
NF_OB_4	No	Output	Yes, 4 destinations

We evaluate the delay of transferring a packet between two routers to show effects of the adopted mechanisms on packet transfers. Then we evaluate the performance of different routers to show why these three mechanisms are adopted by the proposed router. Multi-destination router is firstly proposed in this work and we also compare the total number of packets in routers with and without multi-destination to show the reduction of packets when the proposed multi-destination mechanism is applied. To decide how many destinations are supported in packets and how many entries in the buffers are better, we also evaluate the performance of proposed routers that support a number of different destinations and a number of different entries in buffers.

4.2 Results

The adopted mechanisms can reduce the possibility of congestion in routers and decrease the packet transfer delay in dataflow architecture. Fig.10 shows the average transfer delay in each router. The average transfer delay means how long packets stay in a router in cycles. From F_IB_1 to NF_OB_4, the average delay of different applications is decreased especially from F_IB_1 to NF_IB_1. Flit mechanism can increase router delay significantly. Without flit mechanism, the average delay is decreased from 4.3 cycles to 1.9 cycles, which is decreased by 126%. From input buffer to output buffer, the average delay is decreased by 21.1%. Output buffer can achieve high throughput and can improve the performance. From one destination to four destinations, the average delay is decreased by 13.3%. The adopted mechanisms in the proposed routers can all decrease the transfer delay of routers.

Transfer delay directly affects the performance of dataflow architecture. Fig.11 shows the performance of different routers. In F-IB-1 router with 4 flits, the performance of three kernels is very low and the average performance is only 15.3 GFLOPS. In NF_IB_1,

packets are transferred in a whole without flit. The performance is improved quickly and the average performance is improved to 59.69 GFLOPS. In dataflow architecture, the injection rate is very high and the performance is sensitive to transfer delay of routers. Routers with flit can increase the possibility of congestion significantly and decrease the performance of dataflow architecture even with four mesh networks. From NF_IB_1 to NF_OB_1, the performance is improved by 8.9%. In dataflow architecture, the injection rate is very high and the performance is sensitive to the transfer delay. The output buffer can achieve higher throughput than the input buffer with very little area overhead. Higher throughput means higher performance in dataflow architecture; thus the performance of NF_OB_1 is higher than that of NF_IB_1. With four destinations, NF_OB_4 can improve the performance of dataflow architecture from 64.98 GFLOPS to 70.09 GFLOPS. The multi-destination mechanism can improve the performance by 7.9%. In conclusion, the proposed router can improve the performance of dataflow architecture by 3.6x over a state-of-the-art router.

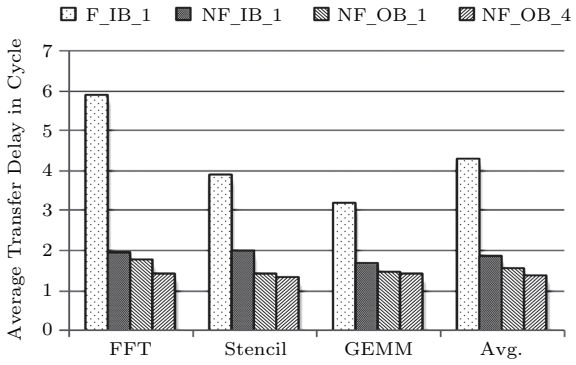


Fig.10. Average transfer delay of routers. The transfer delay means the time packets stay in a router in cycles.

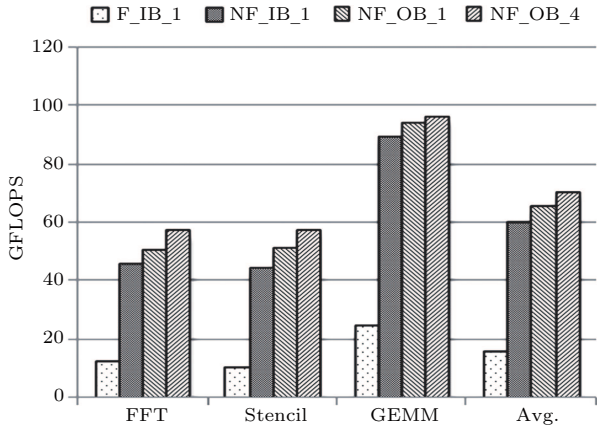


Fig.11. Performance of different routers on typical dataflow applications.

Multi-destination can reduce the packets in NoCs and improve the efficiency of data transfer of routers. Fig.12 shows the total number of packets transferred in NoCs with and without multi-destination. The packet number is normalized to that without multi-destination. From Fig.12, we can see that multi-destination mechanism can reduce the number of packets by 16% on average.

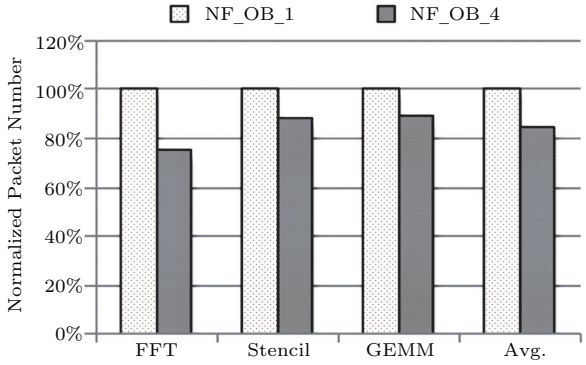


Fig.12. Reduction of packet counts through multi-destination mechanism.

To decide how many destinations supported in packets are optimal, we evaluate the performance of the proposed routers that support a number of different destinations. As Fig.13 shows, when the number of supported destinations is increased from 1 to 2, the average performance of dataflow architecture is increased from 65.0 GFLOPS to 70.2 GFLOPS. However when it is increased from 2 to 4, the average performance almost does not change. It implies that routers supporting two destinations can achieve better performance than those supporting one destination and achieve same performance as those supporting three or four destinations. Therefore routers supporting two destinations can achieve the highest performance per area in dataflow architecture.

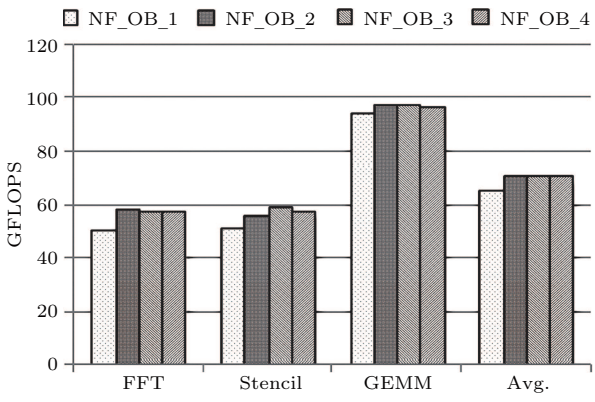


Fig.13. Performance of routers supporting a number of different destinations.

To decide how many entries in each buffer are optimal, we evaluate the performance of the proposed routers that have different numbers of buffer entries. As Fig.14 shows, when the number of entries in each buffer is increased from 2 to 4, the average performance of dataflow architecture is increased from 63.9 GFLOPS to 70.7 GFLOPS. However when it is increased from 4 to 6, the average performance is increased very little. It implies that buffers with four entries can achieve better performance than those with two entries. More than four entries in buffers cannot provide performance improvement but can result in much more chip area. Therefore four entries in each buffer of the proposed routers can achieve the highest performance per area in dataflow architecture.

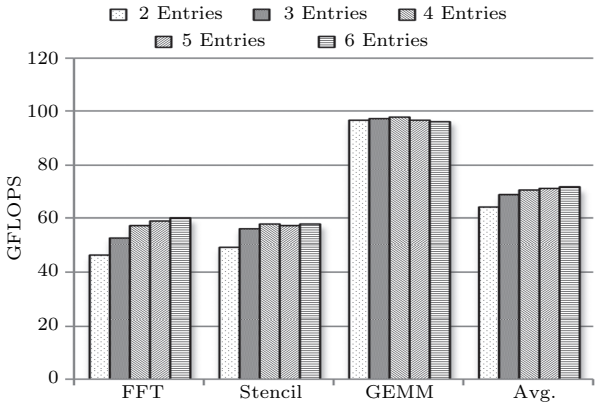


Fig.14. Performance of routers with a number of different entries in each buffer.

We also evaluate the chip area of dataflow processors with different routers. We have implemented the components of dataflow processors in RTL. We use Synopsys Design Compiler to synthesize the design with TSMC 40 nm technology library and estimate the chip area with the synthesizing results. Table 5 shows the area of different routers. It is worth noticing that routers can only affect the area of NoCs and the area of NoCs only occupies about 16.7% of the whole chip area according to the results. From F_IB_1 to NF_IB_1, the area of routers is increased by 217% because non-flit packets result in large buffer size of routers. However the whole chip area is increased only by 10.4%. The performance is improved by about 290%; thus the performance per area of the whole chip is improved by 263%. From NF_IB_1 to NF_OB_1, buffer design is changed from input buffer to output buffer. In the experiment, the number of buffers is the same in NF_IB_1 and NF_OB_1; thus the whole area is almost the same. Area of NF_OB_1 is a little larger than that of NF_IB_1 because output buffer contains a little complex logic to control the buffers. From NF_OB_1 to NF_OB_4, the difference is that they have a number of different destinations in routers. Each destination has 6 bits in the experiment and the data width is 64 bits. Therefore with one more destination, the area of router is increased by 7.4% and the whole chip area is increased only by 1.1%.

Table 5. Areas of Different Routers and Chips

	Router Area (mm^2)	Chip Area (mm^2)
F_IB_1	1.43	29.93
NF_IB_1	4.53	33.03
NF_OB_1	4.88	33.38
NF_OB_2	5.26	33.76
NF_OB_3	5.65	34.15
NF_OB_4	6.04	34.54

McPAT[28] is used to estimate the power consumption of the architecture. We counted the access frequencies of all modules such as ALUs, FPUs, data buffers, instruction queues and buffers in routers. The power consumption of each module of the proposed dataflow architecture is computed in McPAT with the statistical data. Fig.15 shows the normalized power consumption of FFT, 3D stencil and GEMM on dataflow architecture with different routers. From F_IB_1 to NF_IB_1, the average power consumption is increased by 17.5% because the buffers in NF_IB_1 are much larger than those in F_IB_1. Fortunately the performance is increased by 290%; thus the performance per watt is still increased by 231.9%. From NF_IB_1 to NF_OB_1, the power consumption is increased only by 2.0% because the execution time is decreased and the energy consumption is almost the same. From NF_OB_1 to NF_OB_4, the power consumption is decreased by 2.5% because the

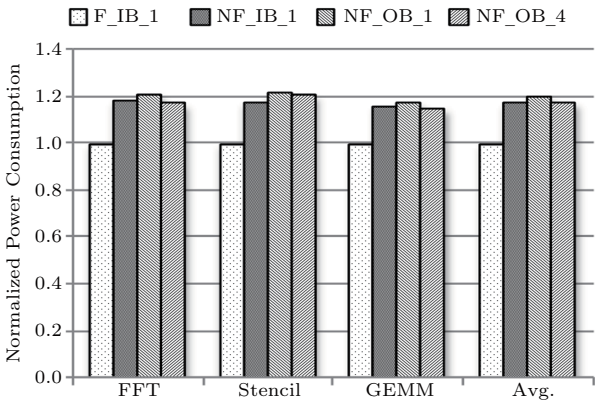


Fig.15. Normalized power consumption of different routers.

multi-destination mechanism can transfer multiple data in a single packet. After all, from F_IB_1 to NF_OB_4, the performance per watt is always increased though the power consumption may be increased. Therefore, the adopted mechanisms can help the proposed router achieve high performance per watt in dataflow architecture.

5 Related Work

Traditional dataflow architecture can be divided into two types: coarse-grained dataflow architectures and fine-grained dataflow architectures. Coarse-grained dataflow architectures, such as TeraFlux[29] and Runnemede[30], partition programs into many program blocks according to their dependency. Different blocks are assigned to different PEs and results of blocks are transferred to PEs that contain their consumer blocks according to their dependency. In coarse-grained dataflow architectures, the PEs generally are von Neumann (non-dataflow) execution model. Coarse-grained dataflow architectures are usually applied in many-core architectures through dataflow programming. Fine-grained dataflow architectures, such as TRIPS[10], WaveScalar[11] and SGMF[3], consist of many PEs which execute instructions in dataflow execution model. Data dependency is represented directly in instructions. The dataflow graph is mapped into the PE array and each PE executes only part of the dataflow graph and transfers their results directly to their consumers. Fine-grained dataflow architectures are usually applied in domain-specific high-performance computing architecture. The dataflow architecture adopted in this work is a fine-grained dataflow architecture. It is a domain-specific architecture designed for parallel applications and very similar to SGMF[3].

Due to the demand of efficient network-on-chip, there has been many router designs[12-22] for multi-core and many-core architecture. All of these designs adopt flit to reduce buffer size and to adapt dynamic packet size. Dynamic packet size and performance insensitive to delay are the features of the general von Neumann many-core architecture. However in the dataflow architecture, the packet size is fixed and the performance is sensitive to transfer delay. Non-flit packets should be more suitable for routers in the dataflow architecture.

Buffer design is important in routers and can be divided into two types: input buffer and output buffer. For the architecture that demands low chip area and power consumption, the input buffer is adopted such as in [12-20]. For the architecture that demands high throughput and low latency, the output buffer is adopted such as in [21-22]. For dataflow architecture, high throughput and low latency are the most important to the performance. Using output buffer in routers can help dataflow architecture achieve high performance. In the proposed router architecture, the output buffers are different from traditional output buffer routers from two aspects. Firstly, packets in the buffers of the proposed router have multiple destinations; thus the buffers are responsible for the encoding and decoding of multiple destinations. Secondly, the proposed router adopts non-flit mechanism; thus the buffers are not necessary to decode the head flit and the tail flit.

Multi-destination mechanism shares some similarities with multi-castschemes[31-33], which can broadcast data to multiple cores. However, our multi-destination scheme is significantly different from these previous proposals since it also adopts non-flit packets and output buffer mechanisms.

Recently, there are a number of proposals which adopt emerging non-volatile memory technologies to build low-power and low-cost NoCs[34-35]. Especially, it will be interesting to dynamically switch non-volatile memory (NVM) based router buffers from SLC (Single-Level Cell) to MLC (Multi-Level Cell) mode to temporarily accommodate more network flits, similar to the notion of proposals[36-37]. In this case, since the data in router buffers will be used within microseconds, the MLC mode NVM buffer can be written with fast but low retention writes[38]. Finally, proactively writing back cached data (i.e., Eager Writebacks[39-40]) may help reduce NOC burst operations and thus improve the effectiveness of NOC.

6 Conclusions

The domain-specific architecture is one of the most important trends in the development of computer architecture. Dataflow architecture has shown its advantages in high-performance computing. In dataflow architecture, a large amount of data are frequently transferred among processing elements through the network-on-chip. Thus the router design has a significant impact on the performance of the dataflow architecture.

Common routers are designed for control-flow multi-core architecture. They adopt flit to reduce buffer size in routers and to transfer dynamic packet size. To reduce the chip area and power consumption, input buffer performs better than output buffer. However features

of dataflow architecture are much different from those of control-flow architecture.

In this work, we proposed a novel and efficient NoC router for dataflow architecture. The proposed router supports multiple destinations; thus it can transfer data with multiple destinations in a single transfer. Moreover, the router adopts output buffer to maximize throughput and adopts non-flit packets to minimize transfer delay. Experimental results showed that the proposed router can improve the performance of dataflow architecture by 3.6x over a state-of-the-art router.

References

[1] Chen T S, Du Z D, Sun N H, Wang J, Wu C Y, Chen Y J, Temam O. DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *Proc. the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar. 2014, pp.269-284.

[2] Liu D F, Chen T S, Liu S L, Zhou J H, Zhou S Y, Temam O, Feng X B, Zhou X H, Chen Y J. PuDianNao: A polyvalent machine learning accelerator. In *Proc. the 20th International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar. 2014, pp.369-381.

[3] Voitsechov D, Etsion Y. Single-graph multiple flows: Energy efficient design alternative for GPGPUs. In *Proc. the 41st Int. Symp. Computer Architecture*, Jun. 2014, pp.205-216.

[4] Oriato D, Tilbury S, Marrocu M, Pusceddu G. Acceleration of a meteorological limited area model with dataflow engines. In *Proc. the Symp. Application Accelerators in High Performance Computing*, Jul. 2012, pp.129-132.

[5] Pratas F, Oriato D, Pell O, Mata R A, Sousa L. Accelerating the computation of induced dipoles for molecular mechanics with dataflow engines. In *Proc. the 21st Annual Int. Symp. Field-Programmable Custom Computing Machines*, Apr. 2013, pp.177-180.

[6] Fu H H, Gan L, Clapp R G, Ruan H B, Pell O, Mencer O, Flynn M, Huang X M, Yang G W. Scaling reverse time migration performance through reconfigurable dataflow engines. *IEEE Micro*, 2014, 34(1): 30-40.

[7] Theobald K B. EARTH: An efficient architecture for running threads [Ph.D. Thesis]. McGill University, Montreal, Que., Canada, 1999.

[8] Milutinovic V, Salom J, Trifunovic N, Giorgi R. Guide to Dataflow Supercomputing (1st edition). Springer International Publishing, 2015.

[9] Sankaralingam K, Nagarajan R, McDonald R, Desikan R, Drolia S, Govindan M S, Gratz P, Gulati D, Hanson H, Kim C, Liu H M, Ranganathan N, Sethumadhavan S, Sharif S, Shivakumar P, Keckler S W, Burger D. Distributed microarchitectural protocols in the TRIPS prototype processor. In *Proc. the 39th Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2006, pp.480-491.

[10] Burger D, Keckler S W, McKinley K S, Dahlin M, John L K, Lin C, Moore C R, Burrill J, McDonald R G, Yoder W. Scaling to the end of silicon with EDGE architectures. *Computer*, 2004, 37(7): 44-55.

[11] Swanson S, Schwerin A, Mercaldi M, Petersen A, Putnam A, Michelson K, Oskin M, Eggers S J. The WaveScalar architecture. *ACM Transactions on Computer Systems*, 2007, 25(2): Article No.4.

[12] Roca A, Flich J, Silla F, Duato J. A latency-efficient router architecture for CMP systems. In *Proc. the 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, Sept. 2010, pp.165-172.

[13] Michelogiannakis G, Dally W J. Router designs for elastic buffer on-chip networks. In *Proc. the Conference on High Performance Computing Networking, Storage and Analysis*, Nov. 2009.

[14] Chang Y Y, Huang Y S, Poremba M, Narayanan V K, Xie Y, King C T. TS-Router: On maximizing the quality-of-allocation in the on-chip network. In *Proc. the 19th Int. Symp. High Performance Computer Architecture*, Feb. 2013, pp.390-399.

[15] Tran A T, Baas B M. Achieving high-performance on-chip networks with shared-buffer routers. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2014, 22(6): 1391-1403.

[16] Poluri P, Louri A. An improved router design for reliable on-chip networks. In *Proc. the 28th Int. Parallel and Distributed Processing Symp.*, May 2014, pp.283-292.

[17] Ben-Itzhak Y, Cidon I, Kolodny A, Shabun M, Shmuel N. Heterogeneous NoC router architecture. *IEEE Transactions on Parallel and Distributed Systems*, 2015, 26(6): 2479-2492.

[18] Zoni D, Flich J, Fornaciari W. CUTBUF: Buffer management and router design for traffic mixing in VNET-based NoCs. *IEEE Transactions on Parallel and Distributed Systems*, 2016, 27(6): 1603-1616.

[19] Singh W, Deb S. Energy efficient and congestion-aware router design for future NoCs. In *Proc. the 29th Int. Conference on VLSI Design*, Jan. 2016, pp.81-85.

[20] Yan P Z, Jiang S X, Sridhar R. A high throughput router with a novel switch allocator for network on chip. In *Proc. the 28th International System-on-Chip Conference*, Sept. 2015, pp.160-163.

[21] Xu Y, Zhao B, Zhang Y T, Yang J. Simple virtual channel allocation for high throughput and high frequency on-chip routers. In *Proc. the 16th Int. Symp. High Performance Computer Architecture*, Jan. 2010, pp.1-11.

[22] Soteriou V, Ramanujam R S, Lin B, Peh L S. A high-throughput distributed shared-buffer NoC router. *IEEE Computer Architecture Letters*, 2009, 8(1): 21-24.

[23] Gu L, Li M, Siegel J. An empirically tuned 2D and 3D FFT library on CUDA GPU. In *Proc. the 24th ACM International Conference on Supercomputing*, Jun. 2010, pp.305-314.

[24] Zhang Y P, Mueller F. Autogeneration and autotuning of 3D stencil codes on homogeneous and heterogeneous GPU clusters. *IEEE Transactions on Parallel and Distributed Systems*, 2013, 24(3): 417-427.

[25] Kuzak J, Tomov S, Dongarra J. Autotuning GEMM kernels for the Fermi GPU. *IEEE Transactions on Parallel and Distributed Systems*, 2012, 23(11): 2045-2057.

[26] Hesse R, Nicholls J, Jerger N E. Fine-grained bandwidth adaptivity in networks-on-chip using bidirectional channels. In *Proc. the 6th IEEE/ACM Int. Symp. Networks-on-Chip*, May 2012, pp.132-141.

[27] Ye X C, Fan D R, Sun N H, Tang S B, Zhang M Z, Zhang H. SimICT: A fast and flexible framework for performance and power evaluation of large-scale architecture. In *Proc. the Int. Symp. Low Power Electronics and Design*, Sept. 2013, pp.273-278.

[28] Li S, Ahn J H, Strong R D, Brockman J B, Tullsen D M, Jouppi N P. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proc. the 42nd Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2009, pp.469-480.

[29] Solinas M, Badia R M, Bodin F, Cohen A, Evripidou P, Faraboschi P, Fenchner B, Gao G R, Garbade A, Girbal S, Goodman D, Khan B, Koliai S, Li F, Luján M, Morin L, Mendelson A, Navarro N, Pop A, Trancoso P, Ungerer T, Valero M, Weis S, Watson I, Zuckermann S, Giorgi R. The TERAFLUX project: Exploiting the dataflow paradigm in next generation teradevices. In *Proc. the Euromicro Conference on Digital System Design*, Sept. 2013, pp.272-279.

[30] Carter N P, Agrawal A, Borkar S, Cledat R, David H, Dunning D, Fryman J, Ganev I, Golliver R A, Knauerhase R, Lethin R, Meister B, Mishra A K, Pinfold W R, Teller J, Torrellas J, Vasilache N, Venkatesh G, Xu J P. Runnemede: An architecture for ubiquitous high-performance computing. In *Proc. the 19th Int. Symp. High Performance Computer Architecture*, Feb. 2013, pp.198-209.

[31] Wei L, Zhou L. An equilibrium partitioning method for multicast traffic in 3D NoC architecture. In *Proc. the IFIP/IEEE International Conference on Very Large Scale Integration*, Oct. 2015, pp.128-133.

[32] Agrawal M, Chakrabarty K. Test-time optimization in NOC-based manycore SOCs using multicast routing. In *Proc. the 32nd IEEE VLSI Test Symposium*, Apr. 2014.

[33] Kamali M, Petre L, Sere K, Daneshtalab M. Formal modeling of multicast communication in 3D NoCs. In *Proc. the 14th Euromicro Conference on Digital System Design*, Aug. 31-Sept. 2, 2011, pp.634-642.

[34] Zhan J, Ouyang J, Ge F, Zhao J S, Xie Y. Hybrid drowsy SRAM and STT-RAM buffer designs for dark-silicon-aware NoC. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2016, 24(10): 3041-3054.

[35] Zhan J, Ouyang J, Ge F, Zhao S, Xie Y. DimNoC: A dim silicon approach towards power-efficient on-chip network. In *Proc. the 52nd ACM/EDAC/IEEE Design Automation Conference*, Jun. 2015.

[36] Zhang L K, Strukov D, Saadeldeen H, Fan D R, Zhang M Z, Franklin D. SpongeDirectory: Flexible sparse directories utilizing multi-level memristors. In *Proc. the 23rd Int. Conf. Parallel Architectures and Compilation*, Aug. 2014, pp.61-74.

[37] Deng Z X, Zhang L K, Franklin D, Chong F T. Herniated hash tables: Exploiting multi-level phase change memory for in-place data expansion. In *Proc. the Int. Symp. Memory Systems*, Oct. 2015, pp.247-257.

[38] Zhang M Z, Zhang L K, Jiang L, Liu Z Y, Chong F T. Balancing performance and lifetime of MLC PCM by using a regionretention monitor. In *Proc. the 23rd Int. Symp. High Performance Computer Architecture*, Feb. 2017. (to be appeared)

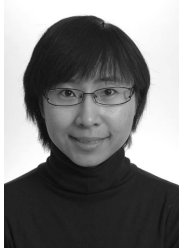
[39] LeeH H S, Tyson G S, Farrens M K. Eager writeback-a technique for improving bandwidth utilization. In *Proc. the 33rd Annual IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2000, pp.11-21.

[40] Zhang L K, Neely B, Franklin D, Strukov D, Xie Y, Chong F T. Mellow writes: Extending lifetime in resistive memories through selective slow write backs. In *Proc. the 43rd Int. Symp. Computer Architecture*, Jun. 2016, pp.519-531.

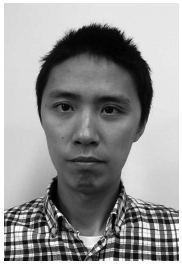
Xiao-Wei Shen received his Bachelor's degree in computer science and technology from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, in 2010. He is currently a Ph. D. candidate in University of Chinese Academy of Sciences, Beijing. His main research interests include processor micro-architecture and high-performance computer systems.

Xiao-Chun Ye received his Ph.D. degree in computer architecture from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2010. He is currently an associate professor in Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His main research interests include high-performance computer architecture and software simulation.

Xu Tan received his Bachelor's degree in computer science and technology from Capital Normal University, Beijing, in 2012. He is currently a Ph. D. candidate in University of Chinese Academy of Sciences, Beijing. His main research interests include high-performance computer architecture and software simulation.



Da Wang received her Ph.D. degree in computer architecture from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2009. She is currently an associate professor in Institute of Computing Technology, Chinese Academy of Sciences, Beijing. Her main research interests include processor micro-architecture and VLSI design.



Lunkai Zhang received his B.S. degree in computer science and technology from Wuhan University, Wuhan, in 2006, and his Ph.D. degree in computer architecture from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2014. He is currently a postdoctoral scholar in the Department of Computer Science, the University of Chicago, Illinois. His research interests include cache/memory architecture, architectural support for emerging device technologies and accelerator design.

Wen-Ming Li received his Ph.D. degree in computer architecture from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2016. He is currently an assistant professor in Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His main research interests include high-performance computer architecture and software simulation.

Zhi-Min Zhang received his Bachelor's degree in computer science and technology from Tsinghua University, Beijing, in 1987. He is currently a professor and Ph.D. supervisor in Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His main research interests include SoC design and computer architecture.

Dong-Rui Fan received his Ph.D. degree in computer architecture from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2005. He is currently a professor and Ph.D. supervisor in Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His main research interests include low-power design and processor micro-architecture.

Ning-Hui Sun received his Ph.D. degree in computer architecture from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 1999. He is currently a professor and Ph.D. supervisor in Institute of Computing Technology, Chinese Academy of Sciences, Beijing. His main research interests include high performance computer systems.