The Performance Evaluation of a 3D Torus Network Using Partial Link-Sharing Method in NoC Router Buffer

Naohisa FUKASE††, Nonmember, Yasuyuki MIURA†, Shigeyoshi WATANABE†, and M.M. HAFIZUR RAHMAN†††, Members

SUMMARY The high performance network-on-chip (NoC) router using minimal hardware resources to minimize the layout area is very essential for NoC design. In this paper, we have proposed a memory sharing method of a wormhole routed NoC architecture to alleviate the area overhead of a NoC router. In the proposed method, a memory is shared by multiple physical links by using a multi-port memory. In this paper, we have proposed a partial link-sharing method and evaluated the communication performance using the proposed method. It is revealed that the resulted communication performance by the proposed methods is higher than that of the conventional method, and the progress ratio of the 3D-torus network is higher than that of 2D-torus network. It is shown that the improvement of communication performance using partial link sharing method is achieved with slightly increase of hardware cost.

key words: interconnection network, network-on-chip (NoC), router, multi-port memory

1. Introduction

Network-on-Chip (NoC) connects hundreds of Intellectual Properties (IPs). These include programmable processors, co-processors, accelerators, application-specific IPs, peripherals, memories, reconfigurable logic, and analog blocks. In spite of various advantages of NoC, area overhead and power consumption are still a challenging problem of NoC. Minimal use of hardware resources in a router design can solve these two challenging drawbacks. Therefore, it is essential to design a high performance router using the minimum hardware resources to minimize the layout area and power consumption.

There are so many methods are already proposed and implemented such that a single memory can be shared by multiple virtual channels for the efficient utilization of router buffer [1]–[3]. However, the utilization is limited to a few virtual channels. We have proposed a buffer sharing method for sharing a buffer by multiple channels or links, and by using the proposed method many channels can be shared and the router can utilize buffers more efficiently. To share the plural links by single memory, multi-port memory can be used. However, the increase of the hardware cost is not reasonable when a usual multi-port memory is used.

The sharing method to share a ring buffer by multiple physical links and a sharing method to share multiple buffer by various links are presented in [4] and [5], respectively. However, it is really very difficult to share a large ring buffer due to use of large crossbar switch. The communication latency is high because wormhole routing is not using here and the number of pipeline stages is increases.

In our past studies, we have proposed the buffer sharing method by multiple physical links for efficient and effective use of router buffer [6]–[12]. It was shown that the conventional Inter-Link Buffer Sharing method increase the hardware cost because of large number of physical links. To mitigate this problem, we proposed Multi-bank Multi-port memory [13], [14] which reduced the hardware cost significantly [6], [9]. In [6], [9], we estimated the hardware cost of the proposed method, and also considered the way to reduce the hardware cost. In [10], we considered 2D torus network for the performance evaluation using the proposed method. It was shown that our proposed method using multi-bank memory resulted almost similar performance as compared to normal multiport memory when the number of banks is more than eight [10]. In [11], we have evaluated the performance using partial sharing, and this reduced the hardware cost. It is shown that the performance of partial sharing is significantly improved with slightly increase of hardware cost [11]. In our past studies, we have estimated proposed method in NoC using 2D torus, mesh and a ring network. In [12], proposed method in a 3D torus network is estimated easily. However, it was the evaluation in the limited condition. In 3D torus, Partial Sharing is not estimating. The main objective of this study is to investigate the performance improvement with respect to communication performance and hardware cost of 3D-torus network using the partial sharing method.

The remainder of this paper is organized as follows. In Sect. 2, we briefly describe the conventional sharing method. Our proposed method is discussed in details in Sect. 3. The hardware cost and communication performance of proposed method is analyzed in Sect. 4. Finally, we conclude this paper in Sect. 5.
is used to connect input/output links in a router. Usually, a physical link is split into multiple virtual channels [15]. And a buffer is used in each virtual channel, especially in the input side of the crossbar switch for smooth flow of flits or packets in data communication. Wormhole routing is a cost-effective packet switching technique, because it can be implemented by using comparatively a small buffer. Wormhole routers have buffers of same capacity in each channel [16]. However, the buffer allocated to the channel is not utilized effectively because some channels and their associated buffers remain idle. To overcome this problem, a memory is sharing by multiple virtual channels of a physical link. This method is proposed and implemented in [1]–[3].

In this sharing method, a block of the shared memory is assigned dynamically and used when the capacity of a virtual channel buffer becomes insufficient. VC Block Info and Free-Pool information is necessary to implement this Inter-Channel Buffer Sharing (Channel Sharing) method. The structure of the Block-Info and Free-Pool is shown in Fig. 1. The ‘Block-Info’ is assigned per channel and the ‘Free-Pool’ is assigned per Link. The connection between memory blocks is expressed by recording the arrangement of memory to ‘VC Block Info’ of the assigning channel from ‘Free-Pool’. When the assigned channel releases a memory block, the information of the released memory block is recorded to ‘Free-Pool’ from ‘VC Block Info’.

3. Inter-Link Buffer Sharing (Link Sharing) Method

3.1 Outline

The sharing of a memory among physical links is not used yet because of hardware cost. We need to use a multi-port memory as a shared memory to support the concurrent access to shared memory from multiple physical links. However, the hardware cost becomes infeasible when conventional multiport memory is used. Here the essential hardware cost is proportional to the square of the number of ports.

The router structure used in proposed Inter-Link Buffer Sharing (Link Sharing) method and the multi-bank multiport memory used in the proposed method are depicted in Fig. 2 and Fig. 3, respectively. As portrayed in Fig. 2, each channel has a ‘Private Buffer’ and a shared memory is laid out between input ports in the router. The ‘Multi-bank Multi-port Memory’ used in our method is to reduce the hardware cost. As illustrated in Fig. 3, the multi-bank multi-port memory has some memory banks which have a few ports. And the shared memory bank is put between two crossbar switches. This multi-port memory reduces the hardware cost because it is not necessary multiple ports for each memory cell. However, the multi-bank multi-port memory cannot access to the addresses in the same bank at the same time.

To solve this problem, we have proposed the ‘block-based control’. Here, a shared memory is divided into multiple blocks and it is controlled according to a block. By associating each block and bank, the physical link which access to each bank is limited to one. This method can reduce hardware cost because the proposed method is block-based control. Therefore, in this paper, the method of controlling a memory by every flit is called ‘Flit-Based Control’. And the method of controlling by every block is called ‘Block-Based Control’.

When the shared memory is fully used, then there is a possibility that no memory block is allocated to a virtual channel or a physical link. In consequence of flit flow in the network, a deadlock situation may occur [17]. To solve this problem, a buffer called the ‘Private Buffer’ with minimum capacity for communication is laid out to each channel. Even if a shared memory is not allocated, each channel can communicate and eventually can avoid the deadlock.

3.2 Hardware Structure

The structure of proposed method is shown in Fig. 4. As shown in Fig. 4, the router of our method has one Shared Memory for all physical links, and it is laid out between Input Port and Private Buffer. A block diagram including the pipeline structure of the Inter-Link Buffer Sharing is portrayed in Fig. 5. As depicted in Fig. 5, the Inter-Link Buffer Sharing has 5 pipeline stages and each stage is shown by
3.3 Pipeline Structure

3.3.1 Pipeline Structure of a Traditional Router

The pipeline structure of a traditional router is shown in Fig. 6. In this paper, traditional router is a router where the memory is not shared. As shown in Fig. 6, a traditional pipeline performs the following four processes in three steps.

1. Routing Computation (RC): An output link is determined from the information provided by the header.
2. Virtual Channel Allocation (VA): The virtual channel to output port is assigned.
3. Switch Allocation (SA): The arbitration and setup of a crossbar switch are performed.
4. Switch Traversal (ST): Flit passes through the crossbar switch.

3.3.2 Pipeline Structure of Proposed Method

In the proposed method, injected flits pass through either of the following two paths. In case of no congestion, the flit passes through the path 1. When there is congestion, the flit passes through the path 2.

- path1: input port → private buffer → output port
- path2: input port → shared memory → private buffer → output port

The path 1 is one of the paths of a flit, whereby the flit is immediately sent to private buffer after receiving it in the input port. The pipeline structure of the path 1 is shown in Fig. 7. As depicted in Fig. 7, the path 1 operates
on the same three-stages pipeline as traditional router. However, In-Judge (IJ) process is executed in stage 1 as shown in Fig. 7. In IJ process, ‘whether the shared memory is used or not in used’ and ‘whether a new block is allocated or not’ are determined. Since the output link of a packet is decided regardless of whether a shared memory is used or not, RC and IJ processes can be processed in parallel.

The path 2 is another way of a flit, whereby the flit is sent into the shared memory and then goes to the private buffer after that. The path 2 needs the stage for a setup and traversal of the switch in the input and output port of the multi-bank multi-port memory. The pipeline of the path 2 is shown in Fig. 8. The path 2 needs the following stages:

1. IJ (In-Judge): ‘Whether the shared memory is used or not’ and ‘whether a new block is allocated or not’ are determined in IJ process.
2. SiA (Switch-i Allocation): Set the input crossbar switch of the multi-bank multi-port memory.
3. SiT (Switch-i Traversal): When it succeeds in a SiA step, a packet passes the input crossbar switch of a multi-bank multi-port memory and it stores to the shared memory.
4. SoA (Switch-o Allocation): A setup of the output crossbar switch of the multi-bank multi-port memory and the block release process are carried out simultaneously.
5. SoT (Switch-o Traversal): When it succeeds in a SoA step, a packet passes the output crossbar switch of multi-bank multi-port memory and it stores to the private buffer.

The pipeline structure of proposed method has the similar structure as the router of [5]. These routers include the access to the sharing memory and next buffer in SiT and SoT respectively. As shown in Fig. 8, the number of stages of path 2 is 2 stages larger than path 1. But the delay by the pipeline of the path 2 is concealed by the following reasons.

- When a network is not congested and the private buffer is not full, it becomes same stages as the traditional since it is processed according to a three-stage pipeline (path 1 as shown in Fig. 7).
- As a network is congested, the private buffer becomes full and the shared memory will be used. The number of flits in the private buffer increases by blocking the packet by the crossbar switch. If the private buffer is designed to permit one blocking (If the number of flits of the private buffer is two or larger), the pipeline using the shared memory will smoothly flow.

An example of a pipeline of the Inter-Link Buffer Sharing is shown in Fig. 9. In Fig. 9, the third flit goes to the path 2 since the header flit is blocked. In this figure, header flit will fail to get a channel of the next router for both the second and third cycles. Therefore, 5 cycles are needed for header and payload 1 who uses path 1 to pass through the router. As a result, delay is concealed because the necessary cycle for Payload-2 is same as Header and Payload-1.

3.3.3 The Detailed Structure of Proposed Method

The structure of proposed router is described in [9] in detail. The structure is shown in Fig. 4. As shown in Fig. 4, the router of our method has one Shared Memory for all physical links, and it is laid out between Input Port and Private Buffer. The structure of IJ Circuit, Flit-Input Circuit, Flit-Output Circuit, Controller, and Shared Memory in Fig. 4 are shown in Figs. 10 - 14 respectively. The behavior of each unit are as follows [9]:

- **IJ Circuit**
  Based on the information about “Private buffer is full” (PB-full) or “a part of shared memory is allocated for the channel” (BI-empty), the signal which indicates that the injected flit uses the shared memory (IJ-sig) is output.

- **Flit-Input Circuit**
  Based on the state of each block in shared memory (Block-full) and the state of tail buffer (which memory block is used for the tail of the channel) in each channel (Tail), the signal which indicates that a block is acquired when a flit is injected (In-req) and the acquired block number of shared memory (In-block) is output.

- **Flit-Output Circuit**
  Flit-Output Circuit outputs the indication signal that the acquired block by channel becomes empty in the next cycle. It checks whether the following conditions are held, and the signal that releases a memory block in the channel is output when all the conditions are held.
1) The number of flits in the block is one (Obtained from Block-edp and Block-stp).
2) A flit is output from private buffer (Obtained from PB-full).
3) A flit is not injected from input port (IJ-sig is used).

- **Controller**
  - Based on the signals from Flit-Input Circuit and Flit-Output Circuit, memory blocks in shared memory are controlled. Mainly, controller consists of one “Free Pool” in the controller circuit and one “Block Info” in each channel. The block number information assigned in each channel is stored in Block Info. And the block information which is not assigned to any channel is stored in Free Pool. These informations are provided to each unit in the router.
  - **Shared Memory**
    - Plural blocks are laid out between two switches.

In the traditional router, the critical path is the arbitration including priority control of the crossbar switch [1], [5], [19]. In the proposed method, IJ, SiA, SoA, SiT and SoT stage are added to pipeline of traditional router. SiT and SoT are the structure equal to a ST stage of the traditional router mostly. Therefore, these stages may be a critical path. To pass these circuits, a signal passes 2 multiplexers, or logical gate of 3 gates and 1 multiplexer. From the above reason, it is thought that the clock frequency does not fall by proposed method.
3.4 Deadlock-Free

When the shared memory is fully used, there is a possibility that no memory block is allocated to a virtual channel or a physical link. In consequence of flit flow in the network, a deadlock situation may occur [17]. To solve this problem, a buffer called the ‘Private Buffer’ with minimum capacity for communication is laid out to each channel, and injected flit selects the following path:

- When Private Buffer is Full, or previous flit of same packet exists in Shared Memory, injected flit selects Shared Memory.
- When it is not the above, injected flit selects Private Buffer.

Even if a shared memory is not allocated, each channel can communicate and eventually can avoid the deadlock.

Figure 15 shows an example of a deadlock situation, where the sharing of a single buffer between multiple links can result in a deadlock where both links block the path if the buffer is filled by a single packet. Deadlocks are thus prevented by using a path that directly connects the private buffers. As shown in Fig. 2, the proposed method is constructed with shared memory inserted in front of the private buffers, so if the private buffers are regarded as the main channel buffers without considering the shared memory, then it is equivalent to the structure of a conventional router. Accordingly, if the interconnected network formed by the private buffers is deadlock-free, then the interconnected network formed by the router that includes shared memory according to the method described herein will also be deadlock-free. According to the idea of an extended channel dependency graph [20], a relationship whereby a packet is sent directly from the private buffers to the private buffers of adjacent routers via shared memory can be regarded as an “indirect dependency”. These relationships are shown in Fig. 16. In this method of this paper, since the shared memory is only located between the private buffers, both direct and indirect dependencies are established between channels with the same combinations (corresponding to the private buffers in this case). We created a directed graph with these dependencies as edges and channels as nodes, and we proved that the graph is deadlock-free as long as it does not have any closed circuits [20]. For example, Fig. 17 shows the channel dependency graph when performing X–Y routing in a 3 × 3 mesh network. In Fig. 17, the large gray rectangles are routers, the small rectangles are the private buffers, and arrows represent links. Also, the black rectangles are nodes (channels), and black arrows represent dependencies. As the figure shows, there are both direct and indirect dependencies between the same nodes. The graph of Fig. 17 has no closed circuits, showing that it is deadlock-free.

3.5 Partial Sharing Method

In our previously proposed partial sharing method also known as all-link Sharing method, a shared memory is shared by all physical links as portrayed in Fig. 2. However, the main problem of the all-link sharing method is the complexity of circuit around a memory cell. Therefore, we have proposed a method where two memories are shared by four links, and it is called partial sharing method. The par-
tial sharing is to share a memory by two physical links. The structure of partial sharing is shown in Fig. 18. As illustrated in Fig. 18, two shared memories are shared by link E and W, and link N and S (or link E and N, and link S and W) respectively.

3.6 Hardware Cost

We estimated the hardware cost in our previous study [9]. In the Inter-Channel Buffer Sharing, most of the hardware cost is ‘a) buffer of the physical link’ except crossbar switch and control circuit. ‘b) Memory element for control information’ is needed for both the Inter-Channel Buffer Sharing and the Inter-Link Buffer Sharing. Memory element includes the buffers for control the shared memory [6]–[12]. Additional hardware costs for the proposed method are ‘c) logic circuit for block control’ and ‘d) surrounded circuits of multiport memory’.

The hardware cost of a physical link can be roughly estimated by estimating the above mentioned elements. In this evaluation, B, C, L, F, and W are defined as follows:

- B: Total number of memory blocks in all links
- C: Total number of channels in all links,
- L: Number of links
- F: Number of flits in a block
- W: The number of bits per a flit

a) Buffer of the physical link

The following formula shows the number of bits \(M_{bu\text{f}}\) in the buffer which a packet is stored.

\[
M_{bu\text{f}} = BFW
\]  

b) Memory element for control information

The control memory elements of the conventional method consist of control queues called Free Pool and Block Info, and a third queue used as a memory block for storing flits. The Free Pool queue needs edp, stp and empty bits that are required for control. In the main storage unit, the \(B/L\) blocks express the number of blocks per link and has \(B/L\) item entries, so the number of memory elements needed for the Free Pool queue is as follows. From left to right, the terms on the right-hand side represent the number of memory elements needed for the main memory, the edp and stp queues, and the empty bits.

\[
M_{tr,FP} = \frac{B}{L} \log B + 2 \log \frac{B}{L} + 1
\]  

(2)

Block Info can be calculated in the same way as Free Pool. Since the number of channels required per link is \(C/L\), the number of memory elements is as follows.

\[
M_{tr,Bl} = \left(\frac{B}{L} \log B + 2 \log \frac{B}{L} + 1\right) \frac{C}{L}
\]  

(3)

The memory block requires pointers edp and stp to indicate the input/output location, and a full bit that indicates whether or not the block is vacant. Since edp and stp represent the number of flits per block, which is the number of entries in the block, they occupy \(\log B\) bits, while full requires one bit. Since each link requires \(B/L\) blocks, the total number required is as follows:

\[
M_{tr,MB} = \frac{B}{L}(2 \log F + 1)
\]  

(4)

From the above, the capacity of the control buffer in the conventional method can be summarized as follows:

\[
M_{tr} = M_{tr,FP} + M_{tr,Bl} + M_{tr,MB}
\]

\[
= \left(\frac{B}{L} + 2\right)(L + C) \log \frac{B}{L} + 2B \log F
\]

\[
+ B' + C + L
\]

(5)

However, since a pointer to the main memory block is not needed under the condition of the conventional method where \(F = 1\), the value of \(B'\) is expressed as follows:

\[
B' = \begin{cases} 
0 & (F = 1) \\
B & (F \neq 1) 
\end{cases}
\]

(6)

Since \(F = 1\) in the conventional method, \(B' = 0\).

As in the conventional method, the memory elements for control in the proposed method consist of three types of queue: a Free Pool queue, a Block Info queue, and a memory block. The implementation of the Free Pool queue is similar to the conventional method, but requires just one queue with \(B\) blocks for all the physical links, so the number of memory elements needed for the Free Pool is as follows:

\[
M_{pr,FP} = B \log B + 2 \log B + 1
\]  

(7)

The same applies to Block Info, and since this is the total required number of channels for all physical links, the number of memory elements needed for Block Info is as follows:

\[
M_{pr,Bl} = (B \log B + 2 \log B + 1)C
\]  

(8)

Similarly, the number of memory elements needed for the memory block is given by

\[
M_{pr,MB} = B(2 \log F + 1)
\]  

(9)

By summing together these results, the overall required
number of memory elements is:

\[ M_{pr} = M_{pr,FP} + M_{pr,BL} + M_{pr,MB} \]
\[ = (B + 2)(C + 1)\log B \]
\[ + 2B\log F + B + C + 1 \]  

(10)

c) logic circuit for block control

The logic circuits for block control consist of the following elements:

1) Flit-Input Circuit
2) Flit-Output Circuit
3) IJ Circuit
4) Crossbar switch for Free Pool control

In the above evaluation, we assume the following:

- A 2-input multiplexer consists of three 2-input NAND gates and an inverter. Here, for simplicity, we configured the multi-input multiplexer from multiple 2-input multiplexers.
- An AND (OR) gate consists of a NAND (NOR) gate and an inverter.
- An XOR (XNOR) gate consists of three 2-input NAND gates and two inverters, and a b-bit XOR (XNOR) gate consists of b 2-input XOR (XNOR) gates and a b-input AND gate.

The number of transistors in each circuit is as follows.

1) The formulae used to calculate the number of transistors and circuits necessary for the Flit-Input Circuit are as follows:

- B-input 1-bit multiplexer
- C/L-input log B-bit multiplexer
- 2-input log B-bit multiplexer

\[ T_{FI} = 14(B - 1) + 14\left(\frac{C}{L} - 1\right)\log B + 14\log B \]
\[ = 14\left(B - 1 + \frac{C}{L}\log B\right) \]  

(11)

In Eq. (12) before transformation, the first term represents the B-input 1-bit multiplexer, the second represents the C/L-input log B-bit multiplexer, and the third represents the 2-input log B-bit multiplexer.

2) The circuits required for the Flit-Output circuit are as follows:

- A circuit that checks whether or not there is one remaining flit in a block
  - Two B-input log F-bit multiplexers
  - Three 2-input log F-bit XNOR gates
  - \( \log F - 1 \) 2-input 1-bit XOR gates
  - \( \log F - 1, \log F - 2, \ldots, 0 \)-input AND gates: one of each
  - One inverter
  - 2-input AND gate
  - 2-input OR gate
- One C/L-input log B-bit multiplexer
- One 2-input log B-bit XNOR gate
- One 2-input NAND gate
- One C/L-input 1-bit multiplexer
- One 3-input AND gate

Accordingly, the formula for calculating the required number of transistors is as follows:

\[ T_{FO} = 28(B - 1)\log F + 54\log F + 6 \]
\[ + 16\log F - 16 + \sum_{i=2}^{\log F} (2i + 2) + 12 \]
\[ + 14\left(\frac{C}{L} - 1\right)\log B + 18\log B + 2 + 4 \]
\[ + 14\left(\frac{C}{L} - 1\right) + 8 \]
\[ = 14(2B + 3)\log F + 2(\log B + 1)\left(\frac{7C}{L} + 2\right) \]
\[ + \sum_{i=2}^{\log F - 1} (2i + 2) \]  

(12)

In Eq. (13) before transformation, the first term represents the two B-input log F-bit multiplexers, the second and third represent the three 2-input log F-bit XNOR gates, the fourth and fifth represent the \( F - 1 \) 2-input 1-bit XOR gates, the sixth represents each of the \( F - 1, \log F - 2, \ldots, 0 \)-input AND gates, the seventh represents an inverter, the eighth represents a 2-input AND gate and 2-input OR gate, the ninth represents a C/L-input log B-bit XNOR gate, the twelfth represents a 2-input NAND gate, the thirteenth represents a C/L-input 1-bit multiplexer, and the fourteenth represents a 3-input AND gate. However, when \( F \leq 4 \), the sixth term is not used, and is omitted from the calculation. (However, when the state of \( F \) is \( F \leq 4 \), the last item of (6) isn’t used.)

3) The circuit required for the IJ Circuit and the number of transistors are calculated using the following formulae:

- Two C/L-input 1-bit multiplexers
- Inverter
- 2-input OR gate

\[ T_{IJ} = 14\left(\frac{C}{L} - 1\right) \times 2 + 2 + 6 \]
\[ = 4\left(\frac{7C}{L} - 5\right) \]  

(13)

In Eq. (14) before transformation, the first term represents the two C/L-input 1-bit multiplexers, the second represents the inverter, and the third represents the 2-input OR gate.

4) The circuit required for the Free Pool control crossbar switch and the number of transistors are calculated using the following formulae:

- log B-bit switch with \( L \times B \) cross points
- \( L \) decoders
  - \( B \log B \)-input AND gates
  - \( \log B \times 2^{\log B/2} \) inverters
Table 1 Implementation cost of inter-link buffer sharing (transistors).

<table>
<thead>
<tr>
<th>W</th>
<th>Topology</th>
<th>L</th>
<th>C</th>
<th>B</th>
<th>F</th>
<th>Traditional (1) Channel Sharing</th>
<th>Link Sharing</th>
<th>Link Sharing by Block Based Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>Ring</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>24576</td>
<td>30732</td>
<td>154318</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>58146</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>16</td>
<td></td>
<td></td>
<td>41710</td>
</tr>
<tr>
<td></td>
<td>2D torus</td>
<td>4</td>
<td>8</td>
<td>32</td>
<td>2</td>
<td>24576</td>
<td>29832</td>
<td>281678</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>4</td>
<td></td>
<td></td>
<td>156034</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>90798</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>58322</td>
</tr>
<tr>
<td></td>
<td>3D torus</td>
<td>6</td>
<td>12</td>
<td>48</td>
<td>2</td>
<td>36864</td>
<td>44748</td>
<td>639786</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>4</td>
<td></td>
<td></td>
<td>343302</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>8</td>
<td></td>
<td></td>
<td>190494</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>8</td>
<td></td>
<td></td>
<td>114594</td>
</tr>
<tr>
<td>128</td>
<td>Ring</td>
<td>2</td>
<td>4</td>
<td>16</td>
<td>4</td>
<td>49152</td>
<td>55308</td>
<td>277198</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>78574</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>16</td>
<td></td>
<td></td>
<td>64214</td>
</tr>
<tr>
<td></td>
<td>2D torus</td>
<td>4</td>
<td>8</td>
<td>32</td>
<td>2</td>
<td>49152</td>
<td>54408</td>
<td>502862</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>4</td>
<td></td>
<td></td>
<td>164526</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>107474</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>8</td>
<td></td>
<td></td>
<td>601350</td>
</tr>
<tr>
<td></td>
<td>3D torus</td>
<td>6</td>
<td>12</td>
<td>48</td>
<td>2</td>
<td>73728</td>
<td>81612</td>
<td>1119018</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>4</td>
<td></td>
<td></td>
<td>337950</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>12</td>
<td>8</td>
<td></td>
<td></td>
<td>208724</td>
</tr>
</tbody>
</table>

\[ T_{FPX} = \begin{align*} 
BL \log B \times 6 + & \left( (2 \log B + 2) B \right) \\
+ & \frac{2 \log B \log B}{2} \times 2 \times L \times 2 \\
= 2L(8B + 2 \log B)\log B + 2B \end{align*} \tag{14} \]

In Eq. (15) before transformation, the first term represents the \( L \times B \) cross points of width \( \log B \) bits, and the second and third terms represent the AND gate and inverter inside the decoder. From the above results, the total number of transistors in the block control combination logic circuit is as follows:

\[ T_{ctl} = (T_{F1} + T_{FO} + T_{I1})L + T_{FPX} \tag{15} \]

However, circuit 1) and 2) are not used when \( F = 1 \), so in this case the total number of transistors is as follows:

\[ T_{ctl} = (T_{I1})L + T_{FPX} \tag{16} \]

d) surrounded circuits of multiport memory

Around the perimeter of the multimode memory, a \( W \)-bit \( L \times B \) crossbar switch is needed at both the input and output. These each require \( L \times B \times W \) crosspoints. Also, to control the crossover switch, \( L \) \( \log B \)-bit decoders are needed at both the input and output. The decoder requires \( B \) \( \log B \)-input AND gates, and \( B \times 2 \log B \) / 2 inverters. Therefore, the total number of transistors is roughly as follows:

\[ T_{Mctl} = \begin{align*} 
6LBW + & \left( (2 \log B + 2) B \right) \\
+ & \frac{2 \log B \log B}{2} \times 2 \times L \times 2 \\
= 2L(6BW + 2B(\log B + 1) + 2^{\log B} \log B) \end{align*} \tag{17} \]

From the above result, the total numbers of transistors in the conventional method and the proposed method are given by Eqs. (19) and (20):

\[ T_{trtotal} = 6(M_{buf} + M_{pr}) \tag{18} \]
\[ T_{prototal} = 6(M_{buf} + M_{pr}) + T_{ctl} + T_{Mctl} \tag{19} \]

The number of channels per link is \( C/L \), and the number of memory block per link in Inter-Channel Buffer Sharing is \( B/L \). Also, in the ‘Flit-Based Control’, \( F \) is set as one. The number of transistors for implementation is counted to evaluate the hardware cost. The cost of memory element is assumed as 6, \( n \)-input NAND (NOR) gate is \( 2n \), inverter is 2, the cross point of crossbar switch is assumed to use a tri-state inverter so the number of transistors is assumed as 6. The implementation costs in terms of the number of transistors for Inter-Channel Buffer Sharing and Inter-Link Buffer Sharing is tabulated in the Table 1. In the evaluation, the total amount of buffer is same \((B \times F = 64) \) and the number of blocks \((B)\) are varying. In both of the ‘Inter-Channel Buffer Sharing’ and ‘Inter-Link Buffer Sharing’ by ‘Flit-Based Control’ in Table 1, the value of \( F \) is equal to 1. In this comparison, “Traditional” is only shown as the capacity of the flit buffers.

It is shown in Table 1 that the hardware cost of the Inter-Link Buffer Sharing method decreases with the decrease of the number of blocks (the value of \( B \) becomes smaller). In this table, ‘Progress Ratio’ is the ratio of ‘Inter-Link Buffer Sharing by block-based control’ to ‘Traditional’. The hardware cost of block-based control can be drastically reduced compared to flit-based control \((F = 1)\). Although the additional logic circuit for block control is needed, the hardware cost reduction effect of the memory element for control information and surrounded circuits of multiport memory exceeds the block-based control. When the router circuit is implemented on the condition of \( B \leq C \), the cost of Inter-Link Buffer Sharing becomes double to that of Inter-Channel Buffer Sharing. As mentioned above, the hardware cost of Inter-Link Buffer Sharing can be reduced by ‘block-based control’.

Furthermore, hardware cost reduction is possible because arbitration and switches for the bank memory can be
reduced. It is to be noted that crossbar switch is used for the shared memory of the Inter-Link Buffer Sharing.

4. Performance Evaluation

4.1 Hardware Cost

4.1.1 Hardware Cost of Partial Sharing

In this section, hardware cost of partial sharing is calculated and evaluated. In this evaluation, a memory is shared by each two physical links. The structure of partial sharing is the similar to the structure of two ‘Link Sharing Method of ring network’. Therefore, the twice of the formula for ring network is applied in this evaluation. The cost is evaluated by prediction formulae. In 2D torus, the number of transistor \((T_{\text{partial/total}})\) for the partial sharing is shown by following formula.

\[
T_{\text{partial/total}} = 2T_{\text{protal}} \quad (L = 2)
\]

The details of evaluation is presented in [6] and [9].

The implementation costs (the number of transistors) of all-link sharing and partial sharing are tabulated in Table 2. In the evaluation, the total amount of buffer is same \((B \times F = 64)\) and the number of blocks \((B)\) is varying. Values of \(B\) in parentheses in partial sharing are the total number of blocks in the entire circuit. (For example, in the case of \(B = 8\) (16), the number of blocks of a shared memory of two links is 8, and the number of total blocks of a router is \(8 \times 2 = 16\).

From the Table 2, it is shown that the number of transistors of partial sharing is smaller than that of all-link sharing at the same \(B\) and \(F\). It is seen that the progress ratio of all-link sharing method considering \(B = 8\) and \(F = 8\) is 2.37, whereas the progress ratio of partial sharing method considering the same value is 1.70. From the comparison of the progress ratio between these two methods, it is shown that partial sharing method reduces the hardware cost significantly.

<table>
<thead>
<tr>
<th>(W)</th>
<th>Method</th>
<th>(L)</th>
<th>(C)</th>
<th>(B)</th>
<th>(F)</th>
<th>Num. of Transistors</th>
<th>Progress Ratio (2)/(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>Partial Sharing</td>
<td>4</td>
<td>8</td>
<td>16(4)</td>
<td>4</td>
<td>57564</td>
<td>2.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>41668</td>
<td>1.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>33684</td>
<td>1.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>All-link Sharing</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>90798</td>
<td>3.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>58322</td>
<td>2.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>42118</td>
<td>1.71</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>34002</td>
<td>1.38</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>Partial Sharing</td>
<td>4</td>
<td>8</td>
<td>16(4)</td>
<td>4</td>
<td>106716</td>
<td>2.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>78532</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16(24)</td>
<td>64404</td>
<td>1.31</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16</td>
<td>164526</td>
<td>3.35</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>107474</td>
<td>2.19</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>78982</td>
<td>1.61</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>64722</td>
<td>1.32</td>
<td></td>
</tr>
</tbody>
</table>

4.1.2 Hardware Cost of Partial Sharing in 3D Torus

In this section, hardware cost of partial sharing is calculated and evaluated in 3D torus. In 3D torus, one memory is also shared by two physical links. Therefore, the triple of the formula for ring network is used in this evaluation. The Table 3 shows the number of transistor of the partial sharing in 3D torus. In this table, all-link sharing in 3D torus is calculated by \(L = 6\) based on the formula(19). As shown in Table 3, the cost of All-Link Sharing is 2.08 times larger than the Traditional when the number of block is 6. And the cost is 5.17 times larger than the Traditional when the number of block is 24. Partial sharing have for about 1.37 times of cost of the Traditional when the number of block is 6, and for about 2.34 times of cost of the Traditional when the number of block is 24. Therefore, hardware cost is also reduced by partial sharing in 3D torus.

<table>
<thead>
<tr>
<th>(W)</th>
<th>Method</th>
<th>(L)</th>
<th>(C)</th>
<th>(B)</th>
<th>(F)</th>
<th>Num. of Transistors</th>
<th>Progress Ratio (2)/(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>Partial Sharing</td>
<td>6</td>
<td>12</td>
<td>8(24)</td>
<td>4</td>
<td>86346</td>
<td>2.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>82502</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>48</td>
<td>50526</td>
<td>1.37</td>
<td></td>
</tr>
<tr>
<td></td>
<td>All-link Sharing</td>
<td>12</td>
<td>12</td>
<td>24</td>
<td>24</td>
<td>190494</td>
<td>5.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>114558</td>
<td>3.11</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>48</td>
<td>76842</td>
<td>2.08</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>Partial Sharing</td>
<td>6</td>
<td>12</td>
<td>8(24)</td>
<td>4</td>
<td>160074</td>
<td>2.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>117798</td>
<td>1.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>48</td>
<td>96606</td>
<td>1.31</td>
<td></td>
</tr>
<tr>
<td></td>
<td>All-link Sharing</td>
<td>12</td>
<td>12</td>
<td>24</td>
<td>24</td>
<td>357950</td>
<td>4.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>24</td>
<td>206118</td>
<td>2.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>48</td>
<td>141354</td>
<td>1.92</td>
<td></td>
</tr>
</tbody>
</table>
A used simulator was written by C language, and hardware structure of the router and the interconnection network was reproduced faithfully by the Functional unit level. The traffic pattern is uniform traffic that is random. In this pattern, Every PE generates packet with a specified probability, and destinations are chosen randomly with equal probability among the nodes in the network. In addition, to evaluate about the various situations, we simulated by the situation that a packet range was set to 16 or 32 or 64 bits, in this evaluation. In NoC and SoC, a lot of memory transfer is assumed, such as DMA transmission in a memory access outside FPGA and A memory copy in GPU.

4.2.1 Evaluation 1: Relation between the Number of Blocks and Communication Performance for 2D Mesh/Torus Network

In evaluation 1, we evaluate the influence of the number of blocks. If the number of blocks is small value, the hardware cost of the Link Sharing will become small. But, communication performance may fall because the utilization efficiency of a memory falls. Figures 19 - 20 portrayed the results of simulations of a torus and mesh network. In our evaluation, we compare the following cases.

- Traditional: Memories are not shared.
- Channel: Memory is shared by some virtual channels in each physical links.
- Flit-Based control: One type of Link Sharing. It does not apply ‘Block-Based control’.
- B2, B4, B8: They are Link Sharing method called ‘Block-Based control’. In those methods, the number of blocks are 2 (B2), 4 (B4), and 8 (B8).

In Figs. 19 - 20, the latency before saturation is almost the same between no-sharing and proposed method. In proposed method, shared memory is used when a block of communication happened continuously, and it’s lacking in private buffer. When network is not crowded, only private buffers are used. The path when not using a sharing memory is mostly same as a traditional method in proposed method. This is written in “Path-1 pipeline” in 3.3.2.

As shown in Figs. 19 - 20, the difference of performance between B8 and Flit-Based control is a little. On the other hand, the performance of B2 and B4 are lower than Flit-Based control in many cases. As stated above, eight is enough as the number of blocks in 2D mesh and torus. Henceforth, eight is used as a basic status of the number of blocks for 2D Mesh/Torus.

4.2.2 Evaluation 2: Evaluation of 3D Torus Network

In this section, we evaluate the performance of the Link Sharing in a 3D torus network. The evaluation result is shown in Fig. 21. B1, B2, B4, B6, B 8, and B12 are the results of the Link Sharing with 1, 2, 4, 6, 8 and 12 blocks respectively. As shown in Fig. 21, the performance of Link Sharing is higher than traditional, and the difference of performance between the Link Sharing and Flit-based control is a little. On the other hand, the difference of performance between B2, B4, B6, B 8 and B12 is a little, unlike 2D network. The progress ratio of the Link Sharing to the traditional in 3D torus is shown in Table 4. The “Throughput Progress Ratio” in percentage is defined as \((T_{th_{prop}}/T_{th_{con}} - 1)\times 100\). ‘\(T_{th_{prop}}\)’ is the throughput of the Link Sharing. ‘\(T_{th_{con}}\)’ is the throughput of the ‘traditional’. The “Throughput Progress Ratio” in 3D torus network is higher than 2D network.
Table 4 Throughput progress ratio of the link sharing for 3D torus.

<table>
<thead>
<tr>
<th>Number of PE</th>
<th>Total Buffer</th>
<th>Packet Length</th>
<th>Throughput Progress Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>48</td>
<td>16</td>
<td>1.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>32</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64</td>
<td>1.16</td>
</tr>
<tr>
<td>96</td>
<td>16</td>
<td>1.35</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>1.12</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>1.19</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 22 The communication performance of the partial sharing: 16 PE, 32 Buffer, and 16 Flits/Packet.

Fig. 23 The communication performance of the partial sharing: 16 PE, 64 Buffer, and 32 Flits/Packet.

4.2.3 Communication Performance of Partial Sharing

The efficiency of the memory of partial sharing is lower than all-link sharing. In this section, we discussed the result of performance evaluation of partial sharing compared with all-link sharing. 2D torus network is used in this evaluation.

The results of simulation are shown in Figs. 22, 23 and Table 5. ‘B2’, ‘B4’, and ‘B8’ are the result of all-link sharing that the number of blocks is 2, 4, and 8, respectively. Also, ‘PS-B4’, ‘PS-B8’, and ‘PS-B16’ are the result of partial sharing that the total number of blocks in router is 4, 8, and 16, respectively. The result shown in the Table 5 represents the “Throughput Progress Ratio” of the ‘all-link sharing’ and ‘partial sharing’ over ‘traditional’.

From those results, it is shown that the partial sharing has almost same performance in ‘PS-B4’, ‘PS-B8’, and ‘PS-B16’. Also, the “Throughput Progress Ratio” of ‘PS-B4’ and ‘PS-B8’ are slightly lower than ‘B8’. However, they were almost same or higher than ‘B2’ and ‘B4’. Because the hardware cost of ‘PS-B4’ and ‘PS-B8’ is almost same as ‘B2’ and ‘B4’, the cost performance of partial sharing is higher than all-link sharing.

Table 5 The throughput progress ratio of communication throughput on partial sharing.

<table>
<thead>
<tr>
<th>PE</th>
<th>Buffer</th>
<th>Packet</th>
<th>All-link</th>
<th>Partial</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B2</td>
<td>B4</td>
<td>B8</td>
<td>PS-B4</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>10.4</td>
<td>11.5</td>
<td>9.9</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>8.9</td>
<td>9.5</td>
<td>8.7</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>6.2</td>
<td>5.4</td>
<td>8.7</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>5.8</td>
<td>6.7</td>
<td>9.7</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>3.8</td>
<td>11.1</td>
<td>8.1</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>17.0</td>
<td>13.1</td>
<td>18.5</td>
</tr>
<tr>
<td>32</td>
<td>16</td>
<td>7.8</td>
<td>13.4</td>
<td>17.9</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>8.8</td>
<td>9.0</td>
<td>16.4</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>8.4</td>
<td>8.5</td>
<td>9.0</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>3.9</td>
<td>7.2</td>
<td>7.5</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>7.8</td>
<td>9.9</td>
<td>16.4</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>15.8</td>
<td>16.6</td>
<td>21.5</td>
</tr>
<tr>
<td>64</td>
<td>16</td>
<td>16.6</td>
<td>21.5</td>
<td>15.3</td>
</tr>
</tbody>
</table>

Fig. 24 The communication performance of a 3D torus: 512 PE, 48 Buffer, and 16 Flits/Packet.

Fig. 25 The communication performance of a 3D torus: 512 PE, 48 Buffer, and 32 Flits/Packet.

4.2.4 Communication Performance of Partial Sharing in 3D Torus

We evaluated the performance of partial sharing in 3D torus.
In this evaluation, the number of physical links which share a memory is two. In this evaluation, we applied Flit-Based control and proposed method which the number of block are 12, 6, 3. Figures 24 - 29 portrayed the results of simulations. A number of B3, B6, B12 in the figure is number of blocks. It is shown in graph that the performance of partial sharing is the intermediate between the no-sharing and all-link sharing. On the other hand, the difference of performance between the several numbers of blocks is a little.

4.2.5 Comparison with Traditional Router

The proposed method and traditional method that a buffer size was enlarged was compared. The used network was 3D torus that number of PE is 512. Packet ranges were 16, 32, or 64 flits/packet. Buffer capacity of traditional router was 48, 60, 72, or 96 flits. Buffer capacity of the all-link sharing and partial sharing method was 48 flits.

The result of evaluation is shown in Figs. 30 - 32. As shown in graph, the all-link sharing was high performance most in all comparative targets. The performance of partial sharing was higher than traditional router of 48 and 60 flits, and was same as traditional router that buffer of 96 flits. As
a result of the above and Tables 1 - 3, it is thought that there is a possibility that the cost performance improves higher than when the capacity of the buffer of a traditional router is increased.

5. Conclusion

In this paper, we have introduced the summary and pipeline structure of Inter-Link Buffer Sharing (Link Sharing) method, and evaluated the communication performance in 3D-torus network. We have also applied partial sharing for the reduction of hardware cost. It is found that performance of proposed method is higher than that of conventional method. In 3D torus, the communication performance of the Link Sharing method is higher than that of traditional method. Also, the difference of performance is a little between block-based control and flit-based control. The progress ratio in 3D torus network is higher than 2D network.

It is also shown that the improvement of communication performance using partial link sharing method is achieved with slightly increase of hardware cost. The cost-performance of partial sharing becomes higher than all-link sharing method. In a 3D-torus network, the performance of partial sharing is in between the no-sharing and all link sharing.

Acknowledgements

A part of this research was supported by JSPS KAKENHI Grant Number JP17K00087.

References

Naohisa Fukase was born in 1986. March in 2015, have completed Shonan Institute of Technology graduate school of engineering doctor’s programs electrical and information engineering. The same year and become Shonan Institute of Technology researcher. study the router structure, a direct combination network and network on chip. Shonan Institute of Technology, 1-1-25, Tsujido Nishikaigan, Fujisawa, Kanagawa, Japan.

Yasuyuki Miura received the bachelor’s degree in Tohoku University in 1997, and the MS and Ph.D degree in Japan Advanced Institute of Science and Technology (JAIST) in 1999 and 2002. Then he had worked in the NICT, Japan until December 2004. From January 2005 to March 2006, he was researcher of the Japan Science and Technology Agency (JST). Since April 2006, he is lecturer in the Shonan Institute of Technology. His research interests include parallel processing, interconnection network, and MPEG video streaming. He is a member of the IEEE and the IEEE Computer Society.

Shigeyoshi Watanabe received the B.E. degree in instrumental engineering from Keio University, Japan, in 1977 and the M.S. degree in applied physics from Tokyo Institute of Technology, Japan, in 1979. He received the Ph.D. degrees in instrumental engineering from Keio University, in 1998. In 1979 he joined Toshiba Corporation, Japan. Since then he has been working on research and development of MOS memories, SOI/BiCMOS/CMOS device technologies, low-power and high-speed technology and architecture of system LSI. On October 2005 he joined Shonan Institute of Technology. He is a professor of the department of information science. He works in multiple areas of LSI: memories, device technologies, low-power/high-speed technology and architecture of system LSI. He has written more than 150 technical papers and reports in these areas and holds more than 65 patents. His research interests are in the area of low-power/high-speed technology, novel circuit/device technology and architecture of system LSI.

M.M. Hafizur Rahman received his B.Sc. degree in Electrical and Electronic Engineering from Khulna University of Engineering and Technology (KUET), Khulna, Bangladesh, in 1996. He received his M.Sc. and Ph.D. degree in Information Science from the Japan Advanced Institute of Science and Technology (JAIST) in 2003 and 2006, respectively. Dr. Rahman is now a JSPS postdoctoral research fellow at Center for Information Science, JAIST. He was also the same at Graduate School of Information Science (GSIS), Tohoku University, Japan. Prior to join at GSIS, Tohoku University, He was an assistant professor in the Dept. of Computer Science and Engineering at KUET. He was also a visiting researcher in the School of Information Science at JAIST in 2008. His current research include hierarchical interconnection networks, optical switching networks, and internet backbone network. Dr. Rahman is member of IEICE of Japan and IEB of Bangladesh.