PAPER

Special Section on Parallel, Distributed, and Reconfigurable Computing, and Networking

Improving the Performance of Circuit-Switched Interconnection Network for a Multi-FPGA System

Kohei ITO††, Kensuke IIJUKA†, Kazuei HIRONAKA†, Yao HUY†, Nonmembers, Michihiro KOIBUCHI††, Senior Member, and Hideharu AMANO†, Fellow

SUMMARY Multi-FPGA systems have gained attention because of their high performance and power efficiency. A multi-FPGA system called Flow-in-Cloud (FiC) is currently being developed as an accelerator of multi-access edge computing (MEC). FiC consists of multiple mid-range FPGAs tightly connected by high-speed serial links. Since time-critical jobs are assumed in MEC, a circuit-switched network with static time-division multiplexing (STDM) switches has been implemented on FiC. This paper investigates techniques of enhancing the interconnection performance of FiC. Unlike switching fabrics for Network on Chips or parallel machines, economical multi-FPGA systems, such as FiC, use Xilinx Aurora IP and FireFly cables with multiple lanes. We adopted the link aggregation and the slot distribution for using multiple lanes. To mitigate the bottleneck between an STDM switch and user logic, we also propose a multi-ejection STDM switch. We evaluated various combinations of our techniques by using three practical applications on an FiC prototype with 24 boards. When the number of slots is large and transferred data size is small, the slot distribution was sometimes more effective, while the link aggregation was superior for other most cases. Our multi-ejection STDM switch mitigated the bottleneck in ejection ports and successfully reduced the number of time slots. As a result, by combining the link aggregation and multi-ejection STDM switch, communication performance improved up to 7.50 times with few additional resources. Although the performance of the fast Fourier transform with the highest communication ratio could not be enhanced by using multiple boards when a lane was used, 1.99 times performance improvement was achieved by using 8 boards with four lanes and our multi-ejection switch compared with a board.

key words: multi-FPGA, multi-FPGA communication, circuit-switched network, STDM switch

1. Introduction

Multi-access edge computing (MEC) has been extensively researched for time-critical services requested from the base stations of the 5th generation mobile communication (5G). Using low-latency and high-bandwidth communication supported by 5G, sophisticated factory control and traffic management of a smart city are possible with MEC. Unlike edge computing limited by tiny devices, image or sound recognition, which requires a much more powerful computational facility, can be carried out with MEC without a large and uncertain delay for communicating with the cloud. However, compared with a server in a data center, the power budget and allowable cost of an MEC server is strictly limited since it must be placed at a base station not in a data center.

Field programmable gate array (FPGA) computing has received attention for MEC servers thanks to its cost and power efficiency. Another important benefit of FPGA computing is that it can cope with time-critical jobs by directly accepting requests from edges with its I/O and executing the job by using the hardwired logic with fixed performance. For processing multiple requests at a base station, multi-FPGA systems are especially advantageous. Flow-in-Cloud (FiC) [1], a bare-metal multi-FPGA cluster, is being developed with a number of tightly connected mid-range FPGAs with high-speed serial GTH links. To handle the time-critical service required for MEC, FiC provides a circuit-switched network with static time-division multiplexing (STDM) switches. The goal with FiC is enabling direct accepting requests from edges with its I/O and executing the job by using the hardwired logic with fixed performance. To achieve this goal, high-speed communication is essential. If communication performance is low, it is impossible to improve application performance by using multiple FPGAs.

Some multi-FPGA systems use extremely high-speed links of more than 40 Gbps [2]–[4], while FiC uses cost-efficient GTH serial links each of which supports 9.9-Gbps transfer speed. It provides 32 bi-directional lanes per board to maintain the total bandwidth. This enables us to use the cost-efficient Firefly™ Micro Flyover™ system by Samtec. Two-meter cables are stably operational in the current prototype without using any special transceivers and receivers by using only Xilinx Aurora IPs in the FPGA. Firefly bundles four serial lanes into a flat cable. Unfortunately, since the training phase for maintaining data transfer is independently inserted to each lane, they cannot be treated as a single 4-bit parallel link, unlike PCIe or Infiniband. Since it is difficult to make the best use of the large aggregated bandwidth, the network with many GTH lanes has not been commonly used. To solve this problem, we adopted two techniques: the link aggregation and the slot distribution. The link aggregation is a straight-forward technique for using multiple links for different data in parallel. The slot distribution uses multiple links to reduce the number of slots, which affect STDM performance [5]. However, it is difficult to take advantages of these techniques with only one port for ejection, which tends to bottleneck the communication. To solve this...
problem, we also propose an STDM switch that provides multiple ejection ports.

Three techniques introduced here are easy to come up for achieving the goal, and similar techniques have been used for other type of network as shown in related work [6], [7]. However, applying them to a circuit-switched network with bundled GTH lanes in multi-FPGA systems has not been reported.

This paper includes our earlier research [8] on the comparison of the link aggregation, which aggregates multiple lanes and regards as one lane [9], and the slot distribution, which distributes time slots to multiple lanes. The new thing we have done in this paper is to clarify the characteristics of the link aggregation and the slot distribution more extensively by comparing them in a larger number of settings than in our earlier study (parts of Fig. 11 and 12), to mitigate the bottleneck in ejection ports (Sect. 4 and 5.3.2), and to combine the link aggregation and the multi-ejection STDM switch (Fig. 16).

The contributions of this paper are as follows:

• We adopt the link aggregation and the slot distribution to use multiple GTH lanes bundled into a cable for cost efficient multi-FPGA networks and evaluated their trade-off.
• We propose a multi-ejection STDM switch to mitigate the bottleneck in ejection ports.
• We evaluated the resource utilization, communication time, and power consumption of our techniques and the combination of the link aggregation and the multi-ejection STDM switch on an FiC prototype with real application programs.

The rest of the paper is organized as follows. In Sect. 2, we describe FiC. In Sect. 3, we describe two techniques using multiple lanes: the link aggregation and the slot distribution. In Sect. 4, we describe the implementation of our proposed multi-ejection STDM switch. In Sect. 5, we discuss the evaluation of our techniques. In Sect. 6, we introduce related work. Finally, we conclude the paper in Sect. 7.

2. Flow-in-Cloud (FiC)

2.1 Overview of FiC

FiC consists of multiple mid-range FPGA boards tightly connected by a number of high-speed GTH serial links. To handle time-critical jobs for MEC, it adopts a circuit-switched network with STDM switches. A prototype with 24 FPGA boards is in use, as shown in Fig. 1, and its network topology is a $4 \times 6$ 2D torus.

2.2 FiC-SW

Figure 2 shows a diagram of an FPGA board called FiC-SW. A FiC-SW consists of an FPGA, two 16GB DRAM cards, high-speed serial links, and Raspberry Pi3. The FPGA mounted on the FiC-SW is either Xilinx Kintex UltraScale XCKU095 or XCKU115, and we call the board with XCKU095 a Mark1 board and that with XCKU115 a Mark2 board. The current FiC prototype has 12 Mark1 boards and 12 Mark2 boards; 24 boards in total. Raspberry Pi3 is connected to the FPGA with GPIO and the host server via Ethernet to manage the configuration and I/O for the FPGA.

The FPGA on FiC-SW is divided into two segments, as shown in Fig. 3. One is a shell that includes SERDES, switches for serial interconnect, and controllers for DDR-SDRAM, GPIO of Raspberry Pi3, and so on. It is configured at the system power-up, and the design does not
change during operation. The other segment is the user-designed accelerator. In FiC, the designer is supposed to use high-level synthesis (HLS) modules, which are connected with the AXI stream interface. These HLS modules are assigned into a single FPGA board or multiple FPGA boards. Various types of applications, e.g., convolutional neural network [10], [11], recurrent neural network [12], genome matching [13], and benchmark application programs have been implemented. Partial configuration [14] is applied to the user-designed HLS segment and the configuration can be carried out without stopping the network in the shell.

Up to eight high-speed serial links can be connected to FiC-SW. The current FiC uses FireFly cables, as shown in Fig. 4, as high-speed serial links. A FireFly cable has four bi-directional lanes, and the bandwidth of each lane is 9.9 Gbps. They are directly connected to the GTH serial interfaces of the FPGA, and Xilinx Aurora IPs are used for the interface of serial links in the FPGA. Unlike the PCIe x4 links, each lane works independently since the training phase (link training) is inserted asynchronously. Therefore, it is impossible to treat them as a \(4 \times 9.9\) Gbps logical link, and an independent switch is provided for each lane.

### 2.3 Static Time Division Multiplexing Switch

An STDM switch is a TDM switch with its switching configuration set beforehand. There have been a few examples of similar switches, mainly in Network on Chips (NoCs) [15], [16]. In FiC, STDM switches are used as network switches. The reasons for adopting STDM switches in FiC systems are as follows. (1) The STDM switch makes it relatively easy to predict communication latency. Since FiC is supposed to be used for time-critical jobs in MEC, this is essential. (2) The resources needed are fewer than those of a packet-switched network. This means that we can keep the resources of FPGA for applications. An STDM switch operates at 100 MHz and can transfer 170-bit data (header: 42-bit, payload: 128-bit) every two clock cycles. Therefore, the maximum bandwidth per lane is 8.5 Gbps.

Figure 5 shows an operation example of an STDM switch on FiC. An input flit enters a FIFO, and if the slot of the flit, which comes at the top of the FIFO, matches the slot of the switch, it flows to the output port(s) in accordance with the routing table set in advance [17]. Each input port generally has several time slots. The volume of one time slot (170 bits) is the size of the data received and accommodated within two clock cycles. The connection between an input port and output port is established before the data transfer. The read or write flit operation on the input or output side tours the time slots one by one in two clock cycles. Therefore, the slot of the switch rotates once at \(2 \times \#_{of\_slots}\) clock cycles. In the example shown in Fig. 5, the data in each slot can be transferred every six clock cycles. A flit transfer is completed at the STDM switch when the write flit operation, which copies flit data at an input port, is finished at an output port. On the same time slot, the read and write flit operations are synchronized. Also, like Slot 1 in Fig. 5, an STDM switch can broadcast or multicast data easily by establishing the connections from an input port to multiple output ports in the same slot.

The worst end-to-end communication latency using a single slot can be estimated using Eq. (1). The theoretical maximum bandwidth of a slot is expressed using Eq. (2) [8], [17], where \(S\), \(L\), \(H\) are the number of slots, number of flits (amount of 170-bit data), and hop count, respectively. The number 60 is the time (clock cycles) for processing in the Aurora IP. Since an STDM switch can only transfer flits in a certain slot at a fixed interval, the number of slots, as well as hop count and the number of flits to be transferred, affects the communication performance. Therefore, reducing the number of slots is important for STDM switches.

\[
2S(L - 1) + (60 + 2S)H\ \text{clock} \\
\frac{8.5}{S}\ \text{Gbps}
\]

The method described in the first part of [5] analyzes the minimum slot by assigning two or more communications that do not interfere with each other to one slot and optimizes the the number of slots required. A tool based on
this method is used to make the routing table for the FiC network and applications. Figure 5 shows an example of a table setting that may combine slots. In this switch, Slot 0 and Slot 2 can be assigned to the same slot because the input and output ports do not overlap. If the switching of Slot 0 and Slot 2 of other switches that are used at the same time does not interfere with each other, the slots can be combined using this method for the entire network.

In FiC, four lanes are bundled into a FireFly cable. One STDM switch is provided for each lane, as shown in Fig. 3. For the current torus network, four $5 \times 5$ (four neighbor FGAs and an HLS module inside the FPGA) switches are provided instead of a large $20 \times 20$ switch because communication between lanes in a link is rarely needed.

3. Using Multiple Lanes

In this section, we explain the link aggregation and the slot distribution for using multiple lanes on FiC. The link aggregation increases the bandwidth by operating multiple lanes in the same way. The slot distribution reduces the number of time slots by operating them individually. As described in Sect. 2, the amount of transferred 170-bit data and number of slots significantly affect communication performance. By increasing the bandwidth using multiple lanes, the amount of 170-bit data transferred in a lane can be reduced. Therefore, these two techniques can improve communication performance. We describe these techniques in detail below.

3.1 The Link Aggregation

The straightforward method for using multiple lanes is the link aggregation. This technique aggregates multiple lanes and regards as one lane [9].

Figure 6 (a) shows an example of communication when only one lane is used. In Fig. 6 (b), the central node is sending data in four slots to neighboring boards: east, west, north, and south. Four connections are established from Node 5 to Node 0 in slot 0, to Node 1 in slot 1, to Node 2 in slot 2, and to Node 3 in slot 3.

Figure 6 (b) shows the case of applying the link aggregation with 2 lanes to the example in Fig. 6 (a). Those four connections are established only in one lane in Fig. 6 (a), while in Fig. 6 (b), eight connections are established using two lanes in parallel. They are established from Node 5 to Node 0 with both upper and lower lanes in slot 0, to Node 1 with both in slot 1, to Node 2 with both in slot 2, and to Node 3 with both in slot 3.

When the number of lanes is $N$, the bandwidth that can be used for each slot can increase $N$ times by applying the link aggregation for multiple lanes. With this technique, all the STDM switches on the same board have the same configuration, and the configuration of the STDM switches and number of slots are also the same when using a single lane.

3.2 The Slot Distribution

The slot distribution distributes slots to multiple lanes to reduce the number of slots for a lane.

Figure 6 (c) shows the case of applying this technique for two lanes to the example in Fig. 6 (a). All four slots are processed in a lane in Fig. 6 (a), while in Fig. 6 (c), the number of slots is halved by distributing slots for each lane. In slot 0, two connections are established from Node 5 to Node 0 with the upper lane, and from Node 5 to Node 1 with the lower lane. In slot 1, two connections are established from Node 5 to Node 2 with the upper lane, and from Node 5 to Node 3 with the lower lane.

If the number of lanes is $N$, the number of slots can be reduced to $1/N$ by applying the slot distribution for multiple lanes. However, the bandwidth that can be used by a board for data transfer is the same as that of one lane. With this technique, the STDM switches on the same board generally must have individual configurations.

4. Proposed Multi-Ejection STDM Switch

In circuit-switched interconnection fabrics, injection/ejection ports tend to be bottlenecks. Figure 7 (a) is an example in which only a port is provided for injection/ejection. In FiC, when connections from/to the user logic are needed,
the same number of time slots are allocated to the link between the injection/ejection port and user logic. For example, when the user logic establishes connections from input port 0 and input port 1, an in Fig. 7 (a), two slots are needed. In FiC, when a user logic sends the same data to the user logic on the other multiple boards, broadcast or multicast can mitigate the injection bottlenecks but does not mitigate the ejection bottlenecks.

We developed our multi-ejection STDM switch to mitigate the ejection bottleneck. Figure 7 (b) shows the concept of this switch. Compared with the base STDM switch (Fig. 5), it provides multiple ejection ports, and each port can establish a connection from a certain input port. For example, ejection port 1 can establish a connection from input port 1. By using our multi-ejection STDM switch, the number of slots can be reduced, as in Fig. 7 (b). We also provided an ejection table to determine whether the connections to ejection ports are established. Its entry is provided for each slot.

5. Evaluation

Unlike a packet switching network, in STDM networks, performance can be easily estimated with parameters (# of slots, # of 170-bit data, # of hops) by using Eq. (1). One-to-one and collective communication in a single lane have been evaluated [8], [17] on real machines and theoretically. Since our techniques and switch just change these parameters, the performance of the synthetic traffic benchmarks can be easily estimated. Thus, we implemented the techniques and switch on application and evaluated their communication performance.

5.1 Applications

We used three applications: conjugate gradient (CG) method, LeNet, and fast Fourier transform (FFT).

5.1.1 CG Method

The CG method is an algorithm that solves linear equations the coefficients of which form a positive definite symmetric matrix by using an iterative method. For an $N \times N$ matrix, it theoretically converges at most $N$ iterations. The following equations are used to solve $b = Ax$ for the iterative part of the CG method. The subscript $k$ indicates the number of iterations.

\[
y_k = A p_k
\]

\[
\alpha_k = \frac{(p_k, r_k)}{(p_k, y_k)}
\]

\[
x_{k+1} = x_k + \alpha_k p_k
\]

\[
r_{k+1} = r_k - \alpha_k y_k
\]

\[
\beta_k = -\frac{(r_{k+1}, y_k)}{(p_k, y_k)}
\]

\[
p_{k+1} = r_{k+1} + \beta_k p_k
\]

where $r$ is a residual vector, and the convergence can be determined by the norm of this vector approaching 0.

On FiC, the CG method uses Eq. (3) in parallel between boards, aggregates the results by all-to-all communication, then executes the other calculations individually. We used $384 \times 384$ and $2048 \times 2048$ 32-bit floating-point dense matrices, and the computation was terminated when 384 and 2048 repetitions were completed, respectively.

5.1.2 LeNet

LeNet[18] is a simple neural network that can recognize handwritten numbers from 0 to 9. As shown in Fig. 8, parallel calculation between the boards is executed in convolution (CONV) 1, CONV 2, fully connected (FC) 1, and FC 2. All-to-all communication is executed after each parallel calculation is completed [11]. This implementation uses 32-bit float numbers to maintain inference accuracy.

5.1.3 FFT

FFT is a high-speed algorithm to solve discrete Fourier transform [19]. The FFT on FiC obtains a part of the array for the butterfly operation from other boards when they are needed. The communication is required twice when divided into four boards, and three times when divided into eight boards. Figure 9 is an example of 8 boards. In this case, board 0 communicates with boards 1, 2, and 4. The 32-bit floating-point vector with 8192 elements is used.

The summary of three applications from the viewpoint
of computation and communication is as follows.

- CG method: generates the same broadcast traffic at a fixed interval. The ratio of computation to communication is relatively high for a large matrix.
- LeNet: generates different traffic for each layer. The ratio of computation to communication is high.
- FFT: generates communication that increases as the number of boards increases because the divided parts to multiple boards require a large amount of communication. That is, the ratio of computation to communication is low.

5.2 Evaluation Setup

Table 1 shows the environment of implementation and evaluation. For the design of 24 boards, we used 12 Mark1 boards (XCKU095) and 12 Mark2 boards (XCKU115). In other evaluations, we used only Mark2 boards, which have more resources than Mark1 boards. Figure 10 shows which boards execute the above applications. Instead of the 4-boards ring topology, we also used a 4-boards fully-connected topology by changing the FireFly cable to confirm that the communication time when using our multi-ejection STDM switch is dependent on topologies (i.e., the communication time when using our multi-ejection STDM switch can be improved by [5]'s generation of optimized topology for a communication pattern). “FC” means that the application is executed on the fully-connected topology. The FiC used in the evaluation has the specification described in Sect. 2. The slot allocation algorithm follows [5]. Hereafter, “base switch” denotes the base STDM switch discussed above, and “M-e switch” denotes our multi-ejection STDM switch.

Table 2 shows the resource utilization of the shell in XCKU115. The resource utilization of two switches increased to about twice that of only one switch, and that of four switches increased to about 4 times. This is because most of the shell of the current FiC is used for STDM switches and Aurora IP. However, the resource utilization in applications such as BRAM, FF, and LUT is still small even with four switches, and enough resources remained for application designers. Although the use of GT is high, it is not a problem for application designers because it is for the GTH links.

5.3 Results

5.3.1 Multiple Lanes

Table 2 shows the resource utilization of the shell in XCKU115. The resource utilization of two switches increased to about twice that of only one switch, and that of four switches increased to about 4 times. This is because most of the shell of the current FiC is used for STDM switches and Aurora IP. However, the resource utilization in applications such as BRAM, FF, and LUT is still small even with four switches, and enough resources remained for application designers. Although the use of GT is high, it is not a problem for application designers because it is for the GTH links.

We describe the communication performance of the applications. The communication time was obtained by subtracting the execution time of the calculation part of an application in the HLS simulation from the execution time of the entire application on actual devices. Hereafter, “Nla” denotes when the link aggregation for N lanes is applied, “Nsd” denotes when the slot distribution for N lanes is applied, and “1lane” denotes when only one lane is used. We also applied both techniques together. That is, both techniques were applied to two pairs of two lanes. The term “2lax2sd” denotes this case.

Figure 11 and 12 show graphs of the execution times of the CG method with the 384 × 384 and 2048 × 2048 matrices, respectively. For 24 boards, the calculation times of XCKU095 and XCKU115 slightly differed. We used the calculation time of XCKU115 as the representative. The execution time on one board and with the CPU is shown in Tables 3 and 4 for comparison. The CG method with the 2048 × 2048 matrix could not be implemented on one board due to the shortage of BRAM. The amount of traffic with the CG method with the 384 × 384 matrix is small, e.g., with the link aggregation on 24 boards, each board sent only one piece of 170-bit data in one iteration.

As shown in Fig. 11 and 12, communication time decreased using multiple lanes in many cases. For example, with the CG method with the 2048 × 2048 matrix on 4 boards, the communication time of 4la (Fig. 12(e)) decreased to 0.43 times that of 1lane (Fig. 12(a)). The communication time of 2sd was slightly worse than that of 1lane for either matrix or the number of boards except for the 384 × 384 matrix on 24 boards. Although the communication time of 2lax2sd decreased compared with 1lane, it was worse than that of 2la except for the 384 × 384 matrix on 24 boards. From these results, we were able to reduce commu-
The link aggregation was superior to the slot distribution for many cases. As described above, the communication time of 2la decreased compared with that of 1lane; however, that of 2sd was worse than that of 1lane in most cases. In all cases, the communication time of 2la was smaller than that of 2sd. The results of using four lanes were slightly different from those of using two lanes. For the $384 \times 384$ matrix on 8 and 24 boards and the $2048 \times 2048$ matrix on 8 boards, the communication time of 4sd was smaller than that of 4la. However, in the other cases, the communication time of 4la was smaller than that of 4sd.

To further compare the link aggregation and the slot distribution, let us look at the results of LeNet. Figure 13 shows the execution time of LeNet. The execution time of the LeNet on CPU (1 threads) was 14 msec. The communication time of LeNet also decreased using four lanes, and the reduction in the communication time of the link aggregation was higher than that with the slot distribution. LeNet communicated more 170-bit data at a certain interval than the CG method with the $2048 \times 2048$ matrix.

In the cases in which the slot distribution was superior to the link aggregation (i.e. the CG method with the $384 \times 384$ matrix on 8 and 24 boards with four lanes, and that with the $2048 \times 2048$ matrix on 8 boards with four lanes), the transferred data size was small and number of slots was large compared with the other cases. With the CG method with the $384 \times 384$ matrix on 24 boards, the communication time of 2lax2sd was smaller than that of 2la. The transferred data size was small and number of slots was large compared with the other cases; the results of which are the opposite. These results indicate that when the number of slots is large and transferred data size is small, the slot distribution is sometimes more effective, while the link aggregation is superior for most other cases.

Table 5 shows the power consumption of the CG method with $384 \times 384$ matrix. Since the I/O of an FPGA uses a different power supply from that of the FPGA core, we can measure the power consumptions of I/O and FPGA by using the clump current meter individually. BSIDE’s ACM91 with 1-mA accuracy was used. The I/O mainly includes the power for transferring data on FireFly cables. The power consumption of I/O increased by the number of lanes used. The power consumption of the FPGA core did not increase steadily, and the power difference between two lanes and four lanes was larger than that between one lane.
Table 5  Power consumption of CG method with $384 \times 384$ matrix on 4 boards

| (value: per board) | FPGA (W) | I/O (W) | total (W) |
|-------------------|----------|---------|-----------|
| 1lane, base       | 1.69     | 2.68    | 4.37      |
| 2la, base         | 1.98     | 4.02    | 6.00      |
| 2sd, base         | 1.97     | 3.98    | 5.95      |
| 2lax2sd, base     | 2.78     | 6.46    | 9.24      |
| 4la, base         | 2.76     | 6.66    | 9.42      |
| 4sd, base         | 2.81     | 6.66    | 9.47      |
| 4lane, M-e        | 1.89     | 2.74    | 4.63      |
| 4la, M-e          | 3.01     | 6.73    | 9.75      |

Table 6  Reduction in number of slots with M-e switch

| # of slots |
|------------|
| 4 board CG(base) | 4 |
| 4 board CG(M-e, ring) | 2 |
| 4 board CG(M-e, FC) | 1 |
| 8 board CG(base) | 8 |
| 8 board CG(M-e) | 3 |
| 24 board CG(base) | 24 |
| 24 board CG(M-e) | 6 |
| 4 board FFT(base) | 2 |
| 4 board FFT(M-e) | 1 |
| 8 board FFT(base) | 4 |
| 8 board FFT(M-e) | 1 |

and two lanes. There was also no clear difference between the link aggregation and the slot distribution.

5.3.2 Multi-Ejection Switch

Table 2 shows the resource utilization of the shell in XCKU115 when using our M-e switch. The resource utilization of this switch increased by only several percentages compared with that of the base switch.

Table 6 shows the reduction in the number of slots with the M-e switch. The number of slots was decreased because of the mitigation of the ejection bottleneck. The number of slots also changed depending on the topology and communication pattern. For example, the number of slots with the CG method on a 4-board ring topology is two, while that on a 4-board FC topology is one.

Figure 14 and 15 show graphs of the execution times of the CG method with the $384 \times 384$ matrix and with the $2048 \times 2048$ matrix, respectively. The communication time decreased because the number of slots decreased. We can confirm that the communication time when using our M-e switch is dependent on topologies. The communication time in Fig. 15 (d) decreased to 0.77 times that in Fig. 15 (c). This improvement is not much affected by the number of hops because the communication time in Fig. 15 (b) is 0.96 times that in Fig. 15 (a).

Figure 16 shows the graph of the execution times of FFT. For comparison, the execution time of FFT on a CPU (8 threads) was 1.3 msec. The communication time of FFT on 8 boards with our M-e switch and multiple lanes (Fig. 16 (f)) decreased to 0.13 times that with the base switch (Fig. 16 (d)). Although FFT with the highest communication ratio was not able to decrease the entire execution time by using multiple boards when a lane was used, 0.50 decrease in the execution time was achieved by using 8 boards when four lanes and our M-e switch were used compared with on a board.

Table 5 shows the power consumption of the CG method with the $384 \times 384$ matrix when using the M-e switches. There was a small difference in the power consumption of I/O between the base switch and M-e switch, while the power consumption of the FPGA core slightly increased by using our M-e switch.
6. Related Work

6.1 Interconnection Network of Multi-FPGA Systems

In most multi-FPGA systems, each FPGA is tightly connected with the host, and for inter-FPGA connection, a straight-forward direct linear network suitable for stream processing is used. A well-known example of a multi-FPGA system is Microsoft’s Catapult-1/2 [20]. FPGAs are connected as a 2D 6 × 8 torus [20]. Each FPGA is connected to its own host. Their main targets are the Bing search engine and AI applications in which the streaming data are transferred between FPGA boards in the order of connection. Some FPGA supercomputers use a PCIe network for interconnection between FPGAs and GPUs [21], [22]. High-speed serial links of more than 40 Gbps are also used [2]–[4] between FPGA boards. Although a large bandwidth is provided, they required expensive optical cables, interface chips, and high-level board-implementation techniques. The goal with interconnection architectures differs from ours which uses many mid-range-speed (9.9 Gbps) links bundled into economical electric cables.

6.2 Multiple Lanes for Circuit-Switched Networks

Link aggregation is a common technique for enhancing the performance of the interconnection network of supercomputers, node of which provides multiple network interface cards [6]. Remote DMA techniques are applied to use multiple links efficiently. This is completely different from the interconnection network used in FiC as follows. (1) Each lane can be directly connected to the user-designed hard-wired logic after transferring through SERDES in multi-FPGA systems. (2) Unlike the interconnection network in supercomputers, FiC uses STDM switches. (3) Four lanes are provided for using the ecumenical Xilinx Aurora IP and FireFly cable, while the degree of link aggregation is mostly two even in supercomputers that use individual network interface cards.

Similar approaches as FiC have been used for NoCs. Lusala and Legat [23] proposed a circuit-switched router for NoCs by combining spatial-division multiplexing (SDM) and TDM. Although SDM is similar to the link aggregation and the slot distribution, they mainly focused on the increasing paths in the circuit-switched network. Since SDM is more flexible available compared with FiC, the trade-off is different. Liu et al. [24] proposed and compared circuit-switched NoCs that can be divided into sub-channels or sub-networks. A sub-network is same as the link aggregation in FiC, while a sub-channel enables the communication between lanes as shown in Fig. 17. Since four lanes are connected to the same board in FiC, the sub-channel is not efficient considering the increase in resources. The trade-off is also different from the NoC they used.

6.3 Multi-Injection/Ejection Router

Multi-injection/ejection routers have been investigated for mitigating the bottleneck of an interconnection network. Lin and Shi [7] proposed a non-contention ejection architecture for NoCs. Contention is avoided by allocating a dedicated channel for every packet, which can be ejected simultaneously. The proposed architecture significantly improves communication performance with a reasonable area increase. Although the basic concept is the same as ours, they focused on the packet-switched network for NoCs.

Sheng et al. [25] proposed a router for FPGA clusters, which are similar to FiC. The router has multiple ports for local injection/ejection. However, it uses a packet-switched router, which is different from an STDM switch of FiC.

7. Conclusion

We investigated techniques for enhancing the interconnection performance of a multi-FPGA system, FiC. Unlike switching fabrics for NoCs or parallel machines, economical multi-FPGA systems use Xilinx Aurora IPs and FireFly cables with multiple lanes, which work asynchronously. We adopted the link aggregation and the slot distribution for using multiple lanes and clarified the characteristics of them. We also proposed a multi-ejection STDM switch for mitigating the bottleneck in ejection ports. We evaluated various combinations of the techniques by using three practical applications on a real FiC prototype with 24 boards.

When the number of slots is large and transferred data size is small, the slot distribution was sometimes more effective, while the link aggregation was superior for most other cases. Our multi-ejection STDM switch mitigated the bottleneck in ejection ports and successfully reduced the number of time slots. By combining the link aggregation and the multi-ejection STDM switch, the communication time decreased to 0.13 times that with only few additional resources in the best case. Although FFT with the highest communication ratio was not able to decrease the entire execution time by using multiple boards when a lane was used, 0.50 decrease in the execution time was achieved by using 8 boards when four lanes and our M-e switch were used compared with on a board.

When the link aggregation and multi-ejection STDM switch are used, the challenge to the programmer is describing the HLS application so as not to degrade the large bandwidth of switches. All application programs we used in this
study were manually optimized, and our next challenge is developing automatic design tools to make the best use of the wide bandwidth of the interconnection in multi-FPGA systems.

Acknowledgments
This work was supported by JST CREST Grant Number JP-MJCR19K1, Japan.

References

[1] K. Hironaka, B.A. Akram, and H. Amano, “Multi-FPGA Management on Flow-in-Cloud Prototype System,” 2019 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp.443–448, 2019.

[2] T. Ueno, T. Miyajima, A. Mondigo, and K. Sano, “Hybrid Network Utilization for Efficient Communication in a Tightly Coupled FPGA Cluster,” 2019 International Conference on Field-Programmable Technology (ICFPT), pp.363–366, 2019.

[3] N. Fujita, R. Kobayashi, Y. Yamaguchi, and T. Boku, “Parallel Processing on FPGA Combining Computation and Communication in OpenCL Programming,” 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp.479–488, 2019.

[4] N. Fujita, R. Kobayashi, Y. Yamaguchi, T. Ueno, K. Sano, and T. Boku, “Performance Evaluation of Pipelined Communication Combined with Computation in OpenCL Programming on FPGA,” 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp.450–459, 2020.

[5] Y. Hu and M. Koibuchi, “Optimizing Slot Utilization and Network Topology for Communication Pattern on Circuit-Switched Parallel Computing Systems,” IEICE Transactions on Information and Systems, vol.E102-D, no.2, pp.247–260, 2019.

[6] T. Watanabe, M. Nakao, T. Hiroyasu, T. Otsuka, and M. Koibuchi, “Impact of topology and link aggregation on a PC cluster with Ethernet,” Int. Conf. Cluster Comput., pp.280–285, Sept. 2008.

[7] S. Lim and J. Shi, “Improving NoC Performance by Non-contention Ejection Architecture,” 2010 IEEE/ACIS 9th International Conference on Computer and Information Science, pp.555–558, 2010.

[8] K. Ito, K. Iizuka, K. Hironaka, Y. Hu, M. Koibuchi, and H. Amano, “Implementing a Multi-ejection Switch and Making the Use of Multiple Lanes in a Circuit-switched Multi-FPGA System,” 2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW), pp.211–217, 2020.

[9] “IEEE Standard for Information Technology - Local and Metropolitan Area Networks - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications-Aggregation of Multiple Link Segments,” IEEE Std 802.3ad-2000, pp.1–184, June 2000.

[10] K. Hironaka, K. Iizuka, A. Ben Ahmed, M.M.I. Ullah, Y. Yamauchi, Y. Sun, M. Yamakura, A. Hiruma, and H. Amano, “Demonstration of Flow-in-Cloud: A Multi-FPGA System,” 2019 29th International Conference on Field Programmable Logic and Applications (FPL), pp.417–418, 2019.

[11] Y. Yamauchi, A.B. Ahmed, K. Hironaka, K. Iizuka, and H. Amano, “Horizontal division of deep learning applications with all-to-all communication on a multi-FPGA system,” 2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW), pp.277–281, 2020.

[12] Y. Sun and H. Amano, “FiC-RNN: A Multi-FPGA Acceleration Framework for Deep Recurrent Neural Networks,” IEICE Transactions on Information and Systems, vol.E103-D, no.12, pp.2457–2462, 2020.

[13] M.M.I. Ullah, A. Ben Ahmed, and H. Amano, “Implementation of FM-Index Based Pattern Search on a Multi-FPGA System,” Applied Reconfigurable Computing. Architectures, Tools, and Applications, ed. F. Rincón, J. Barba, H.K.H. So, P. Diniz, and J. Caba, Cham, pp.376–391, Springer International Publishing, 2020.

[14] M. Yamakura, K. Hironaka, K. Azegami, K. Musha, and H. Amano, “The Evaluation of Partial Reconfiguration for a Multi-Board FPGA System FiCSW,” Proceedings of the 10th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies, HEART 2019, New York, NY, USA, Association for Computing Machinery, 2019.

[15] R. Stefan and K. Goossens, “Multi-path routing in time-division-multiplexed networks on chip,” 2009 17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC), pp.109–114, 2009.

[16] M. Schoberl, F. Brandner, J. Sparso, and E. Kasapaki, “A statically scheduled time-division-multiplexed network-on-chip for real-time systems,” 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip, pp.152–160, 2012.

[17] K. Azegami, K. Musha, K. Hironaka, A. Ben Ahmed, M. Koibuch, Y. Hu, and H. Amano, “A STDM (Static Time Division Multiplexing) Switch on a Multi-FPGA System,” 2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), pp.326–333, Oct. 2019.

[18] Y. LeCun, P. Haffner, L. Bottou, and Y. Benio, “Object Recognition with Gradient-Based Learning,” http://yann.lecun.com/exdb/publ/pdf/lecun-99.pdf

[19] R. Kastner, J. Matai, and S. Neuendorffer, “Parallel Programming for FPGAs,” CoRR, vol.1805.03648, 2018.

[20] A. Putnam, A.M. Caulfield, E.S. Chung, D. Chiuo, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G.P. Gopal, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hornati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P.Y. Xiao, and D. Burger, “A reconfigurable fabric for accelerating large-scale datacenter services,” 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), pp.13–24, June 2014.

[21] I. Firmansyah, Y. Yamaguchi, and T. Boku, “Performance evaluation of Stratix V DE5-Net FPGA board for high performance computing,” 2016 International Conference on Computer, Control, Informatics and Its Applications (IC3INA), pp.23–27, 2016.

[22] R. Kobayashi, N. Fujita, Y. Yamaguchi, A. Nakamichi, and T. Boku, “GPU-FPGA Heterogeneous Computing with OpenCL-Enabled Direct Memory Access,” 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp.489–498, 2019.

[23] A.K. Lusala and J.-D. Legat, “A SDM-TDM based circuit-switched router for on-chip networks,” 6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC), pp.1–8, June 2011.

[24] S. Liu, A. Jantsch, and Z. Lu, “MultiCS: Circuit-switched NoC with Multiple sub-networks and sub-channels,” Journal of Systems Architecture, vol.61, no.9, pp.423–434, 2015.

[25] J. Sheng, C. Yang, T. Wang, and M. Herbozt, “High Performance Dynamic Communication on Reconfigurable Clusters,” 2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp.219–219, 2018.
Kohei Ito received the B.E. degree from Keio University, Japan, in 2020. He is currently a master student in the Department of Information and Computer Science, Keio University. His research interest is reconfigurable architectures.

Kensuke Iizuka received the B.E. degree from Keio University, Japan, in 2018 and M.E. degree from Keio University, Japan, in 2020. He is currently a Ph.D. student in the Department of Information and Computer Science, Keio University. His research interests are Deep learning accelerators and reconfigurable architectures.

Kazuei Hironaka received the B.E. degree from Tokai University, Japan, in 2010 and M.E. degree from Keio University, Japan, in 2012. He is currently a Ph.D. student in the Department of Information and Computer Science, Keio University, and also joined in Central Research Laboratory of Hitachi, Ltd., since 2012. His research interests are reconfigurable architectures and data storage systems.

Yao Hu received the M.S. degree from Beijing University of Posts and Telecommunications, China, in 2009, and received the Ph.D. degree from the Department of Computer Science and Engineering, Waseda University, Tokyo, Japan, in 2015. He is currently working as a project researcher in the National Institute of Informatics, Tokyo, Japan. His main research interests include the area of high-performance computing.

Michihiro Koibuchi received the B.E., M.E. and Ph.D. degrees from Keio University, Yokohama, Japan, in 2000, 2002 and 2003, respectively. Currently, he is an associate professor in the Information Systems Architecture Research Division, National Institute of Informatics and the Graduate University of Advanced Studies, Tokyo, Japan. His research interests include the areas of high-performance computing and interconnection networks. He is a senior member of the IEICE, IEEE and IPSJ.

Hideharu Amano received Ph.D degree from the Department of Electronic Engineering, Keio University, Japan in 1986. He is currently a professor in the Department of Information and Computer Science, Keio University. His research interests include the area of parallel architectures and reconfigurable systems.