OPTWEB: A Lightweight Fully Connected Inter-FPGA Network for Efficient Collectives

Kenji Mizutani, Hiroshi Yamaguchi, Yutaka Urino, and Michihiro Koibuchi

Abstract—Modern FPGA accelerators can be equipped with many high-bandwidth network I/Os, e.g., 64 x 50 Gbps, enabled by onboard optics or co-packaged optics. Some dozens of tightly coupled FPGA accelerators form an emerging computing platform for distributed data processing. However, a conventional indirect packet network using Ethernet's Intellectual Properties imposes an unacceptably large amount of the logic for handling such high-bandwidth interconnects on an FPGA. Besides the indirect network, another approach builds a direct packet network. Existing direct inter-FPGA networks have a low-radix network topology, e.g., 2-D torus. However, the low-radix network has the disadvantage of a large diameter and large average shortest path length that increases the latency of collectives. To mitigate both problems, we propose a lightweight, fully connected inter-FPGA network called OPTWEB for efficient collectives. Since all end-to-end separate communication paths are statically established using onboard optics, raw block data can be transferred with simple link-level synchronization. Once each source FPGA assigns a communication stream to a path by its internal switch logic between memory-mapped and stream interfaces for remote direct memory access (RDMA), a one-hop transfer is provided. Since each FPGA performs input/output of the remote memory access between all FPGAs simultaneously, multiple RDMAs efficiently form collectives. The OPTWEB network provides 0.71-μsec start-up latency of collectives among multiple Intel Stratix 10 MX FPGA cards with onboard optics. The OPTWEB network consumes 31.4 and 57.7 percent of adaptive logic modules for aggregate 400-Gbps and 800-Gbps interconnects on a custom Stratix 10 MX 2100 FPGA, respectively. The OPTWEB network reduces by 40 percent the cost compared to a conventional packet network.

Index Terms—Interconnection architecture, network topology, circuit-switching networks, fiber optics, FPGAs, collectives

1 Introduction

Parallel data processing using multiple field-programmable gate array (FPGA) accelerators with high-bandwidth memory, e.g., HBM2, and high-bandwidth network, becomes a way to compute emerging parallel applications including deep neural networks [1] or columnar database [2]. Increasing the parallelism of the dedicated circuits in the FPGA enables high-throughput data processing, e.g., sorting operation [3], [4], [5]. The more significant number of high-bandwidth memory I/Os would result in the higher-throughput data processing.

Interconnection networks are the heart of such FPGA accelerator systems. However, the communication I/O start-up latency bottlenecks appeared on a traditional FPGA-accelerator system. In the traditional system, an FPGA accelerator is attached to a host compute node via PCIe, OpenCAPI, or Cache Coherent Interconnect for Acc. (CCIX). Each communication stream to a remote FPGA is transferred via a unified off-chip interconnection network shared by inter-processor or storage communication, e.g., InfiniBand or Ethernet. Each communication stream is thus packed into multiple packets that have a complicated structure for the abstraction. Since both packetization latency overhead and PCIe handling overhead are significant, the inter-FPGA communication start-up latency typically reaches tens of μsec order even on a small traditional system [29], [30].

Direct interconnection networks are thus attempted on an FPGA-accelerator system, e.g., Project Catapult v1 [6], Flow in Cloud (FiC) project [8], EuroEXA project [9], Novo-G# cluster [10], [30], and Cygnus supercomputer [12]. Fortunately, some FPGA cards have network ports for fast serial communications [13]. For example, since a NetFPGA SUME card provides multiple 100-Gbps network ports, an application user can design a direct inter-FPGA network. In an FPGA card, Ethernet networks can be constructed by their Interectual Propertys (IPs). Similarly, a flow control IP can be linked to data processing in point-to-point communication [14], [15]. However, the network's hardware resources including flow control IP are costly on an FPGA as a link bandwidth becomes large.

Typical data communications between FPGAs have to be packetized in multiple layers. The data-link layer divides raw data into multiple flits with control information. The network layer then adds the destination and source information to a packet for communication among multiple devices. Packet processing consumes a large amount of logic to extract the following information from the data lines. In the data-link layer, backpressure is needed to temporarily stop data from the source to avoid channel buffer overflow [15]. The backpressure usually requires enough buffers to store packets by the high-speed static random-access memory (SRAM), though

- *Kenji Mizutani and Yutaka Urino are with the Photonics Electronics Technology Research Association, Tsukuba 305-8569, Japan. E-mail: k-mizutani@nec.com, y-urino@petra-jp.org.*
- *Hiroshi Yamaguchi is with the Photonics Electronics Technology Research Association, Tokyo 153-8505, Japan. E-mail: hyamaguch_cw@nec.com.*
- *Michihiro Koibuchi is with the National Institute of Informatics, Tokyo 101-8430, Japan. E-mail: koibuchi@nii.ac.jp.*

Manuscript received 14 Aug. 2020; revised 12 Mar. 2021; accepted 21 Mar. 2021. Date of publication 24 Mar. 2021; date of current version 17 May 2021. (Corresponding author: Kenji Mizutani.)
Recommended for acceptance by L. Chen and Z. Lu.
Digital Object Identifier no. 10.1109/TC.2021.3068715

SRAM is costly. It is reported in [14] that building a reliable packet network using IPs of Ethernet at 40 Gbps requires more than 10 percent of resources in both adaptive logic modules (ALMs) and M20K SRAM blocks (M20Ks) in an Intel Arria 10 FPGA [14]. These amounts correspond to 6 percent of ALMs and 4 percent of M20Ks in Intel's Stratix10 MX2100, which is the largest of the FPGA with HBM2. A similar hardware-resource problem arises in an accelerator of MPI collectives using a custom packet technology [16], [17]. For 40-Gbps inter-FPGA communication, 38 percent of the Stratix V FPGA's ALMs are consumed. This amount corresponds to 14 percent of ALMs in Intel's Stratix10 MX2100. More than 20-times improvement, 800 Gbps, of the bandwidth is required to achieve HBM2 equivalent network bandwidth. However, achieving aggregate 800-Gbps interconnects using the commodity IPs is not implementable in terms of the FPGAs' hardware amount because of the required hardware resource increases in proportion to the bandwidth.

Another problem is the low-radix network topology, e.g., 2-D torus, in existing direct inter-FPGA networks. Existing vast network I/O pluggable ports limit the network radix in an FPGA. The low-radix network has the disadvantage of a large diameter and large average shortest path length. It is known that the throughput and latency performance of a low-radix network is inferior to a high-radix network.

To mitigate the large hardware-resources and the large path-hop problems, we unburden a complicated packet network in this study. We build an FPGA-to-FPGA synchronization circuit using a simple single-cycle block signal. Since the block signal assumes to connect only two devices, a fully connected network topology is designed with advanced optical technology. Recently, high-density many I/Os are enabled by onboard optics or co-packaged optics for hyperscale high-radix top-of-rack switches [18]. We are developing silicon photonics-based onboard optics, which can provide many optical I/Os in FPGAs. e.g., 32 or 64 (32×25 Gbps or 64×25 Gbps) [19]. We design and implement a fully connected network topology for an inter-FPGA network. It supports up to 64 FPGAs that would be the maximum number of optical I/Os at a modern, cost-effective FPGA card due to its areal density. Optical cables are thin to wire than electrical cables and can be used to prepare fully connected wiring sheets for up to dozens of endpoints even though the volume of the cable increases by N^2-N. In the case of a more significant number of FPGAs, we apply wavelength division multiplexing (WDM) to an inter-FPGA network to logically form a fully connected network topology with a shorter aggregate cable length [20], [21].

At the communication operation layer, we provide a point-to-point, i.e., send/recv, sendrecv and six collectives, based on a Single Program Multiple Data (SPMD) model with message passing. Collectives are operations that involve communication of all FPGAs in a group. We provide barrier, scatter, gather, broadcast, alltoall, and allgather as collectives.

The above inter-FPGA network architecture, OPTWEB, is so named because many optical links like spider threads tightly connect all FPGAs. The contribution of this study is summarized as follows.

1. OPTWEB networks provide separate end-to-end paths on a fully connected network topology for simplifying the network structure in each FPGA. It enables collectives using one-step one-hop multicast (Section 3).
2. An OPTWEB network using the custom Intel Stratix 10 MX FPGA cards achieves 0.71-μsec start-up latency of typical collectives. The collectives fully utilize the maximum effective bandwidth of each link (Section 4).
3. We discuss and quantify the OPTWEB network in terms of cost, application-performance estimation, and scalability (Section 5).

The remainder of this paper is organized as follows. Section 2 presents background information on the OPTWEB networks. Section 6 makes our conclusions and future work.

2 Background Information

2.1 Onboard Optics and Co-Packaged Optics (CPO)

Optical technology integration to chip package, co-packaged optics (CPO), is a promising technology for switch ASICs and commodity FPGA/CPU/GPUs. Hyperscale datacenters highly demand a top-of-rack high-radix high-throughput single-chip switch. Broadcom releases the design of Tomahawk 3 Ethernet switch ASIC (12.8 Tbps) in 2018, Tomahawk 4 (25.6 Tbps) in 2019. It is expected that a switch ASIC will reach 51.2 Tbps in the first half of the 2020s. In current Ethernet switches, electric SERDES conversion consumes significant power, and the broad area of the aggregate I/O pluggable ports increases the onboard wire length. To mitigate both problems, the optical technology should be tightly coupled with a switch ASIC. In this context, onboard optics are needed to support up to 40 Tbps switch ASIC [22], and CPO commercially becomes mature before 51.2-Tbps switch ASIC is deployed.

The similar integration occurs in FPGAs. DARPA PIPES project is attempting to enable aggregate signaling rates to 100 Tbps by CPO [23]. In Japan, we have developed onboard optics using Optical I/O Core for 100-Gbps transceiver (4×25 Gbps), and we have integrated these optics into a custom Intel Stratix 10 FPGA card [19]. Optical I/O Core is commercially available from AIO CORE Corporation [24]. In this study, we use our onboard optics for the inter-FPGA communication.

2.2 Target FPGA

This study targets an FPGA that is equipped with high-bandwidth memory and high-bandwidth optical transceivers for distributed data processing. Leading FPGA manufacturers, Intel and Xilinx, have FPGAs with HBM2 as the wide-band memory. The Stratix10 MX 2100 has the most substantial logic of an Intel's FPGA with HBM2. In this study, we design and implement the OPTWEB network on Intel Stratix 10 MX FPGA 2100 cards with our onboard optics.

The maximum number of interconnected FPGAs is currently 64 by our technology-driven design (see Section 5.3) of the OPTWEB network. The existing direct inter-FPGA network usually targets up to dozens of FPGAs (see Section 2.3.2), and a more significant FPGAs are communicated via an inter-CPU network via PCIe [12]. Although we consider that 64 FPGAs are reasonable when an FPGA accelerator system is stored on a chassis, we illustrate the extension to connect a larger number of FPGAs (see Section 5.3).

2.3 Existing Inter-FPGA Networks

2.3.1 Data-Link Layer

An Ethernet frame has two types of control data: one for network and one for link. The former includes the information of destination, source, and data size. The latter includes a preamble to detect the beginning of the packet and an inter-packet gap (IPG) to provide appropriate spacing between packets. These are placed at the beginning of the frame. The frame check sequence (FCS) is added to the end of the frame, and is used for verification with parity-check or cyclic redundancy codes.

Packets could be lost, and out-of-order delivery could occur on an inter-FPGA Ethernet implementation. This becomes a problem when dealing with successive data in direct memory access (DMA). The other implementation issue for distributed data processing is a channel buffer overflow [14], [15]. To avoid a channel buffer overflow, a switching technique should be carefully designed. Cut-through switching that can send the header before receiving the tail is commonly used in high-performance interconnection networks. The cut-through switching requires particular hardware logic to handle a credit-based flow signal between two endpoints of the link for avoiding channel overflow [14], [15]. This is usually achieved by providing dedicated control bits in the payload to achieve synchronization between endpoints. This requires a mechanism to extract the control bits flowing from the data lanes in the FPGA.

2.3.2 Network Layer

The inter-FPGA networks can be built using IPs, e.g., Ethernet PHYs. However, the amount of logic for supporting a commodity packet network becomes a design bottleneck on an FPGA as the network bandwidth becomes large, as described in Section 1. Building a packet network using Ethernet IPs with the flow control would not be feasible with the high-bandwidth I/Os, e.g., 800 Gbps enabled by onboard optics or CPO on an FPGA.

Another approach to designing an inter-FPGA network is to include a lightweight router without Ethernet IPs on an FPGA. The Catapult-v1 node uses SerialLite 3 (SL3) with optical transmission technology, e.g., QSFP 40 Gbps, for the communication. The inter-FPGA network follows a packet structure for a low-radix network topology, 6×82-D torus [6]. Cygnus supercomputer ranked in 264 on top500 as of June 2019 takes an 8×82-D torus for the inter-FPGA network using SL3. Novo-G# also takes a torus in a typical configuration. The torus network requires a router that operates input buffering, routing computation, virtual-channel allocation, switch arbitration, and flit crossbar transfer on each FPGA. A router consumes a vast hardware resource of each FPGA.

Another problem is the path hops on their low-radix torus networks. Path hops directly affect the communication latency. Indeed, Cygnus and Catapult-v1 network consume 200 ns per hop [12] and 400 ns per hop [6], respectively. The low-radix networks take a large diameter and a large average shortest path length (ASPL), and generate large total path hops of packets on a multicast operation [7].

A unique inter-FPGA network relies on circuit switching. Static time-division multiplexing in Flow-in-Cloud (FiC) project [8] and the optical circuit switching in Noctua [11] implement circuit switching. In a Noctua supercomputer, optical circuit switching connects the 40-Gbps QSFP port of each FPGA. It establishes an end-to-end path before starting the communication using S320 optical circuit switches. Both circuit-switching implementations simplify the packet network structure. They are efficient when the number of communication source-and-destination pairs is small and predictable. However, we consider that such a case is not typical in the inter-FPGA network for distributed data processing.

2.3.3 Collective

For distributed data processing, the support of collectives is essential. In [26], a PC cluster consisting of 128 nodes with 1024 GPUs completed a training phase of ImageNet in 15 minutes. It is reported that 20 percent of the execution time is consumed for communication at 128 nodes, and its ratio would be more significant as the scale becomes large. The importance of collectives, e.g., allreduce, is also illustrated for deep learning training by [27].

GPU-to-GPU Remote DMA (RDMA) is often used for improving the performance of AI learning through distributed data processing. Since low start-up latency communication is required to accelerate learning, InfiniBand and NVLINK are frequently used. However, with these technologies, it is reported that the start-up latency is about 10 μsec even for point-to-point communication. The start-up latency of collectives also increases with a large number of connected nodes [28].

The collectives between FPGAs are also being considered. It is reported that speeding up a gather operation is difficult on existing inter-FPGA networks. Surprisingly, it is reported that inter-CPU communication via PCIe is faster than an inter-FPGA communication because of the bottleneck of the data input part in an inter-FPGA 2-D torus network [29]. It is also reported that an inter-FPGA 2-D torus network consumes several tens of μsecs for MPI multicast and reduction operations [30]. The delay usually becomes higher in the collective than that in point-to-point communication.

Unicast-, tree- and path-based multicast techniques are typical implementation methods for multicast in interconnection networks [41]. Unicast-based multicast consists of a large number of unicasts to deliver a message to all the destinations. In the tree-based multicast, a spanning tree whose root is a source node is built. Each message is then forwarded along the spanning tree. The tree-based multicast would minimize the total path hops of the messages [31]. A path-based multicast sends data along a path that includes all destinations, and thus requires an efficient multicast-path search, e.g., Hamiltonian cycle for multicast. It minimizes the number of messages on a multicast. The tree- and path-based methods require a particular function to forward the message on FPGAs, and it would not be lightweight. In this study, we implement the unicast-based multicast for collectives on the inter-FPGA network.

3 OPTWEB Network Architecture

We propose a network architecture called OPTWEB. A typical configuration of conventional and OPTWEB networks is compared in Figs. 1a and 1b. Each key element of the OPTWEB network approach is explained hereafter.

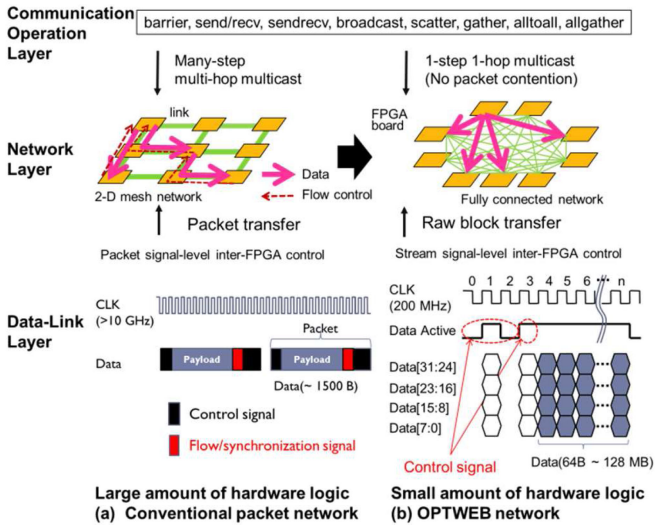

Fig. 1. Outline of network design.

3.1 Network Design

3.1.1 Data-Link Layer

Synchronization is necessary for communication between FPGAs. We introduce simple inter-FPGA communication synchronization in each link for small hardware resources. The simple inter-FPGA communication synchronization distinguishes data and control signals. As shown in Fig. 1b, the link connections have a control lane (Data Active) to determine whether the data lane is activated or not every clock cycle. The control signal assumption is conventional since the equivalent of the start of packet (SOP), or the end of packet (EOP) control signal is provided in Intel's FPGAs, as well as Xilinx FPGAs.

A source FPGA sends data (Clk 3 in Fig. 1b) after it sends a single-cycle active signal (Clks 0-2). When a single-cycle active signal is received at a destination FPGA, it is treated as a notification from the other FPGA. When two-cycle or longer active signals are received at the destination FPGA, the first clock data (Clk 3) is removed, and the rest data (Clks 4∼n) is passed to a DMA controller (DMAC). Unlike a packet structure, we do not have to send the destination and source FPGA identification because we provide a separate end-to-end path (see Section 3.1.2).

We provide no link-level backpressure flow control. Besides the link-level design, the communication operation layer handles communication instructions for avoiding a channel buffer overflow (see Section 3.3).

3.1.2 Network Layer

Network topology would ideally have a low diameter and low average shortest path length. In this context, we take an ideal fully connected network topology by using up to 64 I/Os per FPGA enabled by onboard optics or CPO. The physical implementation of a fully connected network topology depends on the number of FPGAs, as shown in Fig. 2. Optical fiber can transmit broadband signals with low loss and can be easily connected among FPGA cards. The number of optical fibers becomes $N(N-1)$, where the number of FPGAs is N, if direct cabling is taken as shown in Fig. 2a. The direct cabling would support up to a dozen of FPGAs in terms of the cable density.

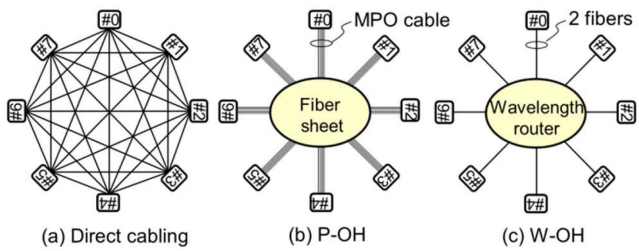

Fig. 2. Three physical implementations of a fully connected network.

When connecting a more significant number of FPGAs, we introduce the concept of Optical Hub in Figs. 2b and 2c [20]. The function of the Optical Hub is an optical shuffle circuit with passive optical components that physically connect the FPGAs in a star structure [33]. The Optical Hub does not provide the arbitration function. It can be a fiber sheet (Fig. 2b) or wavelength router (Fig. 2c). The former, called Parallel Optical Hub (P-OH), is a fiber sheet that realizes the fully connected network topology with high-density fiber optic wires. Connecting nodes and Optical Hub with N 2N-core-MPO (Multi-fiber Push On) /MPO ribbon fiber logically allows for a fully connected network topology. The latter, called Wavelength Router Optical Hub (W-OH), is the cyclic wavelength division multiplexing (WDM) technologies that have been developed in the telecommunications field [34]. With WDM technology, two single-mode fibers connect the node and Optical Hub for bidirectional communication [20], [21]. In this case, only 2N optical fibers or N 2-core-cables are needed between Optical Hub and all FPGAs. Its total length becomes short compared to the case of Fig. 2a.

Our fully connected network topology provides separate end-to-end paths that behave like circuits using the same number of I/Os as the number of FPGAs. A source FPGA thus sends different messages to all the FPGAs simultaneously. Notice that the routing function is simple for any fully connected network implementations because each path can correspond to a cable or a wavelength.

3.1.3 Communication Operation Layer

The fully connected network topology provides a one-hop communication to all nodes simultaneously. Since we take a unicast-based multicast as described in Section 2, we can provide a one-step one-hop multicast. It is expected that the communication latency of the multicast is close to that in point-to-point communication.

We assume the Single Program Multiple Data (SPMD) model with message passing. We support eight point-to-point and collective operations, i.e., barrier, send/recv, sendrecv, scatter, gather, broadcast, alltoall, and allgather. For each inter-FPGA communication, the starting address of the sent data, the starting address of the received data, the data size, the source and the destination nodes, and the group of nodes for collectives are specified by a communication instruction at FPGAs.

3.2 Network Interface Design

Fig. 3 shows a block diagram of the network interface on an FPGA. The HBM2 is located on the silicon interposer beside

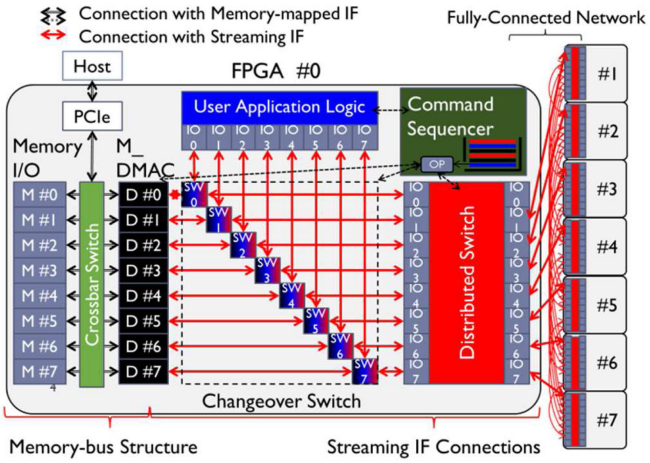

Fig. 3. Block diagram of the network-interface design in the FPGA.

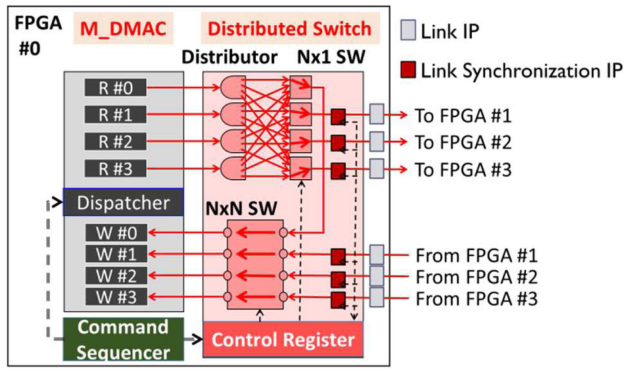

Fig. 4. Block diagram of the distributed switch and M_DMAC.

an FPGA in the FPGA package. The HBM2 has several memory I/Os (M#*i*, $i = 0, \ldots, N\text{-}1$). Each memory I/O is accessed via a memory-mapped interface, and connected to the M_DMAC, which consists of multiple DMACs (D#*i*, $i = 0, \ldots, N\text{-}1$) in order to eliminate the influence of the host by offloading of computation and communication. The DMACs are prepared for the number of connected FPGAs. Then, basically, the D#*n* in the FPGA#*m* is connected to the D#*m* in the FPGA#*n* during collectives. However, in communications where data is copied and distributed, such as broadcast and allgather, any D#i on source FPGA can be used. One DMAC is selected, and then the distributed switch copies the data and then distributes it. These DMACs connect to the distributed switch by stream interfaces (IFs) without address lines. This is because each FPGA can capture address information from communication instructions and because the address space in each FPGA corresponds to one to one. It is connected to the user application logic and the distributed switch through the changeover switch. We can switch the connection of M_DMAC to/from user application logic and that to/from the remote FPGA by using the changeover switch. The command sequencer manages this operation. An FPGA is connected to all the other FPGAs via the distributed switch. Each key element of the network-interface design is explained hereafter.

3.2.1 Distributed Switch

Fig. 4 shows a detailed block diagram of the distributed switch and M_DMAC when four FPGAs are interconnected (we omit the changeover switch). Each D#*i* consists of a reading block (R#*i*) that retrieves data from the memory and a writing block (W#*i*) that writes data to the memory. At the memory side of the distributed switch, $2N$ stream IFs are prepared and connected to these R#*i* and W#*i* with stream IFs where N is the number of FPGAs. Similarly, at the network side of the distributed switch, N stream IFs are prepared for the data transmission, and the remaining N stream IFs are prepared for the data reception. Then, Link IP is placed between the distributed switches of different FPGAs. SL3 or Interlaken is used as link IP. These link IPs allow for signal transmission of several 10 Gbps per line, and enable higher bandwidth communication using multiple signal qualities. These rates are enough for the current network design using a 100(4×25)-Gbps transceiver. The amount of hardware logic for such Link IP can be smaller than that of a conventional packet processing method because no flow control is required.

The distributed switch records the connection pattern between the network port and Memory I/Os in each collective. To support the multiple (local and remote) memory accesses simultaneously, a distributor is placed in each IF at the sending side. Then, an $N\times1$ switch is used in the latter stage to select multiple distributors' data. On the other hand, an $N\times N$ switch is placed at the receiving side to support any memory access pattern. These directional switches enable a direct connection between any DMACs in different FPGAs. In the local data movement within an FPGA, we set to connect an output of the $N\times1$ switch at the sending side to an input of the $N\times N$ switch at the receiving side.

3.2.2 Command Sequencer

The command sequencer is prepared to offload the distributed data processing of the FPGA from the host. The command sequencer is connected to the control register of the distributed switch, the dispatcher of the M_DMAC, and the control part of the application user logic and changeover switch. Instructions from the host to each functional part are stored in the command sequencer. Then, the command sequencer sends instructions to each functional part in turn.

Communication types are specified as instructions in an SPMD model. The starting address of the sent data, the starting address of the received data, the data size, the source and the destination nodes, and the group nodes are specified as the instruction's arguments of the communication. When RDMA is performed, the communication path is set up by specifying the dispatcher, the control register, and the changeover switch.

3.2.3 Memory-Bus Structure

The M_DMAC and memory I/Os are prepared for all FPGAs. A full crossbar switch is a common way to connect all I/Os to retrieve data from any memory space. Xilinx provides crossbar switches as a default in the FPGA with HBM2. Xilinx alternatively provides multiple 4×4 crossbar switches to reduce hardware resources. However, when multiple crossbars are interconnected for memory read/write operation, the same end-to-end bandwidth as I/O bandwidth is not guaranteed [35].

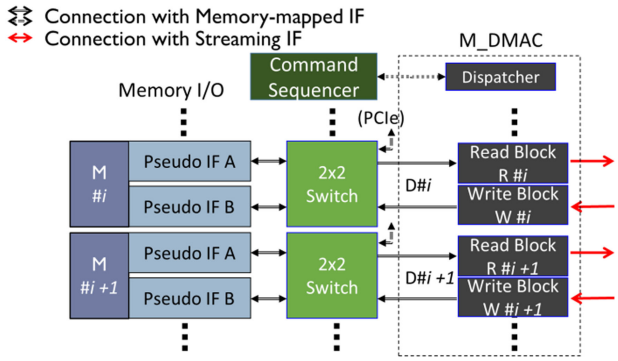

Fig. 5. Memory-bus block diagram between memory I/Os and M_DMAC.

To mitigate this bandwidth problem, we take a different implementation. Fig. 5 shows a block diagram of our memory-bus configuration between memory I/Os and M_DMAC for ensuring high read/write throughput. We prepare two independent IFs with the same memory capacity in one memory I/O. HBM2 has a feature that allows one memory I/O as two pseudo channels for enabling two separate IFs. Two IFs of the memory I/O are connected to one D#*i* and PCIe with a 2×2 crossbar switch. This configuration is considered to be the smallest configuration that can read the written data. Even in this minimal configuration, the occupation of the memory-mapped bus by the DMACs makes it possible to ensure high throughput with simultaneous read/write operations in bidirectional communication in each FPGA.

3.3 Behavior

Fig. 6 illustrates the communication flow. Since we assume the SPMD model, the source and the destination FPGAs follow the same flow. The command sequencer issues and forwards an instruction, i.e., barrier, send/recv, scatter, gather, broadcast, allltoall, or allgather, to the control register of the distributed switch and the dispatcher of the M_DMAC.

When the instruction to the distributed switch is a barrier, no local memory access occurs. Thus, it immediately sends a simple control signal, i.e., the signal at Clk 0-2 in Fig. 1b, to the other FPGAs. Each FPGA completes barrier when it receives control signals from all FPGAs.

When the other instruction is issued, we extract the type of communication, the group to be communicated, and the data size from the communication instructions. We then identify the path for data transfer by the former two parameters. If the data size exceeds the receive buffer prepared for eager communication, rendezvous communication is performed. In rendezvous, the destination FPGAs send simple control to all source FPGAs. At the source FPGA, the data is sent out from the distributed switch after confirming that all relevant control signals are received. Otherwise, eager communication is performed. The distributed switch counts the data on both the sending and receiving sides. It then notifies the command sequencer when all the data has been sent or has been received. After receiving all data, the simple control signal is sent to the source FPGA. Thus, the distributed switch counts the communication data volume to detect the completion of communication. Based on the completion of the communication, the command sequencer executes the next communication. Sequential execution of communication

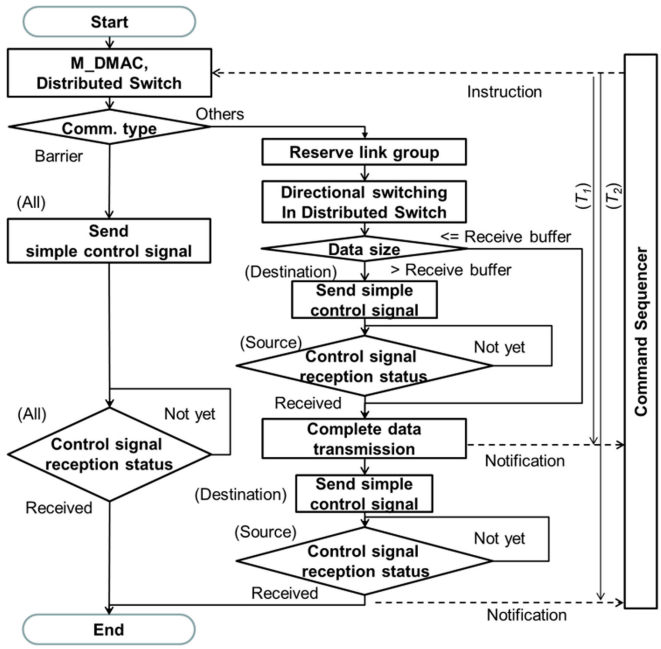

Fig. 6. Communication flow in both source and destination FPGAs.

instructions does not require no extra memory for temporary storing upcoming communication data because all the communication data is stored in the memory of FPGAs in each communication step.

An OPTWEB network achieves collision-free communications because of neither collision on a link nor collision on DMACs. We prepare the DMACs with the same number of connected FPGAs and more. Thus, no collisions occur between DMA and memory controllers by taking different memory spaces for transmitted and received data. No collisions also occur between DMACs and memory.

To achieve high bidirectional throughput, we take striping data placement on multiple memory banks. If the data address is placed consecutively in one memory space, the memory bandwidth is not enough to operate that data with the wire speed. The physical top address (*Pa*) in each striping data is computed from the relative top address (*Ra*) in a communication instruction, as shown in (1).

$$Pa\ \ (i) = \ Ra\ +\ 2K \times i,\ i\ =\ 0,\ 1,\ \cdots,\ N-1, \qquad (1)$$

where K and i is the capacity of memory in each memory IF and the identification number of connected FPGAs, respectively. The striping data placement is applied to multiple memory banks when data is sent to or received from multiple FPGAs simultaneously. This makes it possible to read and write data from multiple D#n in parallel.

The introduction of the striping data placement with multiple D#i does not increase the start-up latency. Each DMAC has the address translation mechanism based on Equation (1). By indicating the relative top address to the dispatcher of the M_DMAC, each DMAC automatically accesses the memory under the physical top address. This placement provides low-latency collectives similar to point-to-point communication, and can maximize the effective bandwidth in some collectives.

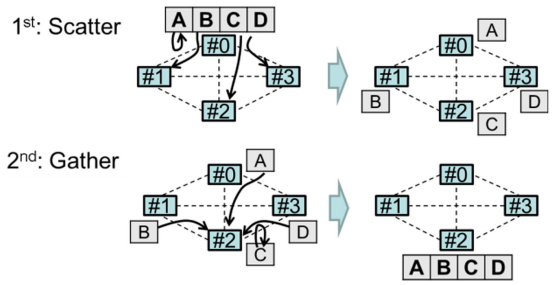

Fig. 7. Multipath routing from FPGA #0 to #2.

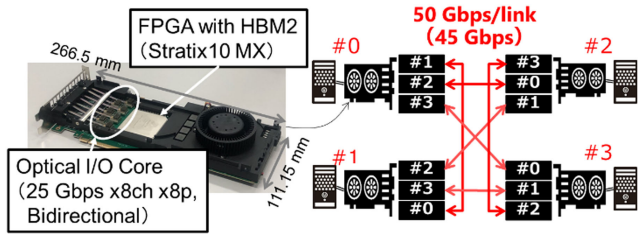

Fig. 8. Evaluation prototype system with four FPGA cards.

3.4 Multipath Routing

The main drawback of an OPTWEB network is a low bandwidth path between two FPGAs due to a fully connected network topology. For example, when an FPGA network bandwidth is 400 Gbps in total, the link bandwidth to another FPGA becomes 25 Gbps on a 16-node OPTWEB network.

To increase the communication bandwidth between two FPGAs, multiple indirect two-hop paths can be optionally taken via intermediate FPGAs in parallel, which is called multipath routing [20]. In the multipath routing, point-to-point communication is replaced with the scatter and gather collectives handled by the command sequencer, as shown in Fig. 7. Then, the command sequencer sequentially performs the two collectives.

Multiple one-hop transfers in the multipath routing use different network resources, e.g., buffers in the distributed switch and the DMAC. Each communication data is stored in the memory of a destination FPGA. Thus, the cyclic channel dependency never occurs on the multi-path routing. Besides, a maximum number of one-hop transfers is limited, e.g., up to two. Therefore, in the multi-path routing, a deadlock or a livelock does not occur.

The two collective communications are executed consecutively by the command sequencer in each FPGA. Then, a distributed routing can be implemented, as shown in the control flow in Fig. 6.

The multipath routing improves the average link utilization, especially when only a few FPGAs communicate with each other because a point-to-point communication can use all links. Since a collective consists of multiple point-to-point communications, the multipath routing can be applied for collectives. In the case of broadcast, the data is divided and scattered to each FPGA, and the divided data is shared by allgather. We hereafter call the original one-hop path the direct path to distinguish it from the multipath enabled by the multipath routing.

4 Performance Evaluation

4.1 Experimental Setup

In an OPTWEB network, the number of memory I/Os is equal to the number of FPGAs. We use a Stratix10 MX FPGA with two HBM2s with a large number of memory I/Os. These HBM2s have 16 memory I/Os (32 pseudo IFs), and a total memory capacity of 8 GiB. This FPGA has 64 GXT transceivers with up to 28.3 Gbps. We fabricated new FPGA accelerator cards, and built a prototype accelerator system as shown in Fig. 8. Each FPGA card has eight small and high-density optical transceivers (4×25- Gbps transmitting and 4×25-Gbps receiving), Optical I/O Core, developed at PETRA [24], and mass-produced by AIO Core. These optical transceivers were connected to the GXT transceivers on an FPGA as pluggable embedded optical modules, and achieve error-free transmission, i.e., the bit error rate was less than 10^{-12}, between two FPGAs without error correction. This FPGA card enables the aggregate 800-Gbps network bandwidth.

We used Intel Quartus Prime Pro version 19.4 for the synthesis and implementation of our designs, which we developed in Verilog. We used the existing memory I/Os for HBM2 and SerialLite3 (SL3) [36] to realize the configuration shown in Figs. 2 and 3. We provided an end-to-end path in which all components provide almost the same bandwidth. There were 256 data lanes in each pseudo channel of HBM2. Therefore, each path in Fig. 4 was unified to 256 lanes at 200 MHz. This makes it possible to connect at the maximum of 51.2 Gbps for each connection. The above design eliminates the need for extra frequency conversion circuitry between the IPs and can reduce hardware resource utilization. Then, we developed the distributed switch, M_DMAC, and the command sequencer. The distributed switch and M_DMAC were designed with a 50-Gbps (2×25-Gbps) for connecting up to eight FPGAs. We evaluated the communication characteristics of the four FPGAs except for the multipath routing in Section 4.3.4.

The latency through the distributed switch was about ten cycles. A 16-KiB reception buffer was provided in the middle of the data line to enable low latency communication by eager communication. An FPGA card used a physical band of bidirectional 50 Gbps by bundling two optical transceivers' maximum bandwidth of 25 Gbps. To entirely provide 50-Gbps throughput, each IP core was used. The operating frequency of the SL3 was set to 355 MHz with 64B/67B physical layer encoding, thus providing 50-Gbps unidirectional bandwidth. Error Correction Code with one-bit correction and two or more bits detection was used in SL3. Therefore, the effective bandwidth of SL3 became 45 Gbps [36].

4.2 Preliminary Results of Barrier Operation

To demonstrate RDMA on collectives, the M_DMAC and the distributed switch have eight ports to connect eight FPGAs. In the prototype accelerator system, four ports are used for four FPGAs. In an OPTWEB network, the command sequencer of the FPGA directly instructs each functional part. Therefore, we implemented a counter function in the command sequencer to measure the communication

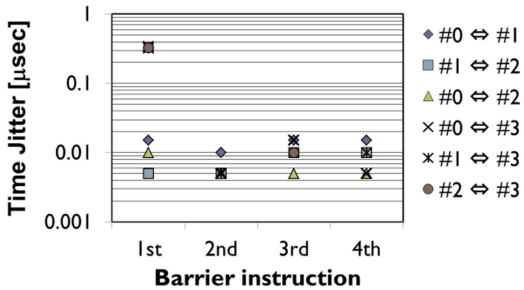

Fig. 9. Synchronization time jitters on barrier operation.

time of each communication instruction. The counter works at 200 MHz. When measuring the execution time at each communication instruction, the time jitter between FPGAs needs to be taken into account. The delay in communication is sub μsec, and the time jitter between the FPGAs should be at most a few tens of nsec.

Our simple synchronization mechanism attempts to minimize the time jitter between FPGAs. Fig. 9 shows the time jitter between FPGAs estimated from the total time when the barrier operation is executed consecutively. Initially, we set to adding 1 msec delay to #2 and 2 msec to #3 for the behavior verification. In the OPTWEB network, the first execution of the barrier operation, the slowest FPGA #3 finishes early, and the other FPGAs finish after receiving the control signal from #3. Therefore, there is a timing difference of about 0.3 μsec between the FPGAs, which is equivalent to the migration delay of the control signal. As shown in Fig. 9, the timing difference becomes less than 20 nsecs.

Then, after running the barrier twice, we start each communication operation multiple times in succession to measuring its performance. In the next subsection, we measure the average time it took to complete the transmission (T_1) and the time it took to complete the communication (T_2) in Fig. 6. The signal migration delay between FPGAs ($T_{migration}$) can be approximated by $(T_2\text{-}T_1)/2$ with minimum data transmission.

4.3 Results

4.3.1 Effective Bandwidth

Each communication was performed four times with 128 MiB and 64 B placed in each memory area and measured $T_{1,128\ MiB}$, $T_{2,128\ MiB}$, $T_{1,64\ B}$ and $T_{2,64\ B}$. The value of $T_{1,128\ MiB}$ represents the transmission delay at 128-MiB data transfer, while the value of $T_{2,64\ B}$ is the communication delay at 64-B data transfer. First, we estimated $T_{migration}$ from $T_{1,64\ B}$ and $T_{2,64\ B}$ in each collective. We then estimated the effective bandwidth in transmission from $T_{1,128\ MiB}$ and the effective bandwidth in reception from $T_{2,128\ MiB}$ - $T_{migration}$. Fig. 10 shows the effective bandwidth in each communication at 128-MiB data transfer. We confirmed that the effective link bandwidth was about 45 Gbps for all the collectives limited by SL3 [36].

In collectives, data are moved across all the relevant links in parallel. In sendrecv operation, which is bidirectional communication between two FPGAs, the bandwidth is doubled to 90 Gbps in total. In the case of scatter/gather/broadcast, the aggregate effective bandwidth per node becomes 135 Gbps, which equals to a multiplication value between

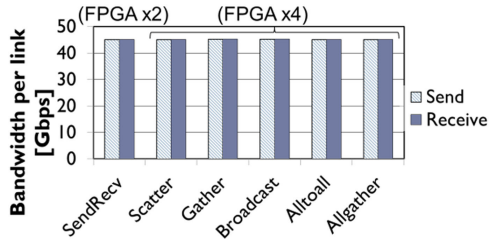

Fig. 10. Effective link bandwidth of each communication (unidirectional).

the number of links per node and the link bandwidth. The effective bandwidth of allgather/alltoall, which is bi-directional communication between all FPGAs, becomes aggregate 270 Gbps per node. Simultaneously, in several FPGAs with internal data movement, there is an additional 90 Gbps of data transfer between internal memories.

Fig. 11 shows the data size versus the effective bandwidth per link for sendrecv, alltoall and allgather operations. The horizontal axis is the data size per unidirectional link. The operation of alltoall/allgather has the same effective bandwidth as that of sendrecv. This interestingly means that the collective has the same effective bandwidth as that of point-to-point communication. The dotted line in the figure shows the theoretical values of the effective bandwidth. It is computed with the start-up latency of 0.8 μsec for the maximum effective bandwidth of 45 Gbps (the breakdown of the 0.8 μsec is given in the next subsection). Since the evaluated values are interestingly on the analytical curve, it can be said that we obtain the expected effective bandwidth.

4.3.2 Start-up Latency

The start-up latency is expressed as the sum of the configuration delay for communication instructions and the migration delay for data/control signal movement. To clarify the latency breakdown for each communication, we measured $T_{1,64B}$ as configuration delay and $T_{migration}$ as migration delay. These delays were also estimated for the barriers in the same way. Fig. 12 shows the results of start-up latency for 64-B data transfer in each communication.

In barrier operation, we obtained a configuration delay of 0.04 μsec and a migration delay of 0.36 μsec. This configuration delay is the preparation time for the distributed switch to recognize the instructions received and sends out the control signal. This indicates that about seven cycles are needed

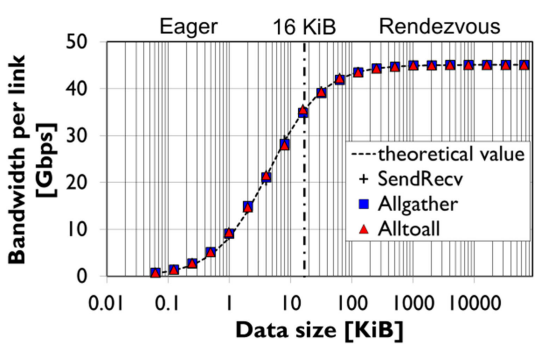

Fig. 11. Effective bandwidth of unidirectional link versus data size.

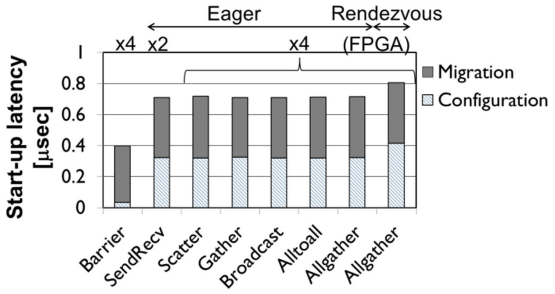

Fig. 12. Start-up latency of collectives and point-to-point communication.

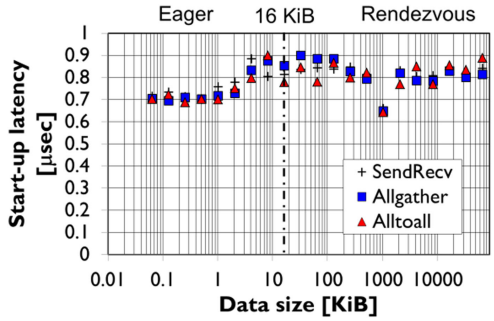

Fig. 13. Start-up latency versus data-transfer size.

for this configuration. One of these cycles is the time actually to send out the control signal. On the other hand, the migration delay is the time it takes for a control signal to move between FPGAs. This Migration delay includes a 0.15-μsec delay in SL3 [36]. The time to transmit the optical cable is as short as several nanoseconds. Therefore, we think the remaining 0.21 μsec is mainly in-FPGA process time, including managing the link.

On the other hand, the start-up latency for the 64-B eager communication is 0.71 μsec, with 0.32 μsec being configuration delay and 0.39 μsec being migration delay. The configuration delay in communication requires a time to read/write data from the memory I/O of HBM2, unlike barrier. In the case of Xilinx, this time requires about 90 cycles stated in the specification [37]. The 90 cycles are 0.45 μsec since the operating frequency of HBM2 is 200 MHz. Interestingly, the HBM2 internals complete it at 0.32 μsec in the prototype system.

We shift to consider the migration delay, i.e., 0.39 μsec. This delay was more significant than that of barrier operation with 0.03 μsec. This difference is because the data size was checked in the module near M_DMAC in a distributed switch. This process adds time for the data to pass through the distributed switch.

We evaluated the start-up latency of eager and rendezvous communications on allgather operation in Fig. 12. It is possible to switch between the two communications by setting the data size to be switched to a dedicated register. Therefore, we set the receive buffer size to 0 B, and rendezvous communication is performed even with 64-B data size. The configuration delay increases for the rendezvous process. However, the increased latency is marginal, about 0.09 μsec. The rendezvous process requires the migration delay for the control signal. By contrast, in our implementation, the rendezvous process was performed during the HBM2 processing. We thus mitigate the latency increases in rendezvous communication.

Fig. 13 shows the results of the start-up latency estimated from the time it takes for the reception and the effective bandwidth of SL3. The start-up latency is almost constant regardless of data size, i.e., 0.71 μsec for eager communications and 0.80 μsec for rendezvous communications, as shown in Fig. 12. However, when data size is more than 1 KiB, we observe a variation in the start-up latency. We consider that the marginal bandwidth difference between 51.2-Gbps internal bus and 50 Gbps of SL3 affects the start-up latency.

We emphasize that the start-up latency with less than 0.90 μsec is achieved on the OPTWEB network for a wide range of data sizes for sendrecv, allgather and alltoall operations.

4.3.3 Scalability

To investigate the effect of the number of FPGAs on communication behavior, we performed allgather operation on two to four FPGAs in terms of the effective bandwidth and start-up latency characteristics shown in Fig. 14. We observe that the communication performance was not affected by the number of FPGAs. This means that the communication performance per link is constant as the number of FPGAs increases.

Cygnus supercomputer using Intel Stratix 10 FPGA including routing module takes 0.5 μsec [12] for a neighboring ping-pong communication. It is difficult to compare these results due to the different evaluation methods. However, in the ping-pong evaluation in [12], the communication of ping-pong is performed in parallel. This means that the data is cut through in the turnaround FPGA. Hence, we consider that the start-up latency in [12] to be almost identical to our 0.71-μsec start-up latency.

Existing direct inter-FPGA networks have high per-hop latency, e.g., 200 ns of Cygnus supercomputer. In the case of 8×82-D mesh, the worst point-to-point communication start-up latency was 1.87 μsec [12]. We expect that collectives require a large number of total path hops in 2-D torus [7]. We expect that the large start-up latency in collective in [29] and [30] is due to these effects. Thus, the existing direct inter-FPGA networks have to accept large start-up latency as the number of FPGAs increase.

By contrast, the OPTWEB network provides the lowest, constant start-up latency of collectives, as shown in Fig. 14. The OPTWEB network increases the effective bandwidth proportionally with the number of FPGAs. In this context, we conclude that the OPTWEB has high scalability for supporting collectives.

4.3.4 Multipath Routing

Fig. 15 shows the experimental results of the multipath routing for point-to-point communication. In the figure, N represents the number of FPGAs. We implemented it by the combination of scatter and gather. In the multipath routing, we take striping data placement on multiple memory banks as described in Section 3.3. We evaluated the multipath routing when four and eight FPGAs. Since the multipath routing uses all network links, it is expected that end-to-end bandwidth becomes large as the number of FPGAs increases.

As shown in Fig. 15a, the start-up latency of the multipath routing is more than twice as high as that of the direct

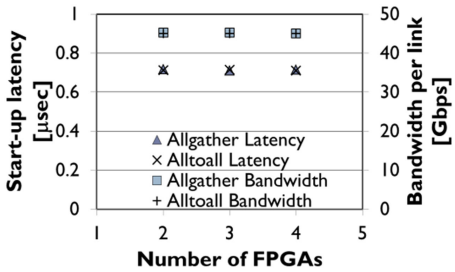

Fig. 14. Start-up latency and bandwidth versus the number of FPGAs.

path, which is 1.87 μsec with four paths. This is due to the additional time for confirming the reception completion. This processing overhead is imposed for collision-free multipath routing communications. The communication time of the multipath routing is shorter than that of the direct path when the data size is larger than 16 KiB. As a result, the effective bandwidth becomes 90 Gbps for four FPGAs and 180 Gbps for eight FPGAs, which is an $N/2$ time higher than the direct path, as shown in Fig. 15b.

4.4 Hardware Resource

The three most valuable resources are ALMs (Adaptive Logic Modules), M20K (high-speed memory), and DSP (Digital Signal Processing) on an FPGA. We evaluated the amount of these hardware resources in the proposed configuration with 400-Gbps bandwidth. We pick up 400-Gbps bandwidth because it is the same bandwidth as a modern NIC card. Evaluation results showed that the utilization of each resource was 31.4, 5.8, and 0.0 percent, respectively. By eliminating the backpressure, an OPTWEB network significantly reduces the required amount of the M20K for the high-bandwidth interconnects.

As for the ALMs, it is crucial to estimate the amount of hardware as the bandwidth increases. Here, we evaluate the case when the aggregate bandwidth, i.e., 800 Gbps, is fully used on the FPGA card. The 800-Gbps configuration was estimated by using two 400-Gbps modules. After extracting the amount of ALMs used by each IP, we estimated the ALM utilization and their breakdown in Fig. 16. Compared to the aggregate 400-Gbps interconnect, the utilization of the other IP cores becomes double while the PCIe for connecting to the host, which is independent of the number of nodes, remained the same.

When the design in literature [14], [16] is directly applied to 800-Gbps bandwidth, the amount of network resource surprisingly exceeds the total amount of ALMs in the

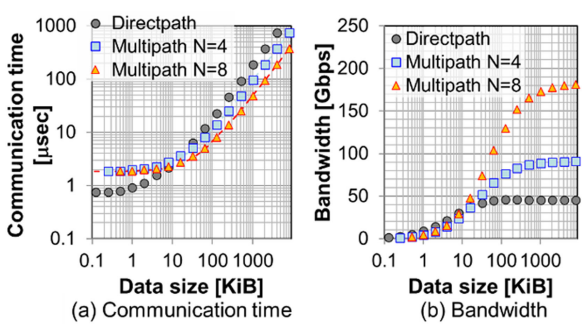

Fig. 15. Communication time and bandwidth of the multipath routing.

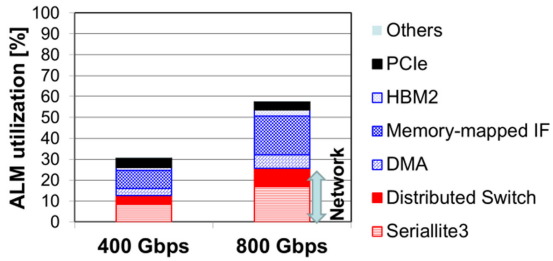

Fig. 16. ALM utilization for an 800-Gbps network on FPGA.

FPGA. This is not implementable. On the other hand, the OPTWEB successfully consumes only 26 percent ALMs for the distributed switch and SL3 due to both simple synchronization and eliminating the backpressure.

ALMs consume 57.7 percent in total by the 800-Gbps network. The remaining part, 42.3 percent of the ALMs, is available for user application logic including the change-over switch and the command sequencer. The ALM utilization of the user application logic is similar to that of the distributed switch. Thus, about 40 percent of ALMs are reasonable for providing user application logic.

5 Discussion

5.1 Cost

Beyond the performance evaluation that we have already investigated in the previous section, another practical concern when deploying an inter-FPGA network is cost. We compare a conventional packet network and the OPTWEB network for interconnecting 32 nodes.

5.1.1 Condition

Based on InfiniBand's price, a 32-node connection of 800 Gbps on each node would be about 9 $/Gbps taken from the Mellanox online site [38]. Since 32 nodes can fit in a rack, we assume to use the copper-cable cost. As for the OPTWEB network, we pick up W-OH using WDM for implementing a fully connected network topology. The cost of the optical transceiver itself is set at 1 $/Gbps, which is possible with silicon photonics [39]. The optical hub itself is an optical passive component and does not require a transceiver, which is located only in each FPGA. The cost of an FPGA is set at 1250 $ per FPGA and the communication cost of FPGA is calculated from the total hardware resource utilization (ALMs, M20K, and DSP). Light sources and WDM filters will be needed for the number of FPGAs. We use multiple WDM filters for cyclic WDM property to ensure link connection. The market price of these is about 400 $ and 90 $ per unit. However, there is an Erbium-doped fiber amplifier (EDFA) that can amplify wavelength-multiplexed light with high efficiency and low noise. By using EDFAs (500 $ per unit) and splitters (432 $ per unit) together, these components can be shared across multiple links, and the cost can be reduced to be 3 $/Gbps.

5.1.2 Results

Fig. 17 shows the cost estimation of both networks. The total cost of W-OH can be reduced to 5.5 $/Gbps. This indicates that the reduced cost of OPTWEB is about 40 percent

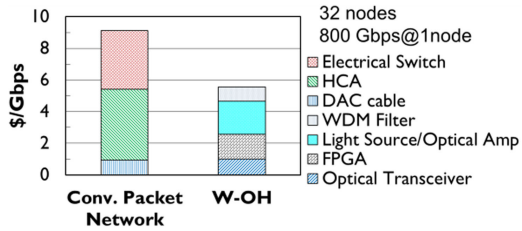

Fig. 17. Cost of conventional packet and OPTWEB networks (32 nodes, 800 Gbps in node).

compared to that of a counterpart InfiniBand network. InfiniBand supports many functions stated in InfiniBand Trade Association (IBTA). Supporting many functions lead to a high cost. By contrast, a custom inter-FPGA network can be simplified, and it becomes cheaper than InfiniBand.

We shift to compare the wiring cost on the OPTWEB network. The wiring cost is almost constant for moderate link bandwidth since an optical passive device is used. The two implementations of a fully connected network topology are evaluated in Fig. 17. P-OH's wiring costs were estimated using fabricating fiber sheets (MPO connections) for 1024 optical fibers. For W-OH, the cost was estimated under the same condition as the optical components shown in Fig. 17. P-OH is cheaper than the direct cable method in Fig. 2a because of the ability of high-density MPO connectors. Then, we omit a direct cabling implementation in Fig. 18. However, the cost of P-OH increases with the square of the number of FPGAs. On the other hand, W-OH's wavelength multiplexing technology makes it possible to consolidate multiplexed wavelength connectors into a single connector. Furthermore, the wiring cost is proportional to the number of FPGAs. Thus, we conclude that P-OH should be used by up to 32 FPGAs, while W-OH is preferred for the larger number of FPGAs.

5.2 Application Performance Estimation

The ultimate evaluation of an OPTWEB network is to estimate and compare the execution time of parallel applications. Since we do not have a conventional packet network of FPGAs, we estimate the relative performance of conventional packet networks (2-D torus, one-switch) and OPTWEB network using the discrete-event simulator SimGrid v3.12 [43]. A one-switch packet network connects all FPGAs to a single switch. As described in Section 2.3.2, to our best knowledge, only Ethernet switches are commercially available for building FPGA indirect networks. However, Ethernet's IPs impose an unacceptably large amount of the logic for high-bandwidth I/O on an FPGA. In the simulation, we assume an ideal lightweight

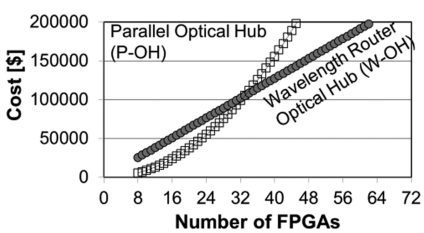

Fig. 18. Wiring cost on an OPTWEB network on parallel and wavelength router Optical Hub networks.

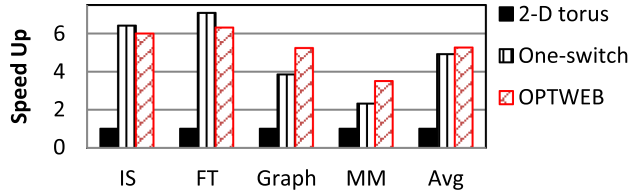

Fig. 19. Relative execution time of parallel applications for conventional packet and OPTWEB networks.

one-switch network that unburdens Ethernet compatibility for comparison purposes.

Modeling a computing system can be coarse grain in SimGrid, and we set a computation power of a node to 50 TFlops. For a fair comparison, node bandwidth is set to 800 Gbps on the three networks, i.e., 50-Gbps link on OPTWEB network while 200 Gbps on 2-D torus. We target the communication behavior of a parallel integer sort (IS), a parallel fast Fourier transform (FFT) of NAS Parallel Benchmarks, and a Graph500, and a parallel matrix multiplication (MM). IS and FT set class C. MM takes an 8192 x 8192 matrix. As for IS and FT, alltoall is frequently used. Graph500 and MM frequently used allgahter and broadcast, respectively. In the evaluation, the OPTWEB network selects direct or multipathdepending on the communication data size and communication type [20]. The larger communication data for point-to-point and some collectives, e.g., broadcast, should be transferred by the multipath routing, while the remaining data is transferred along with a direct path. Alltoall is also performed along with direct paths that use all links on a fully connected network topology. The threshold data size is set based on the observation of Fig.15. According to this policy, only Graph500 and MM use the multipath routing in our evaluation.

Fig. 19 illustrates the application speedup of each network relative to 2-D torus on 16 nodes. The higher value is better in the figure. The OPTWEB and one-switch network achieve six times better than 2-D torus in IS and FT. This is because high-radix networks provide high alltoall performance. On Graph500, the OPTWEB network outperforms by 4.8x and 1.24x the 2-D torus and one-switch networks, respectively. Our implementation of Graph500 generates flat traffic, and the average low-latency communication is preferred. On MM, the OPTWEB network improves by 3.5x and 1.5x the 2-D torus and one-switch networks, respectively, because the OPTWEB network efficiently supports broadcast by the multipath routing. Consequently, the OPTWEB network speeds up 7.1 percent on average compared to the one-switch network.

We conclude that the OPTWEB network is efficient for the parallel applications that frequently generate collectives.

5.3 Limiting Factors of Scalability

A crucial limiting factor in building an OPTWEB network is the number of FPGA card I/Os. In this study, we implement an OPTWEB network on the FPGA card using eight 100-Gbps (4×25-Gbps) links. This enables to connect 32 FPGAs. There is still a small room to increase the I/O's density on an FPGA card by wavelength multiplexing technology [23]. There is a constraint on the wavelength bandwidth of about 40 nm that

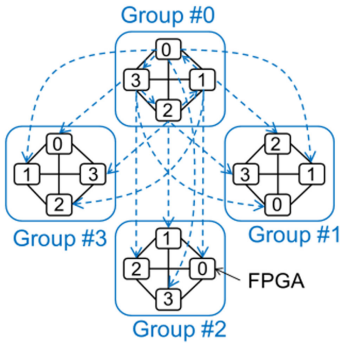

Fig. 20. A 16-node Dragonfly network topology on an OPTWEB network (g = 4, k = 6) [20]. We omit inter-group links except for the inter-group links of Group #0.

can be commonly utilized. In the case of a wavelength interval of 50 GHz (approx. 0.4 nm), we consider that the maximum number of I/Os can be 64 on a cost-effective FPGA.

Another stringent constraint on the number of FPGAs is the number of memory I/Os. In this study, two memory I/Os are used to communicate with a single FPGA. Currently, the maximum number of HBM2 chips in an FPGA is up to two, and this becomes a limiting factor in increasing the number of FPGAs. Fortunately, we would take an alternative to establish a more significant number of pseudo channels by splitting one memory I/O into multiple memory I/Os. We thus consider that the constraint on the memory I/Os can be mitigated.

We shift to discuss the possibility of increasing the network performance with 64 FPGAs (64×50 Gbps). In the early 2020s, faster I/O, larger FPGA packages, and improved circuit design techniques will fill the scale gap from 32/64×25 Gbps to 64×50 Gbps. In the latest FPGAs, the bandwidth of the high-speed transceiver on the network side has been accelerated to 57.8 Gbps using pulse amplitude modulation (PAM) technology [40]. In terms of the number of network I/Os, each FPGA can support 64 x 50 Gbps. This allows 128 optical fibers to be extracted from the FPGA chip using Co-Package Optics, which allows for high-density optical transceiver placement [23]. The number of fiber connectors can be reduced using WDM technology. In addition, the FPGA chips are getting larger: the Stratix10 GX 10M is available in a chip 1.7 times larger than the Stratix10 MX 2100 and with five times the number of ALMs. Considering this background, we will be able to introduce new FPGAs with more ALMs and more HBM2s soon, like the other accelerators [25]. Improvements to the FPGA core architecture are also expected to improve the operating frequency. In Intel's case, the second generation of HyperFlex can be expected to improve by 40 percent [32]. As a result, ALMs are also expected to increase by a factor of four. An OPTWEB network relies on technology-driven design, and each FPGA has up to 64 connections.

Finally, we consider connecting a larger number of FPGAs with a diameter-2 network topology. The two-hop indirect paths can guarantee the reachability on a diameter-2 network topology. Fig. 20 is a diameter-2 network topology, and it is a Dragonfly network topology that consists of fully connected inter-group and intra-group connections [20]. An FPGA can communicate with each other through a one-hop or two-hop path. Given k links of an FGPA and g groups on the above Dragonfly network topology, it supports up to $g(k\text{-}g+2)$ FPGAs on an OPTWEB network. The Dragonfly network should have $k \geq 3(g\text{-}1)$ based on the recommendation in [42]. In this case, up to 22 x (64-22+2) = 968 FPGAs can be interconnected for $k = 64$. Theoretically, allowing n-hop indirect paths enables diameter-n network topology that exponentially increases the number of FPGAs on an OPTWEB network.

As described in Section 3.4, the command sequencer allows collision-free communication on the two-hop indirect paths. Thus, a diameter-2 network topology can be constructed with collision freedom among links and DMACs on an OPTWEB network. Thus, the diameter-2 network extension is thus feasible with our FPGA network design.

6 Conclusion and Future Prospects

Some dozens of tightly coupled FPGA accelerators form a parallel computing platform. However, a conventional indirect packet network using Ethernet's IPs imposes a large amount of the logic on an FPGA for high-bandwidth interconnects. Besides the hardware-resource problem, a low-radix network topology, e.g., 2-D torus, becomes a problem in terms of their large path hops on direct inter-FPGA networks. The path hop directly increases the communication latency.

To mitigate both problems, we propose an OPTWEB network that provides a lightweight, fully connected inter-FPGA network for collectives. We prepare a separate one-hop end-to-end path for every pair of FPGAs. A raw block data is transferred with a simple synchronization. The routing information, i.e., source and destination FPGA addresses, are not needed for each communication block data. Since each FPGA performs input and output of the remote memory access simultaneously, multiple RDMAs enable the collectives with almost the same start-up latency, i.e., 0.71 μsec between Intel Stratix 10 MX FPGA cards with our onboard optics, as that of he point-to-point communication. The start-up latency would become large in the direct inter-FPGA networks with low-radix tori as the number of FPGAs increases. By contrast, the OPTWEB network provides almost constant start-up latency, i.e., 0.71 μsec. We also demonstrate that collectives fully utilize the maximum effective bandwidth of each link on the OPTWEB network.

OPTWEB network with 800-Gbps bandwidth per FPGA consumes only 57.7 percent of ALMs on the FPGA. The OPTWEB network provides a 40 percent lower hardware cost than conventional packet networks for interconnecting 32 FPGAs.

An OPTWEB network can introduce the multipath routing for increasing the end-to-end bandwidth between two FPGAs. We obtain aggregate 180-Gbps point-to-point communication when using eight FPGAs. The two-hop indirect paths are applicable for interconnecting a larger number of FPGAs than the number of connections on an FPGA. A diameter-2 network topology is then supported.

Our future work efficiently integrates simple computation to the communication operations, e.g., in-network reduction. Another future work is to provide a productive

software environment. OpenCL, a high-level language, has made it possible to design FPGA circuits. By using a distributed switch and M_DMAC, we will support arithmetic circuits provided by OpenCL.

Acknowledgments

This article is based on results obtained from a project (JPNP13004) commissioned by the New Energy and Industrial Technology Development Organization (NEDO). The authors would like to thank Mr. Ishida of NIPPON SYSTEMWARE Company Ltd., for his technical suggestion and support on the synthesis and implementation of the custom FPGA.

References

[1] J. Fowers *et al.*, "A configurable cloud-scale DNN processor for real-time AI," in *Proc. Annu. Int. Symp. Comput. Archit.*, 2018, pp. 1–14.
[2] S. Watanabe, K. Fujimoto, Y. Saeki, Y. Fujikawa, and H. Yoshino, "Column-oriented database acceleration using FPGAs," in *Proc. IEEE 35th Int. Conf. Data Eng.*, 2019, pp. 686–697.
[3] N. Samardzic, W. Qiao, V. Aggarwal, M. F. Chang, and J. Cong, "Bonsai: High-performance adaptive merge tree sorting," in *Proc. Annu. Int. Symp. Comput. Archit.*, 2020, pp. 282–294.
[4] H. Miao, M. Jeon, G. Pekhimenko, K. S. McKinley, and F. X. Lin, "Streambox-HBM: Stream analytics on high bandwidth hybrid memory," in *Proc. Int. Conf. Architectural Support Prog.. Lang. Operating Syst.*, 2019, pp. 167–181.
[5] J. Fang, Y. T. B. Mulder, J. Hidders, J. Lee, and H. P. Hofstee, "In-memory database acceleration on FPGAs: A survey," *VLDB J*, vol. 29, no. 1, pp. 33–59, 2020.
[6] A. Putnam *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," *IEEE Micro*, vol. 35, no. 3, pp. 10–22, May/Jun. 2015.
[7] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*, Amsterdam, The Netherlands: Morgan Kaufmann, 2002.
[8] K. Azegami *et al.*, "A STDM (static time division multiplexing) switch on a multi FPGA system," in *Proc. IEEE 13th Int. Symp. Embedded Multicore/Many-Core Syst.–Chip*, 2019, pp. 328–333.
[9] J. Lant, J. Navaridas, M. Jujan, and J. Goodacre, "Toward FPGA-based HPC: Advancing interconnect technologies," *IEEE Micro*, vol. 40, no. 1, pp. 25–34, Jan./Feb. 2020.
[10] A. D. George, M. C. Herbordt, H. Lam, A. G. Lawande, J. Sheng, and C. Yang, "Novo-G#: Large-scale reconfigurable computing with direct and programmable interconnects," in *Proc. IEEE High Perform. Extreme Comput. Conf.*, 2016, pp. 1–7.
[11] T. D. Matteis, J. F. Licht, J. Beranek, and T. Hoefler, "Streaming message interface: High-Performance distributed memory programming on reconfigurable hardware," in *Proc. Int. Conf. High Perform. Comput, Netw. Storage Anal.*, 2019, pp. 1–33.
[12] 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," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, 2020, pp. 450–459.
[13] Xilinx, High Speed Serial, Accessed Aug. 6, 2020. [Online]. Available: https://www.xilinx.com/products/technology/high-speed-serial.html
[14] A. Mondigo, T. Ueno, K. Sano, and H. Takizawa, "Comparison of direct and indirect networks for high-performance FPGA clusters," in *Proc. Int. Symp. Appl. Reconfigurable Comput. Architectures Tools Appl.*, 2020, pp. 314–329.
[15] A. Mondigo, T. Ueno, K. Sano, and H. Takizawa, "Scalability analysis of deeply pipelined tsunami simulation with multiple fpgas," *IEICE Trans. Inf. Syst.*, vol. E102-D, no. 5, pp. 1029—1036, 2019.
[16] J. Stern, Q. Xiong, A. Skjellum, and M. C. Herbordt, "A novel approach to supporting communicators for in-switchprocessing of MPI collectives," in *Proc. Workshop Exascale MPI*, 2019, pp. 1–10.
[17] Q. Xiong, C. Yang, P. Haghi, A. Skjellum, and M. Herbordt, "Accelerating MPI collectives with FPGAS in the network and novel communicator support," in *Proc. 28th IEEE Int. Symp. Field-Programmable Custom Comput. Mach.*, 2020, Art. no. 215.
[18] B. Buscaino, J. M. Kahn, and B. D. Taylor, "Coherent co-packaged optical interfaces for next-generation electrical switches," in *Proc. IEEE Photon. Conf.*, 2019, pp. 1–2.
[19] T. Nakamura *et al.*, "Fingertip-size optical module, "optical I/O core", and its application in FPGA," *IEICE Trans. Electron.*, vol. E102–C, no. 4, pp. 333–339, 2019.
[20] Y. Urino, K. Mizutani, T. Usuki, and S. Nakamura, "Wavelength-routing interconnect "optical hub" for parallel computing systems," in *Proc. HPC Asia*, 2020, pp. 81–91.
[21] R. Proietti et al., "Scalable optical interconnect architecture using AWGR-based TONAK lion switch with limited number of wavelengths," *J. Lightw. Technol.*, vol. 31, no. 24, pp. 4087–4097, 2013.
[22] COBO, Accessed: Aug. 6, 2020. [Online]. Available: https://www.onboardoptics.org/
[23] M. Wade *et al.*, "TeraPHY: A chiplet technology for low-power, high-bandwidth in-package optical I/O," *IEEE Micro*, vol. 40, no. 2, pp. 63–71, Aug. 2019.
[24] AIO CORE, Accessed Aug. 6, 2020. [Online]. Available: http://www.aiocore.com/
[25] NEC SX Aurora TSUBASA, Accessed: Aug. 6, 2020.[Online]. Available: https://www.nec.com/en/global/solutions/hpc/sx/
[26] T. Akiba, S. Suzuki, and K. Fukuda, "Extremely large minibatch SGD: Training resnet-50 on imagenet in 15 minutes," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 1–4.
[27] B. Klenk, N. Jiang, G. Thorson, and L. Dennison, "An in-network architecture for accelerating shared-memory multiprocessor collectives," in *Proc. Int. Symp. Comput. Archit.*, 2020, pp. 996–1009.
[28] A. Li *et al.*, "Evaluating modern GPU interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 1, pp. 94–110, Jan. 2020.
[29] T. Ueno, T. Miyajima, A. Mondigo, and K. Sano, "Hybrid network utilization for efficient communication in a tightly coupled FPGA cluster," in *Proc. Int. Conf. Field-Programmable Technol.*, 2019, pp. 363–366.
[30] J. Sheng, Q. Xiong, C. Yang, and M. C. Herbordt, "Application-aware collective communication (extended abstract)," in *Proc. IEEE Annu. Int. Symp. Field-Programmable Custom Comput. Mach.*, 2016, Art. no. 197.
[31] J. Sheng, Q. Xiong, C. Yang, and M. C. Herbordt, "Collective communication on FPGA clusters with static scheduling," *SIGARCH Comput. Archit. News*, vol. 44, no. 4, pp. 2–7, 2016.
[32] Intel Agilex FPGAs and SoCs, Accessed: Aug. 6, 2020. [Online]. Available: https://www.intel.com/content/www/us/en/products/programmable/fpga/agilex.html
[33] K. Abe and T. Ishigure, "Low loss multimode polymer shuffling optical waveguide for high-density optical circuit," in *Proc. IEEE CPMT Symp.*, 2017, pp. 153–154.
[34] H. Nakamura *et al.*, "40Gbit/s-class-λ-tunable WDM/TDM-PON using tunable B-Tx and cyclic AWG router for flexible photonic aggregation networks," in *Proc. Eur. Conf. Exhib. Opt. Commun.*, 2012, pp. 1–3.
[35] Z. Wang, H. Huang, J. Zhang, and G. Alonso, "Shuhai: Benchmarking high bandwidth memory on FPGAS," in *Proc. IEEE 28th Annu. Int. Symp. Field-Programmable Custom Comput. Mach.*, 2020, pp. 111–119.
[36] "Serial lite III streaming intel FPGA IP core user guide, Updated for Intel® Quartus® Prime Design Suite: 18.1.1," Accessed: Apr. 3. 2021. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/ug/ug_slite3_streaming.pdf
[37] "AXI high bandwidth memory controller v1.0," Accessed: Aug. 6, 2020, [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/hbm/v1_0/pg276-axi-hbm.pdf
[38] Mellanox Switches, Accessed: Aug. 6, 2020, [Online]. Available: https://store.mellanox.com/categories/switches.html
[39] R. Soref, "The achievements and challenges of silicon photonics," *Adv. Opt. Technol.*, vol. 2008, 2008, Art. no. 472305.
[40] "Intel stratix 10 NX FPGA," Accessed: Aug. 6, 2020. [Online]. Available: https://www.intel.com/content/www/us/en/products/programmable/fpga/stratix-10/nx.html
[41] A. Y. Al-Dubai, M. Ould-Khaoua, and L. M. Mackenzie, "Trade-offs between latency, complexity, and load balancing with multicast algorithms," *IEEE Trans. Comput.*, vol. 59, no. 2, pp. 159–173, Feb. 2010.
[42] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proc. Int. Symp. Comput. Archit.*, 2008, pp. 77–88.
[43] "SIMGRID simulation of distributed computer systems," Accessed: Nov. 16, 2020. [Online]. Available: https://simgrid.org/

Kenji Mizutani received the dual PhD degree in semiconductor engineering from Nagoya University, Nagoya, Aichi, Japan, in 2003. In 2003, he joined NEC Corporation, where he worked on optical devices and optical telecommunication systems. Since 2014, he has been a senior researcher with PETRA, developing optical interconnect technology.

Hiroshi Yamaguchi received the BE degree in electronic engineering and the ME degree in electrical engineering from Tokyo Denki University, Tokyo, Japan, in 1986 and 1989, respectively. In 1989, he joined NEC Corporation, Fuchu, Japan, where he worked on the development on electrical circuit. He is currently a chief researcher with Photonics and Electronics Technology Research Association, Fuchu, Japan, on a temporary basis.

Yutaka Urino received the BE degree in communication engineering and the ME degree in electronic engineering from Tohoku University, in 1985 and 1987, respectively. In 1987, he joined NEC Corporation, Kawasaki, Japan, where he worked on research and development of optical waveguide devices and subsystems. He is currently a chief researcher with Photonics and Electronics Technology Research Association on a temporary basis. He is one of the coauthors of *Silicon Photonics III* (Springer). He was the recipient of the Best Paper Award at international conferences, including the OEC'88 and the OECC'98. His current research interests include silicon photonics, optical interconnects, and parallel computing. He is a topical editor of the *Optical Review*.

Michihiro Koibuchi (Senior Member, IEEE) received the BE, ME, and PhD degrees from Keio University, Yokohama, Kanagawa, Japan, in 2000, 2002, and 2003, respectively. He is currently an associate professor with the National Institute of Informatics and SOKENDAI, Tokyo, Japan. His research interests include the areas of high-performance computing and interconnection networks. He has authored or coauthored 100 referred technical conference and journal papers, including nine in the *IEEE Transactions on Parallel and Distributed Systems*, five in the IPDPS, four in the HPCA, and three in the *IEEE Transactions on Computers*. He is a senior member the IEEE Computer Society, the IPSJ, and the IEICE.