PAPER Special Section on Parallel and Distributed Computing and Networking

Novel Chip Stacking Methods to Extend Both Horizontally and Vertically for Many-Core Architectures with ThrouChip Interface

Hiroshi NAKAHARA[†a)], **Tomoya OZAKI**[†], ***Nonmembers***, **Hiroki MATSUTANI**[†], ***Member***, **Michihiro KOIBUCHI**[††], ***Senior Member***, ***and*** **Hideharu AMANO**[†], ***Fellow***

SUMMARY The increase of recent non-recurrent engineering cost (design, mask and test cost) have made large System-on-Chip (SoC) difficult to develop especially with advanced technology. We radically explore an approach for cheap and flexible chip stacking by using Inductive coupling ThruChip Interface (TCI). In order to connect a large number of small chips for building a large scale system, novel chip stacking methods called the linear stacking and staggered stacking are proposed. They enable the system to be extended to ***x*** or/and ***y*** dimensions, not only to ***z*** dimension. Here, a novel chip staking layout, and its deadlock-free routing design for the case using single-core chips and multi-core chips are shown. The network with 256 nodes formed by the proposed stacking improves the latency of 2D mesh by 13.8% and the performance of NAS Parallel Benchmarks by 5.4% on average compared to that of 2D mesh.

key words: *inductive coupling interconnect, interconnection network, network on chip*

1. Introduction

The increase of recent non-recurrent engineering cost have made large System-on-Chip (SoC) difficult to develop especially with advanced technology. Alternatively, various techniques on System-in-Package, which integrates a number of small chips, have been developed. 2.5D implementation with Through Silicon Via (TSV) [1], micro bumps, and a silicon interposer has become a mature technique for building large-scale FPGAs.

We radically explore a different approach for cheap and flexible chip stacking. To connect a large number of small chips for building a large scale system, novel chip stacking methods called the linear stacking and staggered stacking are proposed that enable the system to be extended to x or/and y dimensions, not only to z dimension [2], [3]. They interestingly allow to incrementally add chips to existing stacked chip systems and allows to optimize stacking to a target application on demand.

For such flexible inter-chip communication, we use inductive coupling ThruChip Interface (TCI) [4]. TCI is yet another technique to connect multiple chips with high-speed links. Since links between chips are built with a wireless interconnect, it is easy to insert or replace chips of the chip-stack after fabrication. Since coils for inductor can be built with metal wires, no special process technology is needed other than standard CMOS process. It has been used for memory stacking [5], a dynamically reconfigurable processor [6], and a heterogeneous multi-core system [7]. In all of them, straight-forward 3D stacking is used. However, the number of connected chips is limited to eight considering the total height of the chip stack.

Our challenges in novel chip stacking methods are (1) chip staking layout for TCI using z, x or/and y dimensions, and (2) the routing algorithm for the case using multi-core chips.

2. Inductive Coupling Through Chip Interface

Inductive-coupling TCI uses square coils implemented with common layers of the chip. As shown in Fig. 1, by stacking a transceiver coil on a receiver coil, an inductive coupling channel is formed between two chips. Two coils, one for the clock and the other is for data are usually provided for a channel. A high frequency clock (1GHz to 8GHz) is generated by a ring oscillator, and data are serially transferred synchronized with the clock directly through the driver. The driver and inductor pair for sending data is called the TX channel, while the receiver and inductor pair is called the RX channel. Data can be transferred at most 8Gbps with a low energy dissipation (0.14pJ per bit) and a low bit-error rate (BER$< 10^{-12}$) [8].

Data multicast can be used if a TX channel is placed at the same location of multiple RX channels in different chips. On the other hand, stacked multiple TX channels at the same location cannot send the data simultaneously to avoid interference. Since a coil can be used for both the transmitter and

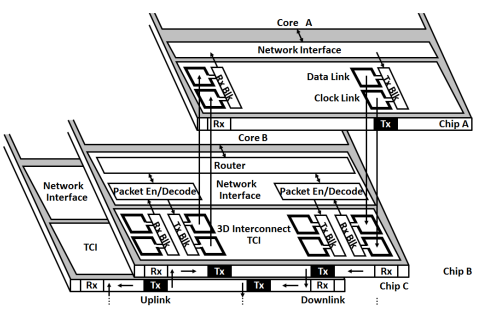

Fig. 1 3D NoC using TCIs

Manuscript received January 8, 2016.
Manuscript revised April 20, 2016.
Manuscript publicized August 24, 2016.

[†]The authors are with Dept. of ICS, Keio University, Yokohama-shi, 223–8522 Japan.
[††]The author is with National Institute of Informatics, Tokyo, 101–8430 Japan.
a) E-mail: blackbus@am.ics.keio.ac.jp

DOI: 10.1587/transinf.2016PAP0033

Copyright © 2016 The Institute of Electronics, Information and Communication Engineers

receiver, the functionality of TX and RX channels can be quickly switched, that is, a half-duplex bi-directional channel can be formed using a single coil.

Although TCI requires a certain amount of logic to form a link between two chips, it has the following benefits. (1) A number of chips can be stacked if a physical environment is allowed. (2) Since chips can be tested before stacking, only known-good-dies can be connected. (3) Since TCI is electrically contact-less, no electro-static-discharge (ESD) protection device is needed. (4) Since the coil uses common wire layers of CMOS process, no extra process is needed. Although a coil has a large footprint, we can implement circuits inside the coil.

Here, we roughly compare TCI and TSV, one of the most common 3D stacking technologies. The paper [4], [8] show the detail. TSV can transfer data at most 10Gbps with the 0.6pJ/bit [9], while the TCI can do at most 8Gbps with the 0.14pJ/bit. The footprint of TCI coil is much larger than that of the TSV. However, while TSV occupies all layers, the TCI uses only two metal layers and the other metal layers and transistors are aveilable for random logic even if they are inside coils. Although TSV can be used to heat dissipation, the TCI, which is electrically contact-less, is not used for it. On the other hand, TCI does not required the ESD device.

Although a number of practical systems have been developed by using TCI, all use simple 3D chip stacking. Although a case has been reported in which more than 10 chips were stacked [10], stacking chips with simple 3D stacking, that is, to z-dimension has certain physical limitations. First, the chip stacking is sometimes physically unstable when more than four chips are stacked, since ground chips are slightly bent because of the difference between the coefficient of thermal expansion of the silicons and that of the wires. Second, the package height is limited. From a practical viewpoint, it is difficult to stack more than eight chips to z-dimension.

3. Linear Stacking

In order to connect more chips with the limitations of z-dimension, the straightforward idea is a linear stacking which extends to only x-dimension. Here, four full-duplex channels are assumed on each corner of a chip as shown in Fig. 2. By using each stacked chip as a bridge between chips, the stacking can be extended to x-dimension as shown in Fig. 3, resulting in more than 64 chips being connected within an eight-chip height. We assume only a single core in a chip first, after that we extend it to multiple cores in a chip.

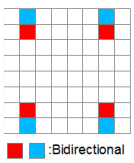

Fig. 2 Chip used in linear stacking

3.1 Single Core Chip Stacking

Definition 3.1: Linear Stacking
Let X be the number of chips on the consecutive two layers and Z be the number of layers, respectively. The position of chip is defined (x, z) where x means the x th chip from left to right, and z means a chip is located on z th layer starting from 0. For layer z, if z is an even number, place the chip on the grid x, where x is an even number. If z is an odd number, place the chip on the grid x, where x is an odd number. The chip on (x, z) is connected with at most four chips $(x-1, z-1)$, $(x-1, z+1)$, $(x+1, z-1)$, and $(x+1, z+1)$ when the following conditions are satisfied : $(0 \leq x < X)$ and $(0 \leq z < Z)$. □

Routing algorithm 1

1. Apply XY routing on the equivalent 2D mesh.

2. If it reaches a boundary before arriving at the destination move the packet a hop to Y direction so that it goes apart from the wall. Then move X direction again so that it is close to that of the destination node.

3. When X of the packet is equal to that of the destination node, then go to Y direction so that it is close to the destination node.

4. If it counters a boundary, go apart from it a hop to X direction and then go to Y direction again until it reaches to the destination node.

The network formed by linear stacking is represented as S$[X, Z]$. S$[X, Z]$ is similar to 2D mesh with stairway boundary, so we call it Stairway Boundary Mesh (SBM). Here, let coordinate of 2D mesh be (i, j) where i is x-axis from left to right, j is y-axis from bottom to up. Note that the coordinate system of linear stacking and 2D mesh is different. S$[X, Z]$ is equivalent to the grid in the mesh surrounded by $(\lfloor \frac{Z}{2} \rfloor, 0)$- $(\lfloor \frac{X}{2} \rfloor + \lfloor \frac{Z}{2} \rfloor, \lfloor \frac{X}{2} \rfloor)$- $(\lfloor \frac{X}{2} \rfloor, \lfloor \frac{X}{2} \rfloor + \lfloor \frac{Z}{2} \rfloor)$- $(0, \lfloor \frac{Z}{2} \rfloor)$. For example, Fig. 4 shows the S[11,7] with solid lines and 9×9 mesh with broken lines. S[11,7] is equivalent to the grid in the mesh surrounded by (3,0)-(8,5)-(5,8)-(0,3).

Since SBM is regarded as a mesh partially, Dimension Ordering Routing (DOR) [11] can be applied except boundary. Modified DOR to apply SBM is represented as Routing algorithm 1.

In the Routing algorithm 1, the number of hops itself is the same as that of the XY routing in the mesh with the flat

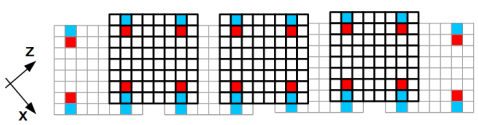

Fig. 3 Example of stacking seven chips in accordance with linear stacking

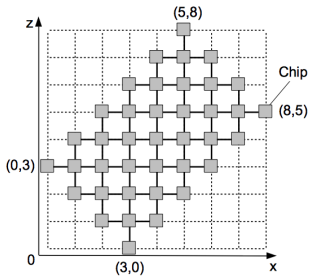

Fig. 4 Network formed by linear stacking (S[11,7]) with solid line and 9×9 mesh with break line

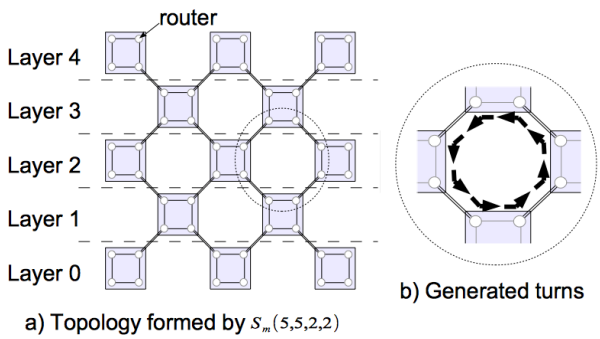

Fig. 5 Linear stacking with multi-core chips

boundaries.

Theorem 3.1: Routing algorithm 1 is deadlock free.

Proof 3.1: In Routing algorithm 1, illegal turns from $Y \rightarrow X$ are only generated at a hop from the boundaries. Since there is no paths outside the boundaries, cycles never formed with such illegal turns. Therefore, it is deadlock free same as XY routing. □

3.2 Multi-Core Chip Stacking

Recent chips have two or more cores even if the chip size is small. Here we extend the linear stacking for single-core chips to stack multi-core chips. Any topology can be applied for inner-chip network if there is deadlock free routing. Assuming that inner-chip topology is 2×2 Mesh as an example. Four routers at each corner of the chip are connected to a router on the other chip with TCI. A general router in 2D mesh has five ports: four for neighbors and one for the core. However, four routers at each corner of the chip use only three of them, thus, remaining one of two ports can be used for up-link or down-link of TCI without changing the router structure. We define the topology formed by linear stacking with multi-core chips as $\mathrm{S_m}[X, Z, X_c, Y_c]$ where X_c and Y_c are the size of x-axis and y-axis on a chip, respectively. For example, Fig. 5 a) shows the $\mathrm{S_m}$[5,5,2,2]. Inter-chip links form the S[5,5] and inner-chip links form the 2×2 Mesh.

Routing for linear stacking with multi-core chips is a little complicated. Routing algorithm 1 can be applied for inter-chip routing, and DOR can be applied for inner-chip routing because inner-chip topology is general 2D mesh. More precisely, routing for the linear stacking with multi-core represented as Routing algorithm 2 is defined as follows.

Routing algorithm 2

1. Apply Routing algorithm 1 on the inter-chip. Here, let C_{next} be the next chip.

2. If a router with a packet is connected to C_{next}, a packet is sent to the next chip through the inter-chip link.

3. Otherwise, apply DOR on the inner-chip to a router connected to C_{next}.

The problem is that turns are generated by inter→inner link and inner→inter link. As a result, deadlock occurs on the 8 routers as shown in Fig. 5 b). Although the cycle can be resolved by prohibiting some turns in inter-chip routing or inner-chip routing, this approach will introduce a pair of nodes which is difficult to communicate each other. So we introduce two virtual channels (VC) and a simple rule to use them as follows.

VC transition for Routing algorithm 2

1. Only if a packet is sent from a chip on (x,z) to the other chip on $(x+1,z+1)$ through inter-chip link, set next VC to be 1.

2. Otherwise, next VC is the same as current VC.

Theorem 3.2: Routing algorithm 2 using two VCs is deadlock free

Proof 3.2: The VC control is the same as well-known method of avoiding deadlock on a ring topology. For example, Fig. 5 b) shows the possible deadlock, and it can be regarded as ring topology with 8 routers. All of cyclic dependencies on the $\mathrm{S_m}[X, Z, X_c, Y_c]$ are generated on the ring topology, and all of them contain the inter-chip link from a chip (x,y) to the other chip $(x+1,y+1)$. Therefore, deadlock never occurs. □

4. Staggered Stacking

As an extension of the simple linear stacking, another stacking method, called staggered stacking is proposed. The aim of the stacking is to extend the system to x and y dimensions, not only z dimension.

4.1 Single Core Chip Stacking

In order to extend chip-stack both to x and y dimensions, we use four coils (two for TX and two for RX) to make a full duplex link between stacked chip. A couple of full duplex links are provided at four corners of a chip instead of the

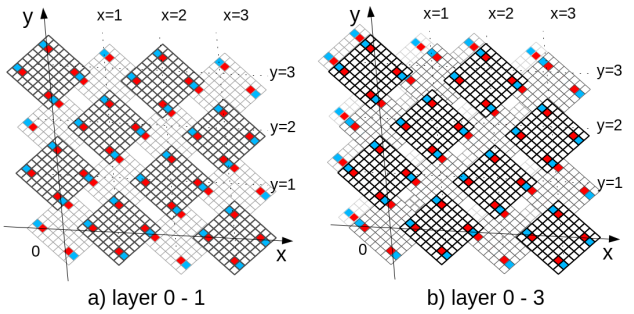

Fig. 6 32-chip stacking with staggered stacking

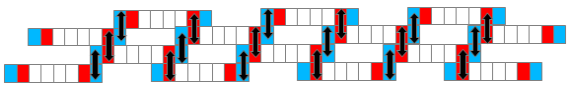

Fig. 7 Cross cutting view of Fig. 6 b)

uni-direction links in Fig. 2. Since a chip can be a bridge of four under-laying or overlaying chips with these coils, we can extend the chip stack to both x and y directions.

Definition 4.1: Staggered Stacking
Chip is placed on a grid turning by 45 degrees. The grid size is fixed so that four TCI links of a chip are just on the under-laying chips and the remaining four TCIs are on the over-laying chips. For layer k, if k is an even number starting from 0, place the chip on the grid (i, j), where $(i + j)$ is an even number. If k is an odd number, place the chip on the grid (i, j), where $(i + j)$ is an odd number. □

For example, considering the case that stacks 32 chips with the staggered stacking, first we stack eight chips for layer 0 and eight chips for layer 1 in accordance with the definition shown in Fig. 6 a). Then, in the same manner, we place layers 2 and 3 as shown in Fig. 6 b). Note that layers 0 and 2 are the same layout, while layers 1 and 3 are also the same.

Note that chips are stacked so that every layer is shifted with the size of the coil to prevent vertical interference. Figure 7 is a cross-cutting view of the chip stack shown in Fig. 6 b). By shifting chips, a coil can be placed just on coils in the next lower and upper layers. This is the reason why the grid tilts. Even if the number of layers is increased, the vertical interference can be avoided in this manner. We call this stacking method staggered stacking.

Staggered stacking generates more spaces between chips than linear stacking. To avoid being physically unstable, spacer chips are sometimes needed. On the other hand, space between chips can be advantageous for heat dissipation.

4.2 The Network Topology

The staggered stacking builds a network between all stacked chips by using a chip as a bridge of other chips. First, for simplicity, all TCI links are connected to a single router which also connects to a core on the chip. That is, a chip

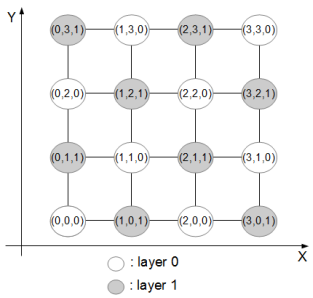

Fig. 8 Topology composed in Fig. 6 a). It is equivalent to the 4×4 mesh

is treated as a node with eight links for connecting to other nodes. A connection topology between nodes composed by the staggered structure is defined as follows.

Definition 4.2: Topology formed by staggered stacking
Network topology formed by the staggered stacking is represented as T$[M, N, H]$ where M and N represent the number of cores in two adjacent layers, and H is the number of layers stacked and must be an even number. A node is identified with (x, y, z), where (x, y) is a coordinate of the grid where the corresponding chip is placed, and z is the layer number placed. A node (x, y, z) is connected to $(x + 1, y, z + 1)$, $(x, y+1, z+1)$, $(x-1, y, z+1)$, $(x, y-1, z+1)$, $(x+1, y, z-1)$, $(x, y+1, z-1)$, $(x-1, y, z-1)$, and $(x, y-1, z-1)$ when the following conditions are satisfied: $0 \le x < N$, $0 \le y < M$, and $0 \le z < H$. □

Note that staggered stacking connects chips through vertical TCI links, a chip and its neighbors have different z. For example, topology shown in Fig. 6 b) is represented as T[4,4,4]. The total number of chips becomes $NMH/2$.

The topology generated by the staggered stacking itself is the same as that of Topology B [12] and the offset cube [13]. We will extend it to stack multi-core chips which have multiple cores in a chip connected with each other.

4.3 The Network Topology for Multi-Core Chips

Like the case of linear stacking, we assume that cores in the chip are connected with a 2D mesh. With this method, we can extend the network for staggered stacking to multi-core chips. Note that the node with eight links in T$[M, N, H]$ is distributed to corner nodes, in which each node has four links, same as a common 2D mesh and the SBM from the linear stacking.

Since the data transfer rate of a TCI link is 8Gbps [4], and the bandwidth of a link can be enhanced by increasing the number of data coils, we assume that inter and inner chip links have the same bandwidth.

When such chips are stacked in the staggered stacking, the network topology T$_m[M, N, H, Mc, Nc]$ is defined.

Definition 4.3: Topology formed by staggered stacking with multi-core chips
Assume that $Mc \times Nc$ nodes in a chip are connected with 2D mesh. Add two TCI links for off-chip connections to

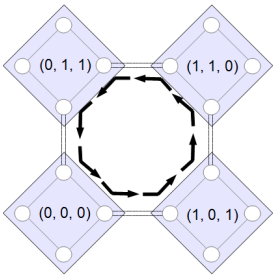

Fig. 9 Topology of [2,2,2,2,2]

four corner nodes. These chips are placed in the staggered stacking T[M, N, H], and then a network topology T$_m$[M,N,H,Mc,Nc] is formed. □

For example, the topology shown in Fig. 9 is represented as T$_m$[2, 2, 2, 2, 2]. Since the stacking method is the same as the staggered stacking, the topology composed by chips is a 2×2 mesh. Here, TCI links are shown with doublet lines, while other single lines show links inside the chip.

4.4 Routing

4.4.1 Routing for T[M, N, H]

First, the routing algorithm for T[M, N, H] is discussed and then it is extended to T$_m$[M,N,H,Mc,Nc]. The positive-Z-first algorithm proposed for the offset cube can be directly applied to T[M, N, H]. However, since it is an adaptive routing, it is difficult to be extended for T$_m$[M,N,H,Mc,Nc]. Here, we use a simpler fixed routing as a basis of the routing.

Although the DOR [11] can be applied on a 2D mesh formed with T[M, N, H], xy-direction and z direction needs to be moved simultaneously to z direction. Thus, we must select two routing methods in accordance with the position of the source node to the destination node.

Let $(x_{\text{cur}}, y_{\text{cur}}, z_{\text{cur}})$ be the source node, and $(x_{\text{dst}}, y_{\text{dst}}, z_{\text{dst}})$ be the destination node. Here, we define the absolute distance between the current node and the destination node as $dx = |x_{\text{cur}} - x_{\text{dst}}|$, $dy = |y_{\text{cur}} - y_{\text{dst}}|$, and $dz = |z_{\text{cur}} - z_{\text{dst}}|$. The number of hops for routing in xy-axes is expressed as $dx + dy$, and that for z-axis is expressed as dz. The routing method is selected on the basis of the relationship between the $dx + dy$ and dz as follows.

- $dx + dy \geq dz$
  In this case, z coordinate reaches z_{dst} before (x, y) becomes the destination $(x_{\text{dst}}, y_{\text{dst}})$ with the DOR. For example, assume that the chip (0,0,0) sends a packet to (3,3,2) in T[4,4,4] topology. As the routing is basically done with the DOR, first, the packet goes in x direction to the x_{dst}. Since a hop in x direction also moves in z direction, we select the direction in which z_{cur} moves closer to z_{dst}. That is, the packet is transferred in the order of (0,0,0)-(1,0,1)-(2,0,2). Now, z coordinate reaches z_{dst} while (x, y) coordinates have not. In this case, we send the packet to z coordinate so that it is not far from the destination while the DOR is applied to xy-direction. That is, if z_{cur} equals z_{dst}, z coordinate is just incremented except when z_{cur} equals $H - 1$. In this case, since the upper neighbor does not exist, z coordinate is decremented. Otherwise, the packet is sent to z direction with the z_{dst}. In accordance with the rules above, the packet is forwarded in the order of (2,0,2)-(3,0,3). Now, since x coordinate is the same as the destination, the packet is sent by the same rules to y direction. Thus, it reaches the destination on the remaining path (3,0,3)-(3,1,2)-(3,2,3)-(3,3,2).
- $dx + dy < dz$
  In this case, (x, y) direction reaches $(x_{\text{dst}}, y_{\text{dst}})$ before z coordinate reaches z_{dst}. For example, assume the case of sending a packet from (0,6,0) to (1,6,7) in the T[8,8,8]. Similar to the case of $dx+dy \geq dz$, routing for xy direction uses the DOR, and routing on z axis makes z_{cur} move closer to the z_{dst}. In the example, the packet goes on the path (0,6,0)-(1,6,1), and x coordinates is equal to x_{cur} even though z coordinate has not arrived yet. Then, in accordance with the DOR the packet is sent to y direction on the basis of the relationship between the dy and dz. When $dy > dz$, the packet is sent to y direction to $y_{\text{cur}} + 1$. When $dy < dz$, the packet is sent to $y_{\text{cur}} - 1$. When $dy = dz$, the packet is sent to closer to y_{dst}. In this example, the remaining routing path from (1,6,1) to (1,6,7) is shown in Fig. 10.

The above routing algorithm is represented as Routing algorithm 3. Here, let $(x_{\text{next}}, y_{\text{next}}, z_{\text{next}})$ be the coordinates of the next chip determined by the algorithm.

Routing algorithm 3

```
if (dx + dy < dz) {
    (x_next, y_next) = DOR to (x_dst, y_dst)

    if (z_cur ≠ z_dst) making z_next move close to z_dst
    else z_next is z_cur + 1
} else {
    if (x_cur ≠ x_dst)
        making x_next move close to x_dst
    else {
        if (dy > dz) y_next is y_cur + 1
        else if (dy < dz) y_next is y_cur−1
        else making y_next move close to y_dst
    }
    making z_next move close to z_dst
}
```

Proof of deadlock-freedom of Routing algorithm 3 is divided into two parts. First, it is shown that all (x, y, z) values must not change simultaneously. Second, the movement of packets is proved to be deadlock free on XY-plane, XZ-plane, and YZ-plane.

Lemma 4.1: When sending a packet to its neighboring chip, all the (x,y,z) values never change simultaneously

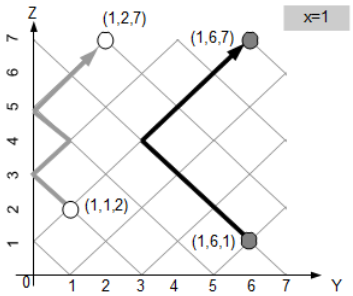

Fig. 10 Routing on YZ-plane in the case of $(dx + dy) < dz$

Proof 4.1: A chip at (x, y, z) coordinates is adjacent to $(x+1, y, z+1)$, $(x, y+1, z+1)$, $(x-1, y, z+1)$, $(x, y-1, z+1)$, $(x+1, y, z-1)$, $(x, y+1, z-1)$, $(x-1, y, z-1)$, and $(x, y-1, z-1)$. All of them have the same value as (x, y, z) at one axis. Thus, all the (x, y, z) values never change simultaneously. □

Lemma 4.2: Routing algorithm 3 is deadlock free on XY-plane, XZ-plane, and YZ-plane.

Proof 4.2: Proofs are divided into three parts for each planes.

1. XY-plane
   Turns on the XY-plane are shown in Fig. 11 a). As the routing on the *xy*-axis is the DOR, two turns $((x+1, y)-(x+1, y+1)-(x, y+1)$ and $(x, y+1)-(x, y)-(x+1, y))$ are prohibited. Cycles are, thus, never formed in this plane.
2. XZ-plane
   Turns on the XZ-plane are shown in Fig. 11 b). Once a packet goes to the $x \pm 1$ direction, it never changes its direction, that is, a packet never goes to $x \mp 1$ the direction. That is, turns $((x+1, z)-(x, z+1)-(x+1, z+2)$ and $(x, z+2)-(x-1, z+1)-(x, z))$ are prohibited. Cycles are, thus, never formed in this plane.
3. YZ-plane
   Turns on the YZ-plane are shown in Figs. 11 c) and 11 d). According to algorithm 3, when $(dx + dy) < dz$, the next direction is determined by the relationship between dy and dz. Here a turn $(y, z)-(y+1, z+1)-(y, z+2)$ in Fig. 11 c) and a turn $(y, z+2)-(y+1, z+1)-(y, z)$ in Fig. 11 d) are prohibited except if y equals 0. If y doesn't equal 0, since these prohibited turns are known to be negative-first routing [14], cycles are never formed in this case. If y equals 0, there is no chip at the $(y-1, z+1)$, thus no cycles are formed. □

Theorem 4.3: Routing algorithm 3 is deadlock free.

Proof 4.3: From Lemma 3.1, Routing algorithm 3 is divided into three parts each of which is on three planes, and routing on each plane is never used twice. From Lemma 3.2, deadlock never occurs in each plane. Thus, the Routing algorithm 3 is deadlock free. □

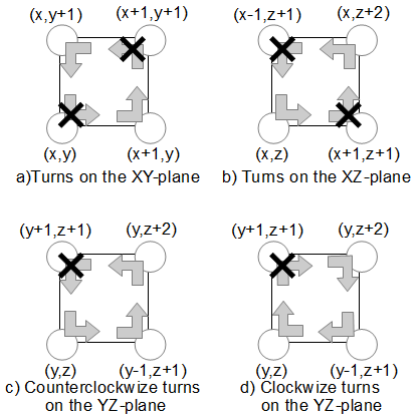

Fig. 11 Turns on each plane

4.4.2 Extension of the Routing

The Routing algorithm 3 can be directly applied for the chip-to-chip communication. However, since an inter-chip link is distributed to nodes, to use another inter-chip link, the inner-chip routing is needed. For inner-chip routing, we can use the common DOR. Here, let $(xc_{\text{cur}}, yc_{\text{cur}})$ and $(xc_{\text{dst}}, yc_{\text{dst}})$ be the coordinates of the current node inside a chip and destination node inside a chip. $(xc_{\text{next}}, yc_{\text{next}})$ shows the coordinates of the next target node inside a chip. Also, let $(xc_{\text{tci}}, yc_{\text{tci}})$ be the coordinates of the node connected with the TCI in the current chip $(x_{\text{cur}}, y_{\text{cur}}, z_{\text{cur}})$ and the next chip $(x_{\text{next}}, y_{\text{next}}, z_{\text{next}})$. Routing algorithm for multi-core chips is shown in Routing algorithm 4.

Routing algorithm 4

($(x_{\text{next}},y_{\text{next}},z_{\text{next}})$ is calculated by Routing algorithm 1)
$(xc_{\text{tci}}, yc_{\text{tci}})$ is determined by the $(x_{\text{next}},y_{\text{next}},z_{\text{next}})$

```
if ((x_cur, y_cur, z_cur) = (x_dst, y_dst, z_dst)) {
      (x_next,y_next,z_next) = (x_cur,y_cur,z_cur)
      (xc_next, yc_next) = DOR to (xc_dst, yc_dst)
} else if ((xc_cur, yc_cur) ≠ (xc_tci, yc_tci)) {
      (x_next,y_next,z_next) = (x_cur,y_cur,z_cur)
      (xc_next, yc_next) = DOR to (xc_tci, yc_tci)
} else {
      // (x_next,y_next,z_next) are not changed.
      (xc_next, yc_next) is determined automatically.
}
```

A problem of routing in $T_m[M, N, H, Mc, Nc]$ is deadlock possibility generated by the combining inner-chip routing and inter-chip routing. An example of cyclic dependency in $T_m[2,2,2,2,2]$ is shown in Fig. 9. In the same manner as linear stacking with multi-core chips, we introduce two virtual channels and a simple rule to use them. Since a cycle is only generated in the XY-plane including four sides of a rectangle consisting of inter-chip TCI links, we can resolve it by

changing the virtual channel on either side. We selected a simple VC transition for Routing algorithm 4 which changes VC when x coordinate is changed first.

VC transition for Routing algorithm 4

1. Only if x_{cur} is not x_{next}, set next VC to be 1.
2. Otherwise, next VC is the same as current VC.

Theorem 4.4: Routing algorithm 4 with the VC transition is deadlock free

Proof 4.4: Inter-chip routing uses Routing algorithm 3, so it is deadlock free. Inner-chip routing uses the DOR, so it is also deadlock free. The cycle is only generated the combination of inner-chip routing and inter-chip routing, thus, it is only generated in XY-plane since inner-chip routing is only done in the XY-plane. A generated cycle includes at least a link for x direction, so by changing the VC at the link, the cycle is removed. □

5. Evaluation

5.1 Evaluation of the Case of Single Core

We compare the network topology formed with the staggered stacking, linear stacking, and common 2D and 3D mesh. Booksim [15] is modified to treat the user defined topology, and used to evaluate the average latency and throughput. Parameters of the network simulation are shown in Table 1. Here, the speed of inner-chip link is assumed to be the same as that of TCI, and the uniform traffic is used. DOR is used for 2D and 3D mesh, Routing algorithm 1 is for topology S, and Routing algorithm 3 is for topology T, respectively. Fig. 12 shows network simulation results with 64 cores. Although the linear stacking denoted as S[11,11] has almost the same latency as the mesh, its bandwidth is worse because of the congestion on the stairway boundaries. On the other hand, the staggered stacking denoted as T[4,4,8] improves the latency by 28.8% compared to the mesh. In the staggered stacking, the packet can move both to x and y directions and z direction at the same time. As a result, the latency of T[4,4,8] is almost the same as that of the 4×4 mesh. The throughput of 3D mesh is greater than that of T[4,4,8] as the average degree of 3D mesh is greater than that of T[4,4,8].

Figure 13 shows network simulation results with 256 cores. The difference becomes larger as the size becomes large. T[8,8,8] improves the latency compared to the 2D mesh by 42.9%, and the throughput by 53.3% compared to the 2D mesh. In addition, T[8,8,8] improves the throughput of 3D mesh as the average degree of T[8,8,8] is greater than that of 3D mesh. These results are not surprising, since a node of T[M, N, H] has eight links, while 2D mesh and the linear stacking uses node with four links. By introducing multi-core chips connected with 2D mesh, we can reduce links of each node to four. Thus, the fair comparison will be done later.

Next, we evaluate the execution time of NAS Parallel Benchmark (NPB) [16] using the GEM5 [17], a full system CMP simulator. GEM5 can deal with both topology and routing defined by the user, but the routing is difficult to tailor. We modified GEM5 to deal with the routing of staggered stacking. Parameters for the full system simulation are shown in Table 2.

Table 1 Parameters for the network simulation

Number of simulation cycles	100000
Number of VCs	4
Buffer size of each VC	8
Number of the pipeline stage	3
Traffic	uniform
VC allocator	Round robin
internal speedup	1.0

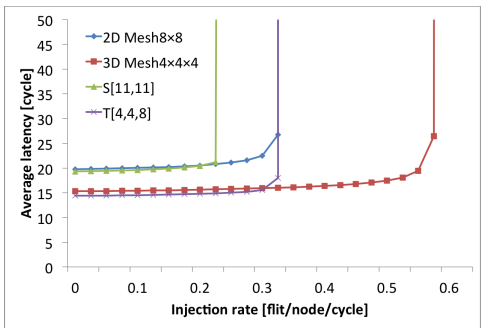

Fig. 12 Network simulation result with 64 cores

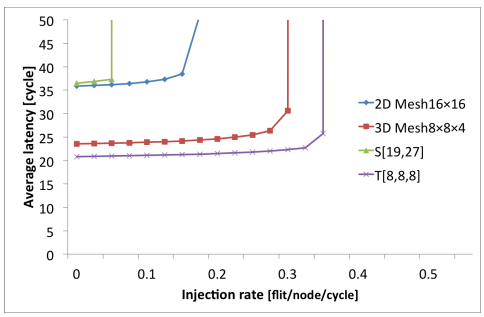

Fig. 13 Network simulation result with 256 cores

Table 2 Parameters for the full system simulation

Processor	X86_64
L1 I/D cache size	64KB
L1 cache latency	1 cycle
L2 cache bank size	256KB
L2 cache latency	6 cycles
Memory size	4GB
Memory latency	160 ± 2 cycles
Router pipeline	3 cycles
Buffer size	5 flits per VC
Flit size	128 bit
Coherency Protocol	MOESI directory
Number of VCs	4
VC allocator	Round robin

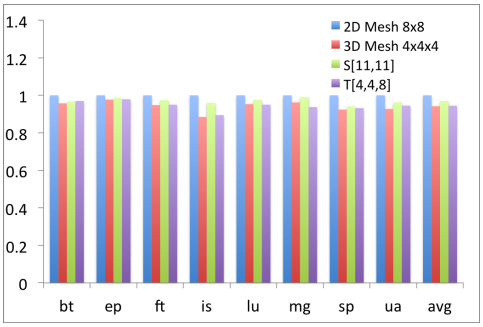

Fig. 14 Application execution time (64 cores)

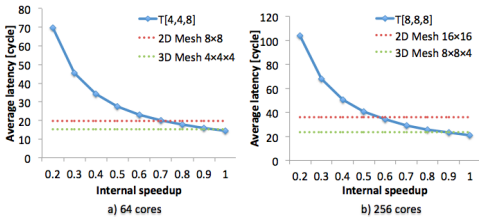

Fig. 15 The performance with various internal speed

Figure 14 shows full system simulation results with 64 routers. In this evaluation, a single CPU is allocated on the chip (0,0), (4,0), (10,0), (4,0), (4,10), (10,0), (10,4), (10,10) on the S[11,11], and (0,0,0), (3,0,1), (0,3,1), (3,3,0), (0,0,6), (3,0,7), (0,3,7), (3,3,6) on the T[4,4,8] to make the best use of the inter-chip network. L2 cache is allocated on the other nodes. We add the evaluation the 3D Mesh with four stacking of 4×4 mesh chips with TCI represented as 3D Mesh(4,4,4) in order to compare the staggered stacking and the stacking in only z direction. The application execution time is normalized to that of the mesh. The linear stacking denoted as S[11,11] reduces execution time by 3.0% on average compared to 2D mesh. Also, the staggered stacking denoted as T[4,4,8] reduces execution time by 5.6% on average compared to the 2D mesh, and achieves almost the same execution time as the 3D mesh.

5.2 Effect of TCI Bandwidth

Although the frequency clock of TCI is from 1GHz to 8GHz as shown in Sect. 2, there is possibility that inner-chip links are faster than inter-chip links. So, we evaluated network performance with serious internal speedup parameters to estimate the effect of TCI bandwidth. Here the internal speedup parameter shows the ratio of the speed of the TCI. In this evaluation, we assumed that all of inner-chip links have enough bandwidth, thus, clock frequency of the router is limited by inter-chip links. Figure 15 shows the network simulation results with various internal speedup parameters in BOOKSIM. The broken line shows the performance of the 2D mesh and 3D mesh with internal speedup 1.0. Injection rate is fixed to be 0.05, and all of other parameters are the same as the Table 1. When internal speedup is 0.5, that is, bandwidth of TCI is a half of inner-chip link, latency of

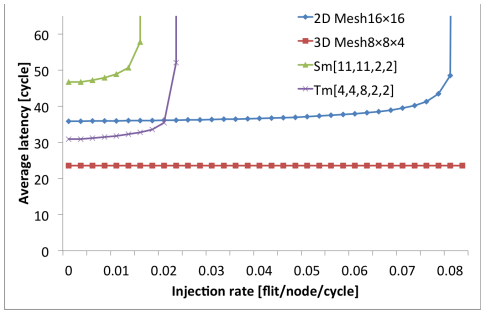

Fig. 16 Network simulation result with S_m and T_m

the T becomes larger than that of 2D and 3D mesh. Note that bandwidth of TCI can be widen by increasing the number of coils. For instance, if there are two TX and RX coils for one link, the bandwidth can be twice that of the communication with a single data coil.

5.3 Network Simulation for the Case of Using Multi-Cores

As mentioned in Sect. 5.1, the evaluation of the case of single core is not fair because the degree of a router of $T[M, N, H]$ is different from that of the mesh. However, a router in the $T_m[M, N, H, Mc, Nc]$ has at most 5 links which is same to a router in the mesh. Figure 16 shows network simulation result with 256 cores. Booksim is used again for simulation. Routing algorithm 2 is used for S_m, and Routing algorithm 4 is for T_m.

Since Routing algorithm 2 needs to use two VCs to remove cycle, the number of VCs becomes two in this evaluation. Other parameters are the same as Table 1. Unlike the case of a single core, the average degree of T_m[4,4,8,2,2] is smaller than 16×16 mesh. The throughput of the T_m[4,4,8,2,2] is, thus, lower than that of 16×16 mesh. However, T_m[4,4,8,2,2] improves the latency by 13.8% compared to the mesh when traffic load is light. Also, the latency and throughput of 3D mesh are better than those of T_m. On the other hand, as T_m is consisting of many tiny chips, it is feasible to form a many core architecture with much lower cost as shown in Sect. 5.5.

5.4 Full System Simulation

The execution time of NPB with multi-core systems with S_m[11,11,2,2] and T_m[4,4,8,2,2] is shown in Fig. 17. CPU allocation of S_m[11,11,2,2] and T_m[4,4,8,2,2] is the same as that of S[11,11] and T[8,8,4], respectively. Each chip has only a CPU attached to the most distant router from the router connected to TCI. Other routers are connected to L2 caches. Same as the simulation in the previous section, a GEM5 full-system simulator is used with the parameters shown in Table 2. The execution results are normalized to the ones with the 16×16 mesh. The arrangement of CPU and L2 cache is same to the simulation with 64 routers in Sect. 5.1. In all application programs, the staggered stack-

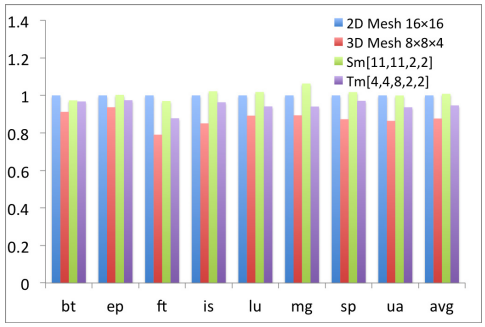

Fig. 17 Application execution time (256 cores)

Table 3 Single chip area evaluation (256 cores)

Topology	Number of chips	Area per chip
16 × 16mesh	1	768mm^2
T$_m$[4,4,8,2,2]	64	15.645mm^2

ing outperforms the 16×16 2D-mesh by 5.4%.

5.5 Chip Area and Cost

Considering the area used for TCI, the total semiconductor area for the staggered stacking is larger than that of the 2D mesh with the same number of cores. However, the cost of the chip is relational to more than the third power, and the system consisting of a small chip-stack can cost less than a large chip. Here, the area and cost of the staggered stacking are evaluated.

First, the area of TCI is evaluated. The coil for the TCI uses only two metal layers, and digital circuits can be implemented in the area of the coil. Thus, the footprint of the coil is not directly a loss of the chip area. However, here, we conservatively assume that the total area of coil is only used for the circuits for the TCI. The size of the coil is determined with the vertical distance to the opposite coil. Here, we assume that the chip is 30μm thick and 7.5μm is needed for glue. In this case, 8Gbps throughput is achieved with a 225μm × 225μm coil. To achieve the same throughput as the inner chip network, four coils for receiving data and four coils for sending data and a coil for the data transfer clock are needed. In staggered stacking, eight TCI links are needed, and thus, the total area of inter-chip communication becomes 225μm × 225μm × 72=3.645mm^2.

We assume a system with 256 cores each of which is implemented in an $a \times b$ size tile. The total area is represented by 256ab. On the other hand, in the case of staggered stacking T$_m$[4,4,8,2,2], a chip requires 4ab+2.645mm^2, so the total silicon area required becomes 256ab+233.28mm^2. That is 233.28mm^2 larger than the case of a single chip. From the Ref. [18], we assume the area of tile to be a=1.5mm and b=2.0mm. The estimated chip area with the above assumption is shown in Table 3. The cost of a chip is relational to more than the third power [19]. Thus, if we directly apply this formula, the semiconductor cost of staggered stacking is less than 1/2000 that of a large single chip. Considering the cost for stacking which is difficult to estimate now, this shows the possibility to build a large system economically by using staggered stacking method.

6. Conclusion

A novel chip stacking methods called linear stacking and staggered stacking are proposed to economically form a large multi-core system from a number of small chips. By using inductive coupling TCI, a large number of chips can be stacked in x, y, and z directions by keeping the height a certain number of chips. The network with 256 nodes formed by the proposed stacking improves the latency of 2D mesh by 13.8% and the performance of NAS Parallel Benchmarks by 5.4% on average compared to 2D mesh. The estimation revealed that the allocation of the chip greatly influences the performance. Investigating an allocation method for building large-scale CMPs by using the chip stacking is our future work.

Acknowledgments

This work was partially supported by JSPS KAKENHI S Grant Number 25220002.

References

[1] J. Burns, L. McIlrath, C. Keast, C. Lewis, A. Loomis, K. Warner, and P. Wyatt, "Three-dimensional integrated circuits for low-power, high-bandwidth systems on a chip," Proceedings of the IEEE International Solid-State Circuits Conference, pp.268–269, Feb. 2001.
[2] H. Amano, "Castle of Chips: A New Chip Stacking Structure with Wireless Inductive Coupling for Large Scale 3-D Multicore Systems," Proceedings of 15th International Conference on Network-Based Information Systems, pp.820–825, 2012.
[3] H. Nakahara, T. Ozaki, H. Matsutani, M. Koibuchi, and H. Amano, "Expandable chip stacking method for many-core architectures consisting of tiny chips," 2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), pp.41–48, 2015.
[4] Y. Take, H. Matsutani, D. Sasaki, M. Koibuch, T. Kuroda, and H. Amano, "3D NoC with Inductive-Coupling Links for Building-Block SiPs," IEEE Trans. Comput., vol.63, no.3, pp.748–763, March 2014.
[5] K. Niitsu, Y. Shimazaki, Y. Sugimori, Y. Kohama, K. Kasuga, I. Nonomura, M. Saen, S. Komatsu, K. Osada, N. Irie, T. Hattori, A. Hasegawa, and T. Kuroda, "An inductive-coupling link for 3d integration of a 90nm cmos processor and a 65nm cmos sram," Proceedings of the IEEE International Solid-State Circuits Conference, pp.480–481,481a, Feb. 2009.
[6] Y. Kohama, Y. Sugimori, S. Saito, Y. Hasegawa, T. Sano, K. Kasuga, Y. Yoshida, K. Niitsu, N. Miura, H. Amano, and T. Kuroda, "A scalable 3D processor by homogeneous chip stacking with inductive-coupling link," Proceedings of the VLSI Circuits Symposium, pp.94–95, June 2009.
[7] N. Miura, Y. Koizumi, Y. Take, H. Matsutani, T. Kuroda, H. Amano, R. Sakamoto, M. Namiki, K. Usami, M. Kondo, and H. Nakamura, "A Scalable 3D Heterogeneous Multicore with an Inductive ThruChip Interface," IEEE Micro, vol.33, no.6, pp.6–15, 2013.
[8] N. Miura, H. Ishikuro, T. Sakurai, and T. Kuroda, "A 0.14pJ/b Inductive-Coupling Inter-Chip Data Transceiver with Digitally-Controlled Precise Pulse Shaping," Proceedings of the International Solid-State Circuits Conference (ISSCC'07), pp.358–359,

Feb. 2007.
[9] D.U. Lee, K.W. Kim, K.W. Kim, K.S. Lee, S.J. Byeon, J.H. Kim, J.H. Cho, J. Lee, and J.H. Chun, "A 1.2 v 8 gb 8-channel 128 gb/s high-bandwidth memory (hbm) stacked dram with effective i/o test circuits," IEEE J. Solid-State Circuits, vol.50, no.1, pp.191–203, Jan. 2015.
[10] M. Saito, Y. Yoshida, N. Miura, H. Ishikuro, and T. Kuroda, "47% power reduction and 91% area reduction in inductive-coupling programmable bus for nand flash memory stacking," Proc. IEEE Trans. Circuits Syst., vol.57, no.9, pp.2269–2278, Sept. 2010.
[11] P.P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," IEEE Trans. Comput., vol.54, no.8, pp.1025–1040, Aug. 2005.
[12] J. Nguyen, J. Pezarts, G. Pratt, and S. Ward, "Three-Dimensional Network Topologies," Parallel Computer and Communication (K.Bolding and L. Synder, eds), vol.853, pp.101–115, 1994.
[13] W.S. Lacy, J.L. Cruz-Rivera, and D.S. Wills, "The Offset Cube: A Three-Dimensional Multicomputer Network Topology Using Through-Wafer Optics," IEEE Trans. Parallel Distrib. Syst., vol.9, no.9, pp.893–908, 1998.
[14] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing," Proceedings of 19th International Symposium on Computer Architecture, pp.278–287, 1992.
[15] W.J. Dally and B.P. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.
[16] H. Jin, M. Frumkin, and J. Yan, "The OpenMP Implementation of NAS Parallel Benchmarks and Its Performane," NAS Technical Report NAS-99-011, Oct. 1999.
[17] N. Binkert, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M.D. Hill, D.A. Wood, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, and T. Krishna, "The gem5 Simulator," ACM SIGARCH Computer Architecture News, vol.39, no.2, pp.1–7, May 2011.
[18] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-tile sub-100-w teraflops processor in 65-nm cmos," IEEE Jounal of Solid-State Circuits, vol.43, no.1, pp.29–41, Jan. 2008.
[19] J.L. Hennessy and D.A. Patterson, Computer Architecture, Fifth Edition: A Quantitative Approach, Morgan Kaufmann, 2011.

Hiroshi Nakahara received B.S. degree from Keio University, Yokohama, Japan, in 2015. He is a master student in Keio university in the presence.

Tomoya Ozaki received B.S., M.E. degree from Keio University, Yokohama, Japan, in 2014 and 2016 respectively.

Hiroki Matsutani received the B.A., M.E., and Ph.D. degrees from Keio University in 2004, 2006, and 2008, respectively. He is currently an assistant professor in the Department of Information and Computer Science, Keio University. From 2009 to 2011, he was a research fellow in the Graduate School of Information Science and Technology, The University of Tokyo, and awarded a Research Fellowship of the Japan Society for the Promotion of Science.

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 Gradua te University of Advanced Studies, Tokyo, Japan. His research interests include the area of high-performance computing and interconnection networks. He is a member of the IEEE and a senior member of IEICE 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.