Expandable Chip Stacking Method for Many-Core Architectures Consisting of Tiny Chips

Hiroshi Nakahara[1], Tomoya Ozaki[1], Hiroki Matsutani[1], Michihiro Koibuchi[2], and Hideharu Amano[1]

[1]Keio University
3-14-1 Hiyoshi, Kohoku-ku,
Yokohama, Japan
blackbus@am.ics.keio.ac.jp

[2]National Institute of Informatcs
2-1-2 Hitotsubashi, Chiyoda-ku,
Tokyo, Japan
koibuchi@nii.ac.jp

***Abstract*—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. To connect a large number of small chips for building a large scale system, a novel chip stacking method called the staggered stacking is proposed that enables the system to be extended to x and y dimensions, not only to z dimension. For such flexible inter-chip communication, we use Inductive coupling ThruChip Interface (TCI). Here, a novel chip staking layout, and its deadlock-free routing design for the case using 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 6.7% on average compared to that of 2D mesh.**

I. 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. They would have a potential to be a competitor or complementary technology to a large SoC. Needless to say, they stack chips only for vertical direction and their structures are fixed and cannot be changed once they are stacked.

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, a novel chip stacking method called the staggered stacking is proposed that enables the system to be extended to x and y dimensions, not only to z dimension. This method interestingly allows 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) [2]. 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. A transfer speed over 8 Gbps was achieved with a low-energy dissipation and a low bit-error rate (BER$< 10^{-12}$) [3]. The TCI has been used for memory stacking [4], a dynamically reconfigurable processor [5], and a heterogeneous multi-core system [6]. 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 the novel chip stacking method are (1) chip staking layout for TCI using x,y and z dimensions, and (2) the routing algorithm for the case using multi-core chips.

The contributions of the paper are as follows:

- The staggered stacking builds more than 64-chip stacking with eight-chip height. It provides a mesh-based 3D topology only with TCI links naturally formed by the chip-stacking.
- The case that a chip has a multi-core connected with 2D mesh is investigated. By distributing TCI links to corner nodes, that achieves better diameter and average shortest path length than 2D mesh with the same degree. An extended deadlock-free routing algorithm with two virtual channels achieves better application performance (Section 4).
- The performance of the various cases of chip stacking is evaluated with both typical traffic and these from the practical applications. (Section 3 and Section 4)

The rest of paper is organized as follows. Section II introduces inductive-coupling TCI technology. Section III proposes a new chip-stacking method, called staggered stacking, corresponding network toplogy and a routing algorithm for the case to stack multi-core. Section IV evaluates these proposed techniques and Section V concludes the paper.

II. Inductive Coupling Through Chip Interface

A. *Inductive coupling channels*

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

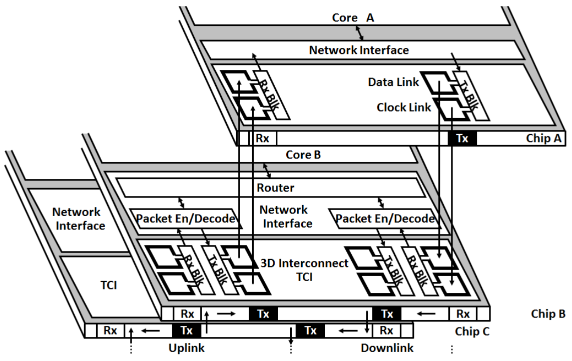

Fig. 1: 3D NoC using TCIs

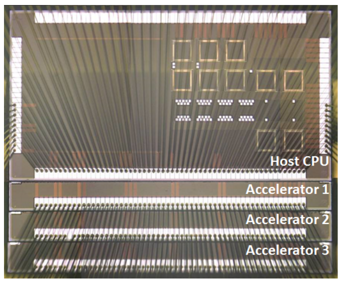

Fig. 2: Chip stacking used in Cube-1

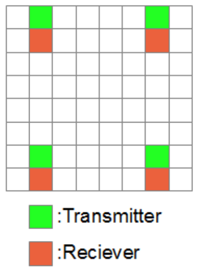

Fig. 3: Chip used in linear stacking

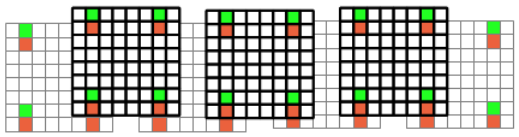

Fig. 4: Example of stacking seven chips in accordance with linear stacking

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$< 10^{-12}$) [3].

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 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.

B. Chip stacking and inter-chip networks

Although a number of practical systems have been developed by using TCI, all use simple 3D chip stacking. Fig. 2 shows the chip stacking used in Cube-1[6]. To place the receiver coil just on the transceiver coil, chips are shifted and stacked. The shifted space is also used to maintain the space for wire bonding. Note that even in TCI, several wires are needed for the power supply. In Cube-1, a ring-like packet switching network is formed just by stacking chips.

Although a case has been reported in which more than 10 chips were stacked [7], 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.

C. Extension of stacking to x-dimension

To connect more chips with the limitations of z-dimension, linear stacking has been proposed [8]. This method assumes four TX and RX channels on a chip as shown in Fig. 3. By using each stacked chip as a bridge between chips, the stacking can be extended to x-dimension as shown in Fig. 4, resulting in more than 64 chips being connected within an eight-chip height. With the linear stacking, a type of mesh network called stairway boundary mesh (SBM) is formed between chips as shown in Fig. 5. Although the dimension order routing (DOR) can be used in the SBM, packets must move around the boundary when the DOR path faces to it. Since the routing introduces congestion around the boundary, the throughput is degraded. Thus, although SBM has almost the same average hop count as common mesh, its throughput is much smaller.

D. Stacking methods for other through-chip interfaces

Three dimensional stacking methods have been studied for various through-chip interface techniques, and some of them are based on the same motivation to extending x, y and

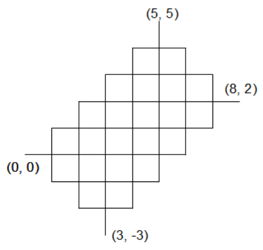

Fig. 5: Stairway boundary mesh with 39 chips

z dimensions[9]. The offset cube[10] proposed for through-wafer optics is 3D stacking method for building ultra-compact massively parallel processors. As shown in later, the generated network topology of the offset cube, the stacking method shown in [9], and that of the Staggered Stacking is the same. However, our stacking method is specialized to the TCI for avoiding the interference between inductors. Moreover, our proposed method is designed for chips which include multiple cores. Our proposed chip stacking method is specialized on these points.

III. Staggered stacking

We propose a new stacking method, called staggered stacking. The aim of the stacking is to extend the system to x and y dimensions, not only z dimension.

A. The stacking method

The limitation of the linear stacking is caused by extending the chip stack to only x-dimension. 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 uni-direction links in Fig. 3. 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 3.1: 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. 6a). Then, in the same manner, we place layers 2 and 3 as shown in Fig. 6b). 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. Fig. 7 is a cross-cutting view of the chip stack shown in Fig. 6b).

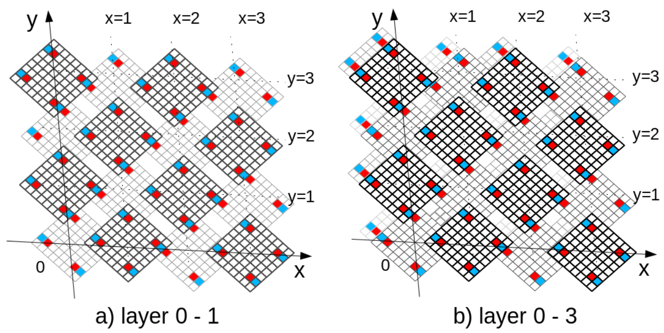

Fig. 6: 32-chip stacking with staggered stacking

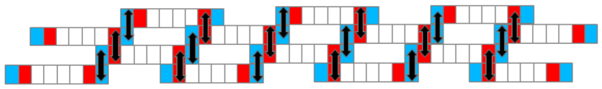

Fig. 7: Cross cutting view of Fig. 6b)

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.

B. 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 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 3.2: 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: $0 \leq x < N$, $0 \leq y < M$, and $0 \leq z < H$. □

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

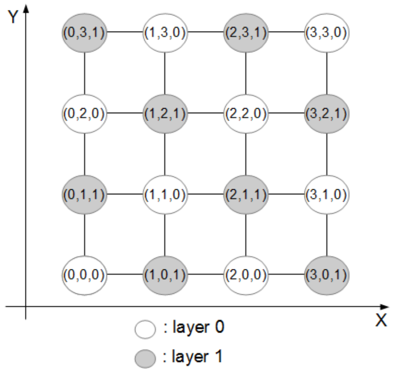

Fig. 8: Topology composed in Fig. 6a) (4×4 mesh)

The topology generated by the staggered stacking itself is the same as that of Topology B[9] and the offset cube[10]. We will extend it to stack multi-core chips which have multiple cores in a chip connected with each other.

C. The Network Topology for Multi-Cores

Recently, even in a small chip, multiple cores connected with a NoC each other are implemented. Here, we assume that cores in the chip are connected with a 2D mesh. A router in 2D mesh usually 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 two can be used for up-link and down-link of TCI without changing the router structure. 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 [2], 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 $\mathrm{T}_m[M, N, H, Mc, Nc]$ is defined.

Definition 3.3: Assume that $Mc \times Nc$ nodes in a chip are connected with 2D mesh. Add two TCI links for off-chip connections to 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 $\mathrm{T}_m[2, 2, 2, 2, 2]$. 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.

D. Routing

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

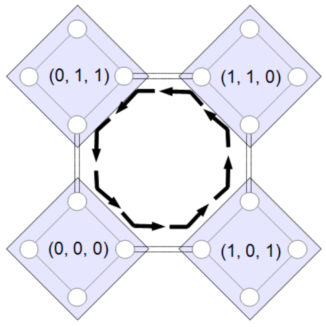

Fig. 9: Topology of [2,2,2,2,2]

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 $(x_{\mathrm{cur}}, y_{\mathrm{cur}}, z_{\mathrm{cur}})$ be the source node, and $(x_{\mathrm{dst}}, y_{\mathrm{dst}}, z_{\mathrm{dst}})$ be the destination node. Here, we define the absolute distance between the current node and the destination node as $dx = |x_{\mathrm{cur}} - x_{\mathrm{dst}}|$, $dy = |y_{\mathrm{cur}} - y_{\mathrm{dst}}|$, and $dz = |z_{\mathrm{cur}} - z_{\mathrm{dst}}|$. 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 $(x_{\mathrm{dst}}, y_{\mathrm{dst}})$ 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 $(x_{\mathrm{dst}}, y_{\mathrm{dst}})$ before z

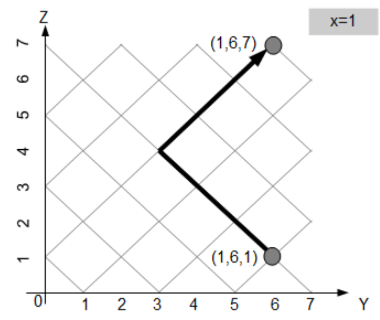

Fig. 10: Routing on YZ-plane in the case of $(dx + dy) < dz$

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 Algorithm 1. Here, let $(x_{\text{next}}, y_{\text{next}}, z_{\text{next}})$ be the coordinates of the next chip determined by the algorithm.

Proof of deadlock-freedom of routing algorithm 1 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 3.1:
When sending a packet to its neighboring chip, all the (x,y,z) values never change simultaneously

Proof 3.1:
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 3.2:
Routing Algorithm 1 is deadlock free on XY-plane, XZ-plane, and YZ-plane.

Proof 3.2:

1) XY-plane
Turns on the XY-plane are shown in Fig. 11a). As the routing on the xy-axis is the DOR, two turns $((x+1, y)-(x+1, y+1)-(x, y+1)$ and $(x, y+1)-(x, y)-(x+1, y))$

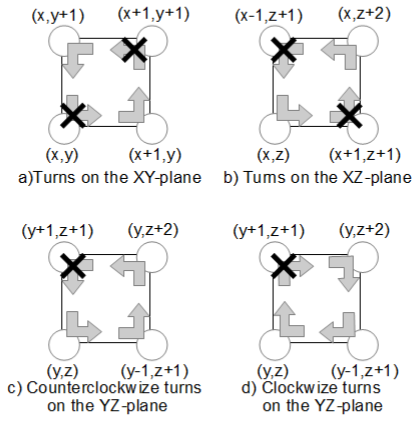

Fig. 11: Turns on each plane

are prohibited. Cycles are, thus, never formed in this plane.

2) XZ-plane
Turns on the XZ-plane are shown in Fig. 11b). 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 Fig. 11c) and 11d). According to Algorithm 1, 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. 11c) and a turn $(y, z+2)-(y+1, z+1)-(y, z)$ in Fig. 11d) are prohibited except if y equals 0. If y doesn't equal 0, since these prohibited turns are known to be negative-first routing [12], 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 3.3: Routing Algorithm 1 is deadlock free.

Proof 3.3: From Lemma 3.1, Routing Algorithm 1 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 1 is deadlock free. □

2) Extension of the routing: The Routing Algorithm 1 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 2.

Routing Algorithm 1

```
if ($dx + dy < dz$) {
    ($x_{\mathrm{next}}, y_{\mathrm{next}}$) = DOR to ($x_{\mathrm{next}}, y_{\mathrm{next}}$)

    if ($z_{\mathrm{cur}} \neq z_{\mathrm{dst}}$) making $z_{\mathrm{next}}$ move close to $z_{\mathrm{dst}}$
    else $z_{\mathrm{next}}$ is $z_{\mathrm{cur}} + 1$
} else {
    if ($x_{\mathrm{cur}} \neq x_{\mathrm{dst}}$)
        making $x_{\mathrm{next}}$ move close to $x_{\mathrm{dst}}$
    else {
        if ($dy > dz$) $y_{\mathrm{next}}$ is $y_{\mathrm{cur}} + 1$
        else if ($dy < dz$) $y_{\mathrm{next}}$ is $y_{\mathrm{cur}-1}$
        else making $y_{\mathrm{next}}$ move close to $y_{\mathrm{dst}}$
    }
    making $z_{\mathrm{next}}$ move close to $z_{\mathrm{dst}}$
}
```

Routing Algorithm 2

```
(($x_{\mathrm{next}}$,$y_{\mathrm{next}}$,$z_{\mathrm{next}}$) is calculated by routing algorithm1)
($xc_{\mathrm{tci}}, yc_{\mathrm{tci}}$) is determined by the ($x_{\mathrm{next}}$,$y_{\mathrm{next}}$,$z_{\mathrm{next}}$)

if (($x_{\mathrm{cur}}$, $y_{\mathrm{cur}}$, $z_{\mathrm{cur}}$) = ($x_{\mathrm{dst}}$, $y_{\mathrm{dst}}$, $z_{\mathrm{dst}}$)) {
    ($x_{\mathrm{next}}$,$y_{\mathrm{next}}$,$z_{\mathrm{next}}$) = ($x_{\mathrm{cur}}$,$y_{\mathrm{cur}}$,$z_{\mathrm{cur}}$)
    ($xc_{\mathrm{next}}, yc_{\mathrm{next}}$) = DOR to ($xc_{\mathrm{dst}}, yc_{\mathrm{dst}}$)
} else if (($xc_{\mathrm{cur}}, yc_{\mathrm{cur}}$) $\neq$ ($xc_{\mathrm{tci}}, yc_{\mathrm{tci}}$)) {
    ($x_{\mathrm{next}}$,$y_{\mathrm{next}}$,$z_{\mathrm{next}}$) = ($x_{\mathrm{cur}}$,$y_{\mathrm{cur}}$,$z_{\mathrm{cur}}$)
    ($xc_{\mathrm{next}}, yc_{\mathrm{next}}$) = DOR to ($xc_{\mathrm{tci}}, yc_{\mathrm{tci}}$)
} else {
    // ($x_{\mathrm{next}}$,$y_{\mathrm{next}}$,$z_{\mathrm{next}}$) are not changed.
    ($xc_{\mathrm{next}}, yc_{\mathrm{next}}$) is determined automatically.
}
```

A problem of routing in $\mathrm{T}_m[M, N, H, Mc, Nc]$ 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. 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 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 2 which changes VC when x coordinate is changed first.

Theorem 3.4: Algorithm 2 with the VC transition is deadlock free

Proof 3.4: Inter-chip routing uses Algorithm 1, 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. □

TABLE I: 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

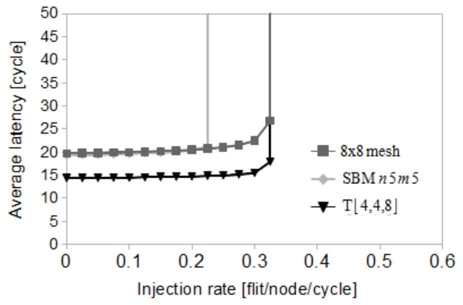

Fig. 12: Network simulation result with 64 cores

VC transition for Routing Algorithm 2

```
if (($xc_{\mathrm{next}}$, $yc_{\mathrm{next}}$) = ($xc_{\mathrm{tci}}, yc_{\mathrm{tci}}$) and ($x_{\mathrm{cur}} \neq x_{\mathrm{dst}}$))
    next VC is 0
else if (($xc_{\mathrm{cur}}$, $yc_{\mathrm{cur}}$) = ($xc_{\mathrm{tci}}, yc_{\mathrm{tci}}$) and ($x_{\mathrm{cur}} \neq x_{\mathrm{next}}$))
    next VC is 1
else
    next VC is same as current VC
```

IV. Evaluation

A. Evaluation of the case of single core

We compare the network topology formed with the staggered stacking, linear stacking, and common 2D mesh. Booksim [13] 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. I.

Fig. 12 shows network simulation results with 64 cores. Although the linear stacking denoted as SBM 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.

Fig. 13 shows network simulation results with 256 cores. The difference becomes larger with as the size becomes larger. The staggered stacking improves the latency compared to the mesh by 42.9%, and the throughput by 53.3% compared to the mesh.

These results are not surprising, since a node of $\mathrm{T}[M, N, H]$ 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) [14] using the GEM5 [15], a full system

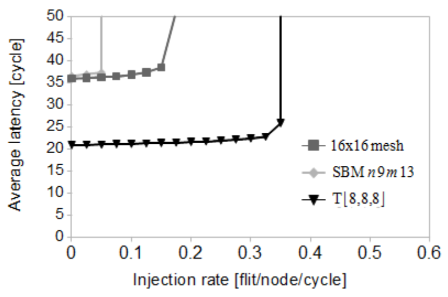

Fig. 13: Network simulation result with 256 cores

TABLE II: 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

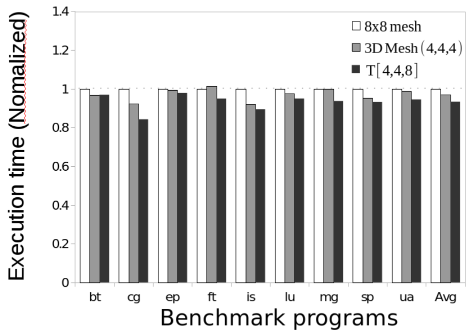

Fig. 14: Application execution time (64 cores)

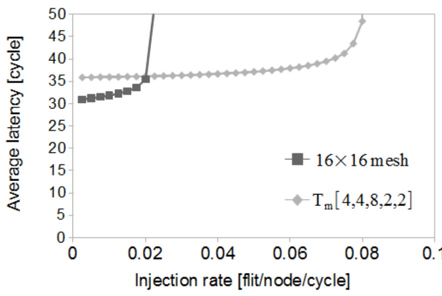

Fig. 15: Network simulation result with T_m[4,4,8,2,2] and 16×16 mesh

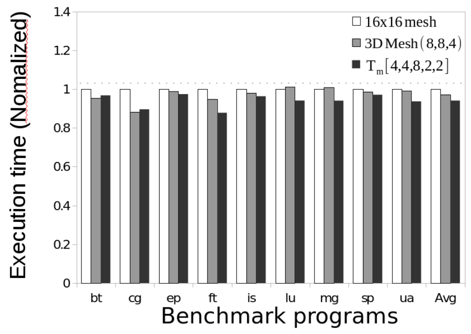

Fig. 16: Application execution time (256 cores)

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. II.

Fig. 14 shows full system simulation results with 64 routers. In this evaluation, a single CPU is allocated on the chip (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 of four stacking of 4x4 mesh chips with TCI represented as TCI(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 staggered stacking denoted as T[4,4,8] reduces execution time by 4.6% on average compared to the mesh, and by 3.4% compared to TCI(4,4,4).

B. *Network simulation for the case of using multi-cores*

As mentioned in the Section IV-A, 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. Fig. 15 shows network simulation result with 256 cores. Booksim is used again for simulation. 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. I. Unlike the case of a single core, the average degree of T_m[4,4,8,2,2] is smaller than 16x16 mesh. The throughput of the T_m[4,4,8,2,2] is, thus, lower than that of 16x16 mesh. However, T_m[4,4,8,2,2] improves the latency by 13.8% compared to the mesh when traffic load is light.

C. *Full system simulation*

The execution time of NPB with multi-core systems with T_m[4,4,8,2,2] is shown in Fig. 16. Same as the simulation in the previous section, a GEM5 full-system simulator is used with the parameters shown in Table. II. The execution results are normalized to the ones with the 16x16 mesh. The arrangement of CPU and L2 cache is same to the simulation with 64 routers in Section IV-A. In all application programs, the staggered stacking performs the best and outperforms the 16x16 2D-mesh by 6.7% on average, and the TCI(8,8,4) by 3.4%.

TABLE III: 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

D. 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 x 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 x 225μm x 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 reference [16], we assume the area of tile to be a=1.5mm and b=2.0mm. The cost of a chip is relational to more than the third power[17]. 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.

V. Conclusion

A novel chip stacking method called staggered stacking is 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 6.7% 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.

Acknowledgment

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," in *Proceedings of the IEEE International Solid-State Circuits Conference*, Feb 2001, pp. 268–269.

[2] Y. Take, H. Matsutani, D. Sasaki, M. Koibuch, T. Kuroda, and H. Amano, "3-D NoC with Inductive-Coupling Links for Building-Block SiPs," *IEEE Transactions on Computers (TC)*, vol. 63, no. 3, pp. 748–763, Mar. 2014.

[3] 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," in *Proceedings of the International Solid-State Circuits Conference (ISSCC'07)*, Feb. 2007, pp. 358–359.

[4] 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," in *Proceedings of the IEEE International Solid-State Circuits Conference*, Feb 2009, pp. 480–481,481a.

[5] 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," in *Proceedings of the VLSI Circuits Symposium*, June 2009, pp. 94–95.

[6] N. Miura and et al, "A Scalable 3D Heterogeneous Multicore with an Inductive ThruChip Interface," in *IEEE Micro, Vol.33, No.6*, 2013, pp. 6–15.

[7] 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," *Proceedings of the IEEE Transactions on Circuits and Systems*, vol. 57, no. 9, pp. 2269–2278, Sept 2010.

[8] H. Amano, "Castle of Chips: A New Chip Stacking Structure with Wireless Inductive Coupling for Large Scale 3-D Multicore Systems," in *Proceedings of 15th International Conference on Network-Based Information Systems*, 2012, pp. 820–825.

[9] 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.

[10] W. Lacy, J. L. Cruz-Rivera, and D. Wills, "The Offset Cube: A Three-Dimensional Multicomputer Network Topology Using Through-Wafer Optics," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 9, no. 0, pp. 893–908, 1998.

[11] P. P. Pande, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers (TC)*, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.

[12] C. J. Glass and L. M. Ni, "The Turn Model for Adaptive Routing," in *Proceedings of 19th International Symposium on Computer Architecture*, 1992, pp. 278–287.

[13] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[14] H. Jin, M. Frumkin, and J. Yan, "The OpenMP Implementation of NAS Parallel Benchmarks and Its Performane," in *NAS Technical Report NAS-99-011*, Oct. 1999.

[15] N. Binkert, B. Beckmann, G. Black, S. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. Hill, and D. Wood, "The gem5 Simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, May 2011.

[16] S. 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.

[17] J. L. Hennessy and D. A. Patterson, *Computer Architecture, Fifth Edition : A Quantitative Approach*. Morgan Kaufmann, 2011.