Three-Dimensional Layout of On-Chip Tree-Based Networks*

Hiroki Matsutani[1], Michihiro Koibuchi[2], D. Frank Hsu[3], and Hideharu Amano[1]

[1]Keio University
3-14-1, Hiyoshi, Kohoku-ku, Yokohama,
JAPAN 223-8522
{matutani,hunga}@am.ics.keio.ac.jp

[2]National Institute of Informatics
2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo,
JAPAN 101-8430
koibuchi@nii.ac.jp

[3]Fordham University
113 West 60th Street, New York, NY 10023, USA
hsu@cis.fordham.edu

Abstract

Three-dimensional Network-on-Chip (3-D NoC) is an emerging research area exploring the network architecture of 3-D ICs that stack several smaller wafers or dice for reducing wire length and wire delay.

Various network topologies such as meshes, tori, and trees have been used for NoCs. In particular, much attention has been focused on tree-based topologies, such as Fat Trees and Fat H-Tree, because of their relatively short hop-count that enables lower latency communication compared to meshes or tori. However, since on-chip tree-based networks in their 2-D layouts have long wire links around the root, they generate serious wire delay, posing severe problems to modern VLSI design. In this paper, we propose a 3-D layout scheme of trees including Fat Trees and Fat H-Tree for 3-D ICs in order to resolve the trees' intrinsic disadvantage. The 3-D layouts are compared with the original 2-D layouts in terms of network logic area, wire length, wire delay, number of repeaters inserted, and energy consumption. Evaluation results show that 1) total wire length is reduced by 25.0% to 50.0%; 2) wire delay is improved and repeater buffers that consume considerable energy can be removed; 3) flit transmission energy is reduced by up to 47.0%; 4) area overhead is at most 7.8%, which compares favorably to those for 3-D mesh and torus.

1 Introduction

As semiconductor technology improves, the number of processing cores integrated on a single chip has continually increased. To connect many cores on a chip, Network-on-Chips (NoCs) [4, 1, 14] that introduce a packet-switched network structure have been widely employed instead of traditional bus-based on-chip interconnects.

On-chip network topology is a crucial factor of the chip in terms of performance, cost, and energy consumption. Various network topologies have been studied for NoCs. Especially, two-dimensional mesh[14] and torus[4] are popularly used in NoCs, because their grid-based regular arrangement is intuitively considered to be matched to the two-dimensional VLSI layout. On the other hand, much attention has been focused on tree-based topologies, such as Fat Trees[10], because of their relatively short hop-count that enables lower latency communication compared to that for mesh or torus. As an extension to Fat Trees, we proposed a novel tree-based topology called Fat H-Tree[12] that provides a torus structure by combining two trees. We also proposed its 2-D layout. Fat H-Tree and its 2-D layout were evaluated in terms of network logic area, wire length, wire delay, and energy consumption[12]. As a result, Fat H-Tree outperforms Fat Trees in terms of cost-performance, and it also achieves comparable performance to torus while its hardware cost is less than torus. However, since on-chip tree-based interconnects have long wire links near the root of the tree, they have been found to generate a serious wire delay which poses severe problems in VLSI design.

An attractive solution to the wire delay problem is 3-D IC technology that stacks multiple wafers or dice using vertical interconnects[2, 5, 6]. Various 3-D interconnect approaches have been proposed: wire-bonding between stacked chips, microbump technology[2], contactless (i.e., wireless), and through-via between stacked wafers[5, 6]. By using these techniques, current concept of NoCs is being extended to 3-D NoCs[11, 9, 13]. In this paper, we assume through-wafer via technology, which is expected to offer both very high density of vertical interconnects and very short distance between wafers. The distance between wafers can range from 5μm to 50μm[11], which is much shorter than the

*This work was supported by Joint Research Fund, "Network-on-Chip Architecture," National Institute of Informatics. We would like to thank VLSI Design and Education Center (VDEC) and Kyoto University for a design flow of ASPLA/STARC 90nm CMOS process.

wire length between cores on a tier, and the pitches of a through-wafer via can range from 1μm to 10μm square[5, 6, 11], depending on the manufacturing process such as wafer-to-wafer alignment.

From the view point of graph theory, 3-D layout methods of de Bruijn and Pyramid networks have been proposed[15, 16]. In this paper, we propose a 3-D layout method of Fat Trees and Fat H-Tree, and the proposed method is evaluated in terms of network logic area, wire length, wire delay, and energy consumption by using NoC circuits. The rest of this paper is organized as follows. Section 2 introduces Fat Trees and Fat H-Tree, and Section 3 proposes their 3-D layouts. Section 4 compares the proposed 3-D layouts with the original 2-D layouts and Section 5 concludes this paper.

2 Two-Dimensional Layout

Figures 1-6 show examples of tree-based topologies with 64 cores, where a white circle represents a network interface of the core and a shaded square represents a router connecting other routers or network interfaces. They have different numbers of routers, different link lengths, and different numbers of ports per router, all of which affect throughput, amount of hardware for network resources, and energy consumption.

2.1 Fat Trees

Fat Trees[10] can be expressed with a tuple (p, q, c), where p is the number of upward connections, q is the number of downward connections, and c is the number of upward connections that each core has. Figure 2 shows a Fat Tree (2,4,1), in which each router (except for top-rank routers) has two upward and four downward connections, and each core has one upward link. Fat Tree (2,4,1) is used in [7]. Note that a Fat Tree (1,4,1) is identical to the H-Tree (Figure 1).

The top-rank link between a top-rank router and its children requires the longest wire whose length is $L/2$ in a tree placed in an $L \times L$ chip. Thus, the wire delay on these links grows quadratically as the chip size enlarges, or repeater buffers that consume additional power will be inserted to mitigate the wire delay.

2.2 Fat H-Tree

Fat H-Tree[12] is a novel tree-based network with a torus structure. It is formed by combining two H-Tree networks, called **red tree** and **black tree**. Similar to the Fat Tree (2,4,2), every processing core in a Fat H-Tree has two ports: one for connecting to the red tree and the other to the black tree. The network interface in a Fat H-Tree has a function which is capable of forwarding packets from the red tree to the black tree and vice versa. This function provides torus-like alternative paths, which greatly improve its performance.

a) Red Tree Figure 3 shows a Fat H-Tree with a red tree, where the number labeled at a router (e.g., 1,2, or 3) is the rank of that router in the tree. Assume that $4^n = 2^{2n}$ cores are aligned in a $2^n \times 2^n$ two-dimensional grid square, and two-dimensional coordinates (x, y) are assigned to each core. We call such a core a rank-0 router from the network point of view. For a rank-0 router (x, y), the red-tree coordinates $R(r_0, r_1, ...r_{n-1})$ are assigned as follows.

$$r_i = ((x/2^i) \bmod 2) + 2 \times ((y/2^i) \bmod 2) \quad (1)$$

For each i from 0 to $n-1$, four rank-i red routers $R(r_i, ...r_{n-1})$ having the same part of coordinates $R(r_{i+1}, ...r_{n-1})$ are connected to the rank-$(i+1)$ red router labeled with $R(r_{i+1}, ...r_{n-1})$. The top-rank router in the red tree has thus coordinates R. Figure 3 shows R, $R(0)$, $R(2,0)$, and $R(2,2,0)$ as examples of red-tree coordinates.

b) Black Tree Figure 4 shows the black tree, which is located to the lower right of the red tree. For a rank-0 router (x, y), the black-tree coordinates $B(b_0, b_1, ...b_{n-1})$ are assigned as follows.

$$b_i = ((((x-1) \bmod 2^n)/2^i) \bmod 2) + 2 \times ((((y-1) \bmod 2^n)/2^i) \bmod 2) \quad (2)$$

For each i from 0 to $n-1$, four rank-i black routers $B(b_i, ...b_{n-1})$ having the same part of coordinates $B(b_{i+1}, ...b_{n-1})$ are connected to the rank-$(i+1)$ black router labeled with $B(b_{i+1}, ...b_{n-1})$. Figure 4 shows B, $B(2)$, $B(1,2)$, and $B(0,1,2)$ as examples of black-tree coordinates.

c) Fat H-Tree On the set of $2^n \times 2^n = N \times N$ rank-0 routers (or cores), a Fat H-Tree FH(N) is formed with an n-rank red tree and an n-rank black tree. Fat H-Tree has a torus structure, which is formed with rank-0 and rank-1 routers in both trees.

In the same manner as a folded two-dimensional torus, a Fat H-Tree can be folded to avoid long feedback links laid across the chip (e.g., links connecting the rightmost/top router and the leftmost/bottom router). As shown in Figure 6, the order of nodes is changed so that every link is connected to the next neighboring node. However, each link, except for top-rank links connecting to the top-rank router, requires twice the wire length of the same-sized H-Tree. The top-rank links become very short because of the folded layout (see Section 4.2). Thus, the next link from the root becomes the longest in a folded Fat H-Tree, and its length is the same as that of the longest link in the H-Tree and Fat Trees.

3 Three-Dimensional Layout

Tree-based topologies such as Fat Trees and Fat H-Tree in their 2-D layouts use relatively large wire resources as their network size increases. Here, we propose a 3-D layout method that divides the original planar network into several parts and connects them by

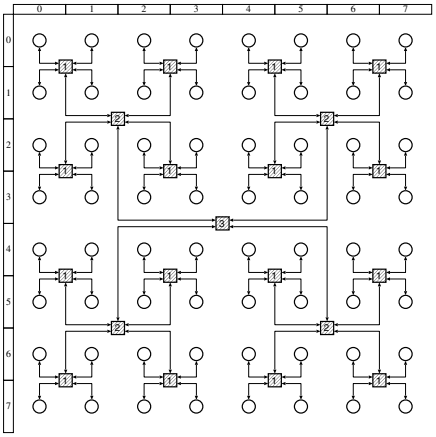


Figure 1. H-Tree

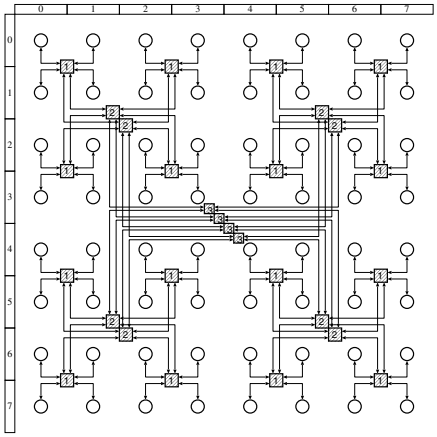


Figure 2. Fat Tree (2,4,1)

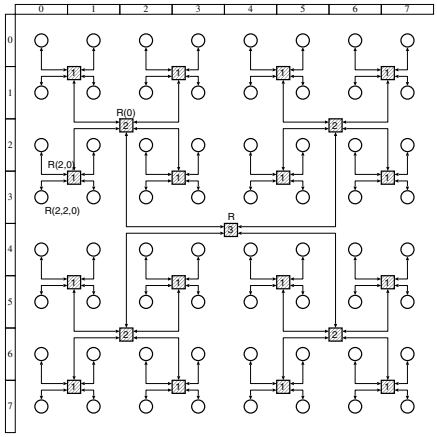


Figure 3. FH(8) (red tree)

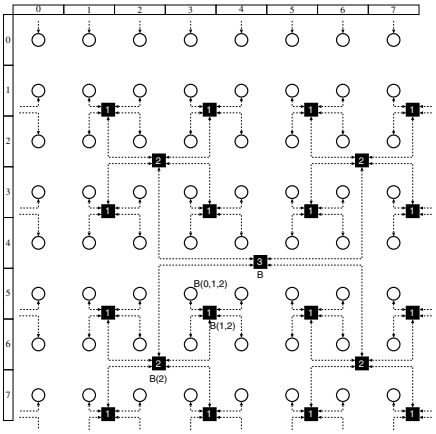


Figure 4. FH(8) (black tree)

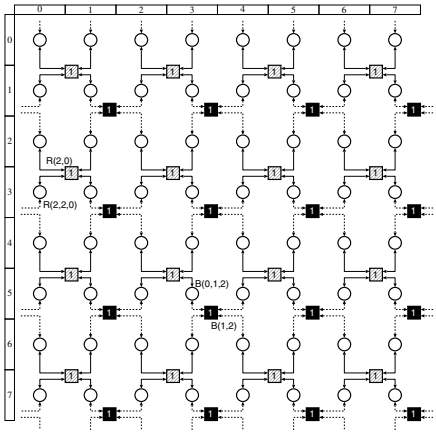


Figure 5. FH(8) with both red and black tree (rank-2 or upper routers are not shown)

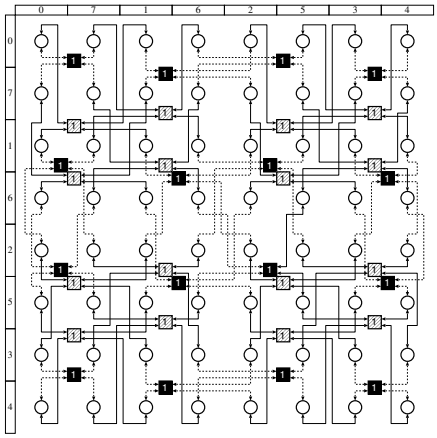


Figure 6. Folded Fat H-Tree (rank-2 or upper routers are not shown)

using vertical links in order to reduce their wire length, wire delay, and energy consumption in the wires.

Three-dimensional layouts of Fat Trees and Fat H-Tree are proposed in Section 3.1 and 3.2 respectively.

3.1 Fat Trees

For Fat Trees, we first show the 4-split method that divides the original planar tree into four parts. Then, we show a more general 2^n-split method.

Four-Split Method The original 2-D layout of a given Fat Tree is divided into four tiers (e.g., tier-0, tier-1, tier-2, and tier-3), and then these tiers are stacked together using vertical interconnects.

The original 2-D layout uses long wires for top-rank links near the root of the tree. Since the distance between neighboring tiers is very short (e.g., 5μm-50μm) in a 3-D IC, we aim to replace these long wires with vertical links. This replacement would shorten the wire length and mitigate the delay on these lines.

Here is the procedure that splits a given Fat Tree into four tiers.

1. Chip partitioning: Assume that the number of cores in a given Fat Tree is $2^n \times 2^n$ and 2-D coordinates (x_{2D}, y_{2D}) are assigned to each core. For the 3-D layout, the 2-D coordinates of each core are transformed into 3-D coordinates (x_{3D}, y_{3D}, z_{3D}), as follows:

$$x_{3D} = x_{2D} \bmod 2^{n-1}$$
$$y_{3D} = y_{2D} \bmod 2^{n-1}$$
$$z_{3D} = 2 \times \lfloor y_{2D}/2^{n-1} \rfloor + \lfloor x_{2D}/2^{n-1} \rfloor$$

   For example, a 64-core Fat Tree (2,4,1) can be divided into four tiers, each of which has 16 cores, as shown in Figure 7.

2. Allocation of routers: Routers are distributed across all tiers evenly so that each tier has the same number of routers for each tree level. In Figure 7, each tier has four rank-1 routers, two rank-2 routers, and a single rank-3 router.

3. Allocation of vertical links: Routers are connected by horizontal wires and/or vertical links so as to

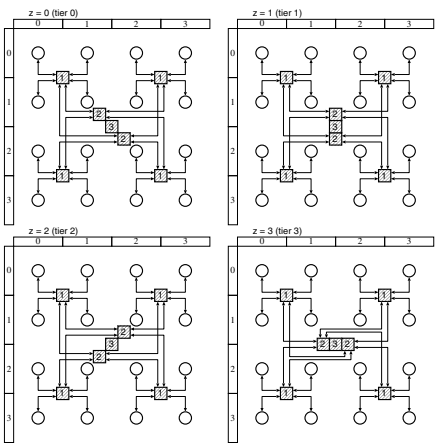


Figure 7. Splitting a 64-core Fat Tree (2,4,1) into four tiers

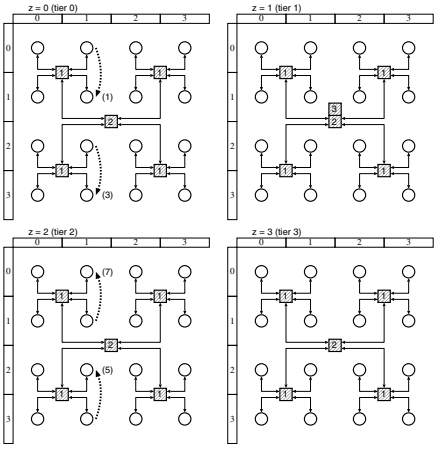


Figure 8. Splitting a 64-core red tree into four tiers

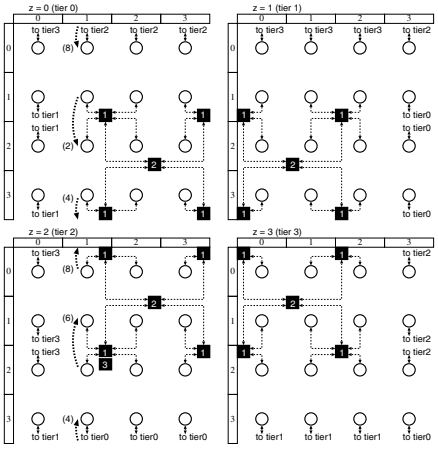


Figure 9. Splitting a 64-core black tree into four tiers

minimize the wire length. In order to connect routers at different tiers, vertical links such as through-wafer vias are placed between the tiers.

4. Reallocation of routers: The locations of routers are adjusted so as not to overlap vertical links, since a vertical interconnect consumes the overhead area. Hence, the overhead area is much smaller than router area with vertical links not overlapping each other. As shown in Figure 7, a single rank-3 router is placed at the center of each tier, and two rank-2 routers are placed closely around the rank-3 router.

As a consequence, every top-rank link that requires the longest wire to connect rank-2 and rank-3 routers is replaced by a very short vertical link that enables low-latency communication. Hence, the connection between rank-1 and rank-2 routers becomes the longest in the 3-D layout. That is, compared with the original 2-D layout, the longest link length in the 3-D layout is reduced by half.

This layout method can be also applied to the other types of Fat Trees, such as (2,4,2). This method is called "4-split method", since the original 2-D layout is divided into four parts.

Two-Split Method More generally, a given Fat Tree can be divided into 2^i tiers, by combining the 4-split method with another method called "2-split method", where i is a positive integer. When a given Fat Tree is divided into two pieces by the 2-split method, the 2-D coordinates of each core are transformed as follows:

$$
\begin{aligned}
x_{3D} &= x_{2D} \\
y_{3D} &= y_{2D} \bmod 2^{n-1} \\
z_{3D} &= \lfloor y_{2D} / 2^{n-1} \rfloor
\end{aligned}
$$

The other steps (i.e., allocations of routers and vertical links) are the same as the 4-split method.

3.2 Fat H-Tree

Since Fat H-Tree network includes a torus structure, its 3-D layout must keep the torus structure intact even though the network is partitioned into several tiers connected by vertical links.

Four-Split Method Here is the procedure to split a given Fat H-Tree into four tiers.

1. Chip partitioning: Assume that the number of cores in a given Fat H-Tree is $2^n \times 2^n$ and 2-D coordinates (x_{2D}, y_{2D}) are assigned to each core. For the 3-D layout, the 2-D coordinates of each core are transformed into 3-D coordinates (x_{3D}, y_{3D}, z_{3D}), as follows:

$$
\begin{aligned}
x_{3D} &= \begin{cases} x_{2D} \bmod 2^{n-1} & x_{2D} < 2^{n-1} \\ 2^{n-1} - (x_{2D} \bmod 2^{n-1}) & x_{2D} \geq 2^{n-1} \end{cases} \\
y_{3D} &= \begin{cases} y_{2D} \bmod 2^{n-1} & y_{2D} < 2^{n-1} \\ 2^{n-1} - (y_{2D} \bmod 2^{n-1}) & y_{2D} \geq 2^{n-1} \end{cases} \\
z_{3D} &= 2 \times \lfloor y_{2D} / 2^{n-1} \rfloor + \lfloor x_{2D} / 2^{n-1} \rfloor
\end{aligned}
$$

   As a result, the original 2-D layout of Fat H-Tree is folded in both vertical and horizontal directions, resulting in four folded pieces. In the case of a 64-core Fat H-Tree, 2-D layouts of red tree and black tree are transformed into their 3-D layouts shown in Figure 8 and 9, respectively.

2. Allocation of routers: Routers are evenly distributed across all tiers. In this example, each tier has eight rank-1 routers (four for red tree and the others for black) and two rank-2 routers (one for red and the other for black). Also, tier-1 has a rank-3 red-tree router, whereas tier-2 has a rank-3 black one.

3. Allocation of vertical links: Vertical links such as through-wafer vias are placed so as to shorten the horizontal wire length between two linked routers

mounted on different tiers. Notice that several links in black tree (Figure 9) are labeled as "to tier n". This means that these links are connected to an associated rank-1 black-tree router mounted on tier-n.

Although we have considered the red tree and the black tree separately, the 3-D layout of Fat H-Tree is formed by superimposing the 3-D layouts of red tree and black tree on the same tiers.

To show that the 3-D layout of Fat H-Tree is keeping its torus (ring) structure, here we illustrate a packet that moves to $y+$ direction from core (1,0,0). This packet goes around the ring and then it reaches the core (1,0,0) again. After moving two hops in $y+$ direction from core (1,0,0), the packet reaches core (1,1,0), as shown with arrow 1 in Figure 8. The next core from core (1,1,0) is core (1,2,0) as shown with arrow 2 (Figure 9), and the next is core (1,3,0) as shown with arrow 3 (Figure 8). Then the packet moves up to core (1,3,2) on tier-2, as shown with arrow 4. On tier-2, in the same way, the packet goes through core (1,3,2), core (1,2,2), core (1,1,2), and core (1,0,2). Finally, the packet gets back to core (1,0,0) on tier-0, as shown with arrow 8.

As mentioned above, the 3-D layout of a given Fat H-Tree is obtained by folding the original Fat H-Tree one or more times until the number of folded pieces meets the number of tiers the 3-D chip has (e.g., two foldings for 4-split). Such a folding-based transformation from 2-D into 3-D layout has been also proposed in 3-D IC placement techniques[3] to generate a 3-D layout with small number of vertical links.

Two-Split Method Like Fat Trees, a given Fat H-Tree can be divided into 2^i tiers, by combining the 4-split and 2-split methods of Fat H-Tree. In the 2-split method, 2-D coordinates are transformed as follows:

$$x_{3D} = x_{2D}$$

$$y_{3D} = \begin{cases} y_{2D} \bmod 2^{n-1} & y_{2D} < 2^{n-1} \\ 2^{n-1} - (y_{2D} \bmod 2^{n-1}) & y_{2D} \geq 2^{n-1} \end{cases}$$

$$z_{3D} = \lfloor y_{2D} / 2^{n-1} \rfloor$$

Note that every ring structure in y direction is formed across two tiers, whereas that in x direction is formed within a single tier. In the 2-split method, therefore, every ring in x direction must be folded within a single tier as well as 2-D layout of rings or tori. That is, a series of nodes in y direction are placed adjacently, whereas those in x direction are interleaved.

The other steps are the same as the 4-split method.

4 Evaluations

In this section, the proposed 3-D layouts of trees are compared to the original 2-D layouts in terms of network logic area, wire length, and energy consumption. Note that we omit the performance evaluations here, because there is no difference of network throughput between 2-D layouts and 3-D layouts of trees. The performance comparisons of trees were shown in [12].

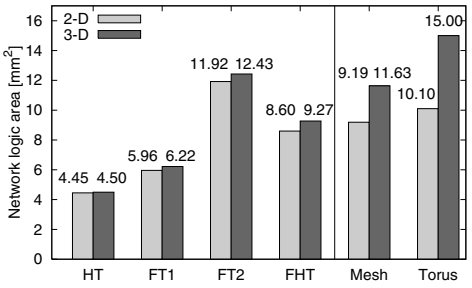


Figure 10. Network logic area [mm^2] (16-core $\times$ 4-tier; via size = 100μm^2/bit/tier)

4.1 Network Logic Area

The network logic area in a 3-D NoC is composed of routers, network interfaces, and vertical links. To obtain the network logic area for each topology, we first estimate the area used in routers and network interfaces and then calculate the area for vertical links.

To estimate the size of routers and network interfaces in a topology, we have implemented a wormhole router that supports various node degrees. We have also developed an NoC generator that automatically connects the routers and network interfaces in the arbitrary network topologies. Using the Synopsys Design Compiler, we synthesized the generated NoC design with ASPLA 90nm standard cell library and estimated the network logic area. The behavior of the synthesized NoC routers was confirmed through a gate-level simulation assuming an operating frequency of 500MHz.

The router architecture is fully pipelined. It transfers a header flit through four pipeline stages consisting of a routing computation, virtual-channel allocation, crossbar allocation, and crossbar traversal. The flit-width is set to 64-bit, and each pipeline stage has a buffer for storing one flit. The routing decisions are stored in the header flit prior to packet injection (i.e., source routing); thus routing tables that require register files for storing routing paths are not needed in each router, resulting a low cost router implementation.

The network interface (NI) has to be designed to interface between a processing core and a network with a minimum hardware amount. We have implemented a simple NI that employs a 2-flit FIFO buffer for both the core-to-network and network-to-core interfaces. In addition, we have implemented a different type of NI for Fat H-Tree, because Fat H-Tree uses 2-port NIs that can forward packets from red tree to black and vice versa, as described in [12].

The pitches of a through-wafer via can range from 1μm to 10μm square[5, 6, 11], depending on the manufacturing process (e.g., accuracy of wafer-to-wafer alignment). In this evaluation, the size of a through-wafer via was set to 10μm square, and the flit-width was set to 64-bit. Then, we calculated the through-via area according to the number of all unidirectional 1-bit links between tiers.

Figure 10 shows the network logic area of vari-

ous topologies: "HT" for H-Tree, "FT1" for Fat Tree (2,4,1), "FT2" for Fat Tree (2,4,2), and "FHT" for Fat H-Tree. In the graph, the hardware amounts of 2-D and 3-D layouts are shown in different colors. In addition, we estimated the hardware amounts of 2-D mesh, 3-D mesh, 2-D torus, and 3-D torus for comparison.

In all topologies, their 3-D layout increases the total network logic area from their original 2-D layout. The 3-D layouts of trees use through-wafer vias according to the number of vertical links that cross tiers, as the additional hardware resources for the 3-D stacking. On the other hand, 3-D mesh and torus require much more hardware resources compared with 2-D mesh and torus, because a router in 3-D mesh (or torus) requires two additional channels for vertical connections (i.e., up and down), which require a larger crossbar switch and more channel buffers, in addition to the through-via area they need.

Although the 3-D layout of Fat H-Tree consumes a relatively larger area for through-vias than other tree-based topologies do and its hardware amount is increased by 7.8%, this area overhead is still smaller than those required for 3-D mesh and torus. Note that we have assumed that the size of a through-wafer via was 10μm square. This area overhead would be further reduced when the pitch becomes 1μm square.

4.2 Total Wire Length

In this section, we calculate the total wire length of Fat Trees, Fat H-Tree, and other typical topologies. Assuming that the distance between neighboring two cores aligned in a 2-D grid square is 1-unit, we define L as the total unit-length of links in a given network.

In Section 4.2.1, we obtain the total unit-length of 2-D layouts. Then in Section 4.2.2, we obtain the total unit-length of 3-D layouts assuming that a given network is divided into four tiers. That is, 16-core, 64-core, and 256-core networks are divided into four tiers, each of which contains 2×2 cores, 4×4 cores, and 8×8 cores, respectively.

4.2.1 Two-Dimensional Layout

The total unit-length of links in an n-rank H-Tree network, $L_{2D,ht}$, can be expressed as

$$L_{2D,ht} = \sum_{i=1}^{n} l_{ht}^{i} \cdot r_{ht}^{i} \tag{3}$$

where l_{ht}^{i} is the total unit-length of links between a rank-i router and its four child routers, and r_{ht}^{i} is the number of rank-i routers in the H-Tree. Assuming that the number of cores is $N = 2^n \times 2^n$, $l_{ht}^{i} = 2^{i+1}$ and $r_{ht}^{i} = N/4^i$, where $1 \le i \le n$. Therefore, Equation 3 can be transformed as follows.

$$L_{2D,ht} = \sum_{i=1}^{n} l_{ht}^{i} \cdot r_{ht}^{i} = \sum_{i=1}^{n} 2^{i+1} \cdot \frac{N}{4^i} = 2(N - 2^n) \tag{4}$$

Table 1. Total link length, L, of 2-D layouts (1-unit = distance between neighboring cores)

	N-core	16core	64core	256core
HT	$2(N-2^n)$	24	112	480
FT1	nN	32	192	1,024
FT2	$2nN$	64	384	2,048
FHT	$8+8(N-2^{n+1})$	72	392	1,800
2Dmesh	$2(N-2^n)$	24	112	480
2Dtorus	$4(N-2^n)$	48	224	960

Table 2. Total link length, L, of 3-D layouts (1-unit = distance between neighboring cores)

	N-core	16core	64core	256core
HT	$2(N-2^{n+1})$	16	96	448
FT1	$(n-1)N$	16	128	768
FT2	$2(n-1)N$	32	256	1,536
FHT	$8+4(N-2^{n+1})$	40	200	904
3Dmesh	$2(N-2^{n+1})$	16	96	448
3Dtorus	$4(N-2^{n+1})$	32	192	896

Similarly, the total unit-length of links in a Fat Tree (2,4,1) network, $L_{2D,ft1}$, is nN. A Fat Tree (2,4,2) has twice the number of routers in the Fat Tree (2,4,1); so $L_{2D,ft2} = 2L_{2D,ft1}$.

A Fat H-Tree has two folded H-Tree networks, in which each link, except for the links connecting to the top-rank router, requires twice the wire resources of an ordinary H-Tree. By folding the H-Tree, only the top-rank router and its four child routers can be placed inside a 1-unit × 1-unit grid square. Therefore, the total unit-length of links in a Fat H-Tree network, $L_{2D,fht}$, can be expressed as follows.

$$l_{fht}^{i} = \begin{cases} 2l_{ht}^{i} & 1 \le i \le n-1 \\ 4 & i = n \end{cases} \tag{5}$$

$$L_{2D,fht} = \sum_{i=1}^{n} l_{fht}^{i} \cdot r_{fht}^{i} = 8 + 8(N - 2^{n+1}) \tag{6}$$

Total unit-lengths of 2-D layouts mentioned above are summarized in Table 1. Although a Fat H-Tree uses slightly more wire resources compared to the Fat Tree (2,4,2) in 16- and 64-core networks, the impact on the chip design is considered to be modest. This is because enormous wire resources are available in an NoC, thanks to the current CMOS technology that has six or more metal layers.

4.2.2 Three-Dimensional Layout

Here we estimate the total unit-length required for the 3-D layout of each topology. Since the distance between wafers (i.e., tiers) can range from 5μm to 50μm[11], which is much shorter than that of horizontal links, we do not consider the vertical link length for simplicity.

Assuming that an n-rank H-Tree network is divided into four tiers, we estimate the total link length of its 3-D layout, $L_{3D,ht}$. Every top-rank link is replaced by a vertical link that connects routers on different tiers

or a very short horizontal link that connects routers in the same tier. Neither do we consider the length of such very short horizontal links for simplicity. In the case of 3-D layout for an n-rank H-Tree, Equation 4 takes the following:

$$L_{3D,ht} = \sum_{i=1}^{n-1} 2^{i+1} \cdot \frac{N}{4^i} = 2N\left(\frac{2^{n-1}-1}{2^{n-1}}\right) = 2(N-2^{n+1}) \quad (7)$$

For Fat H-Tree, we consider red tree and black tree separately. The 3-D layout of a red tree is equivalent to that of a same-sized H-Tree. Therefore its total wire length can be expressed with Equation 7. The layout of a 3-D black tree includes a same-sized H-Tree as well as that of a red tree. Moreover, the black tree requires four 2-unit links for its top-rank links (Figure 9). Thus, the total wire resources required for 3-D layout of a given Fat H-Tree can be expressed as follows:

$$L_{3D,fht} = 8 + 2L_{3D,ht} = 8 + 4(N - 2^{n+1}) \quad (8)$$

Total unit-lengths of 3-D layouts mentioned above are summarized in Table 2. Compared with the original 2-D layouts, the 3-D layouts reduce their total wire length by 44.4%-49.8% for Fat H-Tree and by 25.0%-50.0% for Fat Trees. Although the total unit-length of a Fat H-Tree is much longer than that of a torus in the cases of 2-D layouts, their differences are narrower in the cases of 3-D layouts. Since one of trees' drawbacks is their wire length, it can be mitigated by using the 3-D layout method proposed here. The 3-D layout of a Fat H-Tree can reduce more wire resources compared with that of a Fat Tree (2,4,2), in the cases of large networks such as 64-core or more. Actually, the 3-D layout of a 64-core Fat H-Tree requires 21.9% less wires compared with the same-sized Fat Tree (2,4,2).

Here we discuss why the 3-D layout of Fat H-Tree reduces more wires than those of Fat Trees. In the case of 2-D layout, a given Fat H-Tree must be folded by interleaving a series of nodes in each ring, thus resulting in longer wires between neighboring cores. In its 3-D layout, on the other hand, a given Fat H-Tree does not interleave a series of nodes in each ring, since every ring structure can be formed across two tiers without folding within a single tier (Figure 8 and 9). As a result, the 3-D layout that does not interleave cores requires less wire resources to connect neighboring cores compared with the original 2-D layout. This is the reason why the 3-D layout of Fat H-Tree could effectively reduce its wire length compared with Fat Trees.

4.3 Energy Consumption

The average energy consumption needed to transmit one flit from source to destination can be estimated as

$$E_{flit} = wH_{ave}(E_{sw} + E_{link}) \quad (9)$$

where w is the flit-width, H_{ave} is the average hop count, E_{sw} is the average energy to switch a 1-bit data inside a router, and E_{link} is a 1-bit energy consumed in a link.

We used the Synopsys Power Compiler to extract E_{sw} of the router synthesized with the 90nm standard cell library (the details are shown in Section 4.1). The switching activity of the running router was captured through the gate-level simulation of the synthesized router. Gate-level power analysis based on this switching activity shows that E_{sw} is 0.183pJ when the router is operating at 500MHz with a 1.0V supply voltage.

E_{link} can be calculated as

$$E_{link} = dV^2C_{wire}/2 \quad (10)$$

where d is the average 1-hop distance (in millimeters), V is the supply voltage, and C_{wire} is the wire capacitance per millimeter. C_{wire} can be estimated using the method proposed in [8], and is 300fF/mm in the case of a semi-global interconnect in the 90nm CMOS technology. For instance, E_{link} is 0.150pJ when the 1-hop distance is 1mm on average.

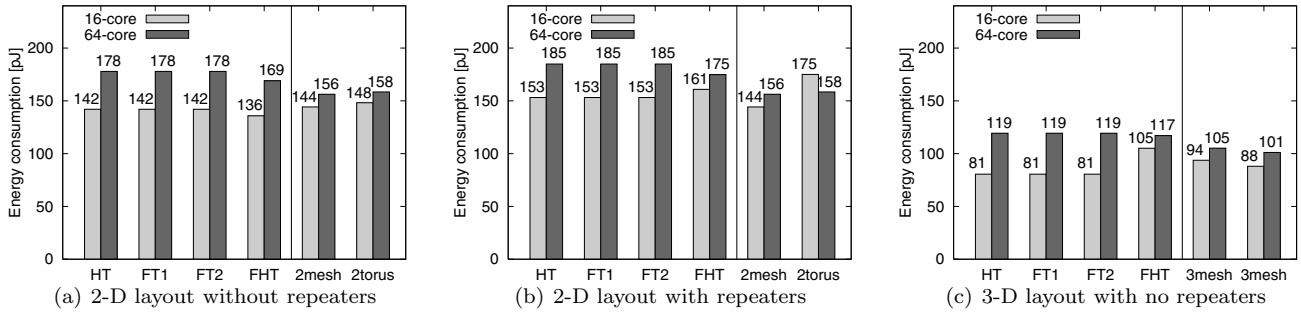
We assume 64-bit 16- and 64-core networks placed in an 8mm × 8mm chip. In the 16-core networks, H_{ave} tends to be short but d becomes long, while they are opposite in the case of 64-core. We estimated the average 1-hop distance, d, using a flit-level network simulator, as in [12]. Then we derived E_{flit} based on Equation 9 with the various parameters mentioned above.

4.3.1 Two-Dimensional Layout

First, we show the average flit transmission energy, E_{flit}, of various topologies in the cases of 2-D layouts without inserting repeater buffers that mitigate the wire delay but add their gate capacitance on the wires. Then, we show that with repeater buffers.

Figure 11(a) shows the results on unrepeated lines. Although the average 1-hop distance, d, of Fat H-Tree is longer than the other tree-based topologies because of its folded layout, its average hop count is the shortest among them[12]. As a result, E_{flit} of Fat H-Tree is less than those of Fat Trees, as shown in the graph. Fat Trees offer relatively short hop counts, but the energy consumption in their network links increases as network size enlarges, because the moving distance of packets is stretched by the long wires around the root.

Since these evaluation results mentioned above assume unrepeated semi-global lines, the wire delay of these lines grows quadratically with the wire length. Figure 11(b) shows the average flit transmission energy on repeated lines, in which repeater buffers are inserted so as to keep their wire delay less than 8 FO4s. As for Fat Trees, since six repeater buffers are inserted on their longest links on average, their E_{flit} is increased especially in the cases of 16-core networks. On the other hand, the impact of repeater buffers on total energy becomes small in the cases of larger networks, because the energy consumption in switches, E_{sw}, becomes large due to their long hop counts. As for Fat H-Tree and torus, their link lengths are stretched due to their folded layout, resulting in larger E_{flit} in 16-core networks. Although repeaters are inserted in a 16-core torus with 4mm links, no repeaters are inserted in a 64-core torus whose wire length is at most 2mm.



(a) 2-D layout without repeaters (b) 2-D layout with repeaters (c) 3-D layout with no repeaters

Figure 11. Average flit transmission energy E_{flit} [pJ]

4.3.2 Three-Dimensional Layout

We estimate the average flit transmission energy on 3-D layouts of Fat Trees and Fat H-Tree. They are placed on four 4mm × 4mm wafers in this experiment. The capacitance of a vertical link is very small (e.g., 4.34fF [6]) compared with that of horizontal links (e.g., 300fF/mm in the 90nm technology). Hence, we do not assess the capacitance of vertical links for simplicity.

Figure 11(c) shows the results. No repeater buffers are inserted on network links of all topologies, since their wire delay is reduced by downsizing each wafer size into half in the 3-D layouts. Compared with the 2-D layouts on unrepeated lines, E_{flit} is reduced by up to 42.9% for Fat Trees, and 30.8% for Fat H-Tree. In addition, comparison with the 2-D layouts on repeated lines shows that E_{flit} is reduced by up to 47.0% for Fat Trees, and 34.8% for Fat H-Tree.

We can see that the 3-D layouts of trees proposed here can shorten the wire length and remove energy-hungry repeaters from these wires. As a result, the energy efficiency of trees can be much improved.

5 Conclusions

Tree-based topologies have been considered to generate a serious wire delay problem because of their long wire links around the root of the tree. And their wire delay problem will be more serious as the process technologies are scaled down beyond 65nm rules or finer. In this paper, we proposed a 3-D layout scheme of Fat Trees and Fat H-Tree. The 3-D layouts of Fat Trees and Fat H-Tree divided into four tiers were compared with original 2-D layouts in terms of network logic area, wire length, wire delay, number of repeaters inserted, and energy consumption. The results show that 1) total wire length is reduced by 25.0% to 50.0%; 2) wire delay is improved and repeater buffers that consume considerable energy and area can be removed; 3) flit transmission energy is reduced by up to 47.0%; 4) area overhead is at most 7.8%, which compares favorably to those for 3-D mesh and torus. Therefore, we have demonstrated that the 3-D layouts proposed here can shorten the trees' long wires and reduce the energy consumption by removing energy-hungry repeaters from these wire lines.

References

[1] L. Benini and G. D. Micheli. *Networks on Chips: Technology And Tools*. Morgan Kaufmann, 2006.

[2] B. Black, D. W. Nelson, C. Webb, and N. Samra. 3D Processing Technology and Its Impact on iA32 Microprocessors. *Proceedings of the International Conference on Computer Design*, pages 316–318, Oct. 2004.

[3] J. Cong et al. Thermal-Aware 3D IC Placement Via Transformation. *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 780–785, Jan. 2007.

[4] W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. *Proceedings of the Design Automation Conference*, pages 684–689, June 2001.

[5] S. Das, A. Fan, K.-N. Chen, C. S. Tan, N. Checka, and R. Reif. Technology, Performance, and Computer-Aided Design of Three-Dimensional Integrated Circuits. *Proceedings of the International Symposium on Physical Design*, pages 108–115, Apr. 2004.

[6] W. R. Davis and et. al. Demystifying 3D ICs: The Pros and Cons of Going Vertical. *IEEE Design and Test of Computers*, 22(6):498–510, Nov. 2005.

[7] C. Grecu et al. Structured Interconnect Architecture: A Solution for the Non-Scalability of Bus-Based SoCs. *Proceedings of the Great Lakes Symposium on VLSI*, pages 192–195, Apr. 2004.

[8] R. Ho, K. W. Mai, and M. A. Horowitz. The Future of Wires. *Proceedings of the IEEE*, 89(4):490–504, Apr. 2001.

[9] J. Kim et al. A Novel Dimensionally-Decomposed Router for On-Chip Communication in 3D Architectures. *Proceedings of the International Symposium on Computer Architecture*, pages 138–149, 2007.

[10] C. E. Leiserson. Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. *IEEE Transactions on Computers*, 34(10):892–901, Oct. 1985.

[11] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir. Design and Management of 3D Chip Multiprocessors Using Network-in-Memory. *Proceedings of the International Symposium on Computer Architecture*, pages 130–141, June 2006.

[12] H. Matsutani, M. Koibuchi, and H. Amano. Performance, Cost, and Energy Evaluation of Fat H-Tree: A Cost-Efficient Tree-Based On-Chip Network. *Proceedings of the International Parallel and Distributed Processing Symposium*, Mar. 2007.

[13] H. Matsutani, M. Koibuchi, and H. Amano. Tightly-Coupled Multi-Layer Topologies for 3-D NoCs. *Proceedings of the International Conference on Parallel Processing*, Sept. 2007.

[14] S. Vangal et al. An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS. *Proceedings of the International Solid-State Circuits Conference*, Feb. 2007.

[15] T. Yamada, N. Fujii, and S. Ueno. On Three-Dimensional Layout of Pyramid Networks. *Proceedings of the Asia-Pacific Conference on Circuits and Systems*, pages 159–164, Oct. 2002.

[16] T. Yamada and S. Ueno. On Three-Dimensional Layout of de Bruijn Networks. *Proceedings of the International Symposium on Circuits and Systems*, pages 779–782, May 2002.