Layout of the Cube-connected Cycles without Long Wires

GUIHAI CHEN[1] AND FRANCIS C. M. LAU[2]

[1]*State Key Lab for Novel Software Technology, Nanjing University, Nanjing 210093, China*
[2]*Department of Computer Science and Information Systems, The University of Hong Kong, Pokfulam Road, Hong Kong, China*
Email: fcmlau@csis.hku.hk

Preparata and Vuillemin proposed the cube-connected cycles (CCC) in 1981, and in the same paper gave an asymptotically-optimal layout scheme for the CCC. While all the known optimal layouts of the CCC, including the Preparata–Vuillemin layout, have long wires, we give a new layout scheme which has no long wires while keeping the asymptotically-optimal area. Hence, we can conclude that the CCC can be laid out optimally (within a constant factor) both in area and in wire length. We also show how large a constant-factor blow-up in area is needed in order not to produce any long wire in the layout.

Received 8 October 1999; revised 28 February 2001

1. INTRODUCTION

The interconnection network is an important part in the design of parallel computers. There are many aspects that need to be considered when choosing a specific interconnection for processors. With the technological progress in very large-scale integrated (VLSI) electronic circuits that has taken place so far, it is reasonable to conceive of a huge number of processors being integrated tightly together to cooperate on the execution of parallel algorithms. As such, one of the major criteria by which to judge the suitability of an interconnection network is how compactly it can be laid out in a VLSI grid. Two most frequently used measures of a layout are the layout area and the maximum wire length.

The cube-connected cycles (CCC), one of the most extensively studied and frequently cited interconnection networks, was proposed by Preparata and Vuillemin in 1981 as a practical substitute for the hypercube [1]. They also gave an asymptotically-optimal layout scheme for the CCC. Their layout scheme, however, has two drawbacks: it is not 'minimal' in area; and it has long wires. Our research aims at finding better layout schemes for the CCC, with two goals: (1) to seek the best minimal layout area of the CCC, and (2) to reduce the long wires of the CCC layout while keeping the asymptotically-optimal area. We have already achieved the first goal, proposing an improved layout [2] which is more compact than the Preparata–Vuillemin layout. This paper addresses the second goal by introducing yet another new layout of the CCC which is free of long wires while keeping the asymptotically-optimal area. Based on this result, we can conclude that the CCC can be laid out optimally both in area and in wire length, thus answering a question posed by Beigel and Kruskal [3].

Wire problems have received much attention from computer designers and chip designers [4]. Wires do cost—they could take up a lot of space, and spend a lot of time in transmitting data [5, 6]. Hence, it is justified that we should look for a layout that is sufficiently small and that uses reasonably short wires. Beigel and Kruskal found a layout of a special class of networks called *bidelta networks* without long wires [3]; Lai and Speangue proposed a layout of the hypercube without long wires [7]; and Lau and Chen showed that some networks can be laid out minimally both in area and in maximum wire length [8]. Note that not all graphs can have a layout that is optimal in both area and wire length [9].

Our work is motivated by and closely related to the paper by Beigel and Kruskal [3], who point out that the laying out of the CCC can be based on the laying out of bidelta networks. Although they gave an optimal layout of bidelta networks without long wires, they did not make use of the relationship between the CCC and bidelta networks to produce a layout for the CCC. In this paper, we first give an improved layout of the bidelta networks, which uses fewer stages and thus less area than the layout by Beigel and Kruskal. Using this improved layout, we then derive our layout for the CCC. This layout has a maximum wire length which is better than that in all existing layouts (including [2]) by a logarithmic factor. We also show how large a constant-factor blow-up in area is necessary in order to keep wires short in a CCC layout.

2. PRELIMINARIES

2.1. The Thompson model

Among the many mathematical models that have been proposed for VLSI computations, the most widely accepted

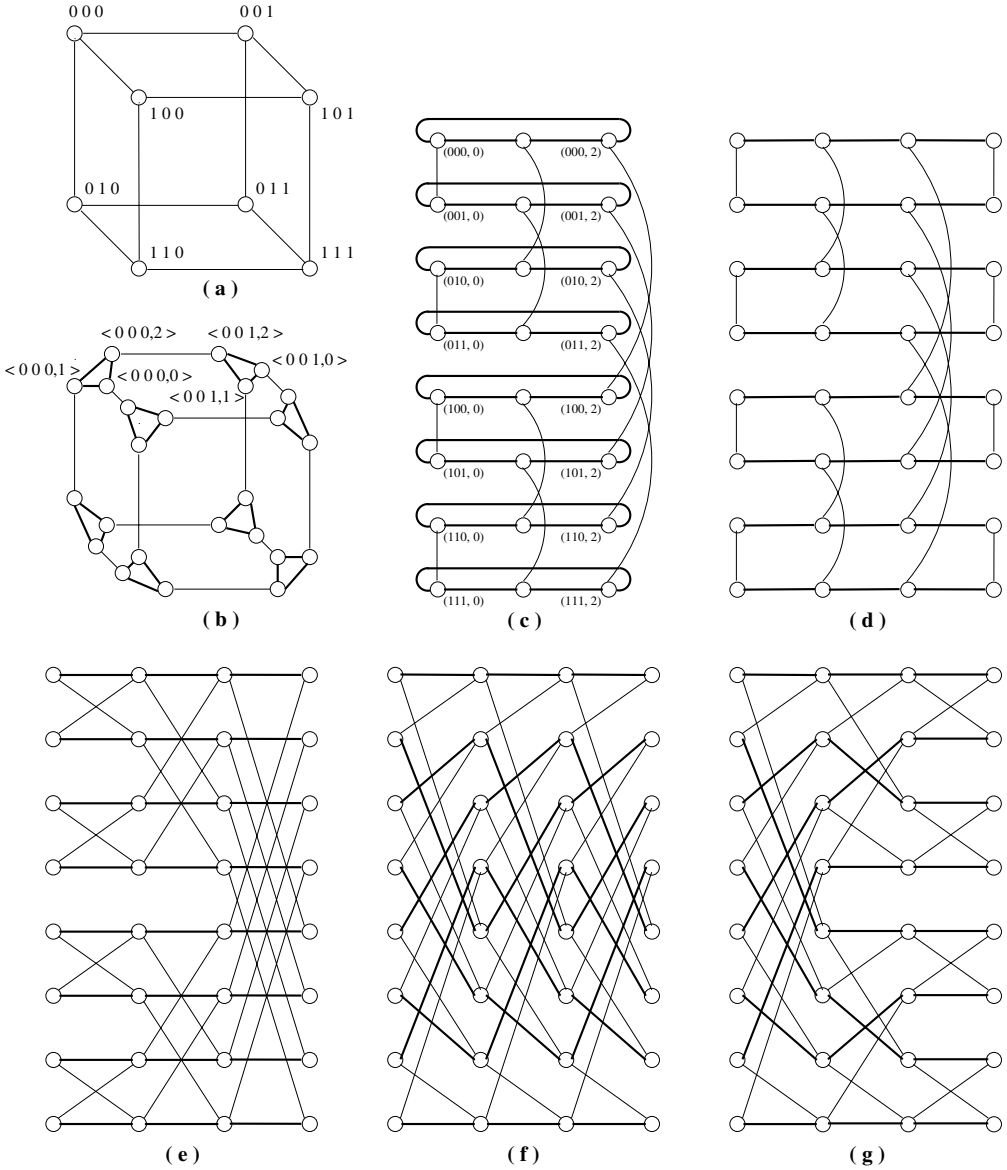

FIGURE 1. (a) A 3-dimensional hypercube. (b) A 3-dimensional CCC. (c) Another drawing of the 3-dimensional CCC: thin lines, cube edges; full lines, cycle-edges. (d) A 3-dimensional CCC unfolded. (e) An 8×8 indirect binary cube. (f) An 8×8 omega network. (g) An 8×8 baseline network.

is the Thompson grid model [10, 11]. In this model, the chip is presumed to consist of a grid of vertical and horizontal tracks which are spaced apart at unit intervals. Two layers of interconnect are used to route the wires. Vertical wires are routed in the top layer of the interconnect and horizontal wires are routed in the bottom layer. Hence, wires may cross each other but cannot overlap for any distance or cross a node to which they are not incident. To change direction, wires may turn into the other layer by contact cuts or vias which facilitate connections between the two layers. The routing of wires in this fashion is also known as *layer per direction routing* or *Manhattan routing*. In our discussion, no knock-knees are allowed—that is, two wires cannot turn at the same grid point [12, 13].

Formally, an embedding or layout of a graph G in a Thompson grid is an assignment of the nodes of G to intersection points in the grid and the edges of G to paths along the grid tracks. The layout area is the product of the number of vertical tracks and the number of horizontal tracks which contain a node or a path segment of the graph. The maximum wire length is the length of the longest wire in the layout.

2.2. The cube-connected cycles

The d-dimensional CCC is constructed from the d-dimensional hypercube by replacing each node of the hypercube with a cycle of d nodes in the CCC [1, 14]. The ith-dimension edge incident to a node of the hypercube is then connected to the ith node of the corresponding cycle of the CCC. For example, see Figures 1a and b. The resulting graph has $d \cdot 2^d$ nodes each of degree 3. By modifying the

labeling scheme of the hypercube, we can represent each node by a pair $\langle w, i\rangle$ where i $(0 \leq i < d)$ is the position of the node within its cycle and w (any d-bit binary string) is the label of the node in the hypercube that corresponds to the cycle. Then, two nodes $\langle w, i\rangle$ and $\langle w', i'\rangle$ are linked by an edge in the CCC if and only if either

(1) $w = w'$ and $i - i' = \pm 1 (\mathrm{mod}\, d)$, or
(2) $i = i'$ and w differs from w' in precisely the ith bit.

Edges due to (1) are cycle-edges and edges due to (2) are cube-edges. As shown in Figure 1c, the CCC is often drawn in the multi-stage format. Alternatively, the CCC can be unfolded along its wraparound links (long links inside cycles); see Figure 1d where the first stage (of nodes) and the last stage are identified. It should be noted that the CCC is very similar to the butterfly network—compare (d) and (e) of Figure 1. One can be embedded into the other with dilation 2 and congestion 2. Because of this similarity, we will show that an optimal layout of the butterfly network without long wires can directly give rise to an optimal layout of the CCC without long wires.

The CCC is closely related to the butterfly network just as the shuffle-exchange network is to the deBruijn network. The group-theoretic relations of the four networks are well studied in [15] where the CCC and the butterfly are proved to be Cayley graphs derivable from the shuffle-exchange network and the deBruijn network respectively; and inversely, the shuffle-exchange network and the deBruijn network are proved to be some coset graph of the CCC and the butterfly network respectively.

In general, we say that a network of N nodes has (asymptotically) optimal area if it can be laid out in area $\Theta(N^2/T^2)$, where T is the time to execute an ascend–descend algorithm. We say that a (constant-degree) network has optimal wire length if the longest wire has length $\Theta(N/T^2)$. Accordingly, for the d-dimensional CCC with $n = 2^d$ cycles, the optimal layout area is $\Theta(n^2)$ and the optimal wire length is $\Theta(n/\log n)$. While all the known optimal layouts of the CCC have maximum wire length $\Theta(n)$, we give a new optimal layout for the CCC whose maximum wire length is $\Theta(n/\log n)$.

2.3. Bidelta networks

There has been a large amount of research on multistage interconnection networks. Kruskal and Snir have found that many of these networks, such as the indirect binary cube (or unfolded butterfly network), the omega network, the SW banyan network, and so on, are isomorphic [16]. That is, one can be produced from the other by simply rearranging the nodes at each stage. These networks are referred to as *bidelta* interconnection networks.

Let $n = 2^d$; an $n \times n$ or d-stage bidelta network is composed of $d + 1$ stages of nodes, interconnected by d stages of connections. At each stage, nodes are numbered from 0 to $n - 1$ and written in a d-bit binary notation. Connections between columns of nodes are based on permutation and correction of the d bits. A correction represents two out-edges (a thin line and a full line in Figures 1e, f and g) connected to the next stage for each node; the permutation represents a permutation of the nodes at the stage (represented by thick lines connecting the last stage to this stage). For the first stage, the d bits are $a_{d-1}a_{d-2}\ldots a_1a_0$. A bidelta network connects each bit a_i to a_i^* $(= a_i$ or $\bar{a}_i)$ one at a stage. The connects do not have to be in order. The final bit pattern is $a^*_{(d-1)}a^*_{(d-2)}\ldots a^*_{(1)}a^*_{(0)}$ where (i) is a permutation of i: $0 \leq i \leq n-1$. That means a_i^* can be at any bit position within the final bit permutation pattern $a^*_{(d-1)}a^*_{(d-2)}\ldots a^*_{(1)}a^*_{(0)}$.

Consider for example 3-stage bidelta networks. These networks begin with the bits $a_2a_1a_0$ and end with the bits $a^*_{(2)}a^*_{(1)}a^*_{(0)}$. There are three bits, and hence three stages of connections are needed to correct them all. For the 3-stage bidelta networks shown in Figures 1e, f and g, the following are their bit changes, respectively. We use a larger font type for the bit being corrected at a stage.

a_2	a_1	a_0
a_2	a_1	$\boldsymbol{a_0^*}$
a_2	$\boldsymbol{a_1^*}$	a_0^*
$\boldsymbol{a_2^*}$	a_1^*	a_0^*

a_2	a_1	a_0
$\boldsymbol{a_0^*}$	a_2	a_1
$\boldsymbol{a_1^*}$	a_0^*	a_2
$\boldsymbol{a_2^*}$	a_1^*	a_0^*

a_2	a_1	a_0
$\boldsymbol{a_0^*}$	a_2	a_1
a_0^*	$\boldsymbol{a_1^*}$	a_2
a_0^*	a_1^*	$\boldsymbol{a_2^*}$

3. LAYOUT OF BIDELTA NETWORKS

Beigel and Kruskal [3] gave an (asymptotically) optimal layout of bidelta networks without long wires. Their main idea is to create a particular bidelta network (to which all other bidelta networks are isomorphic) in which the stages with long wires are spread out, thus amortizing the long wire lengths across intermediary stages. Using the same idea, we give an alternate layout of bidelta networks without long wires. The area and maximum wire length of the alternate layout are the same as the Beigel–Kruskal layout, but the permutations of the nodes are different. As a result, the new layout can be used to generate a layout for the butterfly network (and hence the CCC) using fewer additional stages than the Beigel–Kruskal layout.

The following lemmas are useful in the construction of the new layout.

LEMMA 1. *In a d-stage bidelta network, if the connections at some stage do not involve the upper (most*

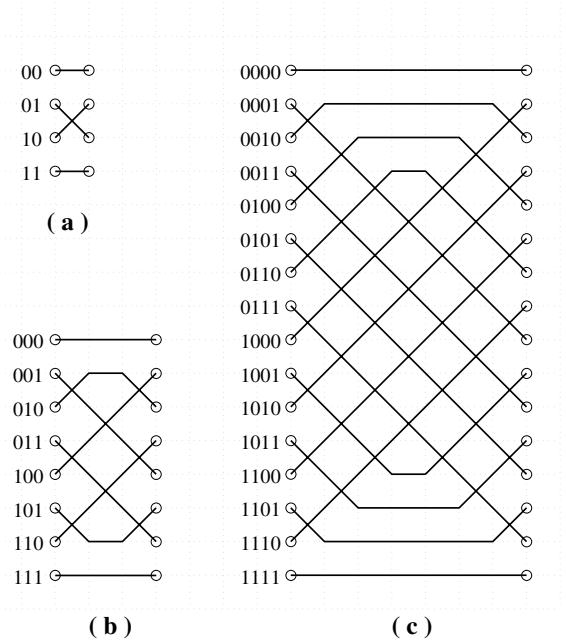

FIGURE 2. End-exchange using local operations.

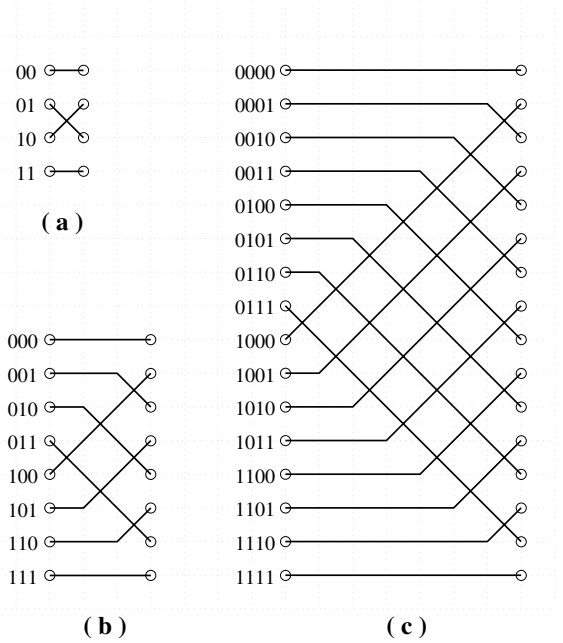

FIGURE 3. Shuffle using local operations.

significant) $\Theta(\log d)$ *bits, then these connections do not contain long wires.*

Proof. The connections involve only the lower $\Theta(d - \log d)$ bits. Therefore, obviously, the longest connection is of length $2^{\Theta(d-\log d)} = \Theta(n/\log n)$. □

We define *end-exchange* to be the operation that permutes r bits from $a_{r-1}a_{r-2}\ldots a_1a_0$ to become $a_0a_{r-2}\ldots a_1a_{r-1}$ (i.e. a butterfly connection of 2^r nodes), and *local operation* to be one that causes a link to go from a node i to a node i or $i+1$ or $i-1$ at the next stage.

LEMMA 2. *The number of stages required to end-exchange r bits using only local operations is* $2^{r-1} - 1$.

Proof. Refer to Figure 2. Note that the wires do not conflict at any stage—that is, every stage is a permutation. □

LEMMA 3. *The number of stages required to shuffle (or unshuffle) r bits using only local operations is* $2^{r-1} - 1$.

Proof. Refer to Figure 3. □

Now we show the construction of the bidelta network whose layout has optimal area and wire length. This network is modified from the F network in [3]. As before, let $n = 2^d$, and assume for convenience that $d + 1$ is a power of two. The results can be generalized for arbitrary d. Let $r = \log(d+1)$. By Lemma 1, we separate the d bits into the upper r bits and the lower $d - r$ bits. Let the stages be numbered $0, 1, \ldots, d-1$. We batch the stages into the first $(d+1)/2$, the next $(d+1)/4$, the next $(d+1)/8$, etc. Then the stages can be represented by the ordered pairs

$$(0,0), (0,1), (0,2), \ldots, (0, ((d+1)/2) - 1),$$
$$(1,0), (1,1), (1,2), \ldots, (1, ((d+1)/4) - 1),$$
$$(2,0), (2,1), (2,2), \ldots, (2, ((d+1)/8) - 1),$$
$$\vdots$$
$$(r-3,0), (r-3,1), (r-3,2), \ldots, (r-3,3),$$
$$(r-2,0), (r-2,1),$$
$$(r-1,0),$$

where the first index represents the batch number and the second index represents the stage number within a batch.

Without loss of generality, we use a 15-stage network to explain the bit changes. The bit changes for this particular network are as given in Table 1. Here, $r = \log(d+1) = 4$. There are 4 (r) batches, comprising respectively the first 8 $((d+1)/2)$ stages, then the next 4 $((d+1)/4)$ stages, then the next 2 $((d+1)/8)$ stages, and the last 1 $((d+1)/16)$ stage. The bits are divided into the upper 4 (r) bits and the lower 11 $(d-r)$ bits. We refer to them as the upper bits and the lower bits respectively. The bit changes are as follows.

- For the first stage of every batch, the lowest of the upper bits is corrected. That is, a_{11}, and then a_{14}, and so on, in the example.
- Within a batch, an end-exchange is applied to the lower $r-j$ bits of the upper bits, where j is the batch number. That is, in the example, an end-exchange is applied to all 4 $(r-0)$ bits in batch 0, and then the lower 3 $(r-1)$ bits in batch 1, and so on.
- For all the stages other than the first within a batch, the lower bits are corrected one at a stage in order (from the lowest to the highest).

Based on the table, we have the layout of the 15-stage bidelta network as shown in Figure 4a. There are $2^{15} = 32768$ nodes in a stage of nodes. The upper bits divide these into 16 (2^r) groups (hence the upper bits represent

TABLE 1. Bit changes for the 15-stage bidelta network.

a_{14}	a_{13}	a_{12}	a_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
a_{14}	a_{13}	a_{12}	a^*_{11}	a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a_0
	≀			a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a_1	a^*_0
	≀			a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a_2	a^*_1	a^*_0
	≀			a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a_3	a^*_2	a^*_1	a^*_0
	≀			a_{10}	a_9	a_8	a_7	a_6	a_5	a_4	a^*_3	a^*_2	a^*_1	a^*_0
	≀			a_{10}	a_9	a_8	a_7	a_6	a_5	a^*_4	a^*_3	a^*_2	a^*_1	a^*_0
	≀			a_{10}	a_9	a_8	a_7	a_6	a^*_5	a^*_4	a^*_3	a^*_2	a^*_1	a^*_0
a^*_{11}	a_{13}	a_{12}	a_{14}	a_{10}	a_9	a_8	a_7	a^*_6	a^*_5	a^*_4	a^*_3	a^*_2	a^*_1	a^*_0
a^*_{11}	a_{13}	a_{12}	a^*_{14}	a_{10}	a_9	a_8	a_7	a^*_6	a^*_5	a^*_4	a^*_3	a^*_2	a^*_1	a^*_0
	≀			a_{10}	a_9	a_8	a^*_7	a^*_6	a^*_5	a^*_4	a^*_3	a^*_2	a^*_1	a^*_0
	≀			a_{10}	a_9	a^*_8	a^*_7	a^*_6	a^*_5	a^*_4	a^*_3	a^*_2	a^*_1	a^*_0
a^*_{11}	a^*_{14}	a_{12}	a_{13}	a_{10}	a^*_9	a^*_8	a^*_7	a^*_6	a^*_5	a^*_4	a^*_3	a^*_2	a^*_1	a^*_0
a^*_{11}	a^*_{14}	a_{12}	a^*_{13}	a_{10}	a^*_9	a^*_8	a^*_7	a^*_6	a^*_5	a^*_4	a^*_3	a^*_2	a^*_1	a^*_0
a^*_{11}	a^*_{14}	a^*_{13}	a_{12}	a^*_{10}	a^*_9	a^*_8	a^*_7	a^*_6	a^*_5	a^*_4	a^*_3	a^*_2	a^*_1	a^*_0
a^*_{11}	a^*_{14}	a^*_{13}	a^*_{12}	a^*_{10}	a^*_9	a^*_8	a^*_7	a^*_6	a^*_5	a^*_4	a^*_3	a^*_2	a^*_1	a^*_0

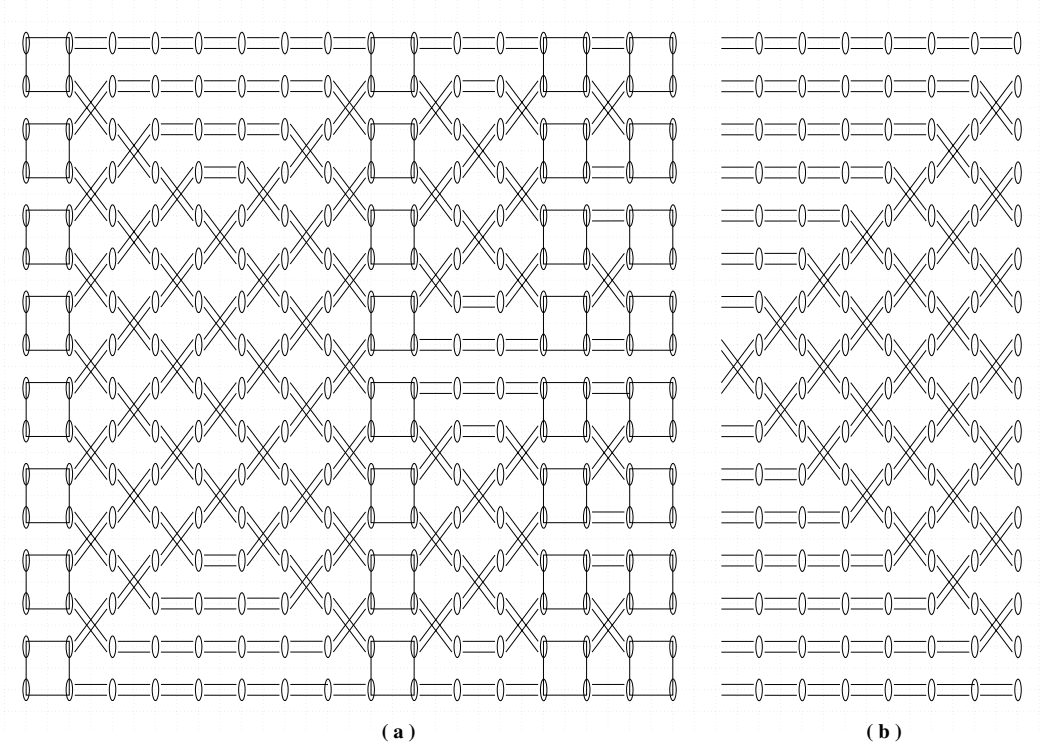

FIGURE 4. (a) Layout of the 15-stage bidelta network without long wires; each oval represents a group of $n/(d+1)$ contiguous nodes. (b) Additional stages for bit shuffling to obtain the desired permutation at the end; each oval represents a group of $3n/2(d+1)$ contiguous nodes.

the number of a group), each consisting of 2048 ($2^{d-r} = n/(d+1)$) nodes. An oval in the figure represents one such group. Ovals are connected to ovals according to the permutations and connections of the upper bits. Since these permutations are actually end-exchanges, the pattern of the connections within a batch is the same as the corresponding one in Figure 2. The detailed connections due to changes of the lower bits are not shown because of their sheer number.

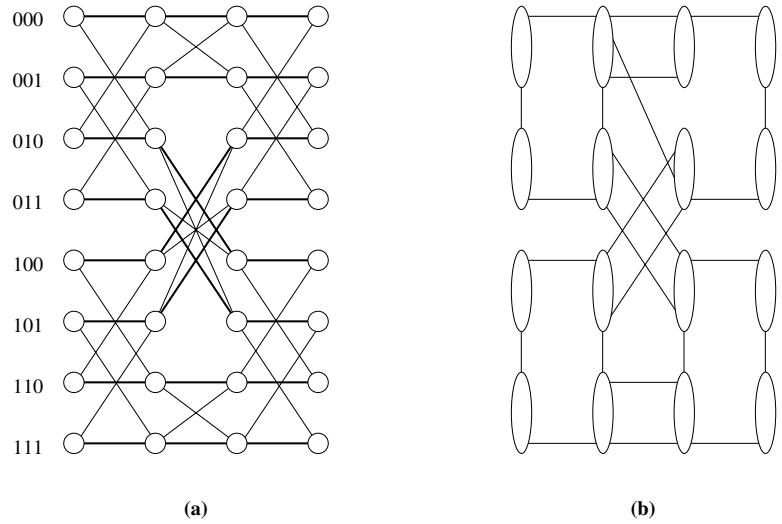

FIGURE 5. (a) Detailed layout of the 3-stage bidelta network without long wires. (b) Abstract layout; each oval represents a group of two contiguous nodes.

We give another example, that of a 3-stage bidelta network, in which all the connections can be clearly shown, as in Figure 5a. The bit changes, based on the strategy given above, are as follows.

a_2	a_1	a_0
a_2	a_1^*	a_0
a_1^*	a_2	a_0^*
a_1^*	a_2^*	a_0^*

Figure 5b gives an abstract view consisting of ovals similar to that for the 15-stage network in Figure 4.

It is easy to see that the new layout has area $\Theta(n^2)$, and, by Lemmas 1 and 2, the longest wire has length $\Theta(n/\log n)$.

4. LAYOUT OF THE BUTTERFLY NETWORK AND THE CCC

A butterfly network can be obtained from an indirect binary cube (a bidelta network) by identifying the last stage of nodes with the first stage. That means the network needs to be folded over horizontally to enable the wraparound connections. This would increase the area and the longest wire length by at worst a constant factor. In order to let the first stage of nodes coincide with the last stage, the final bit pattern (refer to the bit changes table) must be the same as the original in terms of bit positions. For the 15-stage bidelta network whose table is given as Table 1, the final bit pattern is

$$a_{11}^* a_{14}^* a_{13}^* a_{12}^* a_{10}^* a_9^* a_8^* a_7^* a_6^* a_5^* a_4^* a_3^* a_2^* a_1^* a_0^*$$

but we would want it to be

$$a_{14}^* a_{13}^* a_{12}^* a_{11}^* a_{10}^* a_9^* a_8^* a_7^* a_6^* a_5^* a_4^* a_3^* a_2^* a_1^* a_0^*.$$

Hence, the upper 4 (r) bits need to be shuffled. According to Lemma 3, another 7 ($2^{r-1}-1$) extra stages are needed to do this shuffling.

More stages means there are more nodes since $n = 2^d$. Without loss of generality, assume for convenience that $d = (2^r - 1) + (2^{r-1} - 1)$, where the first term is for the bidelta layout, and the second term is for the extra stages as just explained. Each column is divided into 2^r groups (ovals) as we did in Section 3, but each group has $2^{d-r} = 3n/2(d+2)$ contiguous nodes. Figure 4b shows the 'extra stages' of the layout of a 22-stage butterfly network without long wires, and Figure 6b shows the detailed layout of a 4-stage, 16×16 butterfly network (without folding over). The layout consists of a 3-stage layout for the bidelta network (compare this with Figure 5a) and one stage (since $r = 2$) for the shuffling to achieve the desired permutation of bits. Its bit changes are as follows.

a_3	a_2	a_1	a_0
a_3	a_2^*	a_1	a_0
a_2^*	a_3	a_1	a_0^*
a_2^*	a_3^*	a_1	a_0^*
a_3^*	a_2^*	a_1^*	a_0^*

Because of our use of the end-exchange operation to permute the upper r bits instead of the unshuffle pattern in the Beigel–Kruskal layout (see Figure 8 in [3]), we needed only $(2^{r-1} - 1)$ additional stages to shuffle the r upper bits in our layout for the butterfly network, whereas the Beigel–Kruskal layout would need more stages (approximately square that of ours) to reverse the r upper bits.

Due to the affinity of the CCC and the butterfly network, the optimal layout of the $n \times n$ butterfly network without long wires can directly translate into the corresponding optimal layout of the d-dimensional CCC without long wires, where $n = 2^d$. For example, the detailed layout for the 4-dimensional, 16×16 CCC network without folding over is shown in Figure 7b which is a straightforward derivation from Figure 6b.

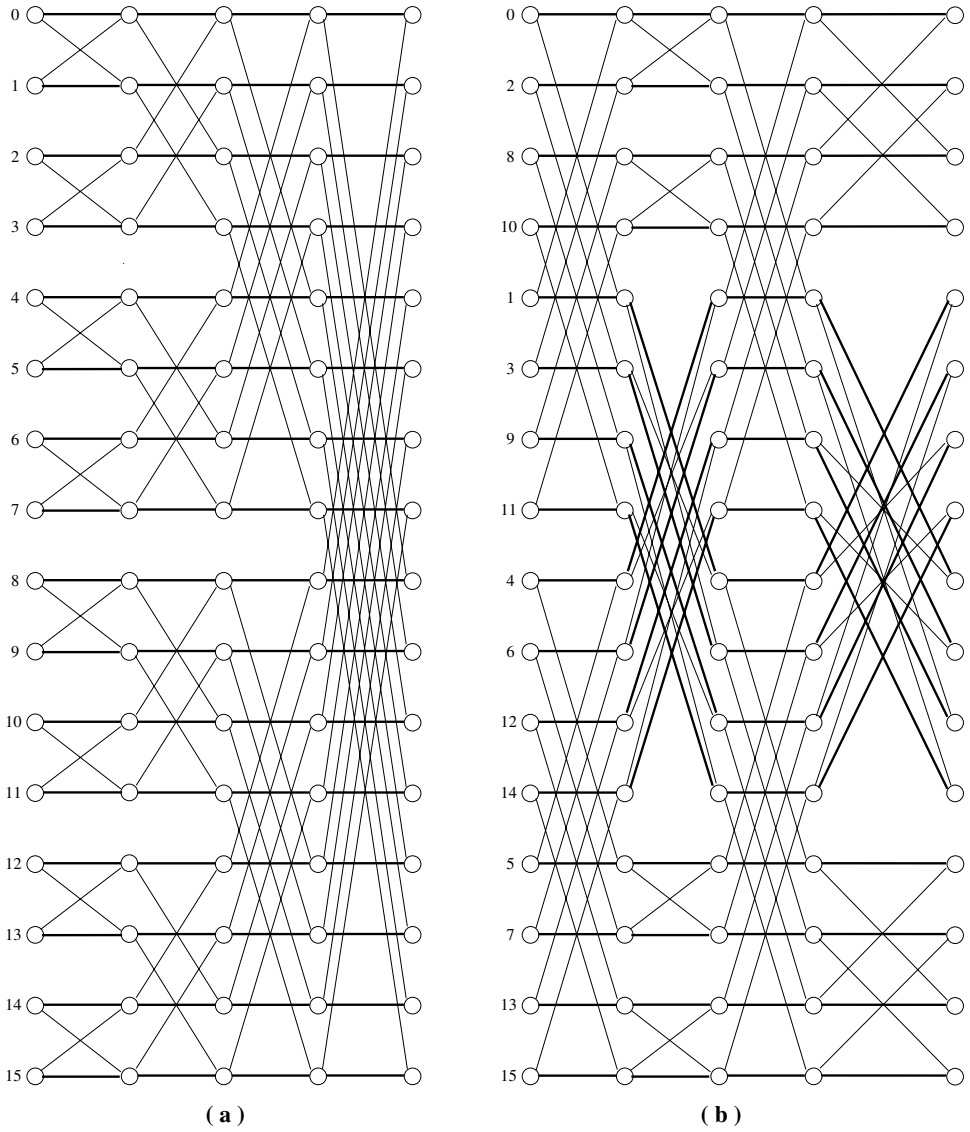

FIGURE 6. (a) A 16 × 16 butterfly network. (b) Layout of the 16 × 16 butterfly without long wires; its isomorphism to (a) can be easily checked with the help of the node numbers at the first stage.

5. LAYOUT IN THE THOMPSON MODEL

We have given in the last section a layout scheme for the butterfly network and the CCC without long wires. Referring to Figure 4 again, all wires are locally arranged—i.e. each oval i (representing a group of $3n/2(d+1)$ contiguous nodes) is connected to oval i or $i+1$ or $i-1$ in the next stage. For the layout of the CCC, oval i is also connected to itself or oval $i+1$ or oval $i-1$ in the same stage. In any case, the wire length is $\Theta(n/\log n)$ and thus optimal.

Our explanation above only shows that the given layout is without long wires in a logical way. In practice, the wire length and the area are model-dependent. Now we adopt the widely-used Thompson model and show that the given layout can be implemented in a Thompson grid with optimal area and optimal wire length. We will compute the exact upper bound for the area to see by how large a constant factor the area is blown up in order to do away with the long wires. Note that in the Thompson model, wires are limited to Manhattan routing.

5.1. Optimal area

THEOREM 1. *The CCC can be laid out optimally both in area and in wire length.*

Proof. Referring to Figure 4, there are three types of connection patterns, as shown in Figures 8a, b and c. Refer also to Figure 7b. For Figure 8a, a total of $3n/(d+2)$ contiguous nodes (in two ovals in the same stage) are connected in pairs by $3n/2(d+2)$ cube-edges. These connections are shown in Figure 8d in thin lines, which occupy $3n/2(d+2)$ columns in the worst case. Then for Figure 8b, $3n/2(d+2)$ contiguous nodes within the same oval are connected in pairs by $3n/4(d+2)$ cube-edges,

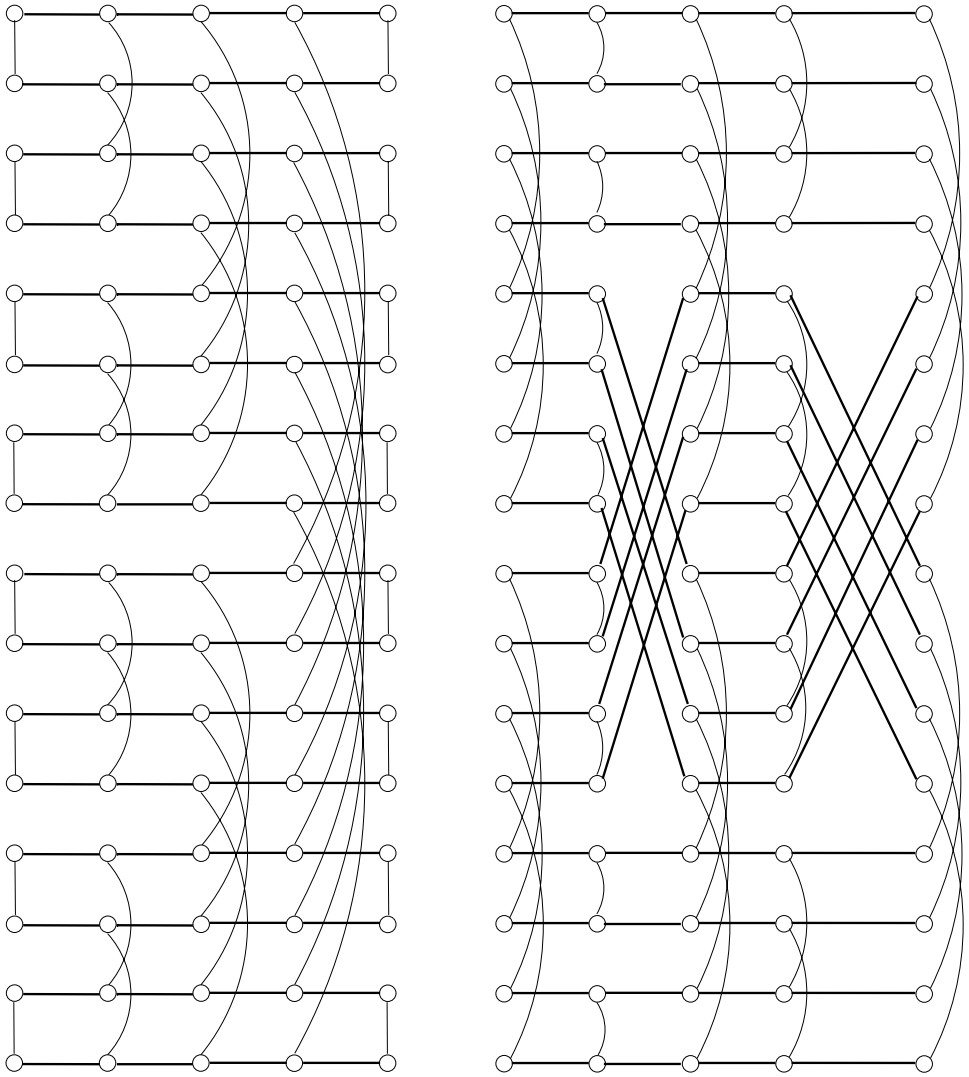

FIGURE 7. (a) A 4-dimensional, 16 × 16 CCC. (b) Layout of the 4-dimensional CCC without long wires.

occupying $3n/4(d+2)$ columns in the worst case. And for Figure 8c, $3n/2(d+2)$ contiguous nodes within the same oval are connected in pairs by $3n/4(d+2)$ cube-edges, occupying $3n/4(d+2)$ columns in the worst case. Among the three cases, only the last case has cycle-edges (in full lines in the figures) that cross. To allow the crossing, an additional $3n/2(d+2)$ columns are needed, as shown in Figure 8f.

Hence, the width of any stage is $\Theta(n/d)$. There are d stages and thus the total width is $\Theta(n)$. Plus the height $\Theta(n)$ of the layout, the total area is $\Theta(n^2)$ and thus optimal. Clearly, the length of any wire is also $\Theta(n/d)$. □

5.2. An exact upper bound on area

There are r stages, i.e. stages (i, j) for $j = 0$, which only contain pattern (a) in Figure 8. For each such stage, the width is $3n/2(d+2)$, referring to Figure 8d.

All the remaining $d - r$ stages contain patterns (b) and (c) in Figure 8. The width of such a stage is determined by pattern (c) since pattern (c) needs additional $3n/2(d+1)$ columns because of the cross connections. The number of columns required to connect the cube-edges within an oval varies from stage to stage. Totally, there are $1+2+4+8+\ldots+2^{d-r-1} = 2^{d-r}-1 = (3n/2(d+2))-1$ such columns.

Hence, the total width of the layout is

$$r\frac{3n}{2(d+2)} + (d-r)\frac{3n}{2(d+2)} + \frac{3n}{2(d+1)} - 1 \simeq \frac{3}{2}n.$$

The total height of the layout is equal to n plus the number of extra rows added because of the connections that cross (i.e. Figure 8f). Note that in Figure 8f, one extra row (at the very bottom) is added when the wires are laid out. According to Figure 4, the cross pattern in our layout happens at all rows except the top and the bottom row. Hence, we have the height of the layout equal to $n + (2^r - 3) \simeq n$.

Finally, the layout has to be folded up to produce the wraparound connections. By the standard technique, the

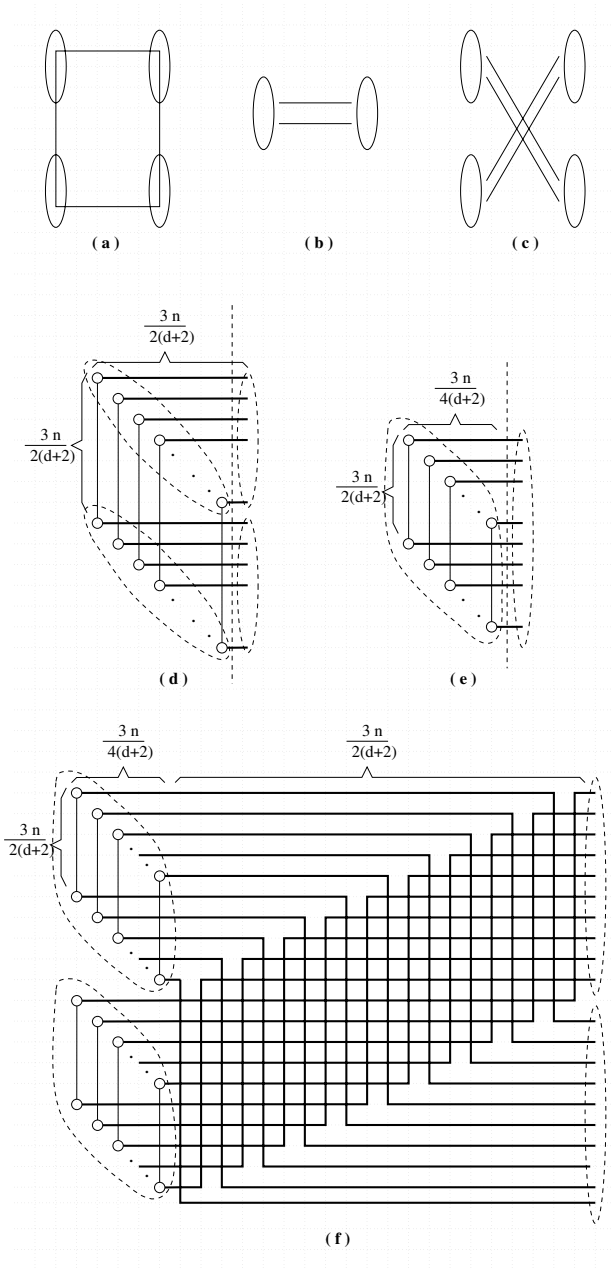

FIGURE 8. (a), (b) and (c) are the three types of connection patterns; each oval represents a group of $3n/2(d+2)$ contiguous nodes. (d), (e) and (f) show the detailed connection patterns of (a), (b) and (c), respectively, in a Thompson grid. Figure 9 shows an example of a complete layout.

original width and height are doubled. Hence, the final area is $6n^2$ (save some lower-order terms).

6. CONCLUSION

It should be noted that the desired layout of the CCC cannot be derived from the layout of the circular shuffle network, although Beigel and Kruskal had expected the opposite based on the similarity between the CCC and the circular shuffle network [3, 17]. In fact, we are doubtful about the existence of an optimal layout of the circular shuffle network

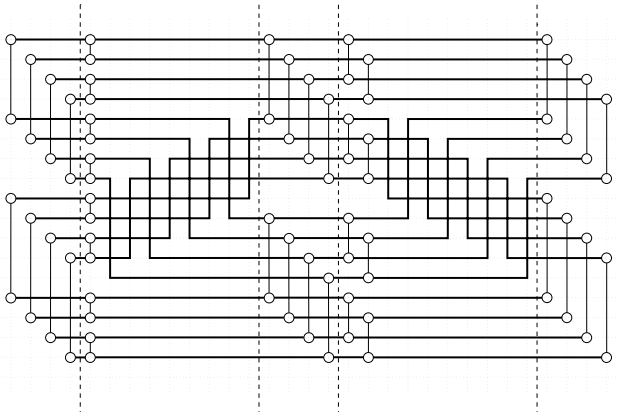

FIGURE 9. Optimal layout of a 4-dimensional CCC in a Thompson grid.

without long wires. Even if one existed, the desired layout of the CCC derived from it could not be better than our layout in area, taking into account constant factors.[3]

Blum has shown that there exist some graphs whose minimal-area layout require much longer wires than a constant factor in comparison with some less-than-optimal layouts [9]. On the other hand, there exist graphs, especially those that are based on the mesh topology, that can be laid out in both minimal area and minimal longest wire [8].

It is interesting and practical to try to analyse the tradeoff between the area and the maximum wire length for important graphs such as the CCC. In this paper, we have given a new layout of the CCC whose maximum wire length is reduced to an optimal level; the cost is a constant-factor blow-up in area. Hence, we say that the CCC can be laid out optimally within a constant factor both in area and in maximum wire length. Specifically, the area of the new layout, which is approximately $6n^2$, represents a blow-up of 12 times since the best minimal layout area of the CCC is $n^2/2$ [2].

Considering certain physical limitations in parallel computers such as the speed of propagation of information, small-diameter networks do not necessarily have an advantage over, say, mesh-connected networks which have a large diameter ($\Theta(N^{1/2})$) [18, 19]. The problem of the small-diameter networks is that they are often not scalable: as they grow in size, the wire length must grow, thus degrading communication performance. Hence, seeking an efficient layout without long wires for the small-diameter networks such as the CCC becomes very important in order to demonstrate their suitability as an interconnection network for the implementation of parallel computers.

Our further research is to establish lower bounds on the area and the maximum wire length for the CCC taking into account constant factors [20]. Must all the minimum-area layouts have long wires? What is the minimal constant blow-up when reducing the length of wires to an optimal level?

[3]The details are beyond the scope of this paper.

ACKNOWLEDGEMENT

Guihai Chen's work is supported by the China National Science Foundation under grants #60073029 and #69803005.

REFERENCES

[1] Preparata, F. P. and Vuillemin, J. (1981) The cube-connected cycles: a versatile network for parallel computation. *CACM*, **24**, 300–309.

[2] Chen, G. and Lau, F. C. M. (2000) Tighter layouts of the cube-connected cycles. *IEEE Trans. Par. Distrib. Syst.*, **11**, 182–191.

[3] Beigel, R. and Kruskal, C. P. (1989) Processor networks and interconnection networks without long wires. In *ACM Symp. on Parallel Algorithm and Architecture*, pp. 42–51. ACM Press.

[4] Feldman, Y. and Shapiro, E. (1992) Spatial machines: a more realistic approach to parallel computation. *Commun. ACM*, **35**, 61–73.

[5] Vitanyi, P. M. B. (1986) Nonsequential computation and laws of nature. *VLSI Algorithms and Architecures. Lecture Notes in Computer Science*, **227**. Springer, Berlin.

[6] Hillis, W. D. (1985) *The Connection Machine*. MIT Press, Cambridge, MA.

[7] Lai, T.-H. and Speangue, A. (1991) Placement of the processors of a hypercube. *IEEE Trans. Comp.*, **40**, 714–722.

[8] Lau, F. C. M. and Chen, G. (1996) Optimal layouts of midmiew networks. *IEEE Trans. Par. Distrib. Syst.*, **7**, 954–961.

[9] Blum, N. (1985) An area-maximum edge length tradeoff for VLSI layout. *Information and Control*, **66**, 45–52.

[10] Thompson, C. D. (1979) Area-time complexity for VLSI. In *Proc. 11th Ann. Symp. on Theory of Computing*, Atlanta, GA, May 1979, pp. 81–88.

[11] Thompson, C. D. (1980) *A Complexity Theory for VLSI*. PhD Thesis, Computer Science Department, Carnegie Mellon University.

[12] Mead, C. and Conway, L. (1980) *Introduction to VLSI Systems*. Addison-Wesley.

[13] Mehlhorn, K., Preparata, F. P. and Sarrafzadeh, M. (1986) Channel routing in knock-knee mode: simplified algorithms and proofs. *Algorithmica*, **1**, 213–221.

[14] Leighton, F. T. (1992) *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercube*. Morgan Kaufmann.

[15] Annexstein, F., Baumslag, M. and Rosenberg, A. L. (1990) Group action graphs and parallel architecture. *SIAM J. Comp.*, **19**, 544–569.

[16] Kruskal, C. P. and Snir, M. (1986) A unified theory of interconnection network structure. *Theoret. Comp. Sci.*, **48**, 75–94.

[17] Jain, B. N. (1986) Equivalence between cube-connected cycles networks and circular shuffle networks. In *Proc. Int. Conf. on Parallel Processing*, pp. 8–11. IEEE Computer Society Press, Los Alamitos, CA.

[18] Mazumder, P. (1986) Evaluation of three interconnection networks for CMOS VLSI implementation. In *Proc. Int. Conf. on Parallel Processing*, pp. 200–207. IEEE Computer Society Press, Los Alamitos, CA.

[19] Vitanyi, P. M. B. (1988) Locality, communication, and interconnection length in multicomputers. *SIAM J. Comput.*, **17**, 659–672.

[20] Lin, F.-C. and Shih, W.-K. (1986) Long edges in the layouts of shuffle-exchange and cube-connected cycles graphs. *Inf. Proc. Lett.*, **23**, 5–9.