VLSI Layout and Packaging of Butterfly Networks

Chi-Hsiang Yeh
Dept. of Electrical & Computer Eng.
Queen's University
Kingston, Ontario K7L 3N6, Canada
chi-hsiang.yeh@ece.queensu.ca

Behrooz Parhami, E.A. Varvarigos, and H. Lee
Dept. of Electrical and Computer Eng.
University of California
Santa Barbara, CA 93106-9560, USA
{parhami, manos, hualee}@ece.ucsb.edu

ABSTRACT
We present a scheme for optimal VLSI layout and packaging of butterfly networks under the Thompson model, the multilayer grid model, and the hierarchical layout model. We show that when L layers of wires are available, an N-node butterfly network can be laid out with area $\frac{4N^2}{L^2 \log_2^2 N} + o\left(\frac{N^2}{L^2 \log^2 N}\right)$, maximum wire length $\frac{2N}{L \log_2 N} + o\left(\frac{N}{L \log N}\right)$, and volume $\frac{4N^2}{L \log_2^2 N} + o\left(\frac{N^2}{L \log^2 N}\right)$, under the multilayer 2-D grid model, where only one active layer (for network nodes) is required and L layers of wires are available. Our layout scheme allows us to partition an N-node butterfly network into $\Theta(N^{1/l} \log^{1-1/l} N)$-node clusters with an average of $d_{\rm ic} \approx \frac{4l-4}{\log_2 N}$ $(= O(\frac{1}{\log N})$ for any constant integer l) inter-cluster links per node, leading to optimal layout and packaging at the same time under the hierarchical layout model. The scalability of our layouts are optimal in that we can allow each of $O(N/\log N)$ nodes to occupy an area as large as $o\left(\frac{N}{L^2 \log N}\right)$ and each of the remaining $N - o(N)$ network nodes to occupy an area as large as $o\left(\frac{N}{L^2 \log^2 N}\right)$, without increasing the leading constants of layout area, volume, or maximum wire length.

1. INTRODUCTION
Under present assumptions on the computing environments, processors are expensive and memory is relatively cheap. Therefore, an important problem is to utilize processors efficiently. For general-purpose applications, the utilization of processors in parallel computers is not as efficient as that in single-processor computers, so that the latter achieve better performance per dollar. In future computing environments, however, the roles are expected to be reversed, so that the majority of chip area is used for memory. Therefore, the optimization problem may then become the efficiency in utilizing the relatively expensive memory. Recent research on processors in memory (PIM), computing in RAM, and smart memory paradigms [9, 14, 18] shows that multiple processors per chip, integrated with memory banks, can increase the memory-processor bandwidth considerably, leading to improved memory utilization. Also, with the rapid advances in VLSI technologies, the number of transistors and the number of processors that can be put onto a chip are expected to continue their exponential growths. Moreover, as pointed out by Dally and Lacy [9], so far there is no efficient alternative to explicit parallelism for exploiting the increased number of transistors and grid points. Therefore, single-chip multiprocessors are expected to achieve better performance per dollar even for general-purpose applications and may achieve mainstream status in the future.

Another related development is the renewed interest in VLSI layouts of switching and sorting networks used in network switches and routers [16]. Many network switches/routers are based on butterfly, Benes, or related interconnection topologies. Efficient VLSI layout is very important to VLSI chips, and potentially significant for other technologies such as multichip modules (MCM) and wafer-scale integration (WSI), implementing the aforementioned multiprocessors and network switches. For larger parallel architectures and switches that are implemented using more than one packaging level (such as chip, board, and cabinet levels), efficient network partition and packaging are also very important. The impact of efficient VLSI layout and packaging on the cost-performance of the resulting parallel architecture is amplified by the lower cost obtained through the reduction in the number of chips, boards, and assemblies, and higher performance achieved by lowering various performance hindrances, such as signal propagation delay, drive power, and fraction of data transfers to off-chip destinations. This explains the reintensified research on finding efficient VLSI layouts and packaging, especially for butterfly networks [1, 10, 11, 16, 21, 26, 27]. Efficient layouts and packaging considerations for other interconnection networks can be found in [7, 8, 12, 13, 24].

Butterfly networks are among the most important topologies for building commercial and experimental parallel computers, special-purpose processors, and network switches. Recently, Avior et al [1] proposed a VLSI layout for butterfly networks under the grid model, showing that an N-node butterfly network can be laid out in $\frac{N^2}{\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$ area using two layers of wires. They also showed that the layout area is optimal within a factor of $1 + o(1)$ when area is defined by an upright encompassing rectangle. In [26, 27], we proposed optimal VLSI layouts for butterfly networks (in asymptotically the same area as in [1]) based on optimal 2-D layouts of complete graphs. In our layout, the network nodes (and clusters of nodes) are aligned as a 2-D grid so that they utilize area efficiently, especially when network nodes (containing processors and memory banks) are large. Our layout is *scalable* in terms of node size since each node can be as large as a square with side $W = o\left(\frac{\sqrt{N}}{\log N}\right)$ without affecting the leading constant of the layout area. In [16], Muthukrishnan et al proposed a butterfly layout under the knock-knee model [5] that uses area $\frac{2N^2}{3\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$ area. Note that knock-knee layouts usually require more than two layers of wires for actual wiring within the same area. In [10], Dinitz et al showed that butterfly networks can be laid out in $\frac{N^2}{2\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right)$ area when area is characterized by a slanted encompassing rectangle (or equivalently, all wires run at 45°), which is optimal within a factor of $1+o(1)$ under this model. In this paper, we propose optimal butterfly layouts under the *Thompson model*, *multilayer grid models*, and *hierarchical layout model* based on optimal collinear layout of complete graphs. Like our previous layouts in [26, 27], our new butterfly layouts align the network nodes and clusters of nodes as a 2-D grid, leading to optimal scalability in terms of node sizes, but improve the maximum wire length of the layouts given in [26, 27] by a factor of 2 when using 2 layers of wires and by a factor of L when using L layers of wires.

We show that, under the multilayer 2-D grid model, a butterfly network can be laid out using L layers of wires, $2 \leq L = o(\sqrt[3]{N})$, with area $\frac{4N^2}{L^2\log_2^2 N} + o\left(\frac{N^2}{L^2\log^2 N}\right)$, maximum wire length $\frac{2N}{L\log_2 N} + o\left(\frac{N}{L\log N}\right)$, and volume $\frac{4N^2}{L\log_2^2 N} + o\left(\frac{N^2}{L\log^2 N}\right)$, where only one active layer is required (for network nodes). In practice, a network node may be much wider than a wire, especially when the node contains one or several processors, memory banks, and a router. Our proposed layouts are scalable in terms of node size, in that the leading constants of the layout area, maximum wire length, and volume are not affected even if each of $O(N/\log N)$ nodes occupies a square of side $W' = o\left(\frac{\sqrt{N}}{L\sqrt{\log N}}\right)$ and each of the remaining $N - o(N)$ network nodes occupies a square of side $W = o\left(\frac{\sqrt{N}}{L\log N}\right)$. Clearly, the scalability of the multilayer layout is asymptotically optimal in terms of node sizes. The multilayer butterfly layouts derived here are, to the best of our knowledge, the best results reported in the literature thus far for $L = 3, 4, \ldots, o(\sqrt[3]{N})$. Note that although we present the layout area, volume, and wire length using asymptotic notation, our layouts are more efficient than those previously proposed, even for networks of modest size.

We propose *hierarchical layouts* of butterfly networks that achieve asymptotically optimal area and packaging at the same time. In particular, our scheme allows us to partition an $R\times R$ butterfly network into $\Theta(R^{1/l}\log_2 R)$-node clusters with an average of $d_{\text{ic}} = \frac{4l-4}{\log_2 R+1} (= O(\frac{1}{\log R}) = O(\frac{1}{\log N})$ for any $l = O(1)$) inter-cluster links per node, leading to efficient packaging. In the process of presenting optimal packaging and layouts for the butterfly networks, we introduce *indirect swap networks*, a class of multistage networks with flexible connectivity and efficient packaging properties, and show how to transform them into corresponding butterfly networks. By appropriately selecting parameters for the indirect swap network to be transformed, the resultant hierarchical layout for the butterfly network can be adapted to various packaging constraints.

The remainder of the paper is organized as follows. In Section 2, we present an efficient partitioning and packaging scheme for butterfly networks. In Section 3, we present optimal layouts for butterfly networks under the Thompson model. In Section 4, we introduce the multilayer grid models and present efficient multilayer layouts of butterfly networks. In Section 5, we introduce the hierarchical layout model and present efficient hierarchical layouts of butterfly networks. In Section 6 we present our conclusions, while in the appendix, we give the definitions of swap networks and indirect swap networks.

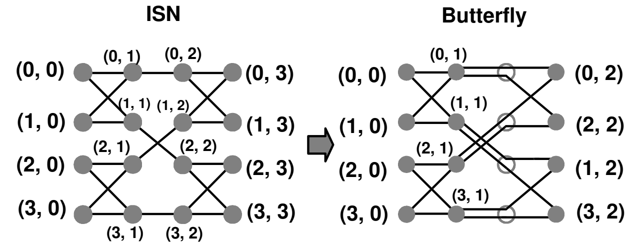

Figure 1: Transforming a 4×4 indirect swap network into a 4×4 butterfly network. The address for each of the network nodes is marked beside or above that node.

2. PACKAGING OF BUTTERFLIES

In this section, we propose an efficient partitioning and packaging scheme for butterfly networks.

2.1 Indirect Swap Networks

An *Indirect swap network (ISN)* is obtained by unfolding a swap network [23, 24, 28], in the same way that many multistage networks are obtained from their direct network counterparts. An ISN is characterized by the parameters $k_1, k_2, \ldots, k_l$. The majority of network nodes in an ISN with $k_1 \geq 3$ have two straight links and two cross links, and the remaining nodes have one straight link, one cross link, and one swap link, except for those in the first or last stage. A useful property of ISNs is that they can be easily and naturally partitioned into clusters so that only a small number of links (i.e., some/all of the swap links) will become inter-cluster links while most of the links (i.e., all straight and cross links) are confined within clusters (see Figs. 1 and 2). We can also derive optimal layout for ISNs based on the collinear layout of complete graphs. Different parameters lead to ISNs with different partitioning and layout characteristics. Figure 1 shows a 4×4 ISN with $k_1, k_2 = 1$. The definitions of swap networks, ISNs, and their parameters are given in Appendix A.

ISNs are closely related to butterfly networks. In the following subsections, we show how to modify ISNs to obtain automorphisms of butterfly networks. A packaging and layout scheme for butterfly networks can then be derived based on this transformation, by first deriving efficient packaging and layout for the modified ISNs. Note that we can present efficient packaging and layout for butterfly networks for several special cases (e.g., when two levels of hierarchy are present and for certain choices of cluster sizes, say, $\Theta(\sqrt{N\log N})$ nodes per cluster) in a simpler manner without resorting to ISNs, but we need the structural features and parameters of ISNs to make the partitioning and layout scheme general and adaptable to various assumptions and technological requirements.

2.2 Transforming ISNs into Butterfly Networks

An automorphism of a butterfly network can be obtained by modifying the stages of an ISN that are connected by swap links in a way to be described shortly. The resultant modified network is called a *swap-butterfly*.

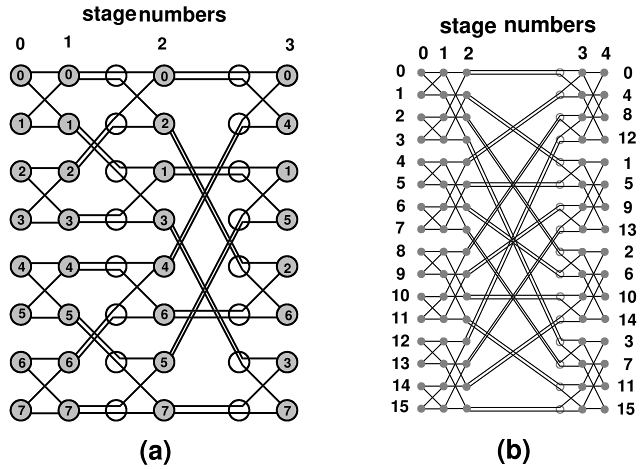

Figure 2: Mapping butterfly networks onto swap-butterflies. (a) A resultant 8×8 butterfly network, where the row numbers are marked in the circles. (b) A resultant 16×16 butterfly network. The first column of numbers represent the row numbers for stages 0, 1, and 2, and the last column of numbers represent the row numbers for stages 3 and 4.

We denote a pair of stages connected by swap links as stages $i-1$ and i. We first double up all the swap links connecting stages $i-1$ and i of the ISN, remove the nodes in the i^{th} stage (through bypassing), and reconnect each of the replicated links to one of the two links between the i^{th} and the $(i+1)^{th}$ stages through a removed node. The row numbers of nodes in the swap-butterfly remain the same as in the original ISN, while the stage numbers of nodes (except for the first k_1+1 stages) are reduced, since some stages in the original ISN are removed in the swap-butterfly. Figures 1 and 2 show the automorphisms of 4×4, 8×8, and 16×16 swap-butterflies. The row number of a node in these swap-butterflies is equal to the position of the row it belongs to; the stage numbers are the same as those indicated in the figures for the corresponding butterfly networks.

A swap-butterfly with $n_l = \sum_{i=1}^{l} k_i$ is an automorphism of an n_l-dimensional butterfly network. To see that, consider the following mapping:

- (1) a node in the swap-butterfly is mapped to a node in the butterfly network that has the same stage number;
- (2) a node in stage 0 of the swap-butterfly is mapped to a node in stage 0 of the butterfly network that has the same row number;
- (3) two nodes in the swap-butterfly connected by a straight link or by a swap link followed by a straight link are mapped to two neighboring nodes with the same row number in the butterfly network.

Note that for a node with stage number larger than k_1 (i.e., to the right of the stages connected by swap links), the row number of the node in the swap-butterfly is different from the row number of the embedded node in a butterfly network. For example, node $(1,2)$ of the swap-butterfly of Fig. 1 (i.e., the one at the intersection of row 1 and stage 2) is mapped to node $(2,2)$ of an n_l-dimensional butterfly network, so the row number of node $(1,2)$ of the swap-butterfly is 2 in a butterfly network.

The readers can verify that the transformed ISNs shown in Figs. 1 and 2 (with renamed node labels) are indeed butterfly networks by looking at their connectivity. An intuitive reason for this is that ISNs can perform fast Fourier transform (FFT) using a variant of an ascend algorithm. (In an ascend algorithm, two nodes whose addresses differ only at bit i exchange packets at step i, $i = 1, 2, 3, ..., \log_2 R$, and the flow graph of such an ascend algorithm is exactly an $R \times R$ butterfly network.) More precisely, at step i, $i = 1, 2, 3, ..., \log_2 R$, if the data are held at a stage (column) of an ISN that connects to the next stage with straight and cross links, two nodes whose addresses differ only at bit i can exchange packets directly; if these two stages are connected with swap links, the data packets have to be forwarded to the next stage using swap links, and then they can be exchanged using straight and cross links connecting to the following stage. We can see that the flow graph of this algorithm is exactly an ISN and the algorithm essentially performs the function of an ascend algorithm, with some additional steps for forwarding packets over swap links. Therefore, if we "by-pass" the downstream stage reachable via swap links, while reconnecting one of the duplicated swap link to a straight link and the other one to a cross link, we obtain an automorphism of the butterfly network.

2.3 Optimal Butterfly Packaging

We are now in a position to present an asymptotically optimal partitioning and packaging scheme for butterfly networks. We first consider the case where there are only two packaging levels and there is a limit on the number of nodes that can be placed onto a single module, such as a chip, board, multi-chip module (MCM), or wafer. The partitioning and packaging methods to be described can be extended to the case where there are more than two levels in the packaging hierarchy. Our goal is to partition butterfly networks and map network nodes onto the modules so that the number of modules and the number of off-module pins required per module are minimized. We also prefer to build the butterfly networks with identical modules or a small number of different modules to eliminate the "number of parts" problem.

Let B_n denote an n-dimensional butterfly network. When $O(R^{1/l}\log_2 R) = O(2^{n_l/l} n_l)$ nodes can be placed in a single module, we begin with the $R \times R$ ISN(l, B_{k_1}) derived from HSN(l, Q_{k_1}) (see Sections A.1 and A.2), where $k_i = n_l/l$ for all $i = 1, 2, ..., l$ and $R = 2^{n_l}$, assuming that the dimension n_l of the butterfly network is a multiple of l first for simplicity. We then transform the ISN into an $R \times R$ butterfly network and place each set of $R^{1/l} = 2^{n_l/l} = 2^{k_1}$ consecutive rows of the swap-butterfly (l, B_{k_1}) (i.e., the one transformed from ISN(l, B_{k_1})) onto the same module, leading to a total of $R^{\frac{l-1}{l}} = 2^{\frac{l-1}{l}n_l} = 2^{(l-1)k_1}$ modules. By partitioning the swap-butterfly (l, B_{k_1}) in this way, we can place each nucleus butterfly network B_{k_1} in the same module so that only swap links will leave the modules, while straight links and cross links are all confined to within modules. Since there are $4(l-1)$ swap links per row in the swap-butterfly (l, B_{k_1}), a fraction $\frac{2^{k_1}-1}{2^{k_1}}$ of which leave the module, and there are n_l+1 nodes per row, the average number of off-module links per node is

$$\frac{4(l-1)(2^{k_1}-1)}{(n_l+1)2^{k_1}} < \frac{4(l-1)}{n_l+1} < \frac{4}{k_1}.$$

An example for partitioning a 9-dimensional butterfly network is given in Section 5. As a basis for comparison, the average number of off-module links per node when placing consecutive rows of a butterfly network onto the same module is approximately equal to 2. Therefore, our packaging scheme outperforms the latter simple scheme by a factor of $\Theta(\log N)$ and is better even for small k_1 (e.g., $k_1 = 3$). As will be shown in Section 3, our method leads to optimal packaging (within a constant factor) and optimal layout area (within a factor of $1 + o(1)$) at the same time.

If the number of nodes that can be placed within a module is even smaller and is

$$O(R^{1/l} \log_2 R/l) = O(2^{n_l/l} k_1) = O(2^{k_1} k_1) = O(N^{1/l} \log N/l),$$

we can place a single nucleus butterfly network B_{k_1} onto a module. Since a node in the first or last stage of the nucleus butterfly network B_{k_1} has 2 swap links connecting to other modules, and there are k_1 nodes per row of the nucleus butterfly network, the average number of off-module links per node is $\frac{4}{k_1}$. Although it may seem that the reduction in module size is not large and the average number of off-module links per node is slightly larger, this variant packaging may perform better for networks of practical sizes. This is because for a given upper bound on the permissible module size, we may choose a larger k_1 and a smaller l using the latter method so that the resultant average number of off-module links per node is smaller. Moreover, each node in the first (or last)stage of the burly network can now have 2 links connecting to processors (or memory modules, respectively) without increasing the average number of off-module links per node. If we include these links in comparing the two methods, the difference in the average numbers of off-module links per node becomes very small (i.e., smaller than $\frac{1}{2^{k_1-1}}$ of the average numbers of off-module links per node), even when using the same values for k_1 and l in both cases.

The preceding partitioning scheme can be extended to the case where $\log_2 R = n_l$ is not a multiple of l. We first transform an ISN with $k_i = k_1$ for $i = 2, 3, 4, \ldots, l-1$ and $k_l < k_1$ (i.e., an ISN derived from an incomplete HSN(l, Q_{k_1}) [23]) into a butterfly network. We then either place 2^{k_1} consecutive rows of the swap-butterfly or a basic building block (B_{k_1} or B_{k_l}) onto the same module. The latter method leads to the following theorem.

Theorem 2.1. *We can partition an $R \times R$ butterfly network into modules that have no more than $2^{k_1} k_1$ nodes and no more than 2^{k_1+2} off-module links per module, which is asymptotically optimal within a constant factor.*

A matching lower bound on the number of off-module links required can be derived by computing the maximum injection rate for performing random routing in a butterfly network with uniformly distributed sources and destinations. The maximum injection rate is $\Theta(1/\log R)$ since the average distance is $O(\log R)$ and the traffic is balanced within a constant factor between the most heavily used nodes/links and the average, leading to a lower bound $\Omega(M/\log R)$ on the number of off-module links required for an M-node module to be able to support such an injection rate. Note that in such a partition, nodes in the first or last stage can have additional links connecting to processors or memory banks; $l-1$ out of l modules contain B_{k_1} and the remaining module (for the last k_l stages of the swap-butterfly) is B_{k_l}. We can see that the greater the number of nodes in a module (or k_1), the more efficient the packaging in terms of the number of off-module links per module.

The proposed partitioning and packaging methods can be extended to the case where there are more than two levels in the packaging hierarchy by selecting appropriate ISNs that can be modified to give an automorphism of the butterfly network. Since the module sizes at higher levels are usually larger, the improvements over the simple partitioning and packaging scheme are even more significant.

3. OPTIMAL LAYOUT OF BUTTERFLIES UNDER THE THOMPSON MODEL

In this section, we present layouts for butterfly networks under the Thompson model that are optimal within a factor of $1 + o(1)$.

3.1 The Thompson Model

In the Thompson model, a network is viewed as a graph whose nodes correspond to processing elements and edges correspond to wires. The graph is then embedded in a 2-D grid, where wires have unit width and a node of degree d occupies a square of side d. The wires can run either horizontally or vertically along grid lines. Two wires can cross each other at a grid point, but cannot overlap or bend at the same grid point, which would form a knock-knee [5].

The area of a layout is defined as the area of the smallest rectangle that contains all the nodes and wires. (In this paper we only consider upright rectangle for this purpose.) When there are two layers of wires, it is guaranteed that we can lay out the network within the area. More precisely, we can use one layer of wires to lay out all the horizontal segments of wires and the other layer to layout all the vertical segments. When a wire makes a turn, its horizontal and vertical parts in different layers are connected by an inter-layer connector known as a via. In Section 4, we modify the butterfly layout derived in this section to obtain more compact layouts that use L layers of wires with $L > 2$.

Note that some authors have assumed that a node occupies a square of side 1 in the layout model they use. Some of such layouts cannot be extended to the Thompson model without a nonnegligible increase in area, while layouts under the Thompson model can usually be extended to the former model using comparable area.

3.2 Butterfly Layout with Dimension $n = 3k_1$

We first lay out n-dimensional butterfly networks assuming that n is a multiple of 3 and then extend the layout method to the general case. We transform an ISN$(3, B_{n/3})$ with $k_3 = k_2 = k_1 = n/3$ to obtain an automorphism of an n-dimensional butterfly network. We place every $\sqrt[3]{R} = 2^{n/3}$ rows of the swap-butterfly into the same block, and arrange them as a 2-D $2^{n/3} \times 2^{n/3}$ grid in the row major order. That is, the first $2^{n/3}$ blocks (which contains the first $2^{2n/3}$ rows) are arranged as the first row of the 2-D grid, the next $2^{n/3}$ blocks are arranged as the second row of the 2-D grid, and so on. By verifying the connectivity of ISNs, we can see that a row or column of the resultant graph, when viewing each block as a supernode, becomes a complete multigraph (i.e., a complete graph with multiple links). More precisely, if we merge each row of an ISN$(3, B_{n/3})$ into a supernode, it becomes the HSN$(3, Q_{n/3})$ it was derived from, where each inter-cluster link is duplicated (corresponding to two swap links); if we continue to merge each nucleus hypercube $Q_{n/3}$ of an HSN$(3, Q_{n/3})$ (without multiple links) into a supernode (corresponding to a block), it becomes a 2-dimensional radix-$2^{n/3}$ generalized hypercube [4], where each pair of (super)nodes belonging to the same row or column are connected by a link. Since each swap link of an

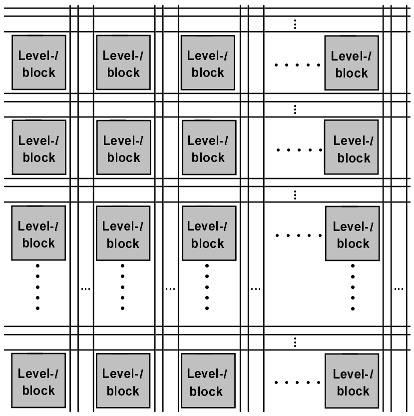

Figure 3: Top-view of a layout based on the recursive grid layout scheme. Level-l blocks are arranged as a 2-D grid.

ISN is duplicated to transform it to a corresponding butterfly network B_n, each pair of blocks belonging to the same row or column in the preceding partition are connected by 4 links.

We use the *recursive grid layout scheme* we proposed in [27] to lay out the swap-butterfly. In this scheme, we arrange the blocks as a 2-D grid, where neighboring rows (or columns) are separated by a sufficient number of horizontal (or vertical, resp.) tracks (see Fig. 3). Then we lay out the swap links connecting these blocks, then lay out each of the block (which contains 3 $n/3$-dimensional butterfly networks) either recursively or using any previous layout [1, 26, 27], and finally connect the links incident to each of the blocks to nodes within that block. Since supernodes within each row (or column) of blocks are completely connected, we can use the collinear layout presented in Appendix B to lay out these swap links. More precisely, we view each block as a supernode in a complete graph and replicate each of the wires of the collinear layout of a $2^{n/3}$-node complete graph four times. The number of tracks required for a row or column is $2^{2n/3}$, for a total of 2^n vertical tracks and 2^n horizontal tracks. Since three $n/3$-dimensional butterfly networks can be laid out in $O(2^{2n/3})$ area [1, 10, 16, 26] and the additional area required for accommodating the links connecting nodes to links incident to the block to which they belong is also $O(2^{2n/3})$, the area for the butterfly network is dominated by links connecting these blocks. Therefore, the area of the layout for an N-node butterfly network is

$$2^{n/3}2^{2n/3} \times 2^{n/3}2^{2n/3} + o(2^{2n}) = \frac{N^2}{\log_2^2 N} + o\left(\frac{N^2}{\log^2 N}\right),$$

which is optimal within a factor of $1 + o(1)$ [1], and the maximum wire length is

$$2^{n/3} \times 2^{2n/3} + o(2^n) = \frac{N}{\log_2 N} + o\left(\frac{N}{\log N}\right),$$

which is a factor-of-2 improvement over that of our previous layouts proposed in [26, 27].

3.3 Butterfly Layout with General Dimension

In what follows we generalize the layout to butterfly networks whose dimension n is not a multiple of 3. For such an n, we transform an ISN with $k_2 = k_1 = (n+1)/3$ and $k_3 = (n-2)/3$ when n modulo 3 is 2, or with $k_1 = (n+2)/3$ and $k_3 = k_2 = (n-1)/3$ when n modulo 3 is 1, to obtain an automorphism of an n-dimensional butterfly network. We can then lay out the swap-butterfly using the preceding layout method. The only difference is that we place every 2^{k_1} rows of the swap-butterfly into the same block, and arrange them as a 2-D $2^{k_3} \times 2^{k_2}$ grid in the row major order. We then use the collinear layout of a 2^{k_2}-node complete graph, each wire of which is replicated $2^{2+k_1-k_2}$ times, to lay out links connecting each row of blocks and use the collinear layout of an 2^{k_3}-node complete graph, each wire of which is replicated $2^{2+k_1-k_3}$ times, to lay out links connecting each column of blocks. It can be easily verified that the area and maximum wire length, remain asymptotically the same (with the same leading constant and lower order terms of the same order) as the case where n is a multiple of 3.

Note that the nucleus butterfly networks B_{k_1}, B_{k_2}, and B_{k_3} within blocks can be laid out using the preceding method recursively. This corresponds to transforming ISNs derived from recursive hierarchical swap networks [23] into butterfly networks and laying out the swap-butterflies. We can also transform $\mathrm{ISN}(l, B_{k_1})$ with $l > 3$ into a butterfly network and then lay it out either using the recursive grid layout scheme [27] or using a bottom-up method as the method we used to lay out hypercubes in [26]. For both methods, the leading constants of the resultant area and maximum wire length. remain the same By selecting an appropriate ISN for transformation to a butterfly network, both the layout area and the number of off-module links can be minimized. Note also that since the network nodes are arranged as a 2-D grid in our layout, the area of the proposed layout is not sensitive to the size of network nodes. When each network node occupies a square of side W for any $W = o\left(\frac{\sqrt{N}}{\log N}\right)$, the leading constants of the layout area and maximum wire length are not affected so the layout is *scalable* in terms of node size. We can also let each of $O(R) = O(N/\log N)$ nodes occupy a square of side W' for any $W' = o\left(\sqrt{\frac{N}{\log N}}\right)$ and each of the remaining $N - o(N)$ nodes occupy a square of side $W = o\left(\frac{\sqrt{N}}{\log N}\right)$, without affecting the leading constants. The latter is particularly useful for butterfly networks with processors and memory banks at the first and/or last stages. It is clear that larger node sizes (by a nonconstant factor) will lead to nonnegligible increase in the layout area. Thus, the scalability of our butterfly layout under the Thompson model is *optimal* in terms of node sizes. Compared with the method proposed in [1, 16], our approach leads to layouts that are not only area-optimal (within a factor of $1 + o(1)$), but also require a factor of $\Theta(\log N)$ fewer pins when there are at least two levels in the packaging hierarchy, given that no more than $O(N^{1/2-\epsilon})$ nodes can be placed in a module for any constant ϵ [e.g., for modules with nodes $\Theta(\sqrt[3]{N})$ or $\Theta(\sqrt[4]{N})$]. By selecting an appropriate ISN for transformation to a butterfly network, both the layout area and the number of off-module links can be asymptotically optimal using our layout and partitioning method. An example for optimization under both layout and packaging considerations is given in Section 5.

Another important advantage of our butterfly layouts is that they can be extended to obtain multilayer layouts whose area, volume, and maximum wire length are all significantly reduced when more than two layers of wires are available, as shown in the following section.

4. LAYOUT OF BUTTERFLIES UNDER THE MULTILAYER GRID MODEL

In this section, we introduce the *multilayer grid model* for VLSI layout of networks. We then derive efficient multilayer layouts for butterfly networks.

4.1 The Multilayer Grid Model

In the multilayer grid model, a network is viewed as a graph whose nodes correspond to processing elements and edges correspond to wires. The nodes and edges of the graph are then embedded in a 3-D grid, where edges have unit width, can run along grid lines, but cannot cross or overlap with each other (i.e., the paths for embedding these edges must be edge- and node-disjoint). The area A of a layout is defined as the area of the smallest upright rectangle along the x-y directions that contains all the nodes and wires. The volume of a layout is equal to the number L of layers times its area A.

In the multilayer 2-D grid model, the nodes of the graph are embedded in the 2-D grid of the first layer (i.e., $z = 1$). The range of actual node sizes must be specified explicitly in this model, and is usually taken to be between the minimum size required to implement a node (e.g., a square of side d, $d/4$, or $\frac{d}{4L}$ for a degree-d node in some technologies) and the maximum allowable size without affecting the leading constants for area, volume, and maximum wire length. A network with area A under the Thompson model can be laid out with area no larger than A under the multilayer 2-D grid model with $L = 2$ layers, so the former can be viewed as a special case of the latter. Note, however, that we may derive layouts under the two-layer 2-D grid model with area smaller than the Thompson model. In the multilayer 3-D grid model, the nodes of the graph are embedded in L_A layers of the 3-D grid. These L_A layers are called "active layers" and do not need to be consecutive layers. The range of actual node sizes is also required to be specified explicitly, which is usually between the minimum size required to implement a node (e.g., a cuboid with sides at least $d/h \times d/h \times h$, $1 \le h \le L_A \le L$, for a degree-d node in some technologies) and the maximum allowable size without affecting the leading constants for area, volume, and maximum wire length. The multilayer 2-D grid model is a special case of the multilayer 3-D grid model with $L_A = 1$ active layer. Note that a $d/h \times d/h \times h$ cuboid node requires h active layers for its implementation, while a $d \times d \times 1$ cuboid node requires only 1 active layer. The cost of a layout under the multilayer grid model is a function of A, L, and L_A, as well as other parameters.

We can extend the multilayer grid model to the *multilayer layout model* by allowing nodes and edges to run in other specified directions. Layouts under this model may have smaller area and volume compared with layouts under its multilayer grid model counterpart. Moreover, wires in this model may have different width and cross area, depending on the technology used. For example, wires along the z direction may have larger cross area in PCB. In what follows, we focus on the multilayer 2-D grid model. Layouts under other models will be reported in the near future.

4.2 Multilayer Butterfly Layout

In this subsection we present efficient multilayer layouts for butterfly networks.

We first derive butterfly layouts with an even number L of layers and then extend them to the case of odd L. The multilayer 2-D grid layout of an n-dimensional butterfly network can be obtained from the 2-D grid layout of an ISN$(3, B_{k_1})$ with $k_1 + k_2 + k_3 = n$ by partitioning all the horizontal (or vertical) tracks above each row (or to the right of each column, resp.) of blocks into $L/2$ groups, each of which has at most $\lceil \frac{2^{k_1+k_2+1}}{L} \rceil$ horizontal tracks (or $\lceil \frac{2^{k_1+k_3+1}}{L} \rceil$ vertical tracks, resp.) and is wired using two layers. More precisely, the vertical segments connecting the horizontal tracks of groups i (above each row) and the vertical tracks of groups i (to the right of each column) are wired using layer $2i-1$, and the horizontal tracks of groups i and the horizontal segments connecting the vertical tracks of groups i are wired using layer $2i$, for $i = 1, 2, \ldots, L/2$. In other words, odd-numbered layers are used to wire vertical tracks and segments and even-numbered layers are used to wire horizontal tracks and segments. When a link makes a turn in the 2-D grid layout, its vertical and horizontal segments, wired in neighboring layers $i-1$ and i in the multilayer layout, should be connected by a wire (or via) along the z direction.

When $L = o(\sqrt[3]{N})$ and L is even, the area of the resultant L-layer layout is reduced from $N^2/\log_2^2 N + o(N^2/\log_2^2 N)$ under the Thompson model (with $L = 2$ layers of wires) to

$$2^{k_3}\frac{2^{k_1+k_2+1}}{L} \times 2^{k_2}\frac{2^{k_1+k_3+1}}{L} + o\left(\frac{2^{2k_1+2k_2+2k_3}}{L^2}\right)$$

$$= \frac{4N^2}{L^2 \log_2^2 N} + o\left(\frac{N^2}{L^2 \log^2 N}\right);$$

the maximum wire length of the L-layer layout is reduced from $N/\log_2 N + o\left(N/\log N\right)$ under the Thompson model to

$$2^{k_3}\frac{2^{k_1+k_2+1}}{L} + o\left(\frac{2^{k_1+k_2+k_3}}{L}\right) = \frac{2N}{L \log_2 N} + o\left(\frac{N}{L \log N}\right);$$

and the volume of the L-layer layout is reduced from $2N^2/\log_2^2 N + o(N^2/\log^2 N)$ under the Thompson model (using two layers of wires) to

$$L \times \frac{2^{2k_1+2k_2+2k_3+2}}{L^2} + o\left(\frac{2^{2k_1+2k_2+2k_3}}{L}\right)$$

$$= \frac{4N^2}{L \log_2^2 N} + o\left(\frac{N^2}{L \log^2 N}\right).$$

Since we can arrange all the network nodes as a 2-D grid with comparable numbers of rows and columns, the leading constants of the layout area, maximum wire length, and volume are not affected when each network node occupies a square of side W for any $W = o(\sqrt{N}/(L \log N))$, so the layout is scalable in terms of node size. This can be easily shown when L is not very large.

When L is odd, we simply partition horizontal tracks outside blocks into $(L+1)/2$ groups, wired on layers $1, 3, \ldots, L$, and partition vertical tracks outside blocks into $(L-1)/2$ groups, wired on layers $2, 4, \ldots, L-1$. We can also partition and wire them the other way around. These layouts lead to the following theorem.

THEOREM 4.1. *An N-node butterfly network can be laid out using L layers of wires, $L = 2, 3, ..., o(\sqrt[3]{N})$, and area*

$$\frac{4N^2}{L^2 \log_2^2 N} + o\left(\frac{N^2}{L^2 \log^2 N}\right)$$

when L is even, or area

$$\frac{4N^2}{(L^2-1) \log_2^2 N} + o\left(\frac{N^2}{L^2 \log^2 N}\right)$$

when L is odd.

Note that N network nodes, each occupying an area of $o\left(\frac{N}{L^2 \log^2 N}\right)$, are allowed to occupy $o\left(\frac{N^2}{L^2 \log^2 N}\right)$ area in total, which is, for any layout method, the maximum possible node size allowed without increasing the leading constant of the layout area. Similar to our butterfly layout under the Thompson model, each of a subset of $O(R) = O(N/\log N)$ nodes can occupy an even larger area $o\left(\frac{N}{L^2 \log N}\right)$ without affecting the leading constants. Thus, the scalability of our multilayer butterfly layout is also optimal in terms of node sizes. Note also that if a network node occupies a $W_1 \times W_2$ rectangle, W_1 and W_2 are not within a constant, and $W_1 W_2$ is very close to $\Theta\left(\frac{N}{L^2 \log^2 N}\right)$, then we should align network nodes as a $\Theta(\sqrt{W_2 N/W_1}) \times \Theta(\sqrt{W_1 N/W_2})$ grid to minimize the layout area. These multilayer butterfly layouts are the best results reported in the literature thus far for $L = 3, 4, \ldots, o(\sqrt[3]{N})$ in terms of both area and volume. Note also that the above layout for $L = 2$ has asymptotically the same area as those of the best previous layouts without knock-knees when computed using an upright encompassing rectangle [1, 10, 26], all of which achieve optimal area within a factor of $1 + o(1)$ under the Thompson model. Similarly, we can show that the above layouts achieve optimal area within a factor of about $2 + o(1)$ under the multilayer grid model. Since we have obtained area-efficient L-layer layouts for butterfly networks, $L = 2, 3, \ldots, o(\sqrt[3]{N})$, we can minimize the cost for implementation, which will be a function of area A, the number L of layers, the number $L_A = 1$ of active layers, as well as other parameters.

If a very large number L of layers and $L_A > 1$ active layers are available, we can design butterfly layouts under the multilayer 3-D grid model to further reduce the layout area, maximum wire length, and volume. To obtain multilayer 3-D layouts for a $(k_1 + k_2 + k_3 + k_4)$-dimensional butterfly network, we can use 2^{k_4} copies of a layout similar to a multilayer 2-D layout for a $(k_1 + k_2 + k_3)$-dimensional butterfly (except for an additional nucleus butterfly B_{k_4} in each block), and connect butterfly building blocks B_{k_3} and B_{k_4} vertically in a way similar to a collinear layout of a 2^{k_4}-node complete graph. To minimize the volume of the multilayer 3-D layout, we should select $L = \Theta\left(\frac{\sqrt{N}}{\log N}\right)$. More details concerning multilayer 3-D layouts of butterfly networks and a variety of other networks will be reported in the near future.

5. LAYOUT OF BUTTERFLIES UNDER THE HIERARCHICAL LAYOUT MODEL

In this section, we introduce the *hierarchical layout model*, which takes into account both layout and packaging issues for parallel architectures with multiple packaging levels. We then derive efficient *hierarchical layouts* for butterfly networks.

5.1 The Hierarchical Layout Model

A parallel computer is typically built from several chips on a board, multiple boards in a cabinet, and several such cabinets interconnected together. Modules at each level of the packaging hierarchy have their respective characteristics in terms of the number of pins, maximum area, minimum wire width, maximum allowable wire length, and the number of wires per link [2]. In what follows, we consider the case where several nodes (processors, routers, and associated memory banks) of a network are implemented on a single chip, or more generally, a single module (e.g., chip, board, wafer, or multi-chip module (MCM)), and several modules are used to built the parallel architecture or a higher-level module. Since the number of available off-module pins per node, in addition to layout area, is one of the major constraints limiting the performance and the number of processors that can be put on a module, efficient partition is crucial to obtaining efficient hierarchical layouts.

In an l-level hierarchical layout, a level-i module, $i = 1, 2, ..., l$, consists of level-$(i-1)$ modules as basic components, interconnected by level-i wires, where each level-0 module is a network node and the single level-l module is the resultant l-level hierarchical layout. We can view each level-$(i-1)$ module as a supernode; then the level-i module becomes a multilayer layout with rules presented in the previous section. Therefore, the multilayer layout model is a special case of the hierarchical layout model with a single level of hierarchy. As we have to specify the range of node sizes in the multilayer layout model, we have to specify the maximum area within and the maximum size of a level-i module for all i; as a network node has degree at most d, we have to specify the maximum number of pins (or links) allowed for level-i modules for all i. Moreover, the minimum wire widths may be different at different levels. Note that there may be more than one type of level-i modules allowed for some i. These parameters are important for computing the final cost of the hierarchical layout, which may be determined by a function of the resultant area, volume, and/or the numbers of level-i modules for some/all values of i. Note that if a certain level is area-limited, we may concentrate on layout at that level. Similarly, if area and volume are less important at a certain level, we mainly focus on the packaging constraints and/or the maximum wire length so that partition of networks and/or the arrangement of nodes become more important issues at that level.

5.2 Hierarchical Butterfly Layout: An Example

In this subsection we present a simple example to illustrate how efficient hierarchical layouts can be derived based on the results in previous sections.

Consider a parallel architecture based on a 9-dimensional butterfly built with two levels of hierarchy: the chip and board levels. We assume that the chip level is pin-limited to simplify the problem and a chip has at most 64 off-chip links. We also assume that a chip is a square with side 20 and a level-2 link has unit width collectively. We would like to compute the number of layers of wires on the board and the total area required to put them onto a single board, as well as the total number of chips required.

To obtain an efficient hierarchical layout under these assumptions, we have to first obtain an efficient partition that utilizes (most of) the 64 links in a chip efficiently to minimize the number of chips required. Using the results given in Section 2, we can obtain a partition where each chip conains 80 nodes by placing 8 consecutive rows of the swap-butterfly onto a chip. To minimize the total area required, we need to arrange these chips appropriately and lay out links efficiently on the board. Using the layout method given in Section 3, we arrange the chips as an 8×8 grid, where each chip is a block in the recursive grid layout scheme (see Fig. 3). We use the collinear layout of a 8-node complete graph with quadruple links (see Appendix B) to connect blocks (i.e., chips) belonging to the same row or column, so that neighboring rows or columns are separated by 64 links. Note that we can split approximately half of the wires belonging to the same link to opposite sides of the chip (to which it is incident) so that a block of side at least 16 is sufficient. We can further improve the layout area by moving links connecting neighboring blocks to the tracks connecting those blocks, so that the number of links is reduced by 4 between neighboring rows or columns of chips. If we use two layers of wires, the total area required is 409.6K, which can be considerably reduced if we use more layers of wires and/or more than one active layer for chips. For example, when 4 layers of wires are available, the area is reduced to 160K; when 8 layers of wires are available, the area is reduced to 78.4K.

It can be seen that the saving in total area diminishes in relative importance when the number L of layers becomes larger. The reason is that for small L, the area is still dominated by wires so additional layers are highly beneficial, while when $L = 8$, e.g., the chip and wires both contribute to the final area (in that the space required by wires between neighboring chips is 15, which is somewhat smaller than the side of a chip). Thus, we can see the advantages of aligning network nodes and node clusters as 2-D grids: when the space for network nodes or clusters of network nodes dominates the layout area, which may be true (1) for smaller networks, or (2) for network nodes with large processors and memory banks, or (3) when a sufficient number of layers for wiring are available, or (4) when lower-level module sizes are relatively large, such arrangement may be quite effective (though arrangements such as the one given in [10] may also tolerate node size to a certain degree). Note that even though the area is only reduced by a factor of about 2, our multilayer layout method may still be preferred when $L = 8$ since the maximum wire length is reduced by a factor of about 1.4 compared to $L = 4$, while the maximum wire length may remain the same (for nonsquare layouts) or become somewhat larger (for square layouts) compared to $L = 4$ by folding a 4-layer layout to obtain an 8-layer layout. Moreover, the folding method requires that we have two active layers (containing chips) and if this can be implemented, we should use multilayer 3-D grid layout to further reduce the area, since the area required for chips is considerably reduced when $L_A = 2$. Note also that when the chips are responsible for a nonnegligible portion of total area, aligning them as a 2-D grid as done in our layout is usually the most effective arrangement, while if we arrange chips as a cross or its variant (e.g., [1, 10]) or on two opposite edges of the board a nonnegligible amount of area may be wasted.

The previous example can be easily extended to layouts with more than two levels of packaging hierarchy, with various assumptions on module sizes and pins, using the techniques introduced in previous sections. For more levels of hierarchy, the proposed partitioning and packaging results are particularly important. Note that if a drill hole or via has diameter larger than unit width, the layout area will be increased. We have shown, using minor modifications to the results presented here, that the layouts for butterfly networks and many other networks, such as hypercubes and k-ary n-cubes, have area, volume, and maximum wire length that are asymptotically the same (within a factor of $1+o(1)$) even for very large drill holes and vias. The details will be reported in the near future.

As another comparison, if we partition the network by placing neighboring rows of the butterfly into a chip, we can place at most three rows into a chip, leading to a total of 171 chips, which approximately triples the number of chips required by our partitioning scheme, resulting in larger board area also (or more layers of wires for similarly limited board area). Previous layouts in [1, 10], although having optimal areas using two layers or wires within a single chip, did not consider packaging and multilayer problems and did not arrange the network nodes (or their clusters) as a 2-D grid. Note that when the number of links per chip is increased, the superiority of our partitioning method is even more pronounced, as shown in Section 2.

6. CONCLUSION

In this paper, we introduced the multilayer 2-D grid model, which can significantly reduce the layout area, volume, and maximum wire length when more than two layers of wires are available, and the hierarchical layout model, which is of practical importance, given that parallel architectures are usually implemented using more than one chip. We presented an efficient layout and packaging scheme for butterfly networks under the Thompson model, the multilayer 2-D grid model, and the hierarchical layout model. Compared to previous layouts reported in the literature, our layouts not only have asymptotically optimal area (within factors of $1+o(1)$ and $2+o(1)$, under the first two models for upright rectangle) but also have an asymptotically optimal number of pins (within a small constant factor) when multiple levels of packaging are involved. Our layouts under the multilayer 2-D grid model and the hierarchical layout model are, to the best of our knowledge, the best results reported in the literature thus far. The fact that butterfly networks can be laid out efficiently based on collinear layout of complete graphs is also interesting.

7. REFERENCES

[1] Avior, A., T. Calamoneri, S. Even, A. Litman, and A. Rosenberg, "A tight layout of the butterfly network," *Theory Comput. Sys.,* vol. 31, no. 4, 1998, pp. 475-488.

[2] Basak, D. and D.K. Panda, "Designing clustered multiprocessor systems under packaging and technological advancements," *IEEE Trans. Parallel Distrib. Sys.,* vol. 7, no. 9, Sep. 1996, pp. 962-978.

[3] Bertsekas, D.P. and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods,* Athena Scientific, 1997.

[4] Bhuyan, L.N. and D.P. Agrawal, "Generalized hypercube and hyperbus structures for a computer network," *IEEE Trans. Comput.,* vol. 33, no. 4, Apr. 1984, pp. 323-333.

[5] Brady, M.L. and M. Sarrafzadeh, "Stretching a knock-knee layout for multilayer wiring," *IEEE Trans. Computers,* vol. 39, no. 1, Jan. 1990, pp. 148-151.

[6] Chen, C. and D.P. Agrawal, "dBCube: a new class of hierarchical multiprocessor interconnection networks with area efficient layout," *IEEE Trans. Parallel Distrib. Sys.,* vol. 4, no. 12, Dec. 1993, pp. 1332-1344.

[7] Chen, G. and F.C.M. Lau, "Tighter layouts of the cube-connected cycles," *IEEE Trans. Parallel Distrib. Sys.,* vol. 11, no. 2, Feb. 2000, pp. 182-191.

[8] Cypher, R., "Theoretical aspects of VLSI pin limitations," *SIAM J. Comput.,* vol. 22, no. 2, Apr. 1993, pp. 356-378.

[9] Dally, W.J. and S. Lacy, "VLSI architecture: past, present, and future," *Proc. Advanced Research in VLSI Conf.,* 1999, to appear.

[10] Dinitz, Y., S. Even, R. Kupershtok, and M. Zapolotsky, "Some compact layouts of the butterfly," *Proc. ACM Symp. Parallel Algorithms and Architectures,* Jun. 1999, to appear.

[11] Even, S., S. Muthukrishnan, M.S. Paterson, and S. Cenk Sahinalp, "Layout of the Batcher bitonic sorter," *Proc. ACM Symp. Parallel Algorithms and Architectures,* 1998, pp. 172-181.

[12] Fernández, A. and K. Efe, "Efficient VLSI layouts for homogeneous product networks," *IEEE Trans. Computer,* vol. 46, no. 10, Oct. 1997, pp. 1070-1082.

[13] Franklin, M.A., D.F. Wann, and W.J. Thomas, "Pin limitations and partitioning of VLSI interconnection networks," *IEEE Trans. Computer*, vol. C-31, Nov. 1982, pp. 1109-1116.

[14] Kogge, P.M., "EXECUBE – a new architecture for scalable MPPs," *Proc. Int'l Conf. Parallel Processing,* vol. I, 1994, pp. 77-84.

[15] Leighton, F.T., *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes,* Morgan-Kaufman, San Mateo, CA, 1992.

[16] Muthukrishnan, S., M.S. Paterson, S. Cenk Sahinalp, and T. Suel, "Compact grid layouts of some multi-level networks," *Proc. ACM Symp. Theory of Computing,* 1999, to appear.

[17] Parhami, B., *Introduction to Parallel Processing: Algorithms and Architectures*, Plenum Press, 1999.

[18] Sterling, T., P. Messina, and P. Smith, *Enabling Technologies for Petaflops Computing, MIT Press.*, 1995.

[19] Thompson, C.D., "Area-time complexity for VLSI," *Proc. ACM Symp. Theory of Computing*, 1979, pp. 81-88.

[20] Thompson, C.D., "A complexity theory for VLSI," Ph.D. dissertation, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, PA, 1980.

[21] Wise, D.S., "Compact layouts of banyan/FFT networks," *VLSI Systems and Computations,* Computer Science Press, 1981, pp. 186-195.

[22] Yeh, C.-H. and B. Parhami, "A class of parallel architectures for fast Fourier transform," *Proc. Midwest Symp. Circuits and Systems,* Aug. 1996, pp. 856-859.

[23] Yeh, C.-H. and B. Parhami, "Recursive hierarchical swapped networks: versatile interconnection architectures for highly parallel systems," *Proc. IEEE Symp. Parallel and Distributed Processing*, Oct. 1996, pp. 453-460.

[24] Yeh, C.-H., "Efficient low-degree interconnection networks for parallel processing: topologies, algorithms, VLSI layouts, and fault tolerance," Ph.D. dissertation, Dept. Electrical & Computer Engineering, Univ. of California, Santa Barbara, Mar. 1998.

[25] Yeh, C.-H. and B. Parhami, "VLSI layouts of complete graphs and star graphs," *Information Processing Letters,* Vol. 68, Oct. 1998, pp. 39-45.

[26] Yeh, C.-H., E.A. Varvarigos, and B. Parhami, "Efficient VLSI layouts of hypercubic networks," *Proc. Symp. Frontiers of Massively Parallel Computation*, Feb. 1999, pp. 98-105.

[27] Yeh, C.-H., B. Parhami, and E.A. Varvarigos, "The recursive grid layout scheme for VLSI layout of hierarchical networks," *Proc. Merged Int'l Parallel Processing Symp. & Symp. Parallel and Distributed Processing*, Apr. 1999, pp. 441-445.

[28] Yeh, C.-H. and B. Parhami, "A unified model for hierarchical networks based on an extension of Cayley graphs," *IEEE Trans. Parallel Distrib. Sys.*, to appear.

APPENDIX

A. INDIRECT SWAP NETWORKS

In this appendix we give the definitions and discuss the structures of swap networks and indirect swap networks (ISNs). Details in swap networks can be found in our papers [23, 24, 28]; a special case of ISNs, the indirect version of swap networks, can be found in [22].

A.1 Structure of Swap Networks

An indirect swap network (ISN) (also called unfolded swapped network (USN) [22]) is a multistage network that can be obtained by unfolding the structure of a swap network [23, 24, 28]. In our previous paper [22], we only dealt with a special case of ISNs. In what follows, we present general ISNs so that we can obtain efficient partition for butterfly networks under various assumptions.

We present the definition of swap networks as follows, where the short form $Z_{i:j}$ is used to represent $Z_i Z_{i-1} \cdots Z_{j+1} Z_j$ for any symbol Z. An l-level swap network based on a k_1-cube, denoted by $\mathrm{SN}(l, Q_{k_1})$, begins with a nucleus k_1-cube Q_{k_1}, which forms an $\mathrm{SN}(1, Q_{k_1})$. To build a 2-level swap network, $\mathrm{SN}(2, Q_{k_1})$, we use 2^{k_2} identical copies of the nucleus k_1-cube, each having 2^{k_1} nodes, where $k_2 \le k_1$. Each nucleus is viewed as a level-2 cluster, and is given a k_2-bit string

$$X_2 = x_{k_2+k_1-1:k_1} = x_{k_2+k_1-1} x_{k_2+k_1-2} \cdots x_{k_1+1} x_{k_1}$$

as its address. We also assign each node a k_1-bit string

$$X_1 = x_{k_1-1:0} = x_{k_1-1} x_{k_1-2} \cdots x_1 x_0$$

as its address within the nucleus k_1-cube to which it belongs. Node X_1 within nucleus X_2 has an n_2-bit string

$$X_2' = X_2 X_1 = x_{k_2+k_1-1:0} = x_{n_2-1:0} = x_{n_2-1} x_{n_2-2} \cdots x_1 x_0$$

as its address within the $\mathrm{SN}(2, Q_{k_1})$, where $n_2 = k_2 + k_1$. Each of the 2^{k_2} nucleus copies has a link connecting it to each of the other $2^{k_2} - 1$ nuclei, via which node $X_2' = X_2 X_1$ is connected to node

$$x_{k_2-1:0} x_{k_1-1:k_2} X_2 = x_{k_2-1:0} x_{k_1-1:k_2} x_{k_2+k_1-1:k_1}$$

$$= x_{k_2-1} x_{k_2-2} \cdots x_1 x_0 x_{k_1-1} x_{k_1-2} \cdots x_{k_2}$$

$$x_{k_2+k_1-1} x_{k_2+k_1-2} \cdots x_{k_1}.$$

In other words, the new neighbor of a node, called the level-2 neighbor, is obtained by swapping the first and last k_2 bits of the address of that node.

To build an l-level swap network, $\mathrm{SN}(l, Q_{k_1})$, we use 2^{k_l} identical copies of $\mathrm{SN}(l-1, Q_{k_1})$, where $k_l \le n_{l-1} = \sum_{i=1}^{l-1} k_i$. Each copy of the $\mathrm{SN}(l-1, Q_{k_1})$ is viewed as a level-l cluster, and is given a k_l-bit string

$$X_l = x_{n_l-1:n_{l-1}} = x_{n_l-1} x_{n_l-2} \cdots x_{n_{l-1}+1} x_{n_{l-1}}$$

as its address, where $n_l = k_l + n_{l-1} = \sum_{i=1}^{l} k_i$. Each node has already been given an n_{l-1}-bit string

$$X_{l-1}' = X_{l-1:1} = x_{n_{l-1}-1:0} = x_{n_{l-1}-1} x_{n_{l-1}-2} \cdots x_1 x_0$$

as its address within the level-l cluster to which it belongs. Node X_{l-1}' within the level-l cluster X_l has an n_l-bit string

$$X_l' = X_l X_{l-1}' = X_{l:1} = x_{n_l-1:0} = x_{n_l-1} x_{n_l-2} \cdots x_1 x_0$$

as its address within the $\mathrm{SN}(l, Q_{k_1})$. Each of the level-l clusters has 2^{n_l-1} nodes and 2^{n_l-2} links connecting it to each of the other $2^{k_l} - 1$ level-l clusters, via which node $X_l' = X_l X_{l-1:1}$ connects to node

$$x_{k_l-1:0} x_{n_{l-1}-1:k_l} X_l = x_{k_l-1:0} x_{n_{l-1}-1:k_l} x_{n_l-1:n_{l-1}}.$$

In other words, the new neighbor, called level-l neighbor, is obtained by swapping the first and last k_l bits of the address of a node.

In summary, the address of a node in an l-level swap network based on a hypercube can be partitioned into l groups of bits, the i^{th} of which (from the right) is composed of k_i bits, $1 \le i \le l$. Two nodes in an $\mathrm{SN}(l, Q_{k_1})$ are connected if and only if

- (a) their addresses differ in exactly one bit in the first group (we say they are connected by a *dimension-i nucleus link* if the addresses differ in the i^{th} least significant bit) , or
- (b) one node's address can be obtained from that of the other by swapping the i^{th} group, $i \in [2, l]$, with the rightmost k_i bits (we say they are connected by a *level-i inter-cluster link*).

Hierarchical swap networks (HSNs) are a special case of swap networks where $k_i = k_j$ for all $i, j \in [1, l]$ [23, 24, 28].

A.2 Structure of Indirect Swap Networks

In this subsection, we first map FFT onto swap networks and then use its flow graph to define indirect swap networks. FFT can be performed in swap networks using the following recursive algorithm:

- Step 1: Perform FFT in each level-l cluster.
- Step 2: Send the data item located at each node $X = X_l x_{n_{l-1}-1:k_l} x_{k_l-1:0}$ to node $x_{k_l-1:0} x_{n_{l-1}-1:k_l} X_l$ via its level-l inter-cluster link.
- Step 3: Perform FFT in each level-l cluster using the same routing paths as those used when performing FFT for the first k_l dimensions.

An FFT algorithm for the nucleus hypercube Q_{k_1} and the steps forwarding packets along inter-cluster links form the building blocks for the preceding recursive algorithm. In what follows, we represent an FFT algorithm for hierarchical swap networks $\mathrm{HSN}(l, Q_{k_1})$ in a bottom-up manner using such building blocks:

- Step 1: Perform FFT in each of the nucleus hypercubes.
- Step 2: For $i = 2$ to l,
  - Step 2.1: Send the data item at each node $X = X_{l:i+1} X_i X_{i-1:2} X_1$ to node $X_{l:i+1} X_1 X_{i-1:2} X_i$ via its level-i inter-cluster link.
  - Step 2.2: Perform FFT in each of the nucleus hypercubes.

An indirect swap network (ISN) is a multistage network obtained by the flow graph for performing FFT on a corresponding swap network [23, 24, 28], similar to the way a butterfly network is derived from a hypercube. Let m be the maximum number of steps for the preceding FFT algorithm in a swap network $\mathrm{SN}(l, Q_{k_1})$. Then an ISN derived from $\mathrm{SN}(l, Q_{k_1})$ has $m + 1$ stages, each having $R = 2^{n_l}$ nodes, where $R = 2^{n_l}$ is the size of the $\mathrm{SN}(l, Q_{k_1})$. A node in the ISN can be represented by a pair (x, y) as its address, where $x \in [0, R-1]$ is an n_l-bit binary row number and $y \in [0, m]$ denotes the stage number. If in step i, $i \in [1, m-1]$, of the FFT algorithm of the SN, we need to forward the data items through nucleus links, then a node $(u, i-1)$ in the ISN derived from that SN has a cross link connecting it to node (v, i), where u and v differ only in the i^{th} bit, and a straight link connecting it to node (u, i); if in step i, $i \in [1, m-1]$, of the FFT algorithm of the SN, we must forward the data items through level-j inter-cluster links (i.e., Step 2 of the recursive FFT algorithm), then a node $(u, i-1)$ in the ISN derived from that SN has a *level-i swap link* connecting it to node (v, i), where u and v can be obtained from each other by swapping the i^{th} group of bits with the rightmost k_i bits (i.e., u and v are connected by a level-i inter-cluster link in the corresponding $\mathrm{SN}(l, Q_{k_1})$). See Fig. 1 for an example of a 4×4 ISN. Figure 2 shows the automorphisms of swap-butterflies (see Subsection 2.2); by combining the duplicated links between each pair of neighboring nodes (including those without numbers), we obtain the automorphisms of an 8×8 ISN and a 16×16 ISN in Figs. 2a and 2b, respectively.

The basic building modules for an ISN derived from an $\mathrm{SN}(l, Q_{k_1})$ are butterfly networks of dimensions no larger than k_1. Basic modules of neighboring stages are connected through swap links using rules similar to the inter-cluster links of the SN it is derived from. ISNs can also be derived by unfolding the structure of swap networks [23, 24, 28] along routing paths, similar to the way butterfly networks are derived from hypercubes. The details are omitted here.

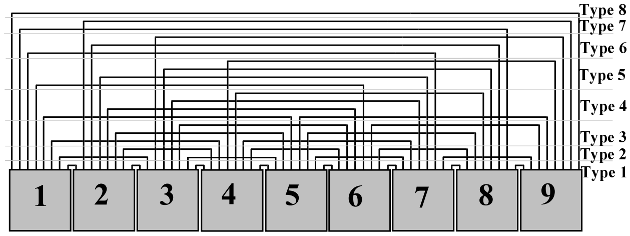

Figure 4: A collinear layout for the 9-node complete graph K_9.

B. STRICTLY OPTIMAL COLLINEAR LAYOUT FOR COMPLETE GRAPHS

In [6], a layout that requires $4(4^{\log_2 N-1} - 1)/3 \approx N^2/3$ tracks is presented for mapping an N-node complete graph, K_N, onto a linear array. In what follows, we show that such a mapping, called a *collinear layout*, can be considerably improved to one that uses $\lfloor N^2/4 \rfloor$ tracks, which exactly matches the bisection-based lower bound.

To obtain the collinear layout, we first place the N nodes, labeled 1 through N, along a row. Let a link be *type-i* if it connects two nodes whose addresses differ by i. Then the $N(N-1)/2$ links in K_N can be classified into types $1, 2, 3, ..., N-1$, and there are $N-i$ type-i links. To lay out K_N, we place all the type-1 links in one track, all the type-2 links in two tracks, where links connecting nodes with odd addresses are put in one track and links connecting nodes with even addresses in the other, and then all the type-i links in $\min(i, N-i)$ tracks for $i = 3, 4, 5, \ldots, N-1$. More precisely, type-i links are placed in i tracks if $i \leq N/2$, where links are put in the same track if the remainder of the node-address modulo i is the same, and each of the $N-i$ type-i links is placed in a different track if $i > N/2$, Clearly, such an arrangement will not result in overlapped links within a track. The resulting layout for a K_9 is illustrated in Fig. 4. Note that we can reverse the order of horizontal tracks so that the maximum wire length is reduced.

The total number of tracks in the layout described above is equal to

$$\sum_{i=1}^{N-1} \min(i, N-i) = \sum_{i=1}^{\lfloor N/2 \rfloor} i + \sum_{i=\lfloor N/2 \rfloor+1}^{N-1} (N-i)$$

$$= \sum_{i=1}^{\lfloor N/2 \rfloor} i + \sum_{i=1}^{\lceil N/2 \rceil - 1} i = \lfloor N^2/4 \rfloor.$$

Since the bisection width of K_N is equal to $N^2/4$ when N is even and $(N^2-1)/4$ when N is odd, this layout is strictly optimal in terms of the number of tracks for collinear layouts of complete graphs. Therefore, the number of tracks required for the minimal collinear layout of an N-node complete graph is $\lfloor N^2/4 \rfloor$. This upper bound is 25% smaller than the one given in [6, Theorem 1] and leads to optimal layouts for butterfly networks under various layout models.