1 OTIS OPTOELECTRONIC COMPUTERS

Chih-fang Wang and Sartaj Sahni*

Department of Computer and Information
Science and Engineering
University of Florida
{wang,sahni}@cise.ufl.edu

Abstract: OTIS computers employ the optical transpose interconnect system for inter package communications and a regular electronic interconnect system (*e.g.*, mesh, hypercube, mesh of trees) for intra package communications. In this chapter, we explore the characteristics of OTIS computers and summarize some of the algorithmic developments that have been made.

INTRODUCTION

It is well known that when communication distances exceed a few millimeters, optical interconnects provide speed (bandwidth) and power advantages over electronic interconnects [1, 5]. Therefore, in the construction of very large multiprocessor computers it is prudent to interconnect physically close processors using electronic interconnect and to use optical interconnect for pairs of processors that are distant. We shall assume that physically close processors are in the same physical package (chip, wafer, board) and processors that are not physically close are in different packages. As a result, electronic interconnects are used for intra package communications while optical interconnect is used for inter package communication.

Various combinations of interconnection networks for intra package (*i.e.*, electronic) communications and inter package (*i.e.*, optical communications

*This work was supported, in part, by the Army Research Office under grant DAA H04-95-1-0111.

) have been proposed. In OTIS computers [3, 6, 13], optical interconnects are realized via a free space optical interconnect system known as the optical transpose interconnection system (OTIS).

In this chapter, we begin by describing the OTIS. Next, we describe the OTIS-Mesh and OTIS-Hypercube parallel computers that result, respectively, when the OTIS optical interconnect system is used for inter package communication and a mesh or hypercube is used for intra package communication. Properties of both the OTIS-Mesh and OTIS-Hypercube computer are also developed. We conclude by summarizing the complexities of OTIS-Mesh and OTIS-Hypercube algorithms for data routing as well as of basic operations and matrix multiplication on OTIS-Mesh computers. In this chapter we are concerned with deterministic algorithms alone. Randomized algorithms for the OTIS-Mesh can be found in [8].

OPTICAL TRANSPOSE INTERCONNECTION SYSTEM (OTIS)

The optical transpose interconnection system (OTIS) was proposed by Marsden *et al.* [6] The OTIS connects $L = MN$ inputs to L outputs using free space optics and two arrays of lenslets. The first lenslet array is a $\sqrt{M} \times \sqrt{M}$ array and the second one is of dimension $\sqrt{N} \times \sqrt{N}$. Thus, a total of $M + N$ lenslets are used. The L inputs and outputs are arranged to form a $\sqrt{L} \times \sqrt{L}$ array. The L inputs are arranged into $\sqrt{M} \times \sqrt{M}$ groups with each group containing N inputs arranged into a $\sqrt{N} \times \sqrt{N}$ array. Figure 1.1 shows the arrangement of the $L = 64$ inputs when $M = 4$ and $N = 16$. The $M \times N$ inputs are indexed (i, j) with $0 \leq i < M$ and $0 \leq j < N$. Inputs with the same i value are in the same $\sqrt{N} \times \sqrt{N}$ block. The notation $(1, *)$, for example, refers to all inputs of the form $(1, j)$.

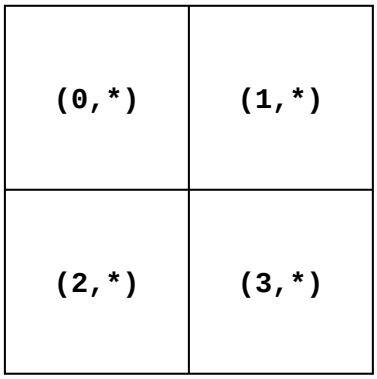

Figure 1.1(a) $\sqrt{M} \times \sqrt{M} = 2 \times 2$ grouping of inputs

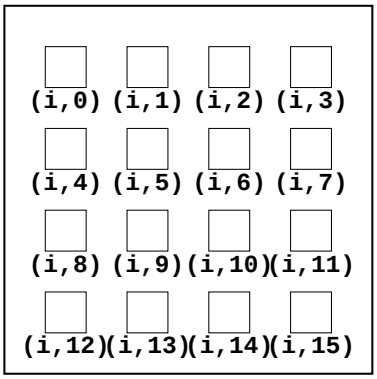

Figure 1.1(b) The $(i, *)$ group, $0 \leq i < M = 4$

Figure 1.1 2-dimensional arrangement of $L = 64$ inputs when $M = 4$ and $N = 16$

In addition to using the two-dimensional notation (i, j) to refer to an input, we also use a four-dimensional notation (i_r, i_c, j_r, j_c) where (i_r, i_c) gives the coordinates (row,column) of the $\sqrt{N} \times \sqrt{N}$ block that contains the input (see Figure 1.1(a)) and (j_r, j_c) gives coordinates of the element within a block (see Figure 1.1(b)). So all elements $(i, *)$ with $i = 0$ have $(i_r, i_c) = (0, 0)$; those with $i = 1$ have $(i_r, i_c) = (0, 1)$; those with $i = 2$ have $(i_r, i_c) = (1, 0)$; and those with $i = 3$ have $(i_r, i_c) = (1, 1)$. Similarly, all inputs with $j = 3$ have $(j_r, j_c) = (0, 3)$, and those with $j = 12$ have $(j_r, j_c) = (3, 0)$.

The L outputs are also arranged into a $\sqrt{L} \times \sqrt{L}$ array. This time, however, the $\sqrt{L} \times \sqrt{L}$ array is composed of $\sqrt{N} \times \sqrt{N}$ blocks with each block containing M outputs that are arranged as a $\sqrt{M} \times \sqrt{M}$ array. The $L = MN$ outputs are indexed (i, j) with $0 \leq i < N$, $0 \leq j < M$. All outputs of the form $(i, *)$ are in the same block, block i. Block i is in position (i_r, i_c) with $i = i_r\sqrt{N} + i_c$ of the $\sqrt{N} \times \sqrt{N}$ block arrangement. Outputs of the form $(*, j)$ are in position (j_r, j_c) of their block, $j = j_r\sqrt{M} + j_c$.

In the physical realization of OTIS, the $\sqrt{L} \times \sqrt{L}$ output arrangement is rotated 180°. We have 4 two-dimensional planes; the first is the $\sqrt{L} \times \sqrt{L}$ input plane; the second is a $\sqrt{M} \times \sqrt{M}$ lenslet plane, the third is a $\sqrt{N} \times \sqrt{N}$ lenslet plane, and the fourth is the $\sqrt{L} \times \sqrt{L}$ plane of outputs rotated 180°. When the OTIS is viewed from the side, only the first column of each of these planes is visible. Such a side view for the case $L = M \times N = 4 \times 16$ is shown in Figure 1.2. Notice that the first column of the input plane consists of the inputs (0,0), (0,4), (0,8), (0,12), (2,0), (2,4), (2,8), (2,12) which in 4D notation are (0,0,0,0), (0,0,1,0), (0,0,2,0), (0,0,3,0), (1,0,0,0), (1,0,1,0), (1,0,2,0), (1,0,3,0). The inputs in the same row as (0,0,0,0) are $(0, *, 0, *)$, those in the same row as (i_r, i_c, j_r, j_c) are $(i_r, *, j_r, *)$. The (i_r, j_r) values top to bottom are (0,0), (0,1), (0,2), (0,3), (1,0), (1,1), (1,2), (1,3). The first column in the output plane (after the 180° rotation) has the outputs (15,3), (15,1), (11,3), (11,1), (7,3), (7,1), (3,3), (3,1) which in 4D notation are (3,3,1,1), (3,3,0,1), (2,3,1,1), (2,3,0,1), (1,3,1,1), (1,3,0,1), (0,3,1,1), (0,3,0,1). The outputs in the same row as (3,3,1,1) are $(3, *, 1, *)$, those in the same row as (i_r, i_c, j_r, j_c) are $(i_r, *, j_r, *)$. The (i_r, j_r) values top to bottom are (3,1), (3,0), (2,1), (2,0), (1,1), (1,0), (0,1), (0,0).

Each lens of Figure 1.2 denotes a row of lenslets and each ◯ a row of inputs or outputs. The interconnection pattern defined by the given arrangement of inputs, outputs, and lenslets connects input $(i, j) = (i_r, i_c, j_r, j_c)$ to output $(j, i) = (j_r, j_c, i_r, i_c)$. The connection is established via an optical ray that originates at input position (i_r, i_c, j_r, j_c), goes through lenslet (i_r, i_c) of the first lenslet array, then through lenslet (j_r, j_c) of the second lenslet array, and finally arrives at output position (j_r, j_c, i_r, i_c).

The basic connectivity provided by the OTIS is an optical connection between input (i, j) and output (j, i), $0 \leq i < M$, $0 \leq j < N$.

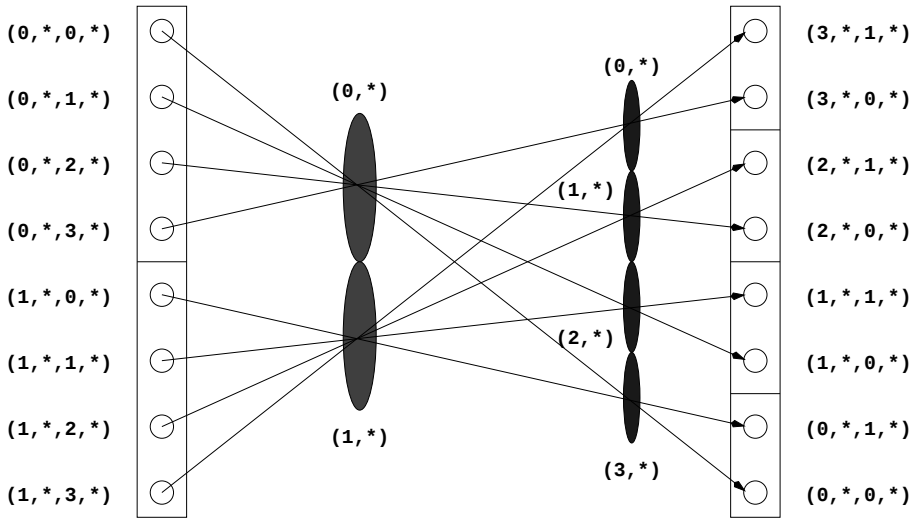

Figure 1.2 Side view of the OTIS with $M = 4$ and $N = 16$

OTIS PARALLEL COMPUTERS

Marsden *et al.* [6] have proposed several parallel computer architectures in which OTIS is used to connect processors in different groups (packages) and an electronic interconnection network is used to connect processors in the same group. Since Krishnamoorthy *et al.* [5] have shown that bandwidth is maximized and power consumption minimized when an $L = N^2$ processor OTIS computer is partitioned into N groups of N processors each, Zane *et al.* [13] limit the study of OTIS parallel computers so that each processor group (package) has N processors and the parallel computer has a total of N groups (packages). Let (i, j) denote processor j of package i, $0 \leq i < N$, $0 \leq j < N$. Processor (i, j), $i \neq j$, is connected to processor (j, i) using free space optics (*i.e.*, OTIS). The only other connections available in an OTIS computer are the electronic intra group connections.

A generic 16 processor OTIS computer is shown in Figure 1.3. The solid boxes denote processors. Each processor is labeled (g, p) where g is the group index and p is the processor index. OTIS connections are shown by arrows. Intra group connections are not shown.

In an OTIS-Mesh, processors in the same group are connected as a 2-D mesh [6, 13, 9]; and in an OTIS-Hypercube, processors in the same group are connected using the hypercube topology [6, 13, 10]. OTIS-Mesh of trees [6], OTIS-Perfect shuffle, OTIS-Cube connected cycles, etc., may be defined in an analogous manner.

When analyzing algorithms for OTIS architectures, we count data moves along electronic interconnects (*i.e.*, electronic moves) and those along optical

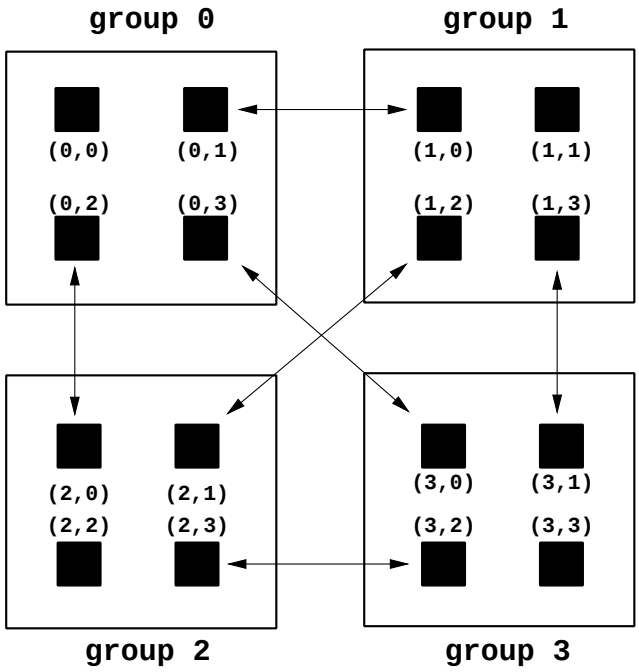

Figure 1.3 Example of OTIS connections with 16 processors

interconnects (*i.e.*, OTIS moves) separately. This allows us to later account for any differences in the speed and bandwidth of these two types of interconnect.

OTIS-MESH

In an N^2 processor OTIS-Mesh, each group is a $\sqrt{N} \times \sqrt{N}$ mesh and there are a total of N groups. Figure 1.4 shows a 16 processor OTIS-Mesh. The processors of groups 0 and 2 are labeled using two dimensional local mesh coordinates while the processors in groups 1 and 3 are labeled in row-major fashion. We use the notation (g, p) to refer to processor p of group g.

Diameter of the OTIS-Mesh

Let (g_1, p_1) and (g_2, p_2) be two OTIS-Mesh processors. The shortest path between these two processors is of one of the form:

(a) The path involves only electronic moves. This is possible only when $g_1 = g_2$.

(b) The path involves an even number of optical moves. In this case the path is of the form $(g_1, p_1) \xrightarrow{\text{E}^*} (g_1, p_1') \xrightarrow{\text{O}} (p_1', g_1) \xrightarrow{\text{E}^*} (p_1', g_1') \xrightarrow{\text{O}} (g_1', p_1') \xrightarrow{\text{E}^*} (g_1', p_1'') \xrightarrow{\text{O}} (p_1'', g_1') \xrightarrow{\text{E}^*} (p_1'', g_1'') \xrightarrow{\text{O}} \cdots \xrightarrow{\text{E}^*} (g_2, p_2)$.

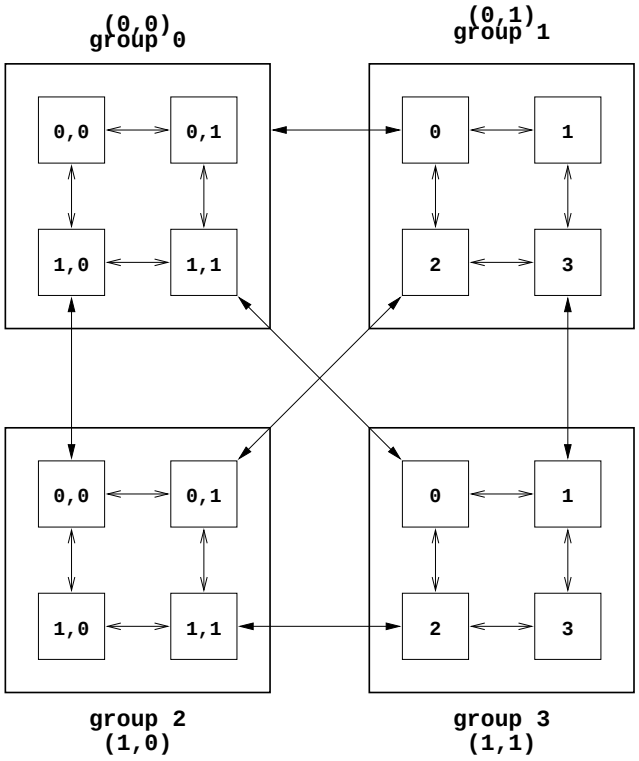

Figure 1.4 16 Processor OTIS-Mesh

Here E* denotes a sequence (possibly empty) of electronic moves and O denotes a single OTIS move. If the number of OTIS moves is more than two, we may compress paths of this form into the shorter path $(g_1,p_1) \xrightarrow{E^*} (g_1,p_2) \xrightarrow{O} (p_2,g_1) \xrightarrow{E^*} (p_2,g_2) \xrightarrow{O} (g_2,p_2)$. So we may assume that the path is of the above form with exactly two OTIS moves.

(c) The path involves an odd number of OTIS moves. In this case, it must involve exactly one OTIS move (as otherwise it may be compressed into a shorter path with just one OTIS move as in (b)) and may be assumed to be of the form $(g_1,p_1) \xrightarrow{E^*} (g_1,g_2) \xrightarrow{O} (g_2,g_1) \xrightarrow{E^*} (g_2,p_2)$.

Let $d(i,j)$ be the shortest distance between processors i and j of a group using a path comprised solely of electronic moves. So, $d(i,j)$ is the Manhattan distance between the two processors of the local mesh group. Shortest paths of type (a) have length $d(p_1,p_2)$ while those of types (b) and (c) have length $d(p_1,p_2)+d(g_1,g_2)+2$ and $d(p_1,g_2)+d(p_2,g_1)+1$, respectively.

From the preceding discussion we have the following theorems:

Theorem 1 *[9] The length of the shortest path between processors* (g_1, p_1) *and* (g_2, p_2) *is* $d(p_1, p_2)$ *when* $g_1 = g_2$ *and* $\min\{d(p_1, p_2) + d(g_1, g_2) + 2, d(p_1, g_2) + d(p_2, g_1) + 1\}$ *when* $g_1 \neq g_2$.

Proof When $g_1 = g_2$, there are three possibilities for the shortest path. It may be of types (a), (b), or (c). If it is of type (a), its length is $d(p_1, p_2)$. If it is of type (b), its length is $d(p_1, p_2)+d(g_1, g_2)+2 = d(p_1, p_2)+2$. If it is of type (c), its length is $d(p_1, g_2)+d(p_2, g_1)+1 = d(p_1, g_1)+d(p_2, g_1)+1 = d(p_1, g_1)+d(g_1, p_2)+1 \geq d(p_1, p_2)+1$. So, the shortest path has length $d(p_1, p_2)$. When $g_1 \neq g_2$, the shortest path is either of type (b) or (c). From our earlier development it follows that its length is $\min\{d(P_1, P_2) + d(G_1, G_2) + 2, d(P_1, G_2) + d(P_2, G_1) + 1\}$. □

Theorem 2 *[9] The diameter of the OTIS-Mesh is* $4\sqrt{N} - 3$.

Proof Since each group is a $\sqrt{N} \times \sqrt{N}$ mesh, $d(p_1, p_2)$, $d(p_2, g_1)$, $d(p_1, g_2)$, and $d(g_1, g_2)$ are all less than or equal to $2(\sqrt{N}-1)$. From Theorem 1, it follows that no two processors are more than $4(\sqrt{N} - 1) + 1 = 4\sqrt{N} - 3$ apart. Hence, the diameter is $\leq 4\sqrt{N} - 3$. Now consider the processors (g_1, p_1), (g_2, p_2) such that p_1 is in position $(0, 0)$ of its group and p_2 is in position $(\sqrt{N}-1, \sqrt{N}-1)$ (*i.e.*, p_1 the top left processor and p_2 the bottom right one of its group). Let g_1 be 0 and g_2 be $N-1$. So, $d(p_1, p_2) = d(g_1, g_2) = d(p_1, g_2) = d(p_2, g_1) = \sqrt{N}-1$. Hence, the distance between (g_1, p_1) and (g_2, p_2) is $4\sqrt{N} - 3$. As a result, the diameter of the OTIS-Mesh is exactly $4\sqrt{N} - 3$. □

Simulation of a 4D Mesh

Zane *et al.* [13] have shown that the OTIS-Mesh can simulate each move of a $\sqrt{N} \times \sqrt{N} \times \sqrt{N} \times \sqrt{N}$ four-dimensional mesh by using either a single electronic move local to a group or using one local electronic move and two inter group OTIS moves. For the simulation, we must first embed the 4D mesh into the OTIS-Mesh. The embedding is rather straightforward with processor (i, j, k, l) of the 4D mesh being identified with processor (g, p) of the OTIS-Mesh. Here, $g = i\sqrt{N} + j$ and $p = k\sqrt{N} + l$.

The mesh moves $(i, j, k \pm 1, l)$ and $(i, j, k, l \pm 1)$ can be performed with one electronic move of the OTIS-Mesh while the moves $(i, j \pm 1, k, l)$ and $(i \pm 1, j, k, l)$ require one electronic and two optical moves. For example, the move $(i, j + 1, k, l)$ may be done by the sequence $(i, j, k, l) \xrightarrow{O} (k, l, i, j) \xrightarrow{E} (k, l, i, j + 1) \xrightarrow{O} (i, j + 1, k, l)$.

The above efficient embedding of a 4D mesh implies that 4D mesh algorithms can be run on the OTIS-Mesh with a constant factor (at most 3) slowdown [13]. Unfortunately, the body of known 4D mesh algorithms is very small compared to that of 2D mesh algorithms. So, it is desirable to consider a 2D mesh embedding. Such an embedding will enable one to run 2D mesh algorithms on the OTIS-Mesh. Naturally, one would do this only for problems for which no 4D algorithm is known or for which the known 4D mesh algorithms are not faster than the 2D algorithms.

Simulation of a 2D Mesh

There are at least two intuitively appealing ways to embed an $N \times N$ mesh into the OTIS-Mesh. One is the *group row mapping* (GRM) in which each group of the OTIS-Mesh represents a row of the 2D mesh. The mapping of the mesh row onto a group of OTIS processors is done in a snake-like fashion as in Figure 1.5(a). The pair of numbers in each processor of Figure 1.5(a) gives the (row,column) index of the mapped 2D mesh processor. The thick edges show the electronic connections used to obtain the 2D mesh row. Notice that the assignment of rows to groups is also done in a snake-like manner. Let (i, j) denote a processor of a 2D mesh. The move to $(i, j+1)$ (or $(i, j-1)$) can be done with one electronic move as (i, j) and $(i, j+1)$ are neighbors in a processor group. If all elements of row i are to be moved over one column, then the OTIS-Mesh would need one electronic move in case of a MIMD mesh and 3 in case of a SIMD mesh as the row move would involve a shift by one left, right, and down within a group. A column shift can be done with 2 additional OTIS moves as in the case of a 4D mesh embedding. GRM is particularly nice for the matrix transpose operation. Data from processor (i, j) can be moved to processor (j, i) with one OTIS and zero electronic moves.

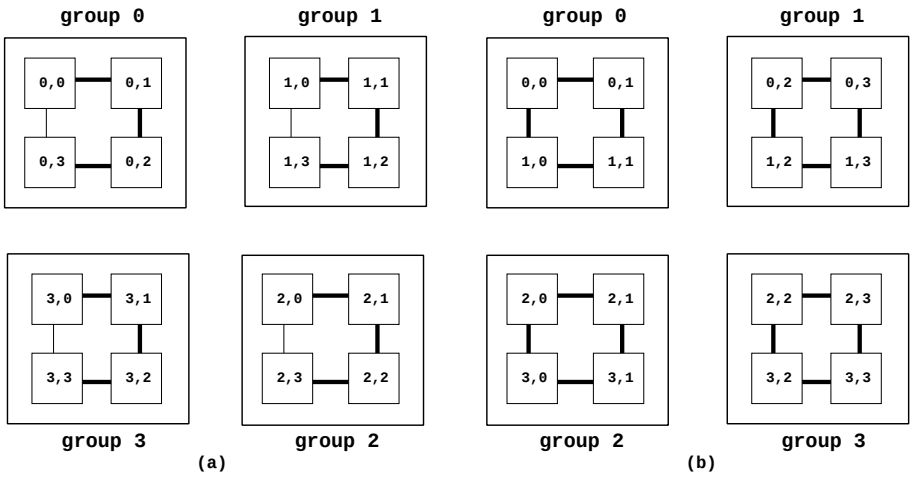

Figure 1.5 Mapping a 4×4 mesh onto a 16 processor OTIS-Mesh: (a) GRM; (b) GSM

The second way to embed an $N \times N$ mesh is to use the *group submesh mapping* (GSM). In this, the $N \times N$ mesh is partitioned into N $\sqrt{N} \times \sqrt{N}$ submeshes. Each of these is mapped in the natural way onto a group of OTIS-Mesh processors. Figure 1.5(b) shows GSM of a 4×4 mesh. Moving all elements of row or column i over by one is now considerably more expensive. For example, a row shift by $+1$ would be accomplished by the following data movements (a boundary processor is one on the right boundary of a group):

Step 1: Shift data in non-boundary processors right by one using an electronic move.

Step 2: Perform an OTIS move on boundary processor data. So, data from (g,p) moves to (p,g).

Step 3: Shift the data moved in Step 2 right by one using an electronic move. Now, data from (g,p) is in $(p,g+1)$.

Step 4: Perform an OTIS move on this data. Now data originally in (g,p) is in $(g+1,p)$.

Step 5: Shift this data left by $\sqrt{N}-1$ using $\sqrt{N}-1$ electronic moves. Now, the boundary data originally in (g,p) is in the processor to its right but in the next group.

The above five step process takes $\sqrt{N}$ electronic and two OTIS moves. Note, however, that if each group is a wraparound mesh in which the last processor of each row connects to the first and the bottom processor of each column connects to the top one, then row and column shift operations become much simpler as Step 1 may be eliminated and Step 5 replaced by a right wraparound shift of 1. The complexity is now two electronic and two OTIS moves.

GSM is also inferior on the transpose operation which now requires $8(\sqrt{N}-1)$ electronic and 2 OTIS moves.

Theorem 3 *[9] The transpose operation of an $N \times N$ mesh requires $8(\sqrt{N}-1)$ electronic and 2 OTIS moves when the GSM is used.*

Proof Let g_xg_y and p_xp_y denote processor (g,p) of the OTIS-Mesh. This processor is in position (p_x,p_y) of group (g_x,g_y) and corresponds to processor (g_xp_x,g_yp_y) of the $N \times N$ embedded mesh. To accomplish the transpose, data is to be moved from the $N \times N$ mesh processor (g_xp_x,g_yp_y) (*i.e.*, the OTIS-Mesh processor $(g,p)=(g_xg_y,p_xp_y)$) to the mesh processor (g_yp_y,g_xp_x) (*i.e.*, the OTIS-Mesh processor (g_yg_x,p_yp_x)). The following movements do this: $(g_xp_x,g_yp_y) \xrightarrow{\text{E}^*} (g_xp_y,g_yp_x) \xrightarrow{\text{O}} (p_yg_x,p_xg_y) \xrightarrow{\text{E}^*} (p_yg_y,p_xg_x) \xrightarrow{\text{O}} (g_yp_y,g_xp_x)$. Once again E* denotes a sequence of electronic moves local to a group and O denotes a single OTIS move. The E moves in this case perform a transpose in a $\sqrt{N} \times \sqrt{N}$ mesh. Each of these transposes can be done in $4(\sqrt{N}-1)$ moves [7]. So, the above transpose method uses $8(\sqrt{N}-1)$ electronic and 2 OTIS moves.

To see that this is optimal, first note that every transpose algorithm requires at least 2 OTIS moves. For this, pick a group g_xg_y such that $g_x \neq g_y$. Data from all N processors in this group are to move to the processors in group g_yg_x. This requires at least one OTIS move. However, if only one OTIS move is performed, data from g_xg_y is scattered to the N groups. So, at least two OTIS moves are needed if the data ends up in the same group.

Next, we shall show that independent of the OTIS moves, at least $8(\sqrt{N}-1)$ electronic moves must be performed. The electronic moves cumulatively

perform one of the following two transforms (depending on whether the number of OTIS moves is even or odd, see previous section about the diameter):

(a) local moves from (p_x, p_y) to (p_y, p_x); local moves from (g_x, g_y) to (g_y, g_x);

(b) local moves from (p_x, p_y) to (g_y, g_x); local moves from (g_x, g_y) to (p_y, p_x).

For $(p_x, p_y) = (g_x, g_y) = (0, \sqrt{N} - 1)$, (a) and (b) require $2(\sqrt{N} - 1)$ left and $2(\sqrt{N} - 1)$ down moves. For $(p_x, p_y) = (g_x, g_y) = (\sqrt{N} - 1, 0)$, (a) and (b) require $2(\sqrt{N} - 1)$ right and $2(\sqrt{N} - 1)$ up moves. The total number of moves is thus $8(\sqrt{N} - 1)$. So, $8(\sqrt{N} - 1)$ is a lower bound on the number of electronic moves needed. □

OTIS-HYPERCUBE

In an N^2 processor OTIS-Hypercube, each group is a hypercube of dimension $\log_2 N$. Figure 1.6 shows a 16 processor OTIS-Hypercube. The number inside a processor is the processor index within its group.

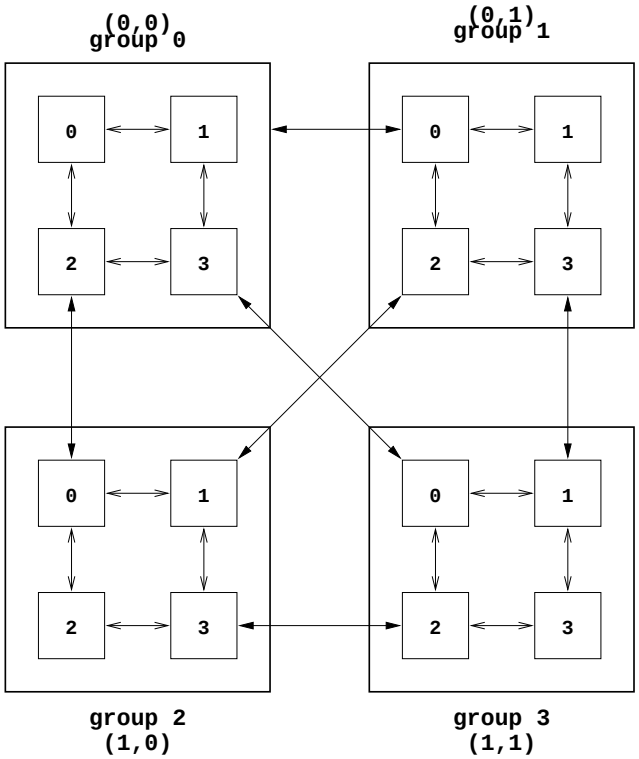

Figure 1.6 16 processor OTIS-Hypercube

Diameter of the OTIS-Hypercube

Let $N = 2^d$ and let $D(i,j)$ be the length of the shortest path from processor i to processor j in a hypercube. Let (g_1,p_1) and (g_2,p_2) be two OTIS-Hypercube processors. Similar to the discussion of the diameter of OTIS-Mesh in the previous section, The shortest path between these two processors fits into one of the following categories:

(a) The path employs electronic moves only. This is possible only when $g_1 = g_2$.

(b) The path employs an even number of OTIS moves. If the number of OTIS moves is more than two, we may compress the path into a shorter path that uses 2 OTIS moves only: $(g_1,p_1) \xrightarrow{\text{E}^*} (g_1,p_2) \xrightarrow{\text{O}} (p_2,g_1) \xrightarrow{\text{E}^*} (p_2,g_2) \xrightarrow{\text{O}} (g_2,p_2)$.

(c) The path employs an odd number of OTIS moves. Again, if the number of moves is more than one, we can compress the path into a shorter one that employs exactly one OTIS move. The compressed path looks like: $(g_1,p_1) \xrightarrow{\text{E}^*} (g_1,g_2) \xrightarrow{\text{O}} (g_2,g_1) \xrightarrow{\text{E}^*} (g_2,p_2)$.

Shortest paths of type (a) have length exactly $D(p_1,p_2)$ (which equals the number of ones in the binary representation of $p_1 \oplus p_2$). Paths of type (b) and type (c) have length $D(p_1,p_2) + D(g_1,g_2) + 2$ and $D(p_1,g_2) + D(p_2,g_1) + 1$, respectively.

The following theorem follows from the preceding discussion:

Theorem 4 *[10] The length of the shortest path between processors* (g_1,p_1) *and* (g_2,p_2) *is* $d(p_1,p_2)$ *when* $g_1 = g_2$ *and* $\min\{D(p_1,p_2)+D(g_1,g_2)+2, D(p_1,g_2)+D(p_2,g_1)+1\}$ *when* $g_1 \neq g_2$.

Theorem 5 *[10] The diameter of the OTIS-Hypercube is* $2d+1$.

Proof Since each group is a d-dimensional hypercube, $D(p_1,p_2)$, $D(g_1,g_2)$, $D(p_1,g_2)$, and $D(p_2,g_1)$ are all less than or equal to d. From theorem 4, we conclude that no two processors are more than $2d+1$ apart. Now consider the processors (g_1,p_1), (g_2,p_2) such that $p_1 = 0$ and $p_2 = N-1$. Let $g_1 = 0$ and $g_2 = N-1$. So $D(p_1,p_2) = D(g_1,g_2) = D(p_1,g_2) = D(P_2,G_1) = d$. Hence, the distance between (g_1,p_1) and (g_2,p_2) is $2d+1$. As a result, the diameter of the OTIS-Mesh is exactly $2d+1$. □

Simulation of an N^2 hypercube

Zane *et al.* [13] have shown that each move of an N^2 processor hypercube can be simulated by either a single electronic move or by one electronic and two OTIS moves in an N^2 processor OTIS-Hypercube. For the simulation, processor q of the hypercube is mapped to processor (g,p) of the OTIS-Hypercube. Here $gp = q$ (*i.e.*, g is obtained from the most significant $\log_2 N$ bits of q and p

comes from the least significant $\log_2 N$ bits). Let $g_{d-1} \cdots g_0$ and $p_{d-1} \cdots p_0$, $d = \log_2 N$, be the binary representations of g and p respectively. The binary representation of q is $q_{2d-1} \cdots q_0 = g_{d-1} \cdots g_0 p_{d-1} \cdots p_0$. A hypercube move moves data from processor q to processor $q^{(k)}$ where $q^{(k)}$ is obtained from q by complementing bit k in the binary representation of q. When k is in the range $[0, d)$, the move is done in the OTIS-Hypercube by a local intra group hypercube move. When $k \geq d$, the move is done using the steps

$$\begin{aligned} & (g_{d-1} \ldots g_j \ldots g_0 p_{d-1} \ldots p_0) \\ \xrightarrow{\text{O}} \quad & (p_{d-1} \ldots p_0 g_{d-1} \ldots g_j \ldots g_0) \\ \xrightarrow{\text{E}} \quad & (p_{d-1} \ldots p_0 g_{d-1} \ldots \overline{g_j} \ldots g_0) \\ \xrightarrow{\text{O}} \quad & (g_{d-1} \ldots \overline{g_j} \ldots g_0 p_{d-1} \ldots p_0) \end{aligned}$$

where $j = k - d$.

PERMUTATION ROUTING ON OTIS COMPUTERS

Suppose we wish to rearrange the data in an N^2 processor OTIS computer according to the permutation $\Pi = \Pi[0] \cdots \Pi[N^2 - 1]$. That is, data from processor $i = gN + p$ is to be sent to processor $\Pi[i]$, $0 \leq i < N^2$. We assume that the interconnection network in each group is able to sort the data in its N processors (equivalently, it is able to perform any permutation of the data in its N processors). This assumption is certainly valid for the mesh, hypercube, perfect shuffle, cube-connected cycles, and mesh of trees interconnections mentioned earlier.

Theorem 6 *Every OTIS computer in which each group can sort can perform any permutation* Π *using at most 2 OTIS moves.*

Proof When 2 OTIS moves are permitted, the data movement can be modeled by a 3 stage MIN (multistage interconnection network) as in Figure 1.7. Each switch represents a processor group which is capable of performing any N input to N output permutation. The OTIS moves are represented by the connections from one stage to the next.

The OTIS inter stage connections are equivalent to the inter stage connections in a standard MIN that uses $N \times N$ switches. From MIN theory [4], we know that when $k \times k$ switches are used, $2 \log_k N^2 - 1$ stages of switches are sufficient to make an N^2 input N^2 output network that can realize every input to output permutation. In our case (Figure 1.7), $k = N$. Therefore, $2 \log_N N^2 - 1 = 3$ stages are sufficient. Hence 2 OTIS moves suffice to realize any permutation.

An alternative proof comes from an equivalence with the preemptive open shop scheduling problem (POSP) [2]. In the POSP we are given n jobs that are to be scheduled on m machines. Each job i has m tasks. The task length of

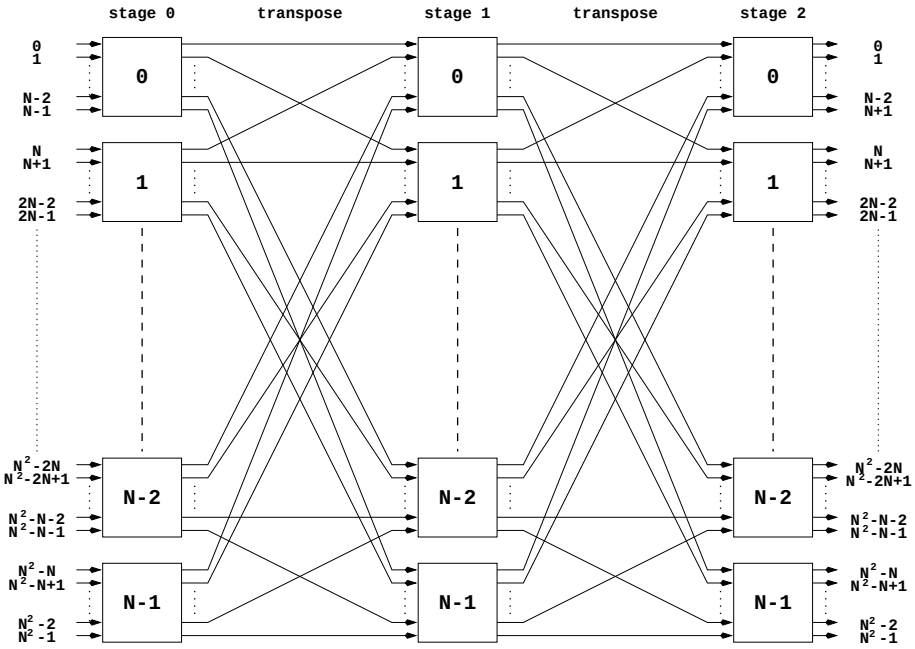

Figure 1.7 Multistage interconnection network (MIN) defined by OTIS

the jth task of job i is the integer $t_{ij} \geq 0$. In a preemptive schedule of length T, the time interval from 0 to T is divided into slices of length 1 unit each. A time slice is divided into m slots with each slot representing a unit time interval on one machine. Time slots on each machine are labeled with a job index. The labeling is done in such a way that (a) Each job (index) i is assigned to exactly t_{ij} slots on machine j, $0 \leq j < m$, and (b) No job is assigned to two or more machines in any time slice. T is the schedule length. The objective is to find the smallest T for which a schedule exists. Gonzalez and Sahni [2] have shown that the length T_{min} of an optimal schedule is

$$T_{min} = \max\{J_{max}, M_{max}\},$$

where $J_{max} = \max_i\{\sum_{j=0}^{m-1} t_{ij}\}$ (*i.e.*, J_{max} is the maximum job length) and $M_{max} = \max_j\{\sum_{i=0}^{n-1} t_{ij}\}$ (*i.e.*, M_{max} is the maximum processing to be done by any machine).

We can transform the OTIS computer permutation routing problem into a POSP. First, note that to realize a permutation Π with 2 OTIS moves, we must be able to write Π as a sequence of permutations $\Pi_0 T \Pi_1 T \Pi_2$ where Π_i is the permutation realized by the switches (*i.e.*, processor groups) in stage i and T denotes the OTIS (transpose) interstage permutation. Let (g_q, p_q) denote processor p_q of group g_q where $q \in \{i_0, o_0, i_1, o_1, i_2, o_2\}$ (i_0 = input of stage 0,

o_0 = output of stage 0, etc.). Then, the data path is $(g_{i_0}, p_{i_0}) \xrightarrow{\Pi_0} (g_{o_0}, p_{o_0}) \xrightarrow{T}$ $(p_{o_0}, g_{o_0}) = (g_{i_1}, p_{i_1}) \xrightarrow{\Pi_1} (g_{o_1}, p_{o_1}) \xrightarrow{T} (p_{o_1}, g_{o_1}) = (g_{i_2}, p_{i_2}) \xrightarrow{\Pi_2} (g_{o_2}, p_{o_2})$.

We observe that to realize the permutation Π, the following must hold:

(i) Switch i of stage 1 should receive exactly one data item from each switch of stage 0, $0 \leq i < N$.

(ii) Switch i of stage 1 should receive exactly one data item destined for each switch of stage 2, $0 \leq i < N$.

Once we know which data items are to get to switch i, $0 \leq i < N$, we can easily compute Π_0, Π_1, and Π_2. Therefore, it is sufficient to demonstrate the existence of an assignment of the N^2 stage 0 inputs to the switches in stage 1 satisfying conditions (i) and (ii). For this, we construct an N job N machine POSP instance. Job i represents switch i of stage 0 and machine j represents switch j of stage 2. The task time t_{ij} equals the number of inputs to switch i of stage 0 that are destined for switch j of stage 2 (*i.e.*, t_{ij} is the number of group i data that are destined for group j). Since Π is a permutation, it follows that $\sum_{j=0}^{N-1} t_{ij}$ = total number of inputs to switch i of stage 0 $= N$ and $\sum_{i=0}^{N-1} t_{ij}$ = total number of inputs destined for switch j of stage 2 $= N$. Therefore, $J_{max} = M_{max} = N$ and the optimal schedule length is N. Since $\sum_{i=0}^{N-1} \sum_{j=0}^{N-1} t_{ij} = N^2$ and the optimal schedule length is N, every slot of every machine is assigned a task in an optimal schedule. From the property of a schedule, it follows that in each time slice all N job labels occur exactly once. The N labels in slice i of the schedule define the inputs that are to be assigned to switch i of stage 1, $0 \leq i < N$. From properties (a) and (b) of a schedule, it follows that this assignment satisfies the requirements (i) and (ii) for an assignment to the stage 1 switches. □

Even though every permutation Π can be realized with just 2 OTIS moves, it takes many more OTIS moves to compute the decomposition $\Pi = \Pi_0 T \Pi_1 T \Pi_2$. Therefore, simulating the 3 stage MIN of Figure 1.7 does not result in an efficient algorithm to perform permutation routing. Consequently, Sahni and Wang [9, 10] have developed customized algorithms for specific as well as generalized BPC permutations. General permutations may be realized using the sorting algorithm of [11], which uses $o(\sqrt{N})$ OTIS moves.

SUMMARY OF OTHER RESULTS

Algorithms for various operations on OTIS computers have been developed in [9, 10, 11, 12, 8]. The complexity of these algorithms is governed by the number of data moves performed. So we summarize these algorithms by providing the number of electronic and OTIS moves they require.

Tables 1.1 and 1.2 give the number of data moves for the commonly used permutations — transpose, perfect shuffle, etc. — and BPC permutations [7] on the OTIS-Mesh [9] and OTIS-Hypercube [10], respectively. Note that in Table 1.2, $d = \log_2 N$.

Table 1.1 Performance of Permutations on OTIS-Mesh[9]

Operation	Electronic	OTIS
Transpose	0	1
Perfect Shuffle	$4\sqrt{N}+6$	2
Unshuffle	$4\sqrt{N}+6$	2
Bit Reversal	$8(\sqrt{N}-1)$	1
Vector Reversal	$8(\sqrt{N}-1)$	2
Bit Shuffle	$\frac{28}{3}\sqrt{N}-4$	$\log_2 N+2$
Shuffled Row-Major	$\frac{28}{3}\sqrt{N}-4$	$\log_2 N+2$
BPC	$12(\sqrt{N}-1)$	$\log_2 N+2$

Table 1.2 Performance of permutations on processor OTIS-Hypercube[10]

Operation	Electronic	OTIS
Transpose	0	1
Perfect Shuffle	$2d$	2
Unshuffle	$2d$	2
Bit Reversal	$2d$	1
Vector Reversal	$2d$	2
Bit Shuffle	$3d$	$d/2+2$
Shuffled Row-major	$3d$	$d/2+2$
BPC	$3d$	$d/2+2$

Table 1.3 gives the number of data moves for the basic operations — broadcast, prefix sum, rank, sorting, etc. — on the OTIS-Mesh with both SIMD and MIMD models [11]. Note that w in window broadcast indicates the window size, where $0 < w \leq \sqrt{N}$ and w divides $\sqrt{N}$; s in regular and circular shifts is the shift distance, $0 \leq s < \sqrt{N}/2$; and M in data accumulation, consecutive sum, and adjacent sum is the block size, $0 \leq M < \sqrt{N}/2$.

Table 1.4 gives the number of data moves for various matrix/vector multiplication operations [12]. Since the mapping of the matrix and/or vector is a determining factor on the outcome, results for both GRM and GSM mappings are listed. The big O besides the matrix $\times$ matrix operation denotes the memory capacity constraints for each processor and K denotes the maximum

Table 1.3 Performance of basic operations on OTIS-Mesh[11]

Operation	SIMD		MIMD	
	Electronic	OTIS	Electronic	OTIS
Broadcast	$4(\sqrt{N}-1)$	1	$4(\sqrt{N}-1)$	1
Window Broadcast	$4\sqrt{N}-2w-2$	2	$4\sqrt{N}-2w-2$	2
Prefix Sum	$7(\sqrt{N}-1)$	2	$7(\sqrt{N}-1)$	2
Data Sum	$8(\sqrt{N}-1)$	1	$4\sqrt{N}$	1
Rank	$7(\sqrt{N}-1)$	2	$7(\sqrt{N}-1)$	2
Regular Shift	s	2	s	2
Circular Shift	$\sqrt{N}$	2	s	2
Data Accumulation	$\sqrt{N}$	2	M	2
Consecutive Sum	$2(M-1)$	2	$M-1$	2
Adjacent Sum	$\sqrt{N}$	2	M	2
Concentrate	$7(\sqrt{N}-1)$	2	$4(\sqrt{N}-1)$	2
Distribute	$7(\sqrt{N}-1)$	2	$4(\sqrt{N}-1)$	2
Generalize	$7(\sqrt{N}-1)$	2	$4(\sqrt{N}-1)$	2
Sorting	$22\sqrt{N}+o(\sqrt{N})$	$o(\sqrt{N})$	$11\sqrt{N}+o(\sqrt{N})$	$o(\sqrt{N})$

amount of data that can be transferred in a single OTIS move (in an electronic move, 1 unit of data may be transferred. Since the optical bandwidth is larger than the electronic bandwidth, $K > 1$ units of data may be transferred in each OTIS move).

The results for randomized routing, sorting, and selection on the OTIS-Mesh [8] are listed in Table 1.5. These results are obtained under the assumption that the cost of OTIS and electronic moves is the same, and that the OTIS-Mesh is a MIMD machine.

SUMMARY

In this chapter, we have described the OTIS family of computer architectures. This family accounts for the fact that computers with a very large number of processors cannot be built using very short connections (*i.e.*, less than a few millimeters) alone. Therefore, at some level of the packaging hierarchy, the inter processor distance will be such as to favor an optical interconnect over an electronic one. The OTIS system is a simple and easy to implement free space optical interconnect system. Using OTIS for the long connections and

Table 1.4 Performance of matrix multiplication on OTIS-Mesh[12]

Scheme	GRM		GSM	
Operation*	Electronic	OTIS	Electronic	OTIS
C × R	$2\sqrt{N}$	2	$4(\sqrt{N}-1)$	2
R × C	$2(\sqrt{N}-1)$	1	$5(\sqrt{N}-1)$	2
R × M	$4(\sqrt{N}-1)$	3	$8(\sqrt{N}-1)$	2
M × C	$4(\sqrt{N}-1)$	1	$8(\sqrt{N}-1)$	2
M × M ($O(N)$)	$4(N-1)$	$N/K+1$	$4(N-1)$	$2\sqrt{N}/K+1$
M × M ($O(1)$)	$8N+O(\sqrt{N})$	$N+1$	$4N+O(\sqrt{N})$	$\sqrt{N}$

*C, R, and M denote column vector, row vector, and matrix respectively.

Table 1.5 Performance of randomized algorithms on OTIS-Mesh[8]

Operation	Complexity
Randomized Routing	$4\sqrt{N}+\tilde{o}(\sqrt{N})$
Randomized Sorting	$8\sqrt{N}+\tilde{o}(\sqrt{N})$
Randomized Selection	$6\sqrt{N}+\tilde{o}(\sqrt{N})$

electronic mesh (say) for the short interconnects results in a computer with a hybrid optoelectronic interconnection network. The development of algorithms for computers using a hybrid interconnection network poses a challenge. Using the simulation methods developed by Zane *et al.* [13], it is possible to obtain efficient OTIS algorithms from efficient algorithms for the 4D-mesh and hypercube architectures. As one might expect, we can do better than this simulation by developing algorithms specifically for the OTIS architecture [9, 10, 11, 12].

References

[1] M. Feldman, S. Esener, C. Guest, and S. Lee. Comparison between electrical and free-space optical interconnects based on power and speed considerations. *Applied Optics*, 27(9):1742–1751, May 1988.

[2] Teofilo Gonzalez and Sartaj Sahni. Open shop scheduling to minimize finish time. *Journal of the Association for Computing Machinery*, 23(4):665–679, October 1976.

[3] W. Hendrick, O. Kibar, P. Marchand, C. Fan, D. V. Blerkom, F. McCormick, I. Cokgor, M. Hansen, and S. Esener. Modeling and optimization of the optical transpose interconnection system. in Optoelectronic Technology Center, Program Review, Cornell University, September 1995.

[4] David M. Koppelman and A. Yavuz Oruç. The complexity of routing in clos permutation networks. *IEEE Transactions on Information Theory*, 40(1):278–284, January 1994.

[5] A. Krishnamoorthy, P. Marchand, F. Kiamilev, and S. Esener. Grain-size considerations for optoelectronic multistage interconnection networks. *Applied Optics*, 31(26):5480–5507, September 1992.

[6] Gary C. Marsden, Philippe J. Marchand, Phil Harvey, and Sadik C. Esener. Optical transpose interconnection system architectures. *Optics Letters*, 18(13):1083–1085, July 1 1993.

[7] David Nassimi and Sartaj Sahni. An optimal routing algorithm for mesh-connected parallel computers. *Journal of the Association for Computing Machinery*, 27(1):6–29, January 1980.

[8] Sanguthevar Rajasekaran and Sartaj Sahni. Randomized routing, selection, and sorting on the OTIS-Mesh optoelectronic computer. *IEEE Transactions on Parallel and Distributed Systems*, 1998. To appear.

[9] Sartaj Sahni and Chih-Fang Wang. BPC permutations on the OTIS-Mesh optoelectronic computer. In *Proceedings of the fourth International Conference on Massively Parallel Processing Using Optical Interconnections (MPPOI'97)*, pages 130–135, 1997.

[10] Sartaj Sahni and Chih-Fang Wang. BPC permutations on the OTIS-Hypercube optoelectronic computer. *Informatica*, 1998. To appear.

[11] Chih-Fang Wang and Sartaj Sahni. Basic operations on the OTIS-Mesh optoelectronic computer. In *Proceedings of the fifth International Conference on Massively Parallel Processing Using Optical Interconnections (MPPOI'98)*, pages 150–157, 1998.

[12] Chih-Fang Wang and Sartaj Sahni. Matrix multiplication on the otis-mesh optoelectronic computer. Technical report, CISE Department, University of Florida, 1998.

[13] Francis Zane, Philippe Marchand, Ramamohan Paturi, and Sadik Esener. Scalable network architectures using the optical transpose interconnection system (OTIS). In *Proceedings of the second International Conference on Massively Parallel Processing Using Optical Interconnections (MPPOI'96)*, pages 114–121, 1996.

Chih-fang Wang is currently a Ph.D. candidate at the Department of Computer and Information Science and Engineering, University of Florida. He got his M.Sc. degree from Mathematics and Computer Science, University of Miami in 1993.

His research interests include distributed and parallel algorithms, reconfigurable networks, optical interconnection computers, and all-optical networks.

Sartaj Sahni is a University of Florida Research Foundation Professor of Computer and Information Sciences and Engineering and a Fellow of IEEE, ACM, AAAS, and Minnesota Supercomputer Institute. He received his B.Tech. (Electrical Engineering) degree from the Indian Institute of Technology, Kanpur, and the M.S. and Ph.D. degrees in Computer Science from Cornell University. Dr. Sahni has published over one hundred and fifty research papers and written several texts. His research publications are on the design and analysis of efficient algorithms, parallel computing, interconnection networks, and design automation.

In 1997, he was awarded the IEEE Taylor L. Booth Education Award "for contributions to Computer Science and Engineering education in the areas of data structures, algorithms, and parallel algorithms". Dr. Sahni is a co-editor of the Journal of Parallel and Distributed Computing and is on the editorial boards of IEEE Parallel and Distributed Technology, and Computer Systems: Science and Engineering. He has served as program committee chair, general chair, and been a keynote speaker at many conferences.