BPC Permutations On The OTIS-Mesh Optoelectronic Computer*

Sartaj Sahni and Chih-fang Wang
Department of Computer and Information Science and Engineering
University of Florida
Gainesville, FL 32611
{sahni,wang}@cise.ufl.edu

Abstract

We show that the diameter of an N^2 processor OTIS-Mesh is $4\sqrt{N}-3$. Two possible embeddings of an $N \times N$ mesh onto an OTIS-Mesh are evaluated. OTIS-Mesh algorithms for some commonly performed permutations – transpose, bit reversal, vector reversal, perfect shuffle, unshuffle, shuffled row-major, and bit shuffle – are developed. We also propose an algorithm for general BPC permutations.

1 Introduction

Marsden *et al.* [6], Hendrick *et al.* [2], and Zane *et al.* [8] have proposed large scale parallel computer architectures in which the processors are divided into groups. Each group of processors is realized using one or more high density chips or modules with electronic interprocessor connections. Interconnections between processors in different groups are realized via free space optics. These optoelectronic architectures use electronic interconnects over short distances (i.e., for the local intra group interconnects) and free space optics for the longer inter group interconnects. This is so as it is known [1] [3] that free space optical connects provide power, speed, and crosstalk advantages over electronic interconnects when the connect distance is more than a few millimeters. Further, when optical interconnects are used, the per group I/O bandwidth increases in proportion to the group's layout area while when electronic interconnects are used, this bandwidth grows in proportion to the layout perimeter.

Krishnamoorthy *et al.* [4] have shown that the bandwidth and power consumption are minimized when the number of processors in a group equal the number of groups. That is when an N^2 processor system is partitioned into N groups each containing N processors. As a result, we limit our study of optoelectronic architectures to those partitioned in this manner. The specific optoelectronic architecture proposed by Marsden *et al.* [6] employs the optical transpose interconnection system

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

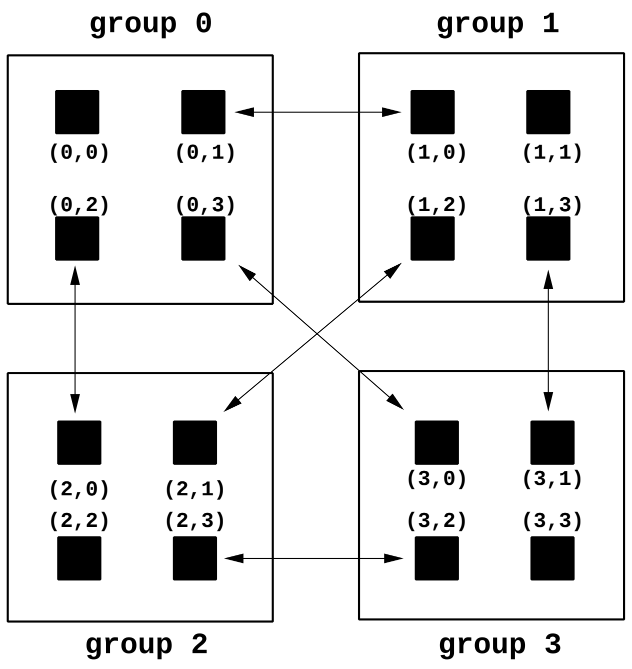

Figure 1: Example of OTIS connections with 16 processors

(OTIS). In this, processor i of group j is connected to processor j of group i via an optical link. Figure 1 shows a 16 processor OTIS architecture. The processor indices are represented as pairs of the form (G, P) where G is the group index and P the processor index (within the group).

Some of the suggested topologies for the intra group electronic connections are mesh, hypercube, and mesh of trees [6]. These, respectively, result in the OTIS-Mesh, OTIS-Hypercube, and OTIS-MT optoelectronic architectures. Figure 2 shows a 16 processor OTIS-Mesh computer. 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.

The OTIS-Hypercube architecture was modeled and its performance characteristics (power, throughput, volume, etc.) were computed in [2]. Zane *et al.* [8] have shown that the OTIS-Mesh can simulate each move of an $\sqrt{N} \times \sqrt{N} \times \sqrt{N} \times \sqrt{N}$ four dimensional mesh by using either an electronic move local to a group or one local electronic move and two inter group moves using the OTIS (or optical) interconnection. We shall refer to the latter as OTIS moves. Further, Zane *et al.* [8] have also shown that each move of an N^2 processor hypercube can be simulated by an N^2 processor OTIS-Hypercube using either one local electronic move or two OTIS and one electronic moves.

In this paper, we study the OTIS-Mesh architecture and obtain properties and basic permutation routing algorithms. These algorithms can in turn be used in the development of efficient application

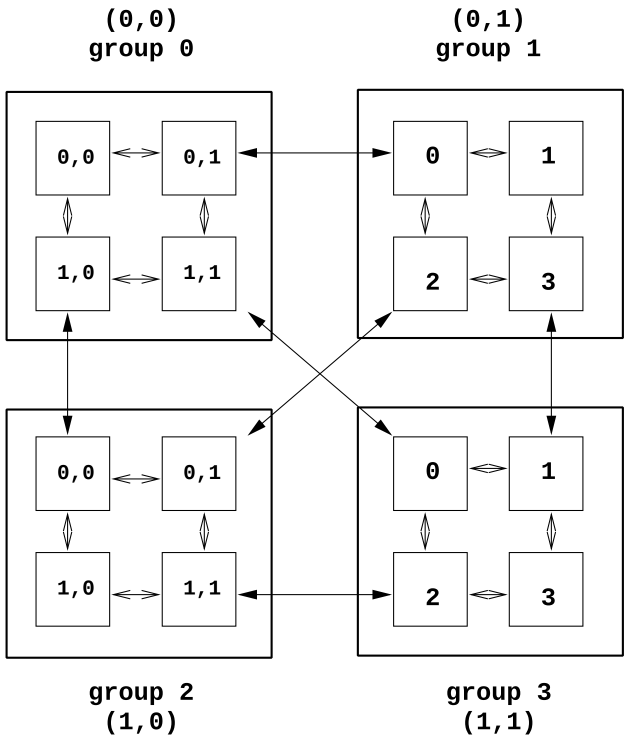

Figure 2: 16 processor OTIS-Mesh

codes. We begin, in Section 2, by deriving the diameter (i.e., maximum distance between any two processors) of the OTIS-Mesh. In Section 3, we consider the problem of embedding a two dimensional $N \times N$ mesh into an N^2 processor OTIS-Mesh. Algorithms to perform popular data rearrangements such as transpose, reversal, and shuffle are developed in Section 4. An algorithm for general BPC (bit permute complement) permutations is presented in Section 5.

2 OTIS-Mesh Diameter

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 forms:

(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 (i.e., OTIS) moves. In this case the path is of the form $(G_1, P_1) \xrightarrow{\text{E}} (G_1, P_1') \xrightarrow{\text{O}} (P_1', G_1) \xrightarrow{\text{E}} (P_1', G_1') \xrightarrow{\text{O}} (G_1', P_1') \xrightarrow{\text{E}} (G_1', P_1'') \xrightarrow{\text{O}} (P_1'', G_1') \xrightarrow{\text{E}} (P_1'', G_1'') \xrightarrow{\text{O}} \cdots \xrightarrow{\text{E}} (G_2, P_2)$.

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 $(G_1, P_1) \xrightarrow{\mathrm{E}} (G_1, P_2) \xrightarrow{\mathrm{O}} (P_2, G_1) \xrightarrow{\mathrm{E}} (P_2, G_2) \xrightarrow{\mathrm{O}} (G_2, P_2)$.

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 $(G_1, P_1) \xrightarrow{\mathrm{E}} (G_1, G_2) \xrightarrow{\mathrm{O}} (G_2, G_1) \xrightarrow{\mathrm{E}} (G_2, P_2)$.

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.

The preceding development results in

Theorem 1 *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 *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 $4\sqrt{N} - 3$. □

3 Embedding Of An $N \times N$ Mesh

Zane *et al.* [8] have shown that a $\sqrt{N}\times\sqrt{N}\times\sqrt{N}\times\sqrt{N}$ mesh may be embedded into an N^2 processor OTIS-Mesh so that each mesh move may be performed by either one electronic OTIS-Mesh move or one electronic and two optical OTIS-Mesh moves. The embedding is rather straightforward with processor (i, j, k, l) of the mesh being identified with processor (G, P), $G = i\sqrt{N} + j$, $P = k\sqrt{N} + l$ of the OTIS-Mesh. 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 $(i, j, k, l) \xrightarrow{\text{O}} (k, l, i, j) \xrightarrow{\text{E}} (k, l, i, j+1) \xrightarrow{\text{O}} (i, j+1, k, l)$.

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 [8]. 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.

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 3(a). The pair of numbers in each processor of Figure 3(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.

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

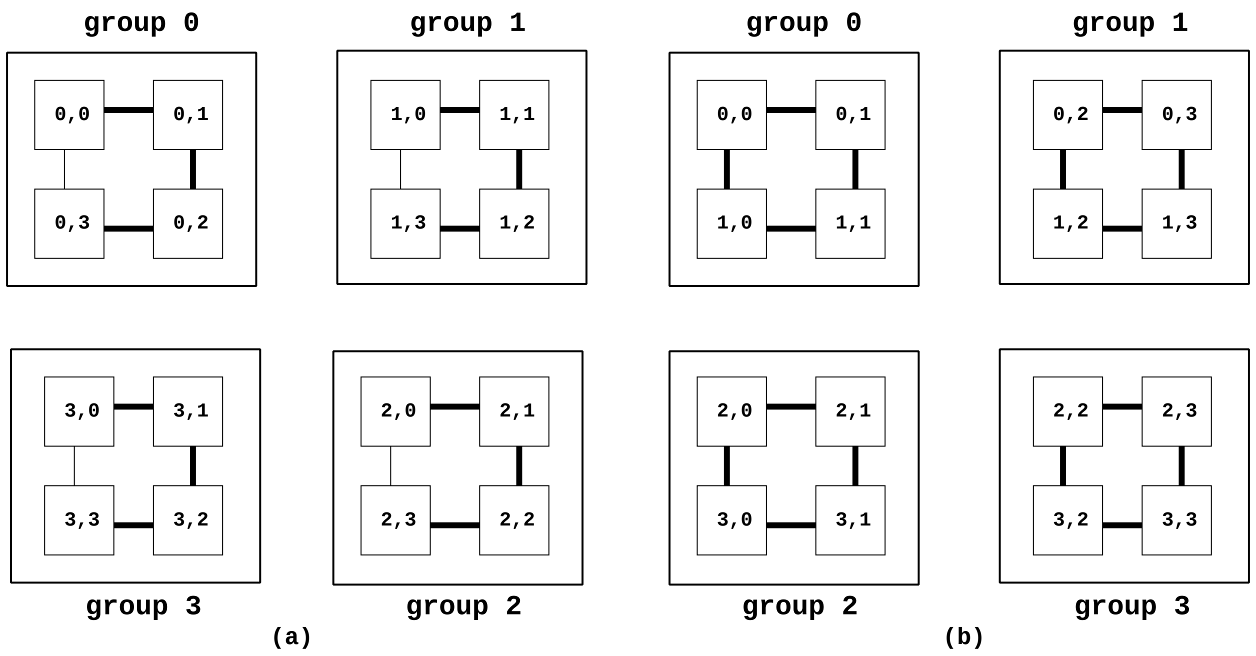

Figure 3: Mapping a 4×4 mesh onto a 16 processor OTIS-Mesh: (a) GRM; (b) GSM

way onto a group of OTIS-Mesh processors. Figure 3(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 *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 Section 2):

(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. □

Permutation	4D mesh			OTIS-Mesh Simulation	
	total	dimension $1+2$	dimension $3+4$	OTIS	electronic
Transpose	$8(\sqrt{N}-1)$	$4(\sqrt{N}-1)$	$4(\sqrt{N}-1)$	$8(\sqrt{N}-1)$	$8(\sqrt{N}-1)$
Perfect Shuffle	$4\sqrt{N}$	$2\sqrt{N}$	$2\sqrt{N}$	$4\sqrt{N}$	$4\sqrt{N}$
Unshuffle	$4\sqrt{N}$	$2\sqrt{N}$	$2\sqrt{N}$	$4\sqrt{N}$	$4\sqrt{N}$
Bit Reversal	$8(\sqrt{N}-1)$	$4(\sqrt{N}-1)$	$4(\sqrt{N}-1)$	$8(\sqrt{N}-1)$	$8(\sqrt{N}-1)$
Vector Reversal	$8(\sqrt{N}-1)$	$4(\sqrt{N}-1)$	$4(\sqrt{N}-1)$	$8(\sqrt{N}-1)$	$8(\sqrt{N}-1)$
Bit Shuffle	$\frac{20}{3}\sqrt{N}-4$	$\frac{8}{3}\sqrt{N}-2$	$4\sqrt{N}-2$	$\frac{16}{3}\sqrt{N}-4$	$\frac{28}{3}\sqrt{N}-4$
Shuffled Row-major	$\frac{20}{3}\sqrt{N}-4$	$\frac{8}{3}\sqrt{N}-2$	$4\sqrt{N}-2$	$\frac{16}{3}\sqrt{N}-4$	$\frac{28}{3}\sqrt{N}-4$
G_yP_x Swap	$4(\sqrt{N}-1)$	$2(\sqrt{N}-1)$	$2(\sqrt{N}-1)$	$4(\sqrt{N}-1)$	$4(\sqrt{N}-1)$

Table 1: Optimal moves for 4D mesh and respective OTIS-Mesh simulations

Based on the relative performance of GRM and GSM with respect to the common mesh operations of shift by row or column and transpose, we recommend the use of GRM.

4 Common Data Rearrangements

Since an N^2 input three stage OTIS multistage interconnection network (MIN) composed of N N-input–N-output switches in each stage is rearrangeable [6] and since the N processor mesh in each group of an N^2 processor OTIS-Mesh is able to realize every permutation of N elements, an N^2 processor OTIS-Mesh can realize any permutation of N^2 data (one to each processor) using at most two OTIS moves. However, additional OTIS moves are needed to determine the local group data rearrangements that must be made. Another way to accomplish arbitrary data permutations is to use sorting. The optimal 4D mesh sorting algorithm of [5] may be run on an OTIS-Mesh using the simulation described in [8]. This yields an OTIS-Mesh sorting algorithm that takes $12\sqrt{N}$ OTIS and $14\sqrt{N}$ electronic moves.

In this section, we are concerned with the realization of permutations such as transpose, shuffle, unshuffle, and vector reversal which arise frequently in applications. Nassimi and Sahni [7] have developed optimal 4D mesh algorithms for several frequently arising permutations. These may be simulated using the method of [8] to obtain algorithms for the OTIS-Mesh. Table 1 gives the number of 4D mesh moves used by the optimal 4D mesh algorithms, a break down of the number of moves in the first two and last two dimensions, and the number of electronic and OTIS moves required by the simulation.

We shall obtain OTIS-Mesh algorithms for the permutations of Table 1, that require far fewer moves than the simulations of the optimal 4D mesh algorithms.

Assume that the N^2 OTIS-Mesh processors are numbered/indexed 0 through N^2-1 such that in the binary representation of a processor index the left half bits give the group number and the right half give the processor number local to a group. So, a processor index I is of the form $I = GP$ where I, G and P are represented in binary and G and P have the same number of bits. G and P may be decomposed into halves to get $G = G_xG_y$ and $P = P_xP_y$ such that G_x and G_y give the group location by row and column in an array layout of groups (as in Figure 1) and P_x and P_y locate processor P of a group by its row and column coordinates.

The permutations of Table 1 are members of the BPC (bit permute complement) class of permutations defined in [7]. In a BPC permutation, the destination processor of each data is given by a rearrangement of the bits in the source processor index. For the case of our N^2 processor OTIS-Mesh we assume that N is a power of two and so the number of bits needed to represent a processor index is $p = \log_2 N^2 = 2\log N$. A BPC permutation [7] is specified by a vector $A = [A_{p-1}, A_{p-2}, \ldots, A_0]$ where

(a) $A_i \in \{\pm 0, \pm 1, \ldots, \pm(p-1)\}$, $0 \le i < p$ and

(b) $[|A_{p-1}|, |A_{p-2}|, \ldots, |A_0|]$ is a permutation of $[0, 1, \ldots, p-1]$.

The destination for the data in any processor may be computed in the following manner. Let $m_{p-1}m_{p-2}\ldots m_0$ be the binary representation of the processor's index. Let $d_{p-1}d_{p-2}\ldots d_0$ be that of the destination processor's index. Then,

$$d_{|A_i|} = \begin{cases} m_i & if \quad A_i \ge 0, \\ 1 - m_i & if \quad A_i < 0. \end{cases}$$

In this definition, -0 is to be regarded as < 0, while $+0$ is ≥ 0.

In a 16 processor OTIS-Mesh, the processor indices have four bits with the first two giving the group number and the second two the local processor index. The BPC permutation $[-0, 1, 2, -3]$ requires data from each processor $m_3m_2m_1m_0$ be routed to processor $(1-m_0)m_1m_2(1-m_3)$. Table 2 lists the source and destination processors of the permutation.

The permutation vector A for each of the permutations of Table 1 is given in Table 3. The permutation vector A for each of the permutations of Table 1 is given in Table 3.

4.1 Transpose $[p/2-1, \ldots, 0, p-1, \ldots, p/2]$

The transpose operation may be accomplished via a single OTIS move and zero electronic moves. The simulation of the optimal 4D mesh algorithm, however, takes $8(\sqrt{N}-1)$ OTIS and $8(\sqrt{N}-1)$

Source			Destination		
Processor	(G, P)	Binary	Binary	(G, P)	Processor
0	(0,0)	0000	1001	(2,1)	9
1	(0,1)	0001	0001	(0,1)	1
2	(0,2)	0010	1101	(3,1)	13
3	(0,3)	0011	0101	(1,1)	5
4	(1,0)	0100	1011	(2,3)	11
5	(1,1)	0101	0011	(0,3)	3
6	(1,2)	0110	1111	(3,3)	15
7	(1,3)	0111	0111	(1,3)	7
8	(2,0)	1000	1000	(2,0)	8
9	(2,1)	1001	0000	(0,0)	0
10	(2,2)	1010	1100	(3,0)	12
11	(2,3)	1011	0100	(1,0)	4
12	(3,0)	1100	1010	(2,2)	10
13	(3,1)	1101	0010	(0,2)	2
14	(3,2)	1110	1110	(3,2)	14
15	(3,3)	1111	0110	(1,2)	6

Table 2: Source and destination of the BPC permutation $[\mathbf{-0, 1, 2, -3}]$ in a 16 processor OTIS-Mesh

Permutation	Permutation Vector
Transpose	$[p/2-1, \ldots, 0, p-1, \ldots, p/2]$
Perfect Shuffle	$[0, p-1, p-2, \ldots, 1]$
Unshuffle	$[p-2, p-3, \ldots, 0, p-1]$
Bit Reversal	$[0, 1, \ldots, p-1]$
Vector Reversal	$[-(p-1), -(p-2), \ldots, -0]$
Bit Shuffle	$[p-1, p-3, \ldots, 1, p-2, p-4, \ldots, 0]$
Shuffled Row-major	$[p-1, p/2-1, p-2, p/2-2, \ldots, p/2, 0]$
G_yP_x Swap	$[p-1, \ldots, 3p/4, p/2-1, \ldots, p/4, 3p/4-1, \ldots, p/2, p/4-1, \ldots, 0]$

Table 3: Permutations and their permutation vectors

electronic moves.

4.2 Perfect Shuffle $[0, p-1, p-2, \ldots, 1]$

Let G represent the first half of the bits in the processor index and P the second half. Let $b_{G(i)}$ and $b_{P(i)}$, respectively, denote the bits in position $G(i)$ and $P(i)$ of G and P. So $b_{G(p/2-1)}$ and $b_{P(p/2-1)}$ are the most significant bits of G and P while $b_{G(0)}$ and $b_{P(0)}$ are the least. Let $G = b_{G(p/2-1)}G'$ and $P = b_{P(p/2-1)}P'$. A perfect shuffle may be performed as below:

Step 1: Perform a local perfect shuffle in each group. This moves data from every processor GP to the corresponding processor $GP'b_{P(p/2-1)}$.

Step 2: This step involves processors in groups G such that $b_{G(p/2-1)} = 0$ only. In these groups, odd processors exchange data with corresponding even processors (note that the processors exchanging data differ only in bit zero). To see the new data arrangement, it is convenient to separete out four cases depending on the values of $b_{G(p/2-1)}$ and $b_{P(p/2-1)}$. Steps 1 and 2 accomplish the following:

$$\begin{array}{ccccc}
0G'0P' & & 0G'P'0 & & 0G'P'1 \\
0G'1P' & & 0G'P'1 & & 0G'P'0 \\
1G'0P' & \xrightarrow{\text{Step 1}} & 1G'P'0 & \xrightarrow{\text{Step 2}} & 1G'P'0 \\
1G'1P' & & 1G'P'1 & & 1G'P'1
\end{array}$$

Step 3: Perform an OTIS move on all processors.

Step 4: Perform a local shuffle in each group. The transformations so far are given below:

$$\begin{array}{ccccccccc}
0G'0P' & & 0G'P'0 & & 0G'P'1 & & P'10G' & & P'1G'0 \\
0G'1P' & & 0G'P'1 & & 0G'P'0 & & P'00G' & & P'0G'0 \\
1G'0P' & \xrightarrow{\text{Step 1}} & 1G'P'0 & \xrightarrow{\text{Step 2}} & 1G'P'0 & \xrightarrow{\text{Step 3}} & P'01G' & \xrightarrow{\text{Step 4}} & P'0G'1 \\
1G'1P' & & 1G'P'1 & & 1G'P'1 & & P'11G' & & P'1G'1
\end{array}$$

Step 5: This step involves only processors in even groups. In these groups, odd processors exchange their data with the corresponding even processors.

$$\begin{array}{ccccccccccc}
0G'0P' & & 0G'P'0 & & 0G'P'1 & & P'10G' & & P'1G'0 & & P'1G'0 \\
0G'1P' & & 0G'P'1 & & 0G'P'0 & & P'00G' & & P'0G'0 & & P'0G'1 \\
1G'0P' & \xrightarrow{\text{Step 1}} & 1G'P'0 & \xrightarrow{\text{Step 2}} & 1G'P'0 & \xrightarrow{\text{Step 3}} & P'01G' & \xrightarrow{\text{Step 4}} & P'0G'1 & \xrightarrow{\text{Step 5}} & P'0G'0 \\
1G'1P' & & 1G'P'1 & & 1G'P'1 & & P'11G' & & P'1G'1 & & P'1G'1
\end{array}$$

Step 6: Perform an OTIS move on all processors.

Step 7: Same as Step 5. The seven step process is shown below:

$$\begin{array}{cccccccccccc}
0G'0P' & & 0G'P'0 & & 0G'P'1 & & P'10G' & & P'1G'0 & & P'1G'0 & \\
0G'1P' & & 0G'P'1 & & 0G'P'0 & & P'00G' & & P'0G'0 & & P'0G'1 & \\
1G'0P' & \xrightarrow{\text{Step 1}} & 1G'P'0 & \xrightarrow{\text{Step 2}} & 1G'P'0 & \xrightarrow{\text{Step 3}} & P'01G' & \xrightarrow{\text{Step 4}} & P'0G'1 & \xrightarrow{\text{Step 5}} & P'0G'0 & \xrightarrow{\text{Step 6}} \\
1G'1P' & & 1G'P'1 & & 1G'P'1 & & P'11G' & & P'1G'1 & & P'1G'1 &
\end{array}$$

$$\begin{array}{lcl} G'0P'1 & & G'0P'0 \\ G'1P'0 & & G'1P'0 \\ G'0P'0 & \xrightarrow{\text{Step 7}} & G'0P'1 \\ G'1P'1 & & G'1P'1 \end{array}$$

The correctness of the seven step algorithm above is readily seen. From the diagram of the data movement operations, we see that data originally in $0G'0P'$ ends up in $G'0P'0$; that in $0G'1P'$ ends up in $G'1P'0$; that in $1G'0P'$ ends up in $0G'1P'$; and that in $1G'1P'$ ends up in $G'1P'1$. In other words, data is moved from GP to $G'b_{P(p/2-1)}P'b_{G(p/2-1)}$ which is precisely what is to be done in a perfect shuffle.

Steps 1 and 4 perform perfect shuffles in $\sqrt{N} \times \sqrt{N}$ meshes. Each of these can be done optimally in $2\sqrt{N}$ electronic moves using the algorithm of [7]. Steps 2, 5, and 7 requires exchanging data between mesh neighbors. Each exchange moves data in opposite directions on the same link and takes two electronic moves. Steps 3 and 6 take one OTIS move each. So, the total number of moves is $4\sqrt{N}+6$ electronic and 2 OTIS only. In contrast, the simulation of the optimal 4D mesh perfect shuffle algorithm takes $4\sqrt{N}$ electronic and $4\sqrt{N}$ OTIS moves.

4.3 Unshuffle $[p-2, p-3, \ldots, 0, p-1]$

This is the inverse of a perfect shuffle and may be done by running the seven step shuffle algorithm backwards (i.e., beginning with Step 7) and replacing the local shuffles of Steps 1 and 4 by local unshuffles. The data movement is shown below ($G = G''b_{G(0)}$, $P = P''b_{P(0)}$).

$$\begin{array}{lclclclclclcl} G''0P''0 & & G''0P''1 & & P''1G''0 & & P''1G''0 & & P''10G'' & & 0G''P''1 & \\ G''0P''1 & & G''0P''0 & & P''0G''0 & & P''0G''1 & & P''01G'' & & 1G''P''0 & \\ G''1P''0 & \xrightarrow{\text{Step 7}} & G''1P''0 & \xrightarrow{\text{Step 6}} & P''0G''1 & \xrightarrow{\text{Step 5}} & P''0G''0 & \xrightarrow{\text{Step 4}} & P''00G'' & \xrightarrow{\text{Step 3}} & 0G''P''0 & \xrightarrow{\text{Step 2}} \\ G''1P''1 & & G''1P''1 & & P''1G''1 & & P''1G''1 & & P''11G'' & & 1G''P''1 & \end{array}$$

$$\begin{array}{lcl} 0G''P''0 & & 0G''0P'' \\ 1G''P''0 & & 1G''0P'' \\ 0G''P''1 & \xrightarrow{\text{Step 1}} & 0G''1P'' \\ 1G''P''1 & & 1G''1P'' \end{array}$$

The number of data moves is the same as for a perfect shuffle.

4.4 Bit Reversal $[0, 1, \ldots, p-1]$

A bit reversal can be done using one OTIS and $8(\sqrt{N}-1)$ electronic moves as below. Note that when the bit reversal is done by simulating the optimal 4D mesh algorithm, $8(\sqrt{N}-1)$ electronic and $8(\sqrt{N}-1)$ OTIS moves are made.

Step 1: Do a local bit reversal in each group.

Step 2: Perform an OTIS move of all data.

Step 3: Do a local bit reversal in each group.

Steps 1 and 3 are done optimally in $4(\sqrt{N}-1)$ electronic moves each using the optimal 2D mesh bit reversal algorithm of [7].

4.5 Vector Reversal $[-(p-1), -(p-2), \ldots, 0]$

A vector reversal can be done using $8(\sqrt{N}-1)$ electronic and two OTIS moves. The steps are:

Step 1: Perform a local vector reversal in each group.

Step 2: Do an OTIS move of all data.

Step 3: Perform a local vector reversal in each group.

Step 4: Do an OTIS move of all data.

Note that Step 1 moves data from GP to $G\overline{P}$ (where $\overline{P}$ is the complement of P). Step 2 moves this data from $G\overline{P}$ to $\overline{P}G$. Next, Step 3 sends that data to $\overline{PG}$ and finally Step 4 sends it to $\overline{GP}$ completing the vector reversal. The number of data moves is easily obtained by noting that the optimal way to perform the local vector reversals takes $4(\sqrt{N}-1)$ electronic moves [7].

4.6 Bit Shuffle $[p-1, p-3, \ldots, 1, p-2, p-4, \ldots, 0]$

Our algorithm to perform this permutation employs a G_yP_x Swap permutation in which data from processor $G_xG_yP_xP_y$ is routed to processor $G_xP_xG_yP_y$. So, let us first see how to perform this permutation.

4.6.1 G_yP_x Swap $[p-1,\ldots,3p/4, p/2-1,\ldots,p/4, 3p/4-1,\ldots,p/2, p/4-1,\ldots,0]$

We present two algorithms for this. The first uses $2(\sqrt{N}-1)$ electronic and $\log_2 N$ OTIS moves. The second uses $6(\sqrt{N}-1)$ electronic and 2 OTIS moves. While the second algorithm uses a larger number of moves, it is to be preferred when the cost of an OTIS move is considerably larger than that of an electronic move.

The first algorithm performs a series of bit exchange permutations of the form $B(i) = [B_{p-1}, \ldots, B_0]$, $0 \le i < p/4$, where

$$B_j = \begin{cases} p/2+i, & j = p/4+i \\ p/4+i, & j = p/2+i \\ j & otherwise \end{cases}$$

The permutaion $B(i)$ may be realized as below:

Step 1: Processors GP with $b_{G(i)} \neq b_{P(p/4+i)}$ route their data to corresponding processors that differ only in bit $b_{P(p/4+i)}$. This requires moving data left and right on rows of $\sqrt{N} \times \sqrt{N}$ meshes by 2^i positions (in each direction).

Step 2: Perform an OTIS move.

Step 3: The data moved in Step 1 is routed from their current processors to corresponding processors that differ only in bit i. This requires data moves left and right along rows of $\sqrt{N} \times \sqrt{N}$ meshes. The distance is 2^i in each direction.

Step 4: Perform an OTIS move.

The total number of moves is 2^{i+2} electronic and two OTIS.

To perform a G_yP_x Swap permutation, we simply perform $B(i)$ permutations for $0 \leq i < p/4$. This takes $p/2 = \log_2 N$ OTIS moves and $\sum_{i=0}^{p/4-1} 2^{i+2} = 4(2^{p/4} - 1) = 4(\sqrt{N} - 1)$ electronic moves.

The second algorithm uses the following six steps:

Step 1: Shift data in group G_xG_y up circularly by G_y rows. This moves data from processor $G_xG_yP_xP_y$ to processor $G_xG_y((P_x - G_y) \bmod \sqrt{N})P_y$.

Step 2: Do an OTIS move. Data from $G_xG_yP_xP_y$ is now in $((P_x - G_y) \bmod \sqrt{N})P_yG_xG_y$.

Step 3: In each group, shift the data right circularly along the rows by an amount given by the left half of the group bits. Data originally in $G_xG_yP_xP_y$ is now in $((P_x - G_y) \bmod \sqrt{N})P_yG_xP_x$.

Step 4: Do an OTIS move. The data is now in $G_xP_x((P_x - G_y) \bmod \sqrt{N})P_y$.

Step 5: Move data up circularly along columns by an amount given by the right half of the group bits. The data is now in $G_xP_x(-G_y \bmod \sqrt{N})Py$.

Step 6: reverse the order of data in each column of each group. The data is now in $G_xP_xG_yP_y$.

While a column or row circular shift takes $\sqrt{N}$ moves in each group, the number of moves in each direction varies from group to group. Assuming that up moves in one group may not be overlapped with down moves in another, Steps 1, 3, and 5 take $2(\sqrt{N} - 1)$ electronic moves each. Step 6 may be combined with Step 5 at no extra cost. So, a total of $6(\sqrt{N} - 1)$ electronic and two OTIS moves are used.

4.6.2 Bit Shuffle

A bit shuffle may be performed following these steps:

Step 1: Perform a G_yP_x swap.

Step 2: Do a local bit shuffle in each group.

Step 3: Do an OTIS move.

Step 4: Do a local bit shuffle in each group.

Step 5: Do an OTIS move.

Using the $4(\sqrt{N}-1)$ electronic and $\log_2 N$ OTIS move algorithm for the G_yP_x Swap and the optimal mesh bit shuffle algorithm of [7], the number of moves becomes (approximately) $\frac{28}{3}\sqrt{N}-4$ electronic and $\log_2 N+2$ OTIS.

4.7 Shuffled Row-major $[p-1, p/2-1, p-2, p/2-2, \ldots, p/2, 0]$

This is the inverse of a bit shuffle and may be done in the same number of moves by running the bit shuffle algorithm backwards. Of course, Steps 2 and 4 are to be changed to shuffled row-major operations.

5 BPC Permutations

Every BPC permutation, A, may be realized by a sequence of bit exchange permutations of the form $B(i,j) = [B_{p-1}, \ldots, B_0]$, $p/2 \le i < p$, $0 \le j < p/2$, and

$$B_q = \begin{cases} j, & q = i \\ i, & q = j \\ q, & otherwise, \end{cases}$$

and a BPC permutation $C = [C_{p-1}, \ldots, C_0] = \Pi_G \Pi_P$ where $|C_q| < p/2$, $0 \le q < p/2$, Π_G and Π_P involve $p/2$ bits each. Let Π'_G be the permutation obtained from Π_G by subtracting $p/2$ from each entry whose absolute value exceeds $p/2-1$. For example, if $\Pi_G = [-3, 5, 4]$, then $p = 6$ and $\Pi'_G = [-0, 2, 1]$.

The transpose permutation may be realized by the sequence $B(p/2+j, j)$, $0 \le j < p/2$; bit reversal is equivalent to the sequence $B(p-1-j, j)$, $0 \le j < p/2$; vector reversal can be realized by performing no bit exchanges and using $C = [-(p-1), -(p-2), \ldots, -0]$ ($\Pi_G = [-(p-1), -(p-2), \ldots, -p/2]$, $\Pi_P = [-(p/2-1), \ldots, -0]$) ; perfect shuffle may be decomposed into $B(p/2, 0)$ and

$C = [p-2, p-3, \ldots, p/2, p-1, p/2-2, \ldots, 1, 0, p/2-1]$ ($\Pi_G = [p-2, p-3, \ldots, p/2, p-1]$, $\Pi_P = [p/2-2, \ldots, 1, 0, p/2-1]$).

A bit exchange permutation $B(i,j)$ may be performed in $2^{i'} + 2^{j'}$ electronic, where

$$i' = \begin{cases} i - p/2, & i < 3p/4 \\ i - 3p/4, & i \geq 3p/4; \end{cases}$$
$$j' = \begin{cases} j, & j < p/4 \\ j - p/4, & j \geq p/4 \end{cases}$$

and 2 OTIS moves following a process silimar to that used for $B(i)$ in Section 4.6.1.

Our algorithm for general BPC permutations is:

Step 1: Decompose the BPC permutation A into the bit exchange permutations $B_1(i_1, j_1)$, $B_2(i_2, j_2)$,. . ., $B_k(i_k, j_k)$ and the BPC permutation $C = \Pi_G \Pi_P$ as above. Do this such that $i_1 > i_2 > \cdots > i_k$, and $j_1 > j_2 > \cdots > j_k$.

Step 2: If $k = 0$, do the following:

Step 2.1: Do the BPC permutation Π_P in each group using the optimal algorithm of [7].

Step 2.2: Do an OTIS move.

Step 2.3: Do the BPC permutation Π'_G in each group using the algorithm of [7].

Step 2.4: Do an OTIS move.

Step 3: If $k = p/2$, do the following:

Step 3.1: Do the BPC permutation Π'_G in each group.

Step 3.2: Do an OTIS move.

Step 3.3: Do the BPC permutation Π_P in each group.

Step 4: If $k < p/4$, do the following:

Step 4.1: Perform the bit exchange permutation $B_1, \ldots, B_k$.

Step 4.2: Do Steps 2.1 through 2.4.

Step 5: If $k \geq p/4$, do the following:

Step 5.1: Perform a sequence of $p/2 - k$ bit exchanges involving bits other than those in $B_1, \ldots, B_k$ in the same orderly fashion described in Step 1. Recompute Π_G and Π_P. Swap Π_G and Π_P.

Step 5.2: Do Steps 3.1 through 3.3.

Consider the permutation $A = [6, 11, 3, 8, 10, 7, 0, 4, 13, 14, 2, 9, 1, 15, 5, 12]$ in a 2^{16} processor OTIS-Mesh (we have omitted complements for simplicity; bit complements can be taken care of when the local BPC permutations Π_G and Π_P are performed). For this, the decomposition of Step 1 yields $B_1 = B(15, 7)$, $B_2 = B(13, 6)$, $B_3 = B(10, 4)$, $B_4 = B(9, 2)$, and $B_5 = B(8, 0)$, $\Pi_G = [13, 11, 14, 8, 10, 9, 15, 12]$, $\Pi_P = [6, 3, 2, 7, 1, 0, 5, 4]$. Since $k = 5 \geq p/4 = 4$, we go to Step 5. First we perform a sequence of bit exchanges on bits not in B_1 through B_5; i.e., $B(14, 5)$, $B(12, 3)$, and $B(11, 1)$. Recomputing Π_G and Π_P, we get $\Pi_G = [6, 2, 3, 1, 5, 7, 0, 4]$ and $\Pi_P = [13, 14, 11, 9, 8, 15, 10, 12]$. Next, Steps 3.1 through 3.3 are done. The sequence of data moves is shown below:

$$\begin{array}{ll}
(b_{15}b_{14}b_{13}b_{12}b_{11}b_{10}b_9b_8b_7b_6b_5b_4b_3b_2b_1b_0) & \xrightarrow{B(14,5)i} \\
(b_{15}b_5b_{13}b_{12}b_{11}b_{10}b_9b_8b_7b_6b_{14}b_4b_3b_2b_1b_0) & \xrightarrow{B(12,3)} \\
(b_{15}b_5b_{13}b_3b_{11}b_{10}b_9b_8b_7b_6b_{14}b_4b_{12}b_2b_1b_0) & \xrightarrow{B(11,1)} \\
(b_{15}b_5b_{13}b_3b_1b_{10}b_9b_8b_7b_6b_{14}b_4b_{12}b_2b_{11}b_0) & \xrightarrow{\Pi_P} \\
(b_{15}b_5b_{13}b_3b_1b_{10}b_9b_8b_2b_6b_7b_0b_{14}b_{11}b_4b_{12}) & \xrightarrow{O} \\
(b_2b_6b_7b_0b_{14}b_{11}b_4b_{12}b_{15}b_5b_{13}b_3b_1b_{10}b_9b_8) & \xrightarrow{\Pi'_G} \\
(b_2b_6b_7b_0b_{14}b_{11}b_4b_{12}b_{10}b_{15}b_1b_8b_{13}b_5b_3b_9) &
\end{array}$$

It can be verified that the resulting position is exactly the destination that the original BPC permutation A dictates.

The local BPC permutations determined by Π_G and Π_P take at most $4(\sqrt{N} - 1)$ electronic moves each [7]; the bit exchanges cumulatively take at most $8(\sqrt{N} - \sqrt[4]{N})$ electronic and $\log_2 N$ OTIS moves. So, the total number of moves is at most $16\sqrt{N} - 8\sqrt[4]{N} - 8$ electronic and $\log_2 N + 2$ OTIS.

6 Conclusion

We have shown that the diameter of an OTIS-Mesh computer is $4\sqrt{N} - 3$, which is very close to that of a $\sqrt{N} \times \sqrt{N} \times \sqrt{N} \times \sqrt{N}$ 4D mesh computer. We have also shown that of the two intuitively appealing ways to embed a 2D mesh into the OTIS-Mesh, GRM is superior for most applications. With respect to frequently performed permutations, it is possible to obtain better routing algorithms than those obtained by simulating the best 4D mesh algorithms using the simulation scheme of [8]. Finally, we have proposed an algorithm for general BPC permutations.

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), May 1988.

[2] 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, Sept. 1995.

[3] F. Kiamilev, P. Marchand, A. Krishnamoorthy, S. Esener, and S. Lee. Performance comparison between optoelectronic and vlsi multistage interconnection networks. *Journal of Lightwave Technology*, 9(12), Dec. 1991.

[4] A. Krishnamoorthy, P. Marchand, F. Kiamilev, and S. Esener. Grain-size considerations for optoelectronic multistage interconnection networks. *Applied Optics*, 31(26), Sept. 1992.

[5] M. Kunde. Routing and sorting on mesh-connected arrays. In *Proceedings of the 3rd Agean Workshop on Computing: VLSI Algorithms and Architectures, Lecture Notes on Computer Science*, volume 319, pages 423–433. Springer Verlag, 1988.

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

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

[8] F. Zane, P. Marchand, R. Paturi, and S. 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.