BPC Permutations On The OTIS-Hypercube Optoelectronic Computer[1]

Sartaj Sahni and Chih-fang Wang
Department of Computer and Information Science and Engineering
University of Florida, Gainesville, FL 32611
Phone: 352-392-1527, Fax: 352-392-1220
{sahni,wang}@cise.ufl.edu

Keywords: OTIS-Hypercube, BPC permutations, diameter

Edited by:

Received: October 30, 1997 **Revised:** January 20, 1998 **Accepted:** December 19, 1997

We show that the diameter of an N^2 processor OTIS-Hypercube computer ($N = 2^d$) is $2d + 1$. OTIS-Hypercube 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

Electronic interconnects are superior to optical interconnects when the interconnect distance is up to a few millimeters (Feldman et. al. 1988, Kiamilev et. al. 1991). However, for longer interconnects, optics (and in particular, free space optics) provides power, speed, and bandwidth advantages over electronics. With this in mind, Marsden et. al. (Marsden et. al. 1993), Hendrick et. al. (Hendrick et. al. 1995), and Zane et. al. (Zane et. al. 1996) have proposed a hybrid computer architecture in which the processors are divided into groups; intra-group connects are electronic, and inter-group interconnects are optical. Krishnamoorthy et. al. (Krishnamoorthy et. al. 1992) have demonstrated that bandwidth and power consumption are minimized when the number of processors in a group equals the number of groups. Marsden et. al. (Marsden et. al. 1993) propose a family of optoelectronic architectures in which the number of groups equals the number of processors per group. In this family – the optical transpose interconnection system (OTIS) – the inter-group connects (or optical interconnect) connect processor p of group g to processor g of group p. The intra-group interconnect (or electronic interconnect) can be any of the standard previously studied connection schemes for electronic computers. This strategy gives rise to the OTIS-Mesh, OTIS-Hypercube, OTIS-Perfect shuffle, OTIS-Mesh of trees, and so forth computers.

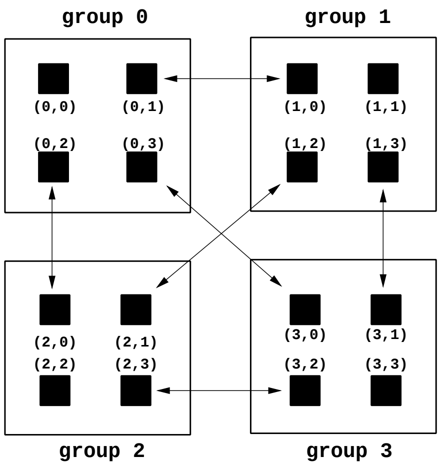

Figure 1: Example of OTIS connections with 16 processors

Figure 1 shows a generic 16 processor OTIS computer; only the optical connections are shown. The solid squares indicate individual processors, and a processor index is given by the pair (G, P) where G is its group index and P the processor or local index. Figure 2 shows a 16 processor OTIS-

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

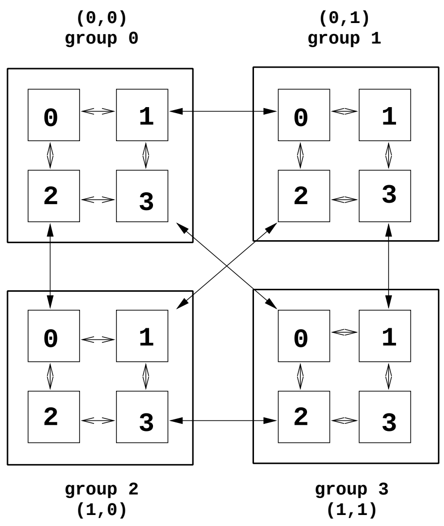

Figure 2: 16 processor OTIS-Hypercube

Hypercube. The number inside a processor is the processor index within its group.

Hendrick et. al. (Hendrick et. al. 1995) have computed the performance characteristics (power, throughput, volume, etc.) of the OTIS-Hypercube architecture. Zane et. al. (Zane et. al. 1996) have 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 one local electronic move and two optical inter group move using the OTIS interconnection. We shall refer the latter as OTIS moves. Sahni and Wang (Sahni & Wang 1997) and Wang and Sahni (Wang & Sahni 1997) have evaluated thoroughly the characteristics of the OTIS-Mesh architecture, developing algorithms for basic data rearrangements..

In this paper, we study the OTIS-Hypercube architecture and obtain basic properties and basic permutation routing algorithms for this architecture. These algorithms can be used to develop efficient application programs.

In the following, when we describe a path through an OTIS-Hypercube, we use the term **electronic move** to refer to a move along an electronic interconnect (so it is an intra-group move) and the **OTIS move** to refer to a move along an optical interconnect.

2 OTIS-Hypercube Diameter

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. 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. Paths of this type look like $(G_1, P_1) \xrightarrow{E} (G_1, P'_1) \xrightarrow{O} (P'_1, G_1) \xrightarrow{E} (P'_1, G'_1) \xrightarrow{O} (G'_1, P'_1) \xrightarrow{E} (G'_1, P''_1) \xrightarrow{O} (P''_1, G'_1) \xrightarrow{E} (P''_1, G''_1) \xrightarrow{O} \cdots \xrightarrow{E} (G_2, P_2)$.

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 the path into a shorter path that uses 2 OTIS moves only: $(G_1, P_1) \xrightarrow{E} (G_1, P_2) \xrightarrow{O} (P_2, G_1) \xrightarrow{E} (P_2, G_2) \xrightarrow{O} (G_2, P_2)$.

(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 as in (b). The shorter path looks like: $(G_1, P_1) \xrightarrow{E} (G_1, G_2) \xrightarrow{O} (G_2, G_1) \xrightarrow{E} (G_2, P_2)$.

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.

As a result, we obtain the following theorem:

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 $G_1 \neq G_2$.*

Theorem 2 *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 1, 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 $2d+1$. □

3 Common Data Rearrangements

In this section, we concentrate on the realization of permutations such as transpose, perfect shuffle, unshuffle, vector reversal which are frequently used in applications. Nassimi and Sahni (Nassimi & Sahni 1982) have developed optimal hypercube algorithms for these frequently used permutations. These algorithms may be simulated by an OTIS-Hypercube using the method of (Zane et. al. 1996) to obtain algorithms to realize these data rearrangement patterns on an OTIS-Hypercube. Table 1 gives the number of moves used by the optimal hypercube algorithms; a break down of the number of moves in the group and local dimensions; and the number of electronic and OTIS moves required by the simulation.

We shall obtain OTIS-Hypercube algorithms, for the permutations of Table 1, that require far fewer moves than the simulations of the optimal hypercube algorithms.

As mentioned before, each processor is indexed as (G, P) where G is the group index and P the local index. An index pair (G, P) may be transformed into a singleton index $I = GP$ by concatenating the binary representations of G and P.

The permutations of Table 1 are members of the BPC (bit-permute-complement) class of permutations defined in (Nassimi & Sahni 1982). 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-Hypercube we know that N is a power of two and so the number of bits needed to represent a processor index is $p = \log_2 N^2 = 2\log N = 2d$. A BPC permutation (Nassimi & Sahni 1982) 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-Hypercube, 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 $m_3m_2m_1m_0$ to 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.

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

The transpose operation may be accomplished via a single OTIS move and no electronic moves. The simulation of the optimal hypercube algorithm, however, takes $2d$ OTIS and $2d$ electronic moves.

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

We can adapt the strategy of (Nassimi & Sahni 1982) to an OTIS-Hypercube. Each processor uses two variables A and B. Initially, all data are in the A variables and the B variables have no data. The algorithm for perfect shuffle is given below:

```
Step 1: Swap A and B in processors with last
        two bits equal to 01 or 10.

Step 2:for ($i = 1; i \le d-1; i++$) {
    (a)  Swap the B variables of processors
         that differ on bit $i$ only;
    (b)  Swap the A and B variables of
         processors with bit $i$ of their
         index I not equal to bit $i+1$ of
         their index; }
```

Permutation	Optimal Hypercube Moves			OTIS-Hypercube Simulation	
	total	group dimension	local dimension	OTIS	electronic
Transpose	$2d$	d	d	$2d$	$2d$
Perfect Shuffle	$2d$	d	d	$2d$	$2d$
Unshuffle	$2d$	d	d	$2d$	$2d$
Bit Reversal	$2d$	d	d	$2d$	$2d$
Vector Reversal	$2d$	d	d	$2d$	$2d$
Bit Shuffle	$2d-2$	$d-1$	$d-1$	$2d-2$	$2d-2$
Shuffled Row-major	$2d-2$	$d-1$	$d-1$	$2d-2$	$2d-2$
G_lP_u Swap	d	$d/2$	$d/2$	d	d

Table 1: Optimal moves for $N^2 = 2^{2d}$ processor hypercube and respective OTIS-Hypercube simulations

Step 3: Perform an OTIS move on the A and B variables.

```
Step 4:for ($i = 0; i \le d-1; i++$) {
    (a)  Swap the $B$ variables of processors
         that differ on bit $i$ only;
    (b)  Swap the $A$ and $B$ variables of
         processors with bit $i$ of their
         index $I$ not equal to bit $i+1$ of
         their index; }
```

Step 5: Perform an OTIS move on the A and B variables.

Step 6: Swap the B variables of processors that differ on bit 0 only.

Step 7: Swap the A and B variables of processors with last two bits equal to 01 or 10.

Actually, in Step 1 it is sufficient to copy from A to B, and in Step 7 to copy from B to A.

Table 4 shows the working of this algorithm on a 16 processor OTIS-Hypercube. The correctness of the algorithm is easily established, and we see that the number of data move step is $2d+2$ ($2d$ electronic moves and 2 OTIS moves; each OTIS move moves two pieces of data from one processor to another, each electronic swap moves a single data between two processors).

The communication complexity of $2d+2$ is very close to optimal. For example, data from the processor with index $I = 0101\ldots0101$ is to move to the processor with index $I' = 1010\ldots1010$ and the distance between these two processors is $2d+1$.

Notice that the simulation of the optimal hypercube algorithm for perfect shuffle takes $4d$ moves.

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

This is the inverse of a perfect shuffle and may be performed by running the perfect shuffle algorithm mentioned above backwards (*i.e.*, beginning with Step 7); the for loops of Steps 2 and 4 are also run backwards. Thus the number of moves is the same as for a perfect shuffle.

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

When simulating the optimal hypercube algorithm, the task requires $2d$ electronic moves and $2d$ OTIS moves. But with the following algorithm:

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.

we can actually achieve the rearrangement in $2d$ electronic moves and 1 OTIS move, since Steps 1 and 3 can be performed optimally in d electronic moves each (Nassimi & Sahni 1982).

The number of moves is optimal since the data from processor $0101\ldots0101$ is to move to processor $1010\ldots1010$, and the distance between these two processors is $2d+1$ (Theorem 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-Hypercube

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

Table 3: Permutations and their permutation vectors

		Step 1	Step 2		OTIS	Step 4				OTIS	Step 6	Step 7	
index	initial		$i = 1$			$i = 0$		$i = 1$					variable
			(a)	(b)		(a)	(b)	(a)	(b)				
0000	0	0	0	0	0	0	0	0	0	0	0	0	A
	-	-	2	2	2	4	4	8	8	8	-	-	B
0001	1	-	-	-	6	6	2	2	2	-	-	8	A
	-	1	-	-	4	2	6	10	10	-	8	-	B
0010	2	-	-	-	8	8	12	12	4	-	-	1	A
	-	2	-	-	10	12	8	4	12	-	1	-	B
0011	3	3	3	1	14	14	14	14	6	9	9	9	A
	-	-	1	3	12	10	10	6	14	1	-	-	B
0100	4	4	4	6	-	-	-	-	-	2	2	2	A
	-	-	6	4	-	-	-	-	-	10	-	-	B
0101	5	-	-	-	-	-	-	-	-	-	-	10	A
	-	5	-	-	-	-	-	-	-	-	10	-	B
0110	6	-	-	-	-	-	-	-	-	-	-	3	A
	-	6	-	-	-	-	-	-	-	-	3	-	B
0111	7	7	7	7	-	-	-	-	-	11	11	11	A
	-	-	5	5	-	-	-	-	-	3	-	-	B
1000	8	8	8	8	-	-	-	-	-	4	4	4	A
	-	-	10	10	-	-	-	-	-	12	-	-	B
1001	9	-	-	-	-	-	-	-	-	-	-	12	A
	-	9	-	-	-	-	-	-	-	-	12	-	B
1010	10	-	-	-	-	-	-	-	-	-	-	5	A
	-	10	-	-	-	-	-	-	-	-	5	-	B
1011	11	11	11	9	-	-	-	-	-	13	13	13	A
	-	-	9	11	-	-	-	-	-	5	-	-	B
1100	12	12	12	14	1	1	1	1	9	6	6	6	A
	-	-	14	12	3	5	5	9	1	14	-	-	B
1101	13	-	-	-	7	7	3	3	11	-	-	14	A
	-	13	-	-	5	3	7	11	3	-	14	-	B
1110	14	-	-	-	9	9	13	13	13	-	-	7	A
	-	14	-	-	11	13	9	5	5	-	7	-	B
1111	15	15	15	15	15	15	15	15	15	15	15	15	A
	-	-	13	13	13	11	11	7	7	7	-	-	B

Table 4: Illustration of the perfect shuffle algorithm on a 16 processor OTIS-Hypercube

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

A vector reversal can be done using $2d$ electronic and 2 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.

The correctness of the algorithm is obvious. The number of moves is computed using the fact that Steps 1 and 3 can be done in d electronic moves each (Nassimi & Sahni 1982).

Since a vector reversal requires us to move data from processor $00\ldots00$ to processor $11\ldots11$, and since the distance between these two processors is $2d+1$ (Theorem 1), our vector reversal algorithm can be improved by at most one move.

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

Let $G = G_uG_l$ where G_u and G_l partition G in half. Same for $P = P_uP_l$. Our algorithm employs a G_lP_u Swap permutation in which data from processor $G_uG_lP_uP_l$ is routed to processor $G_uP_uG_lP_l$. So we need to first look at how this permutation is performed.

3.6.1 G_lP_u Swap $[p-1,\ldots,3p/4, p/2-1,\ldots,p/4, 3p/4-1,\ldots,p/2, p/4-1,\ldots,0]$

The swap is performed by a series of bit exchanges of the form $B(i) = [B_{p-1}, \ldots, B_0]$, $0 \le i < p/4$, where

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

Let $G(i)$ and $P(i)$ denote the ith bit of G and P respectively. So $G(0)$ is the least significant bit in G, and $P(d)$ is the most significant bit in P. The bit exchange $B(i)$ may be accomplished as below:

Step 1: Every processor (G, P) with $G(i) \neq P(d/2+i)$ moves its data to the processor (G, P') where P' differs from P only in bit $d/2+i$.

Step 2: Perform an OTIS move on the data moved in Step 1.

Step 3: Processors (G, P) that receive data in Step 2 move the received data to (G, P'), where P' differs from P only in bit i.

Step 4: Perform an OTIS move on the data moved in Step 3.

The cost is 2 electronic moves and 2 OTIS moves.

To perform a G_lP_u Swap permutation, we simply do $B(i)$ for $0 \le i < d/2$. This takes d electronic moves and d OTIS moves. By doing pairs of bit exchanges $(B(0), B(1))$, $(B(2), B(3))$, etc. together, we can reduce the number of OTIS moves to $d/2$.

3.6.2 Bit Shuffle

A bit shuffle, now, can be performed following these steps:

Step 1: Perform a G_lP_u 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.

Steps 2 and 4 are done using the optimal d move hypercube bit shuffle algorithm of (Nassimi & Sahni 1982). The total number of data moves is $3d$ electronic moves and $d/2+2$ OTIS moves.

3.7 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.

4 BPC Permutations

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

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

and a BPC permutation $C = [C_{2d-1}, \ldots, C_0] = \Pi_G \Pi_P$ where $|C_q| < d$, $0 \le q < d$, Π_G and Π_P involve d bits each.

For example, the transpose permutation may be realized by the sequence $B(d+j, j)$, $0 \le j < d$; bit reversal is equivalent to the sequence $B(2d-1-j, j)$, $0 \le j < d$; vector reversal can be realized by performing no bit exchanges and using $C = [-(2d-1), -(2d-2), \ldots, -0]$ ($\Pi_G = [-(2d-1), -(2d-2), \ldots, -d]$, $\Pi_P = [-(d-1), \ldots, -0]$); and perfect shuffle may be decomposed into $B(d, 0)$ and $C = [2d-2, 2d-3, \ldots, d, 2d-1, d-2, \ldots, 1, 0, d-1]$ ($\Pi_G = [2d-2, 2d-3, \ldots, d, 2d-1]$, $\Pi_P = [d-2, \ldots, 1, 0, d-1]$).

A bit exchange permutation $B(i,j)$ can be performed in 2 electronic moves and 2 OTIS moves using a process similar to that used for the bit exchange permutation $B(i)$. Notice that $B(i) = B(i,i)$.

Our algorithm for general BPC permutations is:

Step 1: Decompose the BPC permutation A into the pair cycle moves $B_1(i_1, j_1)$, $B_2(i_2, j_2)$,..., $B_k(i_k, j_k)$ 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 (Nassimi & Sahni 1982).
- **Step 2.2:** Do an OTIS move.
- **Step 2.3:** Do the BPC permutation Π'_G in each group using the algorithm of (Nassimi & Sahni 1982).
- **Step 2.4:** Do an OTIS move.

Step 3: If $k = d$, 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 $k < d/2$, 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 $k \ge d/2$, do the following:

- **Step 5.1:** Perform a sequence of $d-k$ 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.

The local BPC permutations determined by Π_G and Π_P take at most d electronic moves each (Nassimi & Sahni 1982); and the bit exchange permutations take at most d electronic moves and $d/2$ OTIS moves. So the total number of moves is at most $3d$ electronic moves and $d/2 + 2$ OTIS moves.

5 Conclusion

In this paper we have shown that the diameter of the OTIS-Hypercube is $2d+1$, which is very close to that of an N^2 processor hypercube. However, each OTIS-Hypercube processor is connected to at most $d+1$ other processors; while in an N^2 processor hypercube, a processor is connected to up to $2d$ other processors. We have also developed algorithms for frequently used data permutations. Table 5 compares the performance of our algorithms and those obtained by simulating the optimal hypercube algorithms using the simulation technique of (Zane et. al. 1996). For most of the permutations considered, our algorithms are either optimal or within one move of being optimal. An algorithm for general BPC permutations has also been proposed.

Permutation	Simulation		Our Algorithm	
	OTIS	electronic	OTIS	electronic
Transpose	$2d$	$2d$	1	0
Perfect Shuffle	$2d$	$2d$	2	$2d$
Unshuffle	$2d$	$2d$	2	$2d$
Bit Reversal	$2d$	$2d$	1	$2d$
Vector Reversal	$2d$	$2d$	2	$2d$
Bit Shuffle	$2d-2$	$2d-2$	$d/2+2$	$3d$
Shuffled Row-major	$2d-2$	$2d-2$	$d/2+2$	$3d$
G_lP_u Swap	d	d	$d/2$	d

Table 5: Performance Comparisons

References

[1] Feldman M., Esener S., Guest C., & S. Lee (1988) Comparison Between Electrical and Free-Space Optical Interconnects Based on Power and Speed Considerations. *Applied Optics*, 27, 9, p. 1742-1751.

[2] Hendrick W., Kibar O., Marchand P., Fan C., Blerkom D. V., McCormick F., Cokgor I., Hansen M., & Esener S. (1995) Modeling and Optimization of the Optical Transpose Interconnection System. Optoelectronic Technology Center, Program Review, Cornell University.

[3] Kiamilev F., Marchand P., Krishnamoorthy A., Esener S., & Lee S. (1991) Performance Comparison Between Optoelectronic and VLSI Multistage Interconnection Networks. *Journal of Lightwave Technology*, 9, 12, p. 1674-1692.

[4] Krishnamoorthy A., Marchand P., Kiamilev F., & Esener S. (1992) Grain-Size Considerations for Optoelectronic Multistage Interconnection Networks. *Applied Optics*, 31, 26, p. 5480-5507.

[5] Marsden G. C., Marchand P. J., Harvey P., & Esener S. C. (1993) Optical Transpose Interconnection System Architectures. *Optics Letters*, 18, 13, p. 1083-1085.

[6] Nassimi D. & Sahni S. (1982) Optimal BPC Permutations On A Cube Connected Computer. *IEEE Transactions on Computers*, C-31, 4, p. 338-341.

[7] Sahni S. & Wang C.-F. (1997) BPC Permutations On The OTIS-Mesh Optoelectronic Computer. *Proceedings of the fourth International Conference on Massively Parallel Processing Using Optical Interconnections (MPPOI'97)*, p. 130-135.

[8] Wang C.-F. & Sahni S. (1997) Basic Operations on the OTIS-Mesh Optoelectronic Computer. *Technical Report 97-008*, CISE Department, University of Florida, available by anonymous ftp login from ftp.cise.ufl.edu under directory tech-report/tr97/tr97-008.ps.gz.

[9] Zane F., Marchand P., Paturi R., & Esener S. (1996) Scalable Network Architectures Using the Optical Transpose Interconnection System (OTIS). *Proceedings of the second International Conference on Massively Parallel Processing Using Optical Interconnections (MPPOI'96)*, p. 114-121.