ELSEVIER

Information Processing Letters 51 (1994) 237–243

Information Processing Letters

Realization of an arbitrary permutation on a hypercube

Xiaojun Shen *, Qing Hu, Weifa Liang

Computer Science Telecommunications Program, University of Missouri – Kansas City, 5100 Rockhill Road, Kansas City, MO 64110, USA

Communicated by D. Gries; received 15 October 1993; revised 28 April 1994

Abstract

We present an explicit combinatorial algorithm for constructing a 2-realization for any given permutation on a circuit-switched d-dimensional hypercube (d-cube) such that the total number of directed edges used in the realization (counting every repetition) is bounded by $d2^d$, the total number of directed edges in the d-cube. As a corollary, this result implies a $(2d-3)$ step realization on a packet-switched d-cube ($d \geqslant 3$).

Key words: Circuit-switched network; Hypercube; Interconnection network; Permutation capability; Algorithms; Computer architecture

1. Introduction

The processors in a multi-processor system are connected by *an interconnection network* (or *network* for short). A static network can be represented by a directed graph $G=(V, E)$, where $V=\{0, 1, 2,\dots, n-1\}$ represents n processors and a directed edge (v, w) in E represents a communication link from v to w. G is assumed to be strongly connected, so that at least one path exists from any processor to any other processor. The communication requirement is usually represented by an n-permutation π that specifies a distinct destination vertex $\pi(v)$ for each $v \in V$. Realizing a permutation is to find n paths, $\{path(i, \pi(i)) \mid 0 \leqslant i < n\}$, connecting each vertex i to its destination $\pi(i)$. A permutation π is called *k-realizable* if a realization exists such that any edge in the network is in at most k paths. A 1-realization corresponds to a set of n edge-disjoint paths. A network is called *rearrangeable* if all $n!$ different permutations are 1-realizable [2,6].

A d-dimensional hypercube (d-cube) H^d contains $n=2^d$ vertices such that there is a pair of opposite directed edges between two vertices if and only if their binary representations differ in exactly one bit position. An edge (u, v) is called an ith-dimensional edge if u and v differ in the ith bit, i.e., $u=u_d u_{d-1} \dots u_i \dots u_2 u_1$, and $v=u_d u_{d-1} \dots \bar{u}_i \dots u_2 u_1$. To determine the rearrangeability of a d-cube is an interesting but difficult problem. It has been shown in [9] that any permutation on a 2-*cube* (or a 3-cube) is 1-realizable, with every path being the shortest. It was conjectured in [9] that any permutation is

* Corresponding author.

SSDI 0020-0190(94)00100-D

1-realizable, with every path being the shortest on any d-cube. Lubiw [5] gave a counter-example to the conjecture. However, it is still unknown whether or not a d-cube is rearrangeable if the paths are not required to be the shortest. On the other hand, the *2-realizability* of any permutation has been known [5]. However, the previous algorithm is not explicit.

This paper presents an explicit algorithm for constructing a 2-realization of an arbitrary permutation on a d-cube such that the total number of edges used in the realization (counting every repetition) is bounded by $d2^d$, the total number of edges in the d-cube. In other words, any edge in the d-cube is, on average, used at most once (reaching the same average number as that in the original conjecture [9]). As a corollary, our result implies that any permutation is realizable on a packed-switched d-cube [9] within $2d-3$ $(d \geqslant 3)$ steps, a slight improvement over the previous result, $2d-1$.

It is easy to see that removing all ith-dimensional edges will break a d-cube into two $(d-1)$-cubes. For a given n-permutation π, $n=2^d$, our realization algorithm applies the following divide-and-conquer strategy: (i) break the d-cube into two $(d-1)$-cubes by removing the dth-dimensional edges, and compute two m-permutations, π_1 and π_2 from π, where $m=n/2=2^{d-1}$; (ii) 2-realize π_1 and π_2 on H_1 and H_2 recursively; (iii) use the dth-dimensional edges to combine the two realizations on H_1 and H_2 into a 2-realization of π on the original d-cube.

2. The divide stage

First, we break the d-cube into two $(d-1)$-cubes, H_1 and H_2, by removing the dth-dimensional edges. Therefore, H_1 contains the vertex set $V_1=\{0, 1, \ldots, m-1\}$ and H_2 contains the vertex set $V_2=\{m, m+1, \ldots, 2m-1\}$, where $m=n/2=2^{d-1}$. It is clear that, before the partition, there was a pair of dth-dimensional edges between vertex i in H_1 and vertex $i+m$ in H_2. Let $i'=i+m$. Then, there was a pair of dth-dimensional edges between vertex i and vertex i'.

Second, we show how to compute two m-permutations from n-permutation π, which will be the m-permutations on H_1 and H_2. Permutation π can be viewed as an ordered sequence of n numbers $\pi(0), \pi(1), \ldots, \pi(n-1)$, $0 \leqslant \pi(i) < n$, and $\pi(i) \neq \pi(j)$ if $i \neq j$. Let $M=\{0, 1, \ldots, m-1\}$. We construct two sequences ψ_1 and ψ_2 of m numbers as follows.

For $i \in M$, $0 \leqslant i < m$,

$$\psi_1(i)=\begin{cases}\pi(i) & \text{if } \pi(i)<m,\\ \pi(i)-m & \text{if } \pi(i) \geqslant m,\end{cases}$$

and

$$\psi_2(i)=\begin{cases}\pi(i+m) & \text{if } \pi(i+m)<m,\\ \pi(i+m)-m & \text{if } \pi(i+m) \geqslant m.\end{cases} \tag{2.1}$$

Observation 1. ψ_1 and ψ_2 may not be permutations, since $i \neq j$ does not imply $\psi_1(i) \neq \psi_1(j)$ and $\psi_2(i) \neq \psi_2(j)$.

Observation 2. Each number in M occurs exactly twice among ψ_1 and ψ_2.

We will transform ψ_1 and ψ_2 into two m-permutations by exchanging some numbers between ψ_1 and ψ_2. Specifically, we will find a subset $I \subseteq M$ such that the two sequences π_1 and π_2 defined below are m-permutations.

$$\pi_1(i)=\begin{cases}\psi_1(i) & \text{if } i \notin I,\\ \psi_2(i) & \text{if } i \in I,\end{cases}$$

and

$$\pi_2(i)=\begin{cases}\psi_2(i) & \text{if } i \notin I,\\ \psi_1(i) & \text{if } i \in I.\end{cases} \tag{2.2}$$

Third, we show how to determine the set I. Given ψ_1 and ψ_2 defined in (2.1), we define a binary relation S: (i) $(i, i) \in S$; (ii) $(i, j) \in S$ if $\psi_1(i)=\psi_2(j)$ or $\psi_1(j)=\psi_2(i)$, $0 \leqslant i,j < m$. The transitive closure of S induces an equivalent relation on M that partitions M into equivalent classes, denoted by $\{D_1, D_2, \ldots, D_h\}$.

Let $D=\{p_1, p_2, \ldots, p_k\}$ be an equivalence class such that

$$\begin{aligned}&\psi_1(p_2)=\psi_2(p_1),\\&\psi_1(p_3)=\psi_2(p_2),\\&\ldots\\&\psi_1(p_k)=\psi_2(p_{k-1}).\end{aligned} \tag{I}$$

From the observations, we have two cases:

Case 1: $\psi_1(p_1)\neq\psi_2(p_k)$. Because equivalent classes are disjoint, $\psi_1(p_1)$ cannot be equal to any number in sequence ψ_2. The value of $\psi_1(p_1)$ occurs twice in ψ_1. Similarly, the value of $\psi_2(p_k)$ occurs twice in ψ_2. $\psi_1(p_1)$ is called the head of class D, denoted by $hd(D)$, and $\psi_2(p_k)$ is called the tail of class D, denoted by $tl(D)$.

Case 2: $\psi_1(p_1)=\psi_2(p_k)$. In this case, sequence (I) can be extended to a cyclic sequence by adding $\psi_1(p_1)=\psi_2(p_k)$:

$$\begin{aligned}&\psi_1(p_2)=\psi_2(p_1),\\&\psi_1(p_3)=\psi_2(p_2),\\&\ldots\\&\psi_1(p_k)=\psi_2(p_{k-1}),\\&\psi_1(p_1)=\psi_2(p_k).\end{aligned}$$

Again, $\psi_1(p_1)$ and $\psi_2(p_k)$ are specified as the head and tail of class D, respectively. Obviously, $hd(D)=tl(D)$ and they are not unique in this case. They depend on where this cyclic sequence is broken.

Given a class $D=\{p_1, p_2, \ldots, p_k\}$, we define the set *Marked*(D) as follows:

$$\begin{aligned}&Marked(D)\\&\quad=\{\psi_1(p_j)\mid p_j\in D \text{ and } \psi_1(p_j)=\pi(p_j)\}\\&\qquad\cup\{\psi_2(p_j)\mid p_j\in D \text{ and}\\&\qquad\qquad\psi_2(p_j)=\pi(p_j+m)-m\}.\end{aligned}$$

Now, we construct a vertex-weighted undirected graph $G'=(V', E', \omega)$, where $V'=\{v_1, v_2, \ldots, v_h\}$ corresponds to the set of h equivalent classes, $\{D_1, D_2, \ldots, D_h\}$; $E'=\{(v_i, v_j)\mid hd(D_i)=hd(D_j)$ or $tl(D_i)=tl(D_j)\}$; and ω is a weight function such that $\omega(v_i)=|Marked(D_i)|$. It is easy to see that both ψ_1 and ψ_2 are m-permutations iff $hd(D_i)=tl(D_i)$ for every class D_i, i.e., iff $E'=\emptyset$.

Lemma 2.1. *Let $G'=(V', E', \omega)$ be the graph constructed as above. Then, any connected component of G' can only be an isolated vertex, a single edge, or an even cycle.*

Proof. Let v_i be any vertex in a connected component C (say) and let $D_i=\{p_1, p_2, \ldots, p_k\}$ be the corresponding equivalence class with $\psi_1(p_1)$ and $\psi_2(p_k)$ being $hd(D_i)$ and $tl(D_i)$, respectively. If $hd(D_i)=tl(D_i)$ then, from Observation 2, any other class cannot have the same head or tail. Thus, the corresponding v_i is an isolated vertex. If $hd(D_i)\neq tl(D_i)$, then $\psi_1(p_1)$ occurs exactly twice in sequence ψ_1, $\psi_1(p_1)=\psi_1(x)$, where $x\notin D_i$. Therefore, $x\in D_j$, $i\neq j$, and $\psi_1(x)$ is the head of D_j. Thus, $hd(D_i)=hd(D_j)$. Moreover, from Observation 2, no three or more classes can have the same head value; hence, D_j is unique. Similarly, there is a unique class D_k such that $tl(D_i)=tl(D_k)$. Now, if $D_j=D_k$, then we conclude that C is a single edge. If $D_j\neq D_k$, then every vertex in C has degree 2, and C must be a simple cycle. We mark an edge between v_i and v_j with h if $hd(D_i)=hd(D_j)$, with t if $tl(D_i)=tl(D_j)$. Since an edge marked h cannot be adjacent to an edge marked t, this cycle must be an even cycle. □

Let $C=\{v_1, v_2, \ldots, v_{2p-1}, v_{2p}\}$ be a cycle (or an edge if $p=1$) in the weighted graph G' and $D_1, D_2, \ldots, D_{2p-1}, D_{2p}$, the corresponding classes. We define

$$L(C)=\begin{cases}D_1\cup D_3\cup\cdots\cup D_{2p-1}\\ \quad\text{if } \sum_{i=1}^{p}\omega(v_{2i-1})\leqslant\sum_{i=1}^{p}\omega(v_{2i}),\\ D_2\cup D_4\cup\cdots\cup D_{2p}\\ \quad\text{otherwise}.\end{cases} \tag{2.3}$$

It is easy to see that, if we modify sequences ψ_1 and ψ_2 by exchanging the values of $\psi_1(x)$ and $\psi_2(x)$ for every $x\in L(C)$, we will then have the same set of equivalent classes as before except that $D_1, D_2, \ldots D_{2p-1}, D_{2p}$ are combined into a single class $D=D_1\cup D_2\cup\cdots\cup D_{2p}$, and $hd(D)=tl(D)$. This change shrinks cycle C to a single vertex in G'. Thus, applying this operation to each cycle or single edge in graph G', we obtain two m-permutations. This process is summarized by the following procedure.

Procedure *PARTITION*(π):

1. generate ψ_1 and ψ_2 from the permutation π according to (2.1);
2. construct graph G' and identify all single edges and cycles, $C_1, C_2, \ldots, C_t$;
3. compute $I = L(C_1) \cup L(C_2) \cup \cdots \cup L(C_t)$;
4. generate π_1 and π_2 from ψ_1 and ψ_2 by (2.2).

As discussed, the graph G' based on π_1 and π_2 contains no edges, so π_1 and π_2 are two m-permutations. The relationship between π and (π_1, π_2) is determined by (2.1) and (2.2).

3. The conquer stage

In the previous section, we showed how to break a d-cube into two $(d-1)$-cubes and partition an n-permutation π into two m-permutations, where $m = n/2 = 2^{d-1}$. If $m > 8$, we continue to divide each $(d-1)$-cube into even smaller cubes. If $m = 2^3 = 8$, we stop dividing because we know how to 1-realize any permutation on a 3-cube [9]. Suppose the two m-permutations on H_1 and H_2 have been recursively 2-realized. We shall show how to use the edges between H_1 and H_2 to combine these two realizations into a 2-realization of π on the original d-cube H^d. It is clear that between vertex i in H_1 and vertex j in H_2, there is a pair of edges if and only if $j = i'$.

Procedure *CONSTRUCT*;

for $i = 0$ **to** $m - 1$ **do**
if $i \notin I$ **then**
  Construct path $(i, \pi(i))$ according to the following two cases:
    Case 1: $0 \leqslant \pi(i) < m$.
      In this case, $\pi(i) = \pi_1(i)$ by (2.1).
      Set $path(i, \pi(i)) := path_1(i, \pi_1(i))$.
    Case 2: $m \leqslant \pi(i) < n$.
      In this case, $\pi(i) = \pi_1(i) + m = \pi_1(i)'$ by (2.1).
      Set $path(i, \pi(i)) := path_1(i, \pi_1(i)) + e(\pi_1(i), \pi_1(i)')$.
  Construct $path(i+m, \pi(i+m)) = path(i', \pi(i'))$ according to the following two cases:
    Case 3: $0 \leqslant \pi(i') < m$.
      In this case, $\pi(i') = \pi_2(i)$ by (2.1).
      Set $path(i', \pi(i')) := path_2(i', \pi_2(i)') + e(\pi_2(i)', \pi_2(i))$.
    Case 4: $m \leqslant \pi(i') < n$.
      In this case, $\pi(i') = \pi_2(i) + m = \pi_2(i)'$ by (2.1).
      Set $path(i', \pi(i')) := path_2(i', \pi_2(i)')$.
else $\{i \in I\}$
  Construct $path(i, \pi(i))$ according to the following two cases:
    Case 5: $0 \leqslant \pi(i) < m$.
      In this case, $\pi(i) = \pi_2(i)$ by (2.1) and (2.2).
      Set $path(i, \pi(i)) := e(i, i') + path_2(i', \pi_2(i)') + e(\pi_2(i)', \pi_2(i))$.
    Case 6: $m \leqslant \pi(i) < n$.
      In this case, $\pi(i) = \pi_2(i) + m = \pi_2(i)'$ by (2.1) and (2.2).
      Set $path(i, \pi(i)) := e(i, i') + path_2(i', \pi_2(i)')$.
  Construct $path(i+m, \pi(i+m)) = path(i', \pi(i'))$ according to the following two cases:
    Case 7: $0 \leqslant \pi(') < m$.
      In this case, $\pi(i') = \pi_1(i)$ by (2.1) and (2.2).
      Set $path(i', \pi(i')) := e(i', i) + path_1(i, \pi_1(i))$.
    Case 8: $m \leqslant \pi(i') < n$.
      In this case, $\pi(i') = \pi_1(i) + m = \pi_1(i)'$ by (2.1) and (2.2).
      Set $path(i', \pi(i')) := e(i', i) + path_1(i, \pi_1(i)) + e(\pi_1(i), \pi_1(i)')$.
return $\{path(i, \pi(i)) \mid 0 \leqslant i < n\}$.

Fig. 1.

This pair of edges is denoted by $e(i, i')$ and $e(i', i)$.

Let π_1 and π_2 be the two m-permutations produced by *PARTITION*(π). The 2-realization of π_1 on H_1 is denoted by R_1, and the 2-realization of π_2 on H_2 is denoted by R_2. Let $R_1 = \{path_1(i, \pi_1(i)) \mid 0 \leqslant i < m\}$, and $R_2 = \{path_2(i', \pi_2(i)') \mid 0 \leqslant i < m\}$, where $i' = i + m$, and $\pi_2(i)' = \pi_2(i')$. The 2-realization of π is constructed by the procedure in Fig. 1.

We now present the main algorithm as follows:

Algorithm *REALIZE*(π, H^d);

if $d \leqslant 3$ **then**
  use the algorithm in [9] to produce a 1-realization R
else
  generate π_1 and π_2 by *PARTITION*(π);
  $R_1 := REALIZE(\pi_1, H_1)$;
  $R_2 := REALIZE(\pi_2, H_2)$;
  $R := CONSTRUCT$;
output $\{R\}$.

We shall prove that the realization R constructed by *REALIZE*(π, H^d) is indeed a 2-realization.

Lemma 3.1. *Any path in $R_1 \cup R_2$ is used exactly once in the construction of R.*

Proof. It is obvious that every path in R is obtained by extending one path in $R_1 \cup R_2$. We will prove that any path in $R_1 \cup R_2$ is used at most once by R. Without loss of generality, consider $path_1(i, \pi_1(i))$, where $0 \leqslant i < m$. Note that only $path(i, \pi(i))$ and $path(i', \pi(i'))$ may use $path_1(i, \pi_1(i))$. We observe that $path_1(i, \pi_1(i))$ may be used by $path(i, \pi(i))$ in cases 1 or 2 or by $path(i', \pi(i'))$ in cases 7 or 8. Since cases 1, 2, 7, and 8 are mutually exclusive, it is impossible for $path_1(i, \pi_1(i))$ to be used twice by R. The same conclusion can be drawn for any $path_1(i, \pi_1(i))$, where $m \leqslant i < n$. By symmetry, it is true for any $path_2(i', \pi_2(i'))$ also. □

Lemma 3.2. *Each dth-dimensional edge is used by at most two paths in R.*

Proof. By symmetry, we need consider only an arbitrary dth-dimensional edge $e(v, v')$. According to Procedure *CONSTRUCT*, only cases 2, 5, 6, 8 may need to use $e(v, v')$. Obviously, cases 5 and 6 are mutually exclusive. Cases 2 and 8 are also mutually exclusive because both of them require that $\pi_1(i) = v$ for some i, but case 2 requires $i \notin I$ and case 8 requires $i \in I$. Therefore, $e(v, v')$ is used by at most two paths in R. □

In order to bound the total number of edges used in R, we need the following lemma.

Lemma 3.3. *The total number of dth-dimensional edges used by R is bounded by $n = 2m$.*

Proof. Let $path(j, \pi(j))$ be any path constructed by Procedure *CONSTRUCT*. Let us list the number of dth-dimensional edges used by $path(j, \pi(j))$ according to different cases.

Case 1 or case 4:	zero.
Cases 2, 3, 6, or 7:	one.
Case 5 or case 8:	two.

If we can prove that the number of paths in case 5 or case 8 is not larger than the number of paths in case 1 or case 4, then we are done.

Notice that $path(j, \pi(j))$ belongs to case 5 or case 8 if and only if
(i) $j < m$, $j \in I$, and $\psi_1(j) = \pi(j)$, or
(ii) $j \geqslant m$, $j - m \in I$, and $\psi_2(j - m) = \pi(j) - m$.

Therefore, the number of paths in case 5 or case 8 is $\sum_{D \subset I}^{|marked(D)|}$.

On the other hand, $path(j, \pi(j))$ belongs to case 1 or case 4 if and only if
(i) $j < m$, $j \notin I$, and $\psi_1(j) = \pi(j)$, or
(ii) $j \geqslant m$, $j - m \notin I$, and $\psi_2(j - m) = \pi(j) - m$.

Thus, the number of paths in case 1 or case 4 is $\sum_{D \not\subset I} |marked(D)|$. From (2.3), we have

$$\sum_{D \subset I} |marked(D)| \leqslant \sum_{D \not\subset I} |marked(D)|. \quad \square$$

Theorem 3.4. *Algorithm REALIZE(π, H^d) produces a 2-realization R for any permutation π on H^d such that:*
(i) *at most $d2^d$ edges are used in R,*

(ii) *the length of every path is bounded by* $2d-3$ $(d \geqslant 3)$.

Proof. We prove this theorem by induction on d. Theorem 2 in [9] can serve as the inductive basis that realizes any permutation with the shortest path on a hypercube H^d $(d \leqslant 3)$. By the inductive assumption, $REALIZE(\pi_1, H_1)$ and $REALIZE(\pi_2, H_2)$ produce a 2-realization R_1 of π_1 and a 2-realization R_2 of π_2. According to Lemma 3.1, each edge within H_1 or H_2 will be used at most twice in R. Besides, by Lemma 3.2, each dth-dimensional edge is also used at most twice in R. Thus, R is a 2-realization. Moreover, by the inductive assumption, R_1 and R_2 have used at most $(d-1)2^{d-1}$ edges each. From Lemma 3.3, the number of dth-dimensional edges (counting every repetition) is bounded by $n = 2^d$. Thus, the total number of edges contained in R is bounded by $2(d-1)2^{d-1} + 2^d = d2^d$. The bound on lengths of paths can also be proved by a simple induction. □

Let $T(n)$ be the time complexity of $REALIZE(\pi, H^d)$, $n = 2^d$. It is not difficult to see that $T(n) = O(n \lg n) = O(d2^d)$, because obtaining sequences ψ_1 and ψ_2, computing the equivalent classes, constructing graph $G' = (V', E', \omega)$, and finding set I need only linear time each. (We may need to compute the inverse mappings of π, ψ_1 and ψ_1, which can also be done in linear time.) In addition, Procedure *CONSTRUCT* needs linear time also. Thus, $T(n) = 2T(n/2) + O(n)$, which implies $T(n) = O(n \lg n) = O(d2^d)$.

Now we introduce a corollary on the *packet-switched hypercube*. In a packet-switched hypercube, at each synchronous step, a processor may receive a packet from each of its neighbors through an incoming edge, and/or send a packet to a neighbor through an outgoing edge. Moreover, we do not allow a packet to be buffered in any intermediate processor. An interesting problem occurs in routing packets such that every packet will reach its destination (defined by a permutation π) in a minimum number of steps. We assume that the packet generated by processor i is labeled with pair $(i, \pi(i))$. We use $x \to y$ denote that vertex x sends a packet to vertex y. Currently, the best known result for this problem is $2d-1$, although it is conjectured that d steps are enough [9].

Corollary 3.5. *Any permutation* π *can be realized on the packet-switched* H^d $(d \geqslant 3)$ *within* $2d-3$ *steps.*

Proof. We realize an arbitrary π on a packet-switched H^d by the following algorithm:

If $d = 3$ then realize π in 3 steps by the algorithm provided in [9].
If $d > 3$, do the following:
(1) generate π_1 and π_2 by $PARTITION(\pi)$;
(2) in the first synchronous step, do:
    for each $0 \leqslant i < m$,
    if $i \in I$, then $i \to i'$ and $i' \to i$;
    (Note that every processor still holds a unique packet.)
(3) realize π_1 and π_2 on H_1 and H_2 recursively, which costs $2d-5$ steps without using any edges between H_1 and H_2;
(4) in the last synchronous step, do:
    check every pair $(i, \pi(i))$,
    if it belongs to cases 2 or 8, then $\pi(i) \to \pi(i)'$;
    if it belongs to cases 3 or 5, then $\pi(i)' \to \pi(i)$.

The correctness of this algorithm is evident from the discussion in Sections 2 and 3. □

Acknowledgement

The authors thank David Gries for his helpful comments on the writing.

References

[1] S. Abraham and K. Padmanabhan, Performance of the direct binary n-cube network for multiprocessors, in: *Proc. 1986 Internat. Conf. on Parallel Processing*, pp. 636–639.

[2] E. Choi et al., Hyperswitch network for the hypercube computer, in: *Proc. Ann. Symp. on Computer Architecture* (1988) 90–99.

[3] T.-Y. Feng, A survey of interconnection networks, *IEEE Comput*. December (1981) 12–27.
[4] W.D. Hillis, *The Connection Machine* (MIT Press, Cambridge, MA, 1985).
[5] A. Lubiw, A counter-example to a conjecture by Szymanski on hypercube routing, *Inform. Process. Lett*. **35** (1990) 57–61.
[6] G.M. Masson, G.C. Ginger and S. Makamura, A sampler of circuit switching networks, *IEEE Comput*. June (1979) 32–48.
[7] H.J. Siegel, The multistage cube: A versatile interconnection network, *IEEE Comput*. December (1981) 65–76.
[8] H.J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing* (McGraw-Hill, New York, 1990).
[9] T. Szymanski, On the permutation capability of a circuit-switched hypercube, in: *Proc*. 1989 *Internat. Conf. on Parallel Processing*, Vol. I, pp. 103–110.
[10] T. Szymanski, A fiber-optic hypermesh for SIMD/MIMD machines, *IEEE & ACM Supercomputing* '90 (1990) 710–719.