AN ABSTRACT OF THE THESIS FOR

Shawn M. Larson for the degree of Master of Science in Computer Science
presented on November 26, 1990.

Title: The Möbius Cube: An Interconnection Network for Parallel Computation

Redacted for privacy

Abstract Approved: _____

The binary hypercube has many properties that make it attractive as an interconnection network for parallel computation: expansibility, symmetry, and a diameter that is logarithmic to the number of processors. However, the hypercube is not the minimum-diameter network possible; many other networks exist that have a smaller diameter for exactly the same computation resources. In this paper, we show that by sacrificing symmetry, we can create two interconnection networks - the "0-Möbius" and "1-Möbius" cubes. If the dimension of these Möbius cubes is n, then the networks have a diameter of $\lceil (n+2)/2 \rceil$ and $\lceil (n+1)/2 \rceil$ respectively, for $n \geq 4$, roughly half the diameter of the hypercube, and an expected routing distance of approximately $n/3 - (1/9)[1-(-1/2)^n]$.

We give a simple routing algorithm for both cubes and show that the algorithm runs in time proportional to n. We compare the Möbius cubes to the "Twisted cube" of *Hilbers, et. al.*. We show the vertex and edge asymmetry of the Möbius cube, we map other network topologies to the Möbius cube, and we show that the hypercube can be simulated on the Möbius cube with a communications overhead proportional to n. Finally, we briefly examine general lower bounds on any hypercube variant's diameters.

The "Möbius Cube": An Interconnection Network for Parallel Computation

by

Shawn M. Larson

A THESIS

Submitted to

Oregon State University

In partial fulfillment of
the requirements for the
degree of

Master of Science

Completed November 26, 1990

Commencement June 1991

APPROVED:

Redacted for privacy

Professor of Computer Science in charge of major

Redacted for privacy

Head of Department of Computer Science

Redacted for privacy

Dean of Graduate School

Date Thesis is presented: November 26, 1990

Typed by researcher for ___

Table of Contents

List of Figures

The "Möbius Cube": An Interconnection Network for Parallel Computation

1. Introduction

Parallel computation systems, to have processors work on a problem cooperatively, need a mechanism for exchanging data. Interconnection networks are one method to meet this need. In an interconnection network, each processor has its own memory and resources, and is connected to a number of neighboring processors by communication paths. The processors can work cooperatively by passing messages via the communication paths. The study of message-passing parallel computers has resulted in a rapidly growing field of study - namely, the design of efficient interconnection networks (Feng 1981). The topology of a network determines how well the processors can interact with each other to solve a given problem.

There are many different and often conflicting performance considerations in choosing a particular interconnection network for a parallel computer. These considerations include: a small network diameter; a limited number of connections per processor; uniformity or symmetry of processor connections; expansibility; an efficient processor layout; a simple message routing algorithm between processors; and many others (Hillis 1982). The hypercube network has proved to be one of the most popular interconnection networks (used in both Intel's and NCUBE's computers, among others), partly because it has a relatively small diameter and because the number of connections per processor is logarithmic to the number of processors - hence it is a logarithmic network topology.

A small diameter is often one of the most important considerations in choosing a network topology. If the communication time between processors dominates the computation time, then reducing the diameter in a network should improve the performance of any machine using that network.

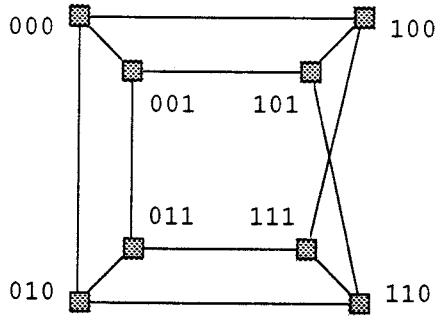

Figure 1. The "Twisted" 3-cube.

The hypercube does not have the smallest diameter possible for a logarithmic network topology. For example, if we exchange or "twist" the endpoints of two edges of the 3-cube as in Figure 1, the diameter of the network reduces to 2. This is the minimum diameter that can be obtained; more twists will not reduce it further.

The first published record of this smaller diameter "cube" occurs in Seitz 19(84), which attributes it to Hillis (1981) in relation to designing connection machine topologies; in these references the twisted 3-cube is given but there is no indication of how to generalize this construction to larger dimension cubes. Hilbers et. al. (1987) give a generalized architecture definition with a maximum routing distance of $\lceil n/2 \rceil+1$, where n is odd, and give an optimal routing algorithm. They also described how to generalize their topology to even n. Abraham and Padmanabhan (1989) and Abraham (1990) compute the expected distance of the twisted cube and compare the dynamic performance of the twisted cube and the hypercubes. Their tests show that the twisted cube shows a somewhat less efficient dynamic performance under heavy loads, possibly because of bottlenecks caused by the twisted cube's asymmetries. They conclude that the twisted cube has better performance than the hypercube, but not quite what would be expected from the $n/2$ diameter.

In this paper, we produce our own generalizations of the topology in figure 1. We call our topologies the Möbius cubes - largely because the flattened 3-cube resembles the outline of a Möbius strip.

This paper covers the following topics. In section 2, we give an introduction to the 0-Möbius and 1-Möbius cubes; we give the connection rules between processors in the networks, and demonstrate the network's expansibility. In section 3 we give a set of definitions used throughout the paper (with examples for each) and set up preliminaries for the proofs to follow. In section 4, we give a distance metric for determining a lower bound on the routing steps between processor addresses by using a stronger model of the Möbius cube. We then give the calculations for the exact distance between processors and show that in some circumstances one more step than predicted by the distance metric is needed. In section 5 we describe and prove an optimal routing algorithm based on the sufficiency arguments given in the proof of Section 4, and we demonstrate the Möbius cube's diameter and bounds on the expected distance. In section 6, we compare empirical measurements of both Möbius cubes to Hilbers' Twisted cube. In section 7, we show that the edge and vertex symmetry of the hypercube do not to extend to the Möbius cubes. In section 8, we discuss the embedding of other networks into the Möbius cubes. We specifically show that the ring network and binomial tree network can be embedded. We also consider simulating hypercube algorithms on the Möbius cube and vice-versa, and show that a slowdown at worst linear to n occurs in both directions. Finally, in section 9 we show that any n-dimensional logarithmic network must have diameter at least $\Omega(n/\log n)$ and that a tree network will have this diameter.

2. The 0-Möbius and 1-Möbius cubes

The Möbius cube of degree n has $k = 2^n$ processor nodes. Each node has a unique n-component binary vector for an address and has connections to n other distinct nodes. The node $\mathbf{X} = x_{n-1}\, x_{n-2} \ldots x_0$ connects to one of its neighbors $\mathbf{Y_i}$, $n\text{-}1 \geq i \geq 0$, where $\mathbf{Y_i}$ satisfies one of the following relations:

$$\mathbf{Y_i} = (x_{n-1} \cdots \; 0\, \overline{x_i} \ldots x_0)$$
$$\mathbf{Y_i} = (x_{n-1} \cdots \; 1\, \overline{x_i \ldots x_0})$$

Stated more informally, $\mathbf{X}$ connects to $\mathbf{Y_i}$ by complementing x_i if $x_{i+1} = 0$, or by complementing $x_i \ldots x_0$ if $x_{i+1} = 1$. For the connection between $\mathbf{X}$ and $\mathbf{Y_{n-1}}$, we can assume the unspecified x_n to be either 0 or 1, giving slightly different topologies. If we assume x_n to be 0, the network generated is called the "0-Möbius cube" and if we assume x_n to be 1, the network is called a "1-Möbius cube." For most of this paper, we will consider only the topology generated by $x_n = 0$ in proving properties of the network, because the properties of both networks prove to be very similar.

Figure 2 illustrates the connections of the 0-Möbius cube of dimension 4. It also illustrates the expansibility of the Möbius cube networks by showing how a 0-Möbius cube of dimension 3 connects to a 1-Möbius cube of dimension 3 to create a 0-Möbius cube The new connections are shown in shaded lines. It can be inferred from the above definition that both the 0-Möbius cube and 1-Möbius cube of dimension n+1 can be constructed from a 0-Möbius cube and a 1-Möbius cube of dimension n by adding 2^{n+1} edges. The 0-Möbius (n+1)-cube is constructed by connecting all pairs of nodes that differ only in the x_n component, and the 1-Möbius (n+1)-cube is constructed by connecting all pairs of nodes that differ in every component.

There are some properties that are immediately apparent. As with the conventional hypercube, the Möbius cube topologies are undirected graphs; we can return to $\mathbf{X}$ along the same edge after moving to any neighbor $\mathbf{Y_i}$. The Möbius cubes

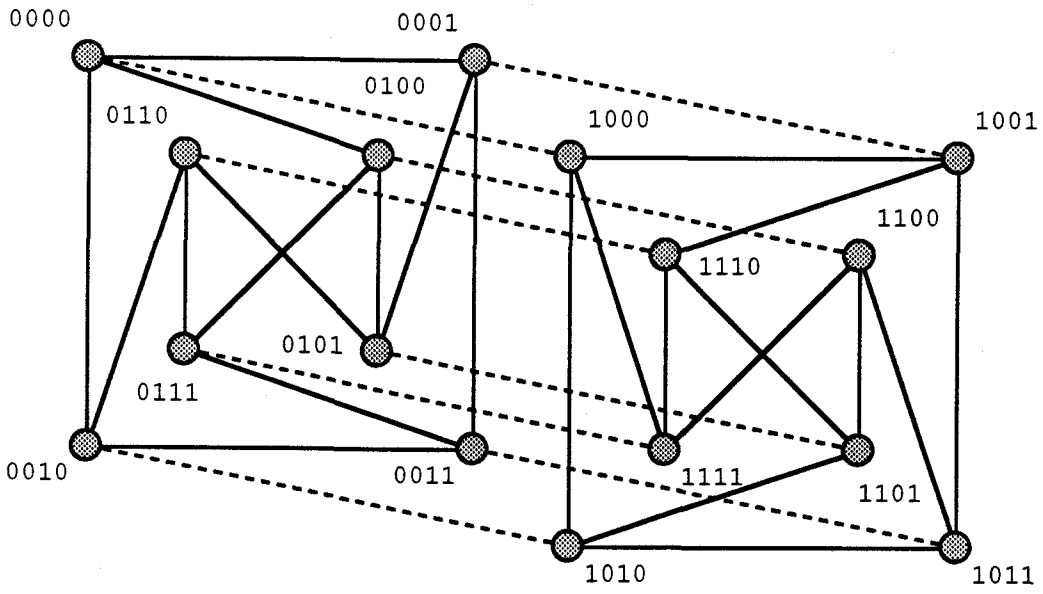

Figure 2. The 0-Möbius 4-cube.

also have exactly the same number of vertices and edges as the hypercube, so the Möbius cubes are also logarithmic network topologies.

If the fault tolerance of a network is the maximum number of nodes that can safely be removed before the network becomes disconnected, then the Möbius cubes have exactly the same fault tolerance as the hypercube; an inductive argument shows that this fault tolerance is n-1. A single processor has a fault tolerance 0. For $n > 1$, we note that a 0- or 1-Möbius cube of dimension n is simply a 0-Möbius and a 1-Möbius subcube of dimension n-1 joined together by 2^{n-1} edges. If the faults are divided between these subcubes, then neither contains more than n-2 faults and by induction both subcubes are connected and joined together by at least one edge. If all the faults occur in one subcube, then that subcube may be disconnected, but a path exists between every node of that subcube through the other subcube. Thus the fault tolerance is at least n+1.

3. Definition of notation and terms

Definition: Given two n-component binary vectors $\mathbf{X} = X_{n-1} X_{n-2} \ldots X_0$ and $\mathbf{Y} = Y_{n-1} Y_{n-2} \ldots Y_0$, we define the *mod 2 sum* of $\mathbf{X}$ and $\mathbf{Y}$ as $\mathbf{Z} = Z_{n-1} Z_{n-2} \ldots Z_0$, where each $Z_i = X_i + Y_i$, $n > i \geq 0$, under mod 2 addition. This is the same as the component-wise exclusive OR of the two vectors.

Example: If $\mathbf{X} = 001001010110$ and $\mathbf{Y} = 010100110011$, then $\mathbf{Z} = 011101100101$.

Definition: We define a *block* as a sequence of contiguous components $Z_r \ldots Z_s$, $n > r \geq s \geq 0$. The block begins at any chosen r and satisfies one of the following conditions:

- **0-block**:..A 0-block is a component sequence of the form 01...1 or 10...0. That is, $Z_r \neq Z_i$, $r > i \geq s$, and either $s = 0$ or $Z_s \neq Z_{s-1}$.

  Example: If $\mathbf{Z} = 011101100101$, then $(Z_{11} \ldots Z_8)$, $(Z_8\ Z_7)$, $(Z_7 \ldots Z_5)$, $(Z_5 \ldots Z_3)$, $(Z_3\ Z_2)$, $(Z_2\ Z_1)$, and $(Z_1\ Z_0)$ all are 0-blocks.

- **1-block**: A 1-block is a component sequence of the form 00...0 or 11...1. That is, $Z_r = Z_i$, $r > i \geq s$, and either $s = 0$ or $Z_s \neq Z_{s-1}$.

  Example: If $\mathbf{Z} = 011101100101$, then $(Z_{10} \ldots Z_8)$, $(Z_9\ Z_8)$, $(Z_6\ Z_5)$, and $(Z_4\ Z_3)$ all are 1-blocks.

- **End-block**: An end-block is is either a 0 or a 1 in the last component of the vector. That is, $r = s = 0$.

  Example: If $\mathbf{Z} = 011101100101$, then (Z_0) is an end-block.

Definition: A sequence of components $Z_r \ldots Z_0$ *contains k blocks* iff we can divide $Z_r \ldots Z_0$ into k adjacent blocks. A vector $\mathbf{Z}$ *contains k blocks* iff Z_r is the

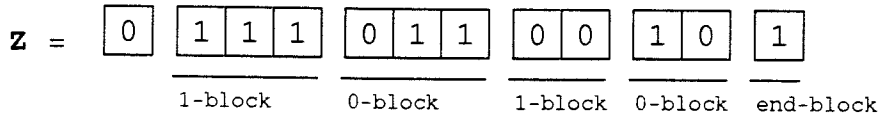

Figure 3. An example of dividing a vector into blocks.

highest order component in **Z** with $Z_r = 1$ and $Z_r \ldots Z_0$ contains k blocks. The *highest order block* in either **Z** or $Z_r \ldots Z_0$ is the block that begins at Z_r.

Example: If **Z** = 011101100101, then $Z_{10} = 1$ and $Z_{10} \ldots Z_0$ contains five blocks, as in Figure 3.

There are several items immediately apparent from the definitions above:

- There is only one way to divide a sequence of components in $Z_r \ldots Z_0$ into adjacent blocks. This is because the type and length of a block starting at Z_r are uniquely determined by their definitions.
- If we complement $Z_r \ldots Z_0$, then $Z_r \ldots Z_0$ still contains k blocks, because the definitions are independent of the parity of the components.
- If $Z_r \ldots Z_0$ contains $k > 1$ blocks and $Z_r \ldots Z_s$ is the highest order block, then $Z_{s-1} \ldots Z_0$ contains k-1 blocks.
- If a vector **Z** = 0, then **Z** contains no blocks, because there is no $Z_r = 1$.

We can describe the transition rules on either **X** or **Y** as mod 2 additions on **Z,** because application of the same transition rule on either one will have the same effect on **Z**:

Definition: We define a *0-reduction* at a component i as

$$\mathbf{Z} + e_i$$

where e_i is the n-dimensional (0, 1) vector with the ith element equal to one and the rest set to zero. We define a *1-reduction* at a component i as

$$\mathbf{Z} + e_i + e_{i-1} + \cdots + e_1 + e_0 = \mathbf{Z} + E_i$$

where E_i is the n-dimensional (0, 1) vector with the i-th through 0-th elements equal to 1 and the rest zero.. On the Möbius cube, performing a 0-reduction or a 1-reduction operation corresponds to a transition along an edge of the network. But if $\mathbf{X}$ is a processor address then exactly one of e_i and E_i give an allowed transition - which one is allowed depends on the value of X_{i+1}.

Example: If $\mathbf{Z} = 011101100101$, then $\mathbf{Z} + e_8 = 011001100101$ and $\mathbf{Z} + E_8 = 011010011010$.

Definition: The function $\alpha(n)$ returns a value whose binary representation is an n-component vector of the form 101010 ... 1. The function $\alpha(n)$ satisfies:

$$\alpha(n) = \begin{cases} 2\sum_{i=1}^{n/2} 2^{2i} + 1 & \text{if } n \text{ even} \\ \sum_{i=1}^{(n-1)/2} 2^{2i} + 1 & \text{if } n \text{ odd} \end{cases}$$

Example: The first eight values for $\alpha(n)$ are:

$$\begin{array}{ll} \alpha(0) = 1_2 = 1 & \alpha(1) = 1_2 = 1 \\ \alpha(2) = 11_2 = 3 & \alpha(3) = 101_2 = 5 \\ \alpha(4) = 1011_2 = 11 & \alpha(5) = 10101_2 = 21 \\ \alpha(6) = 101011_2 = 43 & \alpha(7) = 1010101_2 = 85 \end{array}$$

This function will prove important in calculating the exact distance between any two nodes, and also in calculating the expected distance. We note that the function $\alpha(n)$ has the property:

$$\alpha(n) + \alpha(n-1) = 2^n \qquad (1)$$

This is a difference equation that we can solve for $\alpha(n)$:

$$\alpha(n) = \frac{2^{n+1}}{3} + \frac{(-1)^n}{3}$$

4. The distance between two nodes

In order to calculate the diameter and expected routing distance of the Möbius cubes, and to derive an optimal routing algorithm, we will need some distance metric to calculate the minimum number of communication steps between nodes. For the hypercube, the Hamming distance gives the number of steps between two nodes. That is, the distance between **X** and **Y** is given by:

$$D_{IST}(\mathbf{X}, \mathbf{Y}) = \left|\{i : X_i \oplus Y_i = 1\}\right|$$

Each routing step on the hypercube reduces the Hamming distance by exactly one. The Hamming distance shows that the diameter is n, that the expected distance is $n/2$, and that the routing algorithm is optimal because the number of steps between two nodes is the number of differing bits in their addresses.

Similarly, we need a distance metric for the Möbius cube. The "block," as we have defined it in section 3, turns out to be a useful measure of distance; we show in Theorem 1 that if the mod 2 sum of two vectors contains k blocks, then at least k steps are necessary to route a message between the two vectors. This gives us a lower, though not an upper, bound on the number of steps between any two nodes, because in considering blocks we allow both e_i and E_i as changes, whereas only one of either e_i or E_i is allowed as an edge in the Möbius cube.

Theorem 1: Let **Z** be the mod 2 sum of **X** and **Y**, let r be the highest ordered (leftmost) component in **Z** with $Z_r = 1$, and let $Z_r \ldots Z_0$ contain k blocks; then at least k steps are necessary to move from **X** to **Y**, that is, to transform **Z** to the zero vector 0.

Proof: We show that a stronger model of the Möbius cube also requires at least k steps. This model allows us to use either a 0-reduction or a 1-reduction at any component of **Z**, disregarding whether that reduction corresponds to an actual edge on the Möbius cube.

To simplify the proof of optimality, we will consider only the results of a 0- or 1-reduction on **Z**, because applying a reduction on either **X** or **Y** will give the same change as applying that same reduction to **Z**. Note that **X, Y** and **Z** are elements of $(\mathbb{Z}_2)^n$, an n-dimensional vector space over 2 elements. When **X** = **Y**, their mod 2 sum **Z** is 0. Any transformation of **X** and **Y** to a common vector **W** is then equivalent to a transformation of **Z** to the zero vector 0.

To transform **Z** to the zero vector 0, there must be some combination of 0- and 1-reductions we can apply so that:

$$0 = \mathbf{Z} + (e_{i_1} + e_{i_2} + \cdots + e_{i_j}) + (E_{i_1} + E_{i_2} + \cdots + E_{i_J})$$

And then:

$$\mathbf{Z} = (e_{i_1} + e_{i_2} + \cdots + e_{i_j}) + (E_{i_1} + E_{i_2} + \cdots + E_{i_J})$$

Because modulo 2 addition is commutative, the order of these reductions does not matter.

To transform **Z** to 0, we must complement Z_r by an odd number of reductions (i.e., at least one) regardless of what order we do the reductions in. We need to consider only 0-reductions at r or 1-reductions at i, $i \geq r$, as a possible step towards 0, because no other reduction will affect Z_r.

We needed to consider how many blocks may be in a sequence of components $Z_{i-1} \ldots Z_0$ or $Z_{i+1} \ldots Z_0$, based on the number of blocks in $Z_i \ldots Z_0$. Intuitively, the number of blocks should differ by at most one, but proving this is not entirely trivial. Lemma 1 and its corollary shows that this assumption is true. It also states the conditions under which the number of blocks increases, by using the function $\alpha(n)$ as defined in section 3.

Lemma 1: If for any $r \geq 0$, $Z_r \ldots Z_0$ contains k blocks, then $Z_{r+1} \ldots Z_0$ contains k+1 blocks iff $\alpha(r\text{-}1) \leq Z_r \ldots Z_0 < \alpha(r)$, and contains k blocks otherwise.

Example: The sequence $Z_5 \ldots Z_0 = 100110$ contains three blocks. Because $\alpha(4) \leq Z_5 \ldots Z_0 < \alpha(5)$, $Z_6 \ldots Z_0$ must contain four blocks.

Proof: We first note that we can rearrange equation (1) to get:

$$\alpha(n) = \alpha(n\text{-}2) + 2^{n-1} \qquad (2)$$

Equation (2) will prove important in the inductive step of this proof.

We proceed by induction on k, the number of blocks in $Z_r \ldots Z_0$:

Base Case: Let $k = 1$. Then $Z_r \ldots Z_0$ can be one of three blocks:

- If Z_0 is an end-block, then $Z_1 Z_0$ is either a 0-block (if $Z_1 \neq Z_0$) or a 1-block (if $Z_1 = Z_0$), so $Z_1 Z_0$ contains 1 block. Also, either $Z_0 = 0 < \alpha(0)$ or $Z_0 = 1 \geq \alpha(1)$.

- If $Z_r \ldots Z_0$ is a 1-block, then $Z_{r+1} \ldots Z_0$ is either a 0-block (if $Z_{r+1} \neq Z_r$) or a 1-block (if $Z_{r+1} = Z_r$), so $Z_{r+1} \ldots Z_0$ contains 1 block. Also, either $Z_r \ldots Z_0 = 0\ 0 \ldots 0 < \alpha(r\text{-}1)$ or $Z_r \ldots Z_0 \geq 1\ 1 \ldots 1 \geq \alpha(r)$.

- If $Z_r \ldots Z_0$ is a 0-block, then $Z_{r+1} Z_r$ is either a 0-block (if $Z_{r+1} \neq Z_r$) or a 1-block (if $Z_{r+1} = Z_r$), and $Z_{r-1} \ldots Z_0$ is a 1-block (if $r > 1$) or an end-block (if $r = 1$), so $Z_{r+1} \ldots Z_0$ contains 2 blocks. Also, either $\alpha(r\text{-}1) \leq Z_r \ldots Z_0 = 0\ 1 \ldots 1 < \alpha(r)$ or $\alpha(r\text{-}1) \leq Z_r \ldots Z_0 = 1\ 0 \ldots 0 < \alpha(r)$.

Thus $Z_{r+1} \ldots Z_0$ contains 2 blocks iff $\alpha(r\text{-}1) \leq Z_r \ldots Z_0 < \alpha(r)$, and 1 block otherwise.

Inductive Step: Assume that for any $i \geq 0$ where $Z_i \ldots Z_0$ contains k-1 blocks, that $Z_{i+1} \ldots Z_0$ contains k blocks iff $\alpha(i\text{-}1) \leq Z_i \ldots Z_0 < \alpha(i)$, and k-1 blocks otherwise. Then let $Z_r \ldots Z_0$ contain k blocks, and let the highest order block in $Z_r \ldots Z_0$ be $Z_r \ldots Z_s$: There are then two cases for $Z_r \ldots Z_s$:

- If $Z_r \ldots Z_s$ is a 1-block, then $Z_{r+1}\, Z_s$ is either a 0-block (if $Z_{r+1} \neq Z_r$) or a 1-block (if $Z_{r+1} = Z_r$), so $Z_{r+1} \ldots Z_0$ contains k blocks. Also, either $Z_r \ldots Z_s = 0\ 0 \ldots 0 < \alpha(r\text{-}1)$ or $Z_r \ldots Z_0 = 1\ 1 \ldots 1 \geq \alpha(r)$.

- If $Z_r \ldots Z_s$ is a 0-block, then $Z_{r+1}\, Z_r$ is either a 0-block (if $Z_{r+1} \neq Z_r$) or a 1-block (if $Z_{r+1} = Z_r$).

  If $r\text{-}s > 1$, then $Z_{r-1} \ldots Z_s$ is a 1-block, so $Z_{r+1} \ldots Z_0$ contains $k+1$ blocks. Also, either $\alpha(r\text{-}1) \leq Z_r \ldots Z_0 = 0\ 1\ 1 \ldots < \alpha(r)$ or $\alpha(r\text{-}1) \leq Z_r \ldots Z_0 = 1\ 0\ 0 \ldots < \alpha(r)$.

  However, if $r\text{-}s = 1$, then because $Z_{s-1} \ldots Z_0$ contains k-1 blocks, by induction there are k blocks in $Z_s \ldots Z_0$ iff $\alpha(s\text{-}2) \leq Z_{s-1} \ldots Z_0 < \alpha(s\text{-}1)$. Either $Z_r\, Z_s = 0\ 1$ or $Z_r\, Z_s = 1\ 0$; in either case, we substitute equation (2) for $\alpha(s\text{-}2)$ and $\alpha(s\text{-}1)$ to show that $Z_{r+1} \ldots Z_0$ has $k+1$ blocks iff $\alpha(s\text{-}2) + 2^{r-1} = \alpha(r\text{-}1) \leq Z_r \ldots Z_0 < \alpha(s\text{-}1) + 2^r = \alpha(r)$ and k blocks otherwise. ∴

Corollary: If for any $r > 0$, $Z_r \ldots Z_0$ contains k blocks, then $Z_{r-1} \ldots Z_0$ contains k blocks iff $\alpha(r\text{-}2) \leq Z_{r-1} \ldots Z_0 < \alpha(r\text{-}1)$, and contains k-1 blocks otherwise.

Proof: This corollary follows immediately from Lemma 1 above. ∴

Now, to show the results for Theorem 1. Let the highest order block in $\mathbf{Z}$ be $Z_r \ldots Z_s$. If there is no such r, then $\mathbf{Z} = 0$ and zero steps are needed. If $Z_r \ldots Z_0$ contains 1 block, then it immediately follows that at least 1 reduction is necessary (in fact, exactly one at Z_r is sufficient). However, if $Z_r \ldots Z_0$ contains $k > 1$ blocks, then we must consider one of several cases:

- If $Z_r \ldots Z_s$ is a 0-block, then $\mathbf{Z'} = \mathbf{Z} + e_r$ has $Z'_{s-1} = 1$ and $Z'_{s-1} \ldots Z'_0$ contains k-1 blocks, and $\mathbf{Z'} = \mathbf{Z} + E_r$ has $Z'_{r-1} = 1$ and $Z'_{r-1} \ldots Z'_0$ contains at least k-1 blocks, by the corollary of Lemma 1 below.

- If $Z_r \ldots Z_s$ is a 1-block, then $\mathbf{Z'} = \mathbf{Z} + e_r$ has $Z'_{r-1} = 1$ and $Z'_{r-1} \ldots Z'_0$ contains at least k-1 blocks, by the corollary of Lemma 1, and $\mathbf{Z'} = \mathbf{Z} + E_r$ has $Z'_s = 1$ and $Z'_s \ldots Z'_0$ contains k-1 blocks.

- For any $Z_r \ldots Z_s$, $\mathbf{Z'} = \mathbf{Z} + E_{r+1}$ has $Z'_{r+1} = 1$ and $Z'_{r+1} \ldots Z'_0$ contains at least k blocks, by Lemma 1, and $\mathbf{Z'} = \mathbf{Z} + E_i$, i > r+1, has $Z'_i = 1$ and $Z'_i \ldots Z'_0$ contains k+1 blocks, because $Z'_i \ldots Z'_{r+1}$ is a 1-block and $Z'_r \ldots Z'_0$ contains k blocks.

Because in every case above, $\mathbf{Z'}$ contains at least k-1 blocks, at least k steps are necessary to transform $\mathbf{Z}$ to 0. ∴

Because the number of blocks is at least the number of steps necessary to transform $\mathbf{Z}$ to 0, we might suppose that it is also the sufficient number of steps. This is true in only some cases; in other cases one more step than the number of blocks is needed. This is because we used a stronger model then the Möbius cube to prove lower bounds on distance.

The reason for an extra step lies in the fact that we may not always be able to perform the reduction that Theorem 1 states we need. The reduction might not correspond to a legal edge in the Möbius cube. For example, Theorem 1 might call for a 1-reduction at component i to set $\mathbf{X} = \mathbf{Y}$ in the optimal number of steps , but if $X_{i+1} = Y_{i+1} = 0$, we can only legally perform a 0-reduction at i . We may be able to delay that reduction for several steps until either $X_{i+1} = 0$ or $Y_{i+1} = 0$, or we may be able to do a 0-reduction at i and still use only k steps. Occasionally, however, both of these alternatives may be impossible and at least one extra step is needed.

Theorem 2 shows that occasionally one extra step is necessary and sufficient, and shows that determining when that extra step is is needed is also easy to compute, by following the steps that the sufficiency arguments state. Theorem 2 is an inductive proof that inducts on the number of blocks k in $\mathbf{Z}$ and examines only reductions at the highest component r in $\mathbf{Z}$ with $Z_r = 1$. There are four cases that can occur in the inductive step, because there are two conditions for $X_{r+1} = Y_{r+1}$ and two conditions for the highest order block $Z_r \ldots Z_s$.

Each case requires slightly different considerations. The first two cases depend upon Lemma 2, which follows Theorem 2. The last two cases use induction to show that we can either reduce **Z** to a simpler problem with one less block or one of the first two cases in just one step.

Theorem 2: Let r be the highest ordered component in **Z** with $Z_r = 1$, and let $Z_r \ldots Z_0$ contain k blocks; then either exactly k or $k+1$ steps are necessary to set $\mathbf{Z} = 0$. Furthermore, we can decide when $k+1$ steps are needed.

Proof: We note first that any reduction above Z_r will result in at least k blocks, that is:

- $\mathbf{Z}' = \mathbf{Z} + e_{r+1}$ or $\mathbf{Z}' = \mathbf{Z} + E_{r+1}$ has $Z'_{r+1} = 1$ and $Z'_{r+1} \ldots Z'_0$ contains at least k blocks, by Lemma 1.

- $\mathbf{Z}' = \mathbf{Z} + e_i$ or $\mathbf{Z}' = \mathbf{Z} + E_i$, $i > r+1$, has $Z'_i = 1$ and $Z'_i \ldots Z'_0$ contains k blocks, because $Z'_i \ldots Z'_{r+1}$ is a 1-block and $Z'_r \ldots Z'_0$ contains k blocks.

Thus a reduction above r will not allow us to set $\mathbf{Z} = 0$ in less than $k+1$ steps, and we only need to consider reductions at r and below for this proof. We proceed by induction on k, the number of blocks in **Z**:

Base Case: Let the highest order block in **Z** be $Z_r \ldots Z_s$, where r is the highest order component in **Z** with $Z_r = 1$:

Case 1: Let $k = 0$: There is no such r. This is trivial, because then $\mathbf{Z} = 0$.

Case 2: Let $k = 1$: One of five possibilities can occur:

- If $Z_r \ldots Z_0$ is an end-block, then $\mathbf{Z}' = \mathbf{Z} + e_0 = \mathbf{Z} + E_0 = 0$.

- If $Z_r \ldots Z_0$ is an 0-block and $X_{r+1} = Y_{r+1} = 0$, then $\mathbf{Z}' = \mathbf{Z} + e_r = 0$.

- If $Z_r \ldots Z_0$ is an 1-block and $X_{r+1} = Y_{r+1} = 1$, then $\mathbf{Z}' = \mathbf{Z} + E_r = 0$.

- If $Z_r \ldots Z_0$ is an 0-block and $X_{r+1} = Y_{r+1} = 1$, then $\mathbf{Z}' = \mathbf{Z} + E_{r-1} + E_r = 0$.

- If $Z_r \ldots Z_0$ is an 1-block and $X_{r+1} = Y_{r+1} = 0$, then $\mathbf{Z}' = \mathbf{Z} + E_{r-1} + e_r = 0$.

In each case above, either one or two steps are sufficient to transform $\mathbf{Z}$ to 0. Also for the last two cases, two steps are sufficient because E_{r-1} is a legal reduction before the reduction on r-1 (because $X_r \neq Y_r$). Two steps are necessary because it is impossible to transform $\mathbf{Z}$ to 0 in just 1 step.

Inductive Step: Let the highest order block in $\mathbf{Z}$ be $Z_r \ldots Z_s$ where r is the highest order component in $\mathbf{Z}$ with $Z_r = 1$. Assume that exactly k-1 or k steps are necessary and sufficient to transform any $\mathbf{Z}'$ with k-1 blocks to 0.

Then there are four cases to consider:

Case 1: $X_{r+1} = Y_{r+1} = 1$ and $Z_r \ldots Z_s$ is a 1-block:

By Lemma 2, we can first transform $\mathbf{Z}$ into $\mathbf{Z}'$ with $Z'_{s-1} \ldots Z'_0 = 1 \ldots 1$ in k-1 steps, so the resulting $\mathbf{Z}'$ contains 1 block. Then $\mathbf{Z}'' = \mathbf{Z}' + E_r = 0 \ldots 0$, so that k steps are sufficient. Theorem 1 shows that k steps are necessary.

Case 2: $X_{r+1} = Y_{r+1} = 1$ and $Z_r \ldots Z_s$ is a 0-block:

If the corollary of Lemma 1 shows that $Z_{r-1} \ldots Z_0$ contains k-1 blocks, then by Lemma 2, we can first transform $\mathbf{Z}$ into $\mathbf{Z}'$ with $Z'_{r-1} \ldots Z'_0 = 1 \ldots 1$ in k-1 steps, so the resulting $\mathbf{Z}'$ contains 1 block. Then $\mathbf{Z}'' = \mathbf{Z}' + E_r = 0 \ldots 0$, so that k steps are sufficient. Theorem 1 shows that k steps are necessary.

If the corollary of Lemma 1 shows that $Z_{r-1} \ldots Z_0$ contains k blocks, then by Lemma 2, we can first transform $\mathbf{Z}$ into $\mathbf{Z}'$ with $Z'_{r-1} \ldots Z'_0 = 1 \ldots 1$ in k steps, so the resulting $\mathbf{Z}'$ contains 1 block. Then $\mathbf{Z}'' = \mathbf{Z}' + E_r = 0 \ldots 0$, so that k+1 steps are sufficient. Any $\mathbf{Z}' = \mathbf{Z} + E_r$ has $Z'_{r-1} = 1$ and $Z'_{r-1} \ldots Z'_0$ contains k blocks by the corollary of Lemma 1, so k+1 steps are necessary, because Theorem 1 shows k

steps are necessary to transform $\mathbf{Z}'$ to 0. Any other reduction would still give us a situation with at least k blocks, requiring at least k more steps.

Case 3: $X_{r+1} = Y_{r+1} = 0$ and $Z_r \ldots Z_s$ is a 0-block:

$\mathbf{Z}' = \mathbf{Z} + e_r$ has $Z'_{s-1} = 1$ and $Z'_{s-1} \ldots Z'_0$ contains k-1 blocks. If by induction exactly k-1 steps are sufficient to transform $\mathbf{Z}'$ to 0, then k steps are sufficient to transform $\mathbf{Z}$ to 0. Theorem 1 then shows that k steps are necessary. If by induction exactly k steps are sufficient to transform $\mathbf{Z}'$ to 0, then k+1 steps are sufficient. It may be possible that some $\mathbf{Z}' = \mathbf{Z} + E_i$, or $\mathbf{Z}' = \mathbf{Z} + e_i$, $r > i \geq s$, has $Z'_r = 1$ and $Z'_r \ldots Z'_0$ contains at least k-1 blocks, but then a 0-reduction at r won't affect Z'_i, $Z'_i = 1$, and $Z'_i \ldots Z'_0$ contains at least k-1 blocks. Thus at least k+1 steps are necessary.

Case 4: $X_{r+1} = Y_{r+1} = 0$ and $Z_r \ldots Z_s$ is a 1-block:

$\mathbf{Z}' = \mathbf{Z} + e_r$ has $Z'_{r-1} = 1$ and $Z'_{r-1} \ldots Z'_0$ contains either exactly k or exactly k-1 blocks. It is possible to apply a 0-reduction at e_r so that either $X'_r = Y'_r = 0$ or $X'_r = Y'_r = 1$. Because a reduction must affect Z_r, we perform a 0-reduction at r, but we have to examine the two alternatives for $\mathbf{X}'$ and $\mathbf{Y}'$ for each possible situation in $Z'_{r-1} \ldots Z'_0$ and always choose the shorter alternative:

- If r-$s > 1$, then $Z'_{r-1} \ldots Z'_0$ contains k blocks by the corollary of Lemma 1. If $X'_r = Y'_r = 1$, then by Case 1 above exactly k steps are necessary and sufficient to transform $\mathbf{Z}'$ to 0, because $Z'_{r-1} \ldots Z'_s$ is a 1-block and $Z'_s \ldots Z'_0$ contains k-1 blocks. However, if $X'_r = Y'_r = 0$, then by induction either k or k+1 steps are necessary and sufficient to transform $\mathbf{Z}'$ to 0.

- If r-$s = 1$ and $Z'_{r-1} \ldots Z'_0$ contains k blocks by the corollary of Lemma 1, then if $X'_r = Y'_r = 1$, by Case 2 above exactly k steps are necessary and sufficient to transform $\mathbf{Z}'$ to 0, because $Z'_{r-1} \ldots Z'_{s-1}$ is a 0-block and $Z'_{s-1} \ldots Z'_0$ contains k-1 blocks. However, if $X'_r = Y'_r = 0$, then by induction k or k+1 steps are necessary and sufficient to transform $\mathbf{Z}'$ to 0.

- If r-$s = 1$ and $Z'_{r-1} \ldots Z'_0$ contains k-1 blocks by the corollary of Lemma 1, then if $X'_r = Y'_r = 0$, by induction exactly k-1 or k steps are necessary and sufficient to transform $\mathbf{Z}'$ to 0. If $X'_r = Y'_r = 1$, then by case 2 above, k steps are necessary to transform $\mathbf{Z}'$ to 0, because $Z'_{r-1} \ldots Z'_{s-1}$ is a 0-block and $Z'_{s-1} \ldots Z'_0$ contains k blocks.

In every alternative, either k-1 or k steps are necessary and sufficient to transform $\mathbf{Z}'$ to 0, so either k or k+1 steps are necessary and sufficient to transform $\mathbf{Z}$ to 0.

Hence, either exactly k or exactly k+1 steps are necessary and sufficient to transform $\mathbf{Z}$ to 0 in every situations; the fast algorithm for deciding which is necessary and sufficient is to apply to $\mathbf{X}$ and $\mathbf{Y}$ the steps taken for the sufficiency arguments in each possible case. ∴

Lemma 2 is a fairly important result that shows we can set two vectors $\mathbf{X}$ and $\mathbf{Y}$ to differ in every component from i to 0 in an optimum number of steps if $X_{i+1} \neq Y_{i+1}$. It is based on noticing that if $X_{i+1} \neq Y_{i+1}$, then we can choose to perform a 0-reduction on X_i or a 1-reduction on Y_i, or vice versa. This choice allows us to make an optimal move at every step.

Lemma 2: For any given vector $\mathbf{Z}$ with $Z_{r+1} = 1$, $Z_r = 0$ and $Z_r \ldots Z_0$ contains k blocks, then k steps are sufficient to transform $\mathbf{Z}$ to a $\mathbf{Z}'$ where $Z'_r \ldots Z'_0 = 1 \ldots 1$ and $Z'_{n-1} \ldots Z'_{r+1} = Z_{n-1} \ldots Z_{r+1}$.

Proof: This proof depends on the fact that if $Z_{r+1} = 1$, then $X_{r+1} \neq Y_{r+1}$ and either a 0- or 1-reduction is possible at r. We proceed by induction on k, the number of blocks:

Base case: Let $k = 1$:

- If $Z_r \ldots Z_0$ is an 0-block, then $\mathbf{Z}' = \mathbf{Z} + e_r$ has $Z'_r \ldots Z'_0 = 1 \ldots 1$;
- If $Z_r \ldots Z_0$ is an 1-block, then $\mathbf{Z}' = \mathbf{Z} + E_r$ has $Z'_r \ldots Z'_0 = 1 \ldots 1$;

- If Z_0 is an end-block, then either $\mathbf{Z'} = \mathbf{Z} + e_0$ or $\mathbf{Z'} = \mathbf{Z} + E_0$ has $Z'_0 = 1$.

Inductive Step: Assume that for any given vector $\mathbf{Z'}$ where $Z'_{i+1} = 1$ and $Z'_i \ldots Z'_0$ contains k-1 blocks, then k-1 steps are sufficient to transform $\mathbf{Z'}$ to a $\mathbf{Z''}$ where $Z''_i \ldots Z''_0 = 1 \ldots 1$ and $Z''_{n-1} \ldots Z''_{i+1} = Z'_{n-1} \ldots Z'_{i+1}$. Then let the highest order block in $Z_r \ldots Z_0$ be $Z_r \ldots Z_s$. If $Z_r \ldots Z_s$ is a 0-block, then $\mathbf{Z'} = \mathbf{Z} + e_r$ has $Z'_i = 1$, $r \geq i \geq s$; or if $Z_r \ldots Z_s$ is a 1-block, then $\mathbf{Z'} = \mathbf{Z} + E_r$ has $Z'_i = 1$, $r \geq i \geq s$. In either case, $Z'_s = 1$ $Z'_{s-1} = 0$ and $Z'_{s-1} \ldots Z'_0$ contains k-1 blocks. By induction on $\mathbf{Z'}$, k steps are sufficient to set $Z_{r-1} \ldots Z_0 = 1 \ldots 1$ and $Z'_{n-1} \ldots Z'_{r+1} = Z_{n-1} \ldots Z_{r+1}$. $\therefore$

5. Routing algorithm, diameter, and expected distance

An important characteristic of interconnection networks is a simple and optimal algorithm to route messages between processors. The hypercube has such an algorithm; the left to right (LR) , i.e., highest to lowest, bit correction algorithm. We proceed from left to right, changing a component in the source address **X** whenever it differs from the destination address **Y**. However, if we used a similar LR algorithm on the Möbius cubes, the algorithm would not give an optimum routing path between processors, and performance in general would not be improved over the hypercube.

A simple routing algorithm for the Möbius hypercube exists, and though it is not optimal, it still allows a smaller maximum routing distance and expected routing distance than the hypercube. Informally, the algorithm is as follows: Separate the components of the vectors **X** and **Y** into adjacent groups of three components. When routing between nodes, we can deal with each group of three components as a routing on either a 0-Möbius or a 1-Möbius cube of dimension 3, depending on the last component of the preceding group; the last group will have 1, 2 or 3 components. We need at most two steps to set each group of three components equal. We also need at most two steps if the last group has two or three components, or one step if it has one component. The maximum routing distance of this algorithm is then $\lceil 2n/3 \rceil + 1$, and because the expected routing distance of the 0-Möbius 3-cube is $11 \times 3 / 24$, the expected distance for n in general is $(11n + n \bmod 3) / 24$. This gives a slight improvement over the hypercube's expected distance of $n/2$. However, we would like an optimal routing algorithm for the Möbius cubes.

The reductions that occur in an optimal path between source and destination must occur in a specific order. Thus any routing algorithm for the Möbius cube must specify both what reductions to perform and what order to perform them in. The sufficiency arguments in Theorem 2 specify reductions in highest-to-lowest order on the source **X** and destination **Y**, except for the one reduction on **X** that occurs "out of order" in cases 1 and 2. This ordering allows us to encode a routing path compactly in

two n-component binary vectors - call them **DX** and **DY** - that specify on which components to do a reduction, and a (log n)-bit integer to represent the "out of order" reduction - call it *Unordered.* To route from **X** to **Y**, we perform the specified reductions on **X** in order from highest to lowest, if necessary perform the one "out of order" reduction, and then perform the reductions on **Y** in order from lowest to highest.

We encode the messages this way because the sufficiency arguments in Theorem 2 examine the components of both vectors **X** and **Y** in highest to lowest order, allowing reductions on either **X** or **Y**, until both **X** and **Y** have been changed to a common address **W**, as in Figure 4 (a) The "out of order" reduction is the last step on **X** that sets **X** and **Y** equal. Unfortunately, most routing algorithms work by performing a number of transformations on the source address **X** until it becomes equal to the destination address **Y** as in Figure 4 (b). Each transformation corresponds to a legal edge of the network. If a message is to be routed from **X** to **Y** on the Möbius cube, we must perform the reductions (in order) on **X** to route from **X** to **W**, including the "out of order" reduction, then perform reductions (in reverse order) on **Y** to route from **W** to **Y**.

In the "change vector" **DX**, $DX_i = 1$ indicates that a reduction should occur at component X_i, and $DX_i = 0$ indicates that a reduction should not occur. The change vector **DY** records the same information for **Y**. We do not need to specify whether to use a 0-reduction or a 1-reduction at each step, because the address vectors determine the type of reduction. The integer *Unordered* must be large enough to record $n + 1$ values, because the "out of order" reduction could occur at any of the n components, or not occur at all. Thus the reduction stored in *Unordered* can be represented in log $(n + 1)$ bits, with *Unordered* = -1 if no "out of order" reduction occurs. If the reduction in *Unordered* is needed, it will always occur between the reductions specified by **DX** and the ones specified by **DY**. Thus this representation stores the routing path in $2n + \log (n + 1)$ bits.

(a) (b)

Figure 4. The optimality proof (a) and the routing algorithm (b).

An algorithm for computing the path from this information is illustrated in Figure 5 as the Pascal function *ComputeDestination*. It takes as input the source address **X**, the number of components k in the source address, the change vectors **DX** and **DY** and the "out-of-order" reduction *Unordered*. It returns as output the destination vector **Y**. The algorithm computes the message's destination from the input using calls to the procedures *ZeroReduce(X, i)* and *OneReduce(X, i)*, which are 0- and 1-reductions on the source vector **X** at component i. Each of these reductions corresponds to a transition from one processor to its i-th neighbor, the next processor on the routing path.

The algorithm to produce the routing information is illustrated in Figure 6 as the Pascal procedure *ComputeMöbius*. It takes as arguments the source address **X**, the destination address **Y**, and the number of components k in **X** and **Y**. It returns as output the vectors **DX** and **DY** and the integer *Unordered*. The algorithm is correct and runs in linear time, as outlined below.

The algorithm begins (in lines 10-11) by initializing **DX**, **DY** and *Unordered*, then performs a call to *ComputeAlphas*, which computes whether the mod 2 sum $Z_i \ldots Z_0 = (X_i \oplus Y_i) \ldots (X_0 \oplus Y_0)$ is between $\alpha(i\text{-}1)$ and $\alpha(i)$ for each i between k and 0, as in Lemma 1. Figure 7 illustrates *ComputeAlphas* as a Pascal function. *ComputeAlphas* executes in linear time because it computes in a constant number of bit comparisons whether $Z_i \ldots Z_0$ is between $\alpha(i\text{-}1)$ and $\alpha(i)$ given only Z_i, $Z_{i\text{-}1}$, and whether $Z_{i\text{-}2} \ldots Z_0$ is between $\alpha(i\text{-}3)$ and $\alpha(i\text{-}2)$. It achieves this by using comparisons based on the recurrence relation in equation (2).

```
function  ComputeDestination (X:BinaryVector;
                k: integer;
                DX, DY: BinaryVector;
                Unordered: integer): BinaryVector;
var
  i: integer;
begin
  for i := k downto 0 do
    if DX[i] = 1 then
        case X[i + 1] of
          0:  ZeroReduce(X, i);
          1: OneReduce(X, i);
        end;
  if Unordered < -1 then OneReduce(X, Unordered);
  for i := 0 to k do
    if DY[i] = 1 then
        case X[i + 1] of
          0: ZeroReduce(X, i);
          1: OneReduce(X, i);
        end;
  ComputeDestination := X;
end;
```

Figure 5. The function *ComputeDestination* .

The first part of the algorithm in lines 12-24 record the reductions given by the sufficiency arguments in cases 3 and 4 of Theorem 2. The algorithm finds the highest order component below k where **X** differs from **Y** and updates k to that value (line 13). Then, using the comparison results from *ComputeAlphas*, it chooses to set either $X_k = Y_k = 0$ (line 17-19) or $X_k = Y_k = 1$ (line 20-22) so that the algorithm can continue on an optimal path, as specified in case 4. This same code also trivially covers Case 3 and the base case for end-blocks because in those cases it doesn't matter whether the algorithm sets $X_k = Y_k = 0$ or $X_k = Y_k = 1$. The algorithm repeats the steps in the lines 12-24 until it either examines all the components of **X** and **Y**, or a situation from case 1 or case 2 occurs.

In this first part of the algorithm, it is possible that a previous iteration of lines 14-24 recorded that a 0-reduction occurs, say, on X_{i+1} , and the current iteration will record that a 0-reduction occurs on X_i. The algorithm needs to know the value of X_{i+1} after the 0-reduction on it. It can find this value by using the mod 2 sum of X_{i+1} and DX_{i+1}.

```
1   procedure   ComputeMöbius (X, Y: BinaryVector;
2                    k: integer;
3                    var DX, DY: BinaryVector;
4                    var Unordered: integer);
5   var
6     i: integer;
7     Alpha: AlphaVector;
8     XP, YP: BinaryValues;
9   begin
10    for i := 0 to k do begin DX[i] := 0; DY[i] := 0 end;
11    Unordered:= -1; Alpha := ComputeAlphas(X, Y, k);
12    repeat
13      repeat k := k-1 until (k < 0) or (X[k] <> Y[k]);
14      if (k >= 0) and ((X[k+1]+DX[k+1]) mod 2 = 0)
15        then  begin
16          if (Alpha[k] = InBetween)
17            then  begin
18              if X[k] = 1 then DX[k] := 1 else DY[k] := 1
19            end
20            else  begin
21              if X[k] = 0 then DX[k] := 1 else DY[k] := 1
22            end
23        end
24    until (k < 0) or ((X[k+1]+DX[k+1]) mod 2 = 1);
25    if (k >= 0) and ((X[k+1]+DX[k+1]) mod 2 = 1)
26      then  begin
27        Unordered := k; XP := 0; YP := 0;
28        repeat
29          repeat
30            k := k-1
31          until (k < 0) or ((X[k-1]+XP+Y[k-1]+YP) mod 2 = 1)
32          if k = 0
33            then DX[k] := 1
34          else if (k > 0) and ((X[k-1]+XP+Y[k-1]+YP) mod 2 = 1)
35            then  begin
36              if (X[k+1]+XP) mod 2 = 0
37                then DX[k] := 1
38                else DY[k] := 1
39            end
40          else if (k > 0) and ((X[k-1]+XP+Y[k-1]+YP) mod 2 = 0)
41            then begin
42              if (X[k+1]+XP) mod 2 = 1
43                then begin DX[k] := 1; XP := (XP+1) mod 2 end
44                else begin DY[k] := 1; YP := (YP+1) mod 2 end
45            end
46        until k < 0
47      end
48  end;
```

Figure 6. The procedure *ComputeMöbius*.

```
function ComputeAlphas (X, Y: BinaryVector;
                        k: integer): AlphaVector;
  var
    TempAlpha: AlphaVector;
    i: integer;
begin
  case (X[0] + Y[0]) mod 2 of
    0:  TempAlpha[0] := Below;
    1:  TempAlpha[0] := Above;
  end;
  case (X[1] + Y[1]) mod 2 of
    0:  case (X[0] + Y[0]) mod 2 of
          0:  TempAlpha[1] := Below;
          1:  TempAlpha[1] := InBetween;
        end;
    1: case (X[0] + Y[0]) mod 2 of
          0:  TempAlpha[1] := InBetween;
          1:  TempAlpha[1] := Above;
        end;
  end;
  for i := 2 to k do
    case (X[i] + Y[i]) mod 2 of
      0:  case (X[i - 1] + Y[i - 1]) mod 2 of
            0:  TempAlpha[i] := Below;
            1:  if (TempAlpha[i - 2] = Below) then
                  TempAlpha[i] := Below
                else
                  TempAlpha[i] := InBetween;
          end;
      1:  case (X[i - 1] + Y[i - 1]) mod 2 of
            0:  if (TempAlpha[i - 2] = Above) then
                  TempAlpha[i] := Above
                else
                    TempAlpha[i] := InBetween;
            1:  TempAlpha[i] := Above;
          end;
    end;
  ComputeAlpha := TempAlpha;
end;
```

Figure 7. The function *ComputeAlphas*.

The second part of the algorithm in lines 25-47 records the reductions given by the sufficiency arguments in cases 1 and 2 of Theorem 2. The algorithm first records the single, out-of-order 1-reduction in *Unordered.* Although it is the last reduction performed by the sufficiency arguments, we can record it before we calculate the rest of the reductions, because the algorithm doesn't actually perform any reductions. The

algorithm then finds the highest order component below k where **X** equals **Y** and updates k to that value (line 29-31). The algorithm then records the reductions given by the sufficiency arguments given by Lemma 2. The lines 32-33 cover the case of end-blocks, lines 34-39 cover the cases of 0-blocks, and lines 40-45 cover the cases of 1-blocks. The algorithm repeats the steps in the lines 25-47 until it examines all the components of **X** and **Y**.

In this second part of the algorithm, 1-reductions are possible at any step. A 1-reduction at any component i complements all the components between i and 0. This could take up to i bit operations to perform explicitly - which would not lead to a linear runtime algorithm. Because the algorithm will need to examine the values of these components after the 1-reduction, the binary variables *XP* and *YP* record when a 1-reduction complements the components below i in **X** and **Y**, respectively. If we do a 1-reduction at the component i, we can compute the current value of any X_j, $j < i$ by mod 2 addition between X_j and *XComp*. We can compute the current value of any Y_j similarly. An even number of 1-reductions on **X** or **Y** will simply reset *XP* or *YP*. Thus we can record the effects of a 1-reduction at the component i using only 1 bit operation, rather than i bit operations.

Because we follow the sufficiency arguments given by each case in Theorem 2, our algorithm is correct . We can also show that the algorithm has a runtime linear in the dimension of the Möbius cube by noticing that the variable k limits the algorithm's runtime. Initially the value of k is set to n, the number of components in the address vectors, and the dimension of the Möbius cube. The variable k is strictly decreasing and for each value of k, a constant number of bit operations are done. The only exception is the assignment to *Unordered*, which requires log n bit operations. However, the algorithm executes this assignment for at most one value of k. The algorithm terminates when $k < 0$, so the algorithm's runtime is bounded by n.

There are two ways to use this routing algorithm on a Möbius cube. The first way is to precompute the path using the algorithm above, and send the routing information along with the message. The cost of this is small, because as mentioned before, the number of extra bits sent is at most $2n + \log (n + 1)$. Each processor computes the neighbor to route the message to by looking up the next reduction to

perform. If the routing information specifies a reduction on component i, the processor sends the message and the updated routing information to its i-th neighbor. When the routing information gives no next reduction to perform, the message is at its destination.

The second way to use this algorithm is to distribute it and send only the destination address with the message, so that each processor computes which neighbor to route the message to next. It is not entirely clear how to modify the algorithm above to give a distributed routing algorithm, unless each processor runs the algorithm until it discovers the next executable reduction. This will give the routing algorithm a total runtime of at worst $O(n^2)$, because each processor along the routing path could run the algorithm to completion. If we transmit the current state of the algorithm along with the message, we could reduce this to a linear runtime.

Diameter

The diameter of a graph is the maximum number of edges on a shortest path between any two nodes in that graph. A small diameter is a desirable performance characteristic of a network topology; if the time to route a message along a connection between two nodes is a constant, the maximum delay in sending a message is proportional to the network's diameter.

The hypercube has a diameter of n; this follows directly from the maximum Hamming distance between two nodes on the hypercube. However, the Möbius cube has a diameter approximately half that of the hypercube, as Theorem 3 shows. The necessary and sufficient arguments of Theorem 2 directly show the diameter of the 0-Möbius and 1-Möbius cubes. It also shows that the diameters of the two network topologies differ when the degree of the cube is odd:

Theorem 3: The diameter of the 0-Möbius cube is:

$$\left\lceil \frac{(n+2)}{2} \right\rceil \quad n \geq 4$$

and the diameter of the 1-Möbius cube is:

$$\left\lceil \frac{(n+1)}{2} \right\rceil \quad n \geq 4$$

where n is the dimension of the cube.

Proof: We can trivially show that the diameter for both the 0-Möbius cube and 1-Möbius cube is 1, 2 and 2 for $n = 1$, $n = 2$, and $n = 3$ by enumerating all possible paths. For $n \geq 4$, however, we need a more general proof:

For the 0-Möbius cube, the particular address pairs:

$$\mathbf{X} = 1(11)^{(n-1)/2} \quad \mathbf{Y} = 0(01)^{(n-1)/2} \quad , n \text{ odd}$$
$$\mathbf{X} = 1(11)^{(n-2)/2}1 \quad \mathbf{Y} = 0(01)^{(n-2)/2}0 \quad , n \text{ even}$$

require at least $\lceil (n+2)/2 \rceil$ steps by Theorem 2. In the 0-Möbius cube of dimension n, $Z_{n-1} \ldots Z_0$ can contain at most $\lceil n/2 \rceil$ blocks; but Theorem 2 states that one extra step may be needed. Then at most $\lceil n/2 \rceil + 1 = \lceil (n+2)/2 \rceil$ steps are required.

For the 1-Möbius cube, the address pairs:

$$\mathbf{X} = (11)^{(n-1)/2}1 \quad \mathbf{Y} = (01)^{(n-1)/2}0 \quad , n \text{ odd}$$
$$\mathbf{X} = (11)^{n/2} \quad \mathbf{Y} = (01)^{n/2} \quad , n \text{ even}$$

require at least $\lceil (n+1)/2 \rceil$ steps by Theorem 2. The 1-Möbius cube of dimension n does not have a diameter of quite $\lceil (n+2)/2 \rceil$, and examining the particular possibilities shows why. Because the 1-Möbius cube has an implied $Z_n = 1$, only cases 1 and 2 in Theorem 2 above can apply; if we didn't have $Z_{n-1} = 1$, we would be dealing with a Möbius cube of dimension n-1. $Z_{n-1} \ldots Z_0$ can contain at most $\lceil n/2 \rceil$ blocks, but $Z_{n-2} \ldots Z_0$ can contain at most $\lceil (n-1)/2 \rceil$ blocks by Case 2 above, which we can reduce in at most $\lceil (n-1)/2 \rceil$ steps by Lemma 2. Then at most $\lceil (n-1)/2 \rceil + 1 = \lceil (n+1)/2 \rceil$ steps are required. $\therefore$

Expected Distance

A second performance characteristic is the expected distance between any nodes in a given network. The expected distance for the hypercube is $n/2$; because the

Möbius cube has a diameter roughly half that of the hypercube, we expect it to also have a smaller expected distance. This expected distance is not half of the hypercube's expected distance, however; a majority of node pairs in the Möbius cube have distances closer to the diameter, making the expected distance somewhat higher than $n/4$. Though we do not calculate the expected distance exactly, we can show that it lies within bounds that are significantly below the hypercube's expected distance.

If we can calculate the expected number of blocks in an address pair, then by Theorem 2 we have an upper bound and a lower bound on the expected distance between nodes, since the theorem states that the distance is between the number of blocks and the number of blocks plus one. Theorem 4 calculates the expected number of blocks, thereby bounding the expected distance:

Theorem 4: The expected distance between nodes in both the 0-Möbius cube and the 1-Möbius cube is between $n/3 + [1 - (-1/2)^n]/9$ and $n/3 + [1 - (-1/2)^n]/9 + 1$.

Proof: This is a straightforward argument, done by counting the expected number of blocks in the vector $\mathbf{Z} = \mathbf{X} \oplus \mathbf{Y}$. Assume that each pair of addresses **X**, **Y** occur with equal probability. Consider a particular **X**. The mod 2 sum of **X** and each **Y** will give each **Z** exactly once. Hence each **Z** will have the same probability for a given **X**, and because each **X** has equal probability, each **Z** will occur with equal probability.

Consider any particular $\mathbf{Z} = Z_{n-1} \ldots Z_0$ of n components. We notice that prepending a component n, $Z_n = 0$, onto a vector **Z** will not increase the number of blocks in **Z**, because the highest order component r with $Z_r = 1$ is not equal to n. We notice also that prepending a component n, $Z_n = 1$, onto a vector **Z** will increase the number of blocks in **Z** iff the components $Z_n \ldots Z_0 < \alpha(n)$. This is because if $0 \leq \mathbf{Z} < 2^{n-1}$, then $Z_{n-1} \ldots Z_{r+1}$ will form a single 0-block, giving us $k+1$ blocks. If $2^{n-1} \leq \mathbf{Z} < 2^n$, then by Lemma 1 the number of blocks in **Z** will increase by one iff $Z_n \ldots Z_0 < \alpha(n)$.

The expected number of blocks $E(n)$ in an n component vector can be described as the average of the expected number of blocks in a $(n-1)$-component vector when prepended by a 0 and when prepended by a 1:

$$E(n) = \frac{\left[E(n\text{-}1) + E(n\text{-}1) + \frac{\alpha(n\text{-}1)}{2^{n-1}}\right]}{2}$$
$$= E(n\text{-}1) + \frac{\alpha(n\text{-}1)}{2^n}, \; E(0) = 0$$

Solving this difference equation will give us the expected number of blocks in an n-component vector:

$$E(n) = \frac{n}{3} + \frac{1}{9}\left[1 - \left(-\frac{1}{2}\right)^n\right]$$

Because $E(n)$ is the expected number of blocks, by Theorem 2, the number of steps is bounded below by $E(n)$, and bounded from above by $E(n) + 1$:

$$\frac{n}{3} + \frac{1}{9}\left[1 - \left(-\frac{1}{2}\right)^n\right] \leq E(n) \leq \frac{n}{3} + \frac{1}{9}\left[1 - \left(-\frac{1}{2}\right)^n\right] + 1$$

The expected distance is then approximately $(3n + 1)/9$, as n grows larges. Also, $E(n)/n$ approaches 1/3 as n grows large. $\therefore$

6. Comparing the Möbius cube to the Twisted cube

The twisted cube topology (Hilbers et. al. 1987) has the same diameter as the 1-Möbius cube and a slightly smaller diameter than the 0-Möbius cube. An empirical comparison of the three networks made this difference clear. We wrote a program to compute the diameters of each of the hypercube, twisted cube and 0-Möbius and 1-Möbius cubes. The comparison of the diameters is shown in Figure 8. The diameters of the 1-Möbius cube and twisted cube are the same. The diameter of the 0-Möbius cube was one more for odd-dimensioned networks of at least dimension 5.

We also wrote a program to compare the expected distance of the hypercube, the twisted cube, and the Möbius cubes. For $n \leq 2$, all the cubes are identical. For $n > 2$, the hypercube variants show an improvement on the hypercube's expected distance. The 1-Möbius cube has a smaller expected distance than the 0-Möbius cube, due partly to its smaller diameter. The twisted cube shows a larger expected distance than both, though it is comparable to the 0-Möbius cube. By dividing the expected distance by the number of nodes in the cube, the differences in the rate of growth becomes much more apparent, as shown in Figure 9.

Though the twisted cube has the same maximum routing distance as the 1-Möbius cube, the expected distance shows that the topologies of the two networks are distinct. Abraham (1990) showed that for large n the twisted cube has an expected distance of about $3n/8$. However, for large n the Möbius cube has expected distance of about $n/3$. Thus the Möbius cubes have the advantages of having conceptually simpler definitions and of having slightly smaller expected distances between processors.

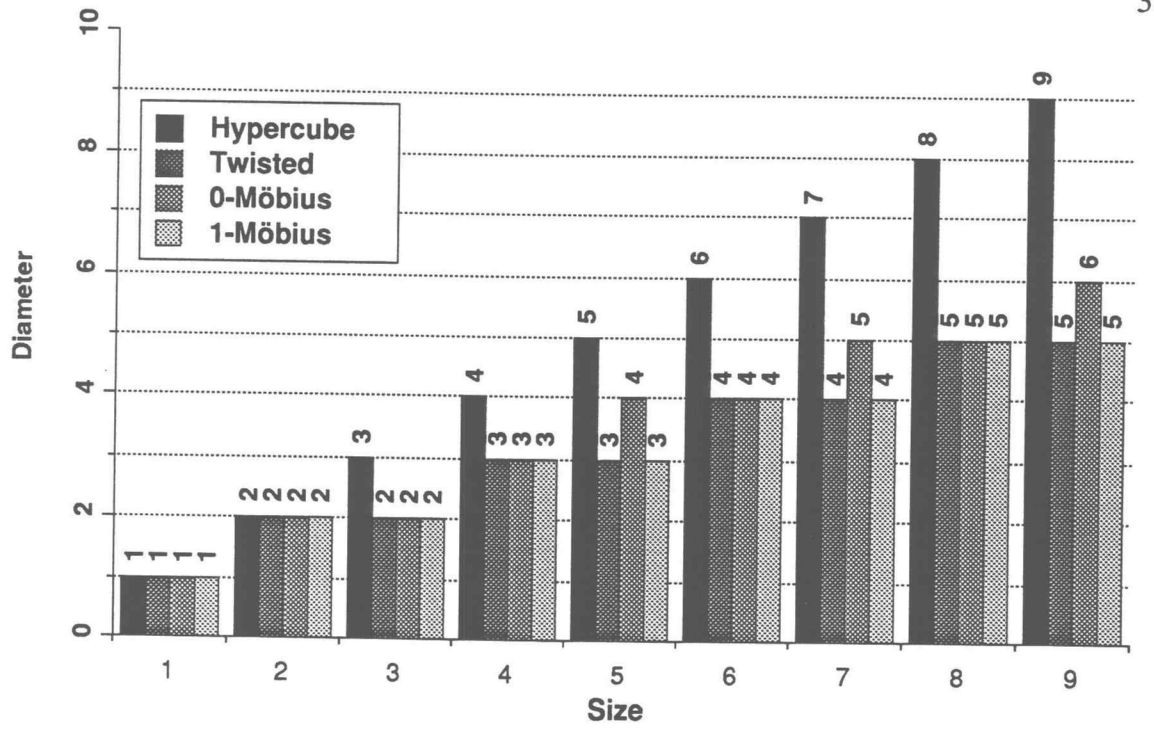

Figure 8. The diameters of the hypercube and variants.

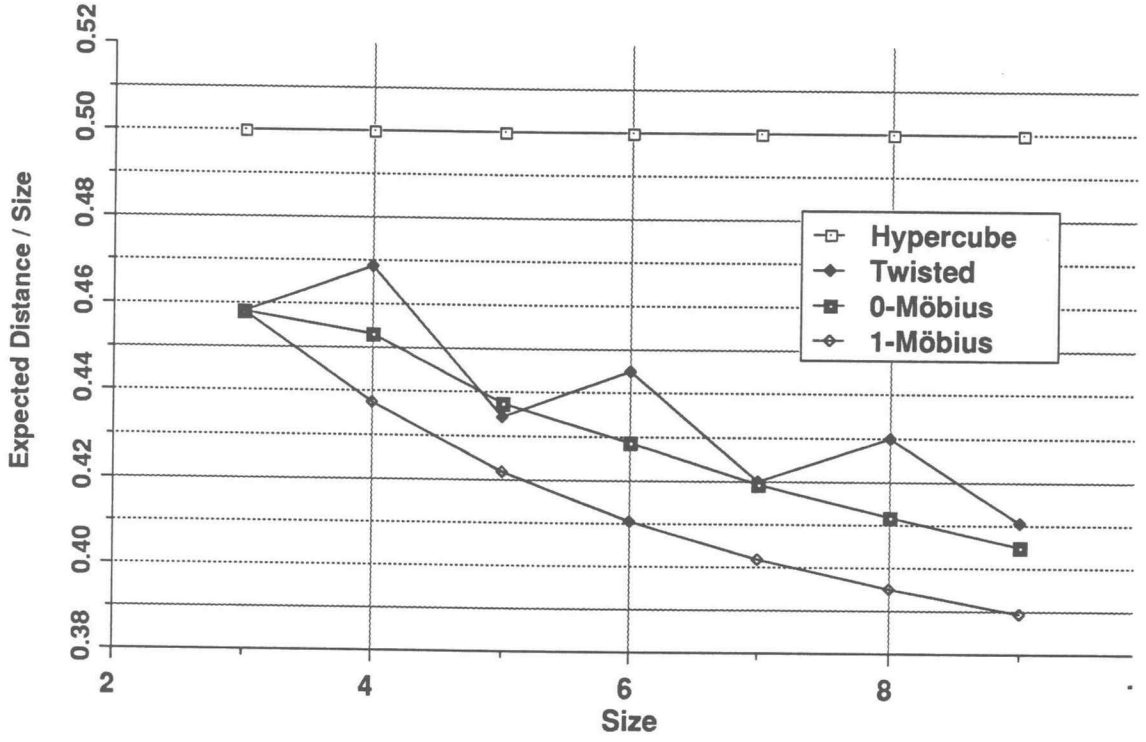

Figure 9. The expected distances of the hypercube and variants.

7. Vertex and edge asymmetry in the Möbius cube

Among other properties, the hypercube has the property of symmetry; the network as viewed from any one vertex looks the same if viewed from any other vertex on the network. Because all nodes are identical, with relative addressing, software written for one node can work from any node in the network without change. There are actually two symmetric properties that the hypercube has:

Definition: A graph is *vertex symmetric* if for every pair of vertices a and b in graph G there is an automorphism of G that maps a into b. A graph is *edge symmetric* if for every pair of edges a and b in graph G there is an automorphism of G that maps a into b. (Akers, Balakrishnan 1989)

The hypercube is both vertex symmetric and edge symmetric; it would be desirable if the Möbius cube also retained these two properties, so that algorithms can easily be remapped from node to node. However, the Möbius cube is neither vertex symmetric nor edge symmetric.

We demonstrate the asymmetry of the Möbius cube in Theorems 5 and 6. Theorem 5 uses the number of cycles of length four through a single node to show the Möbius cube is not vertex symmetric. Because the number of 4-cycles differ between nodes, there exist an a and b that has no automorphism G. Theorem 6 uses a similar argument to count the number of 4-cycles through a single edge and show edge asymmetry. In order to count the number of 4-cycles, Lemma 3 below is used to show the conditions under which a 4-cycle can exist on the Möbius cube, namely that a 4-cycle can only be traced by doing alternate reductions on two components.

Theorem 5: The Möbius cubes are not vertex symmetric for $n > 3$.

Proof: A necessary, but not sufficient, condition for vertex symmetry is that local graph properties must hold around each vertex; for instance, the number of 4-cycles through a vertex must be the same for all vertices in the graph. Otherwise any

mapping which maps a vertex to another vertex with a different number of 4-cycles will fail to be an automorphism, and hence no automorphism exists which maps between these two vertices.

Assume $n > i > j \geq 0$. For $j = 0$, there are n-1 possible values for i that form a 4-cycle. However, for $j > 0$, the number of values for i is at most n-j-2, but is dependent upon the number of i such that $X_{i+1} = 0$, from the conditions of Lemma 3 below. Therefore the Möbius cube for $n > 3$ cannot be vertex symmetric. ∴

Theorem 6: The Möbius cubes are not edge symmetric for $n > 2$.

Proof: We will use the argument of counting the number of 4-cycles through an edge. Each edge (**X**, **Y**) is in a 4-cycle iff both **X** and **Y** are in the same 4-cycle. As Lemma 3 shows, we must apply alternate reductions to two components i and j, $n > i > j \geq 0$. Assume that edge (**X**, **Y**) corresponds to a reduction on j; then all $X_i = Y_i$, $i > j$. If $j = 0$, then the number of 4-cycles through (**X**, **Y**) is n-1. If $j > 0$, then the number of 4-cycles through (**X**, **Y**) is at most n-j-2, but is dependent upon the number of $X_i = Y_i = 0$. Therefore the Möbius cube cannot be edge symmetric for n > 2. ∴

Lemma 3: Any 4-cycle in the Möbius hypercube through a node **X** can only be traced by alternate reductions on exactly two components X_i, X_j, n-1 $\geq i > j \geq 0$ with either (1) $X_{i+1} = 0$ and i-1 $> j > 0$ or (2) $j = 0$.

Example: Choosing $i = 3$ and $j = 1$ and alternately perform reductions on X_i, X_j for **X** = 01011, we get the 4-cycle:

$$01011 \rightarrow 00011 \rightarrow 00000 \rightarrow 01000 \rightarrow 01011$$

Proof: Tracing a four-cycle from **X** will lead back to the original vertex **X**. All that we need to show is that a return to **X** in four reductions will require the reductions be done alternately on only two components. Obviously four reductions on one component does not create a 4-cycle, because it retraces one edge. Nor will four reductions on four components suffice, because in so doing we complement the highest order component only once, forcing the final vector to be different from **X**. To show

that four reductions on three components do not suffice, assume that there are i, j, k such that $n > i > j > k \geq 0$. **X** will need two reductions at i, because reductions at j and k cannot affect X_i. Then we can do no more than one reduction each on X_j and X_k. Because we complement X_i twice and because a reduction at k does not affect X_j, X_j will always have its value complemented an odd number of times after four steps. Thus we must use exactly two components and we must perform reductions on them alternately, because a reduction on the same component twice in succession will retrace an edge - which is not allowed in a cycle.

To prove the conditions (1) and (2) for i and j:

Case 1: $X_{i+1} = 1$, so that X_i must be 1-reduced, and $j > 0$. A reduction at i also complements X_{j+1}, forcing j to be 1-reduced and 0-reduced once each. After four steps, $X_{j-1} \ldots X_0$ is complemented, hence a 4-cycle does not exist.

Case 2: $X_{i+1} = 0$, $i = j+1$ and $j > 0$. A reduction at i complements $X_i = X_{j+1}$, forcing X_j to be 1-reduced and 0-reduced once each, as in case 1 above, hence a 4-cycle does not exist. $\therefore$

This lack of vertex symmetry removes Möbius cube from the class of Cayley graphs (Akers, Balakrishnan 89). It is trivial to show that the Möbius cubes are vertex symmetric for $n \leq 3$ and edge symmetric for $n \leq 2$. There are a number of automorphisms for mapping the Möbius cube onto itself. An immediately apparent one is to map each node **X** to its neighbor $\mathbf{Y_0}$. This preserves edges because a 0-reduction on the 0-th component of a Möbius cube has the same effect as a 1-reduction on the 0-th component. However, the number of automorphisms for the Möbius cube probably remains small as n grows large.

8. Embedding and simulating other networks

The hypercube can have a number of other network topologies embedded into it, so it can simulate other common interconnection networks, such as the mesh and ring networks. It is desirable that the Möbius hypercube also has a number of such networks embedded into it. We examine three common hypercube-embeddable topologies to see if we can embed them into the Möbius cube as well.

Hamiltonian circuit

One useful graph embedding is the Hamiltonian circuit. If the Möbius cube of dimension n has a Hamiltonian cycle, then we can embed a ring network of length 2^n into the network, and then use any ring network algorithms on the Möbius cube.

Theorem 7 shows that the Möbius cube has a Hamiltonian circuit. It does so by example - we give an algorithm to generate a sequence of steps or reductions that follow a Hamiltonian circuit on a Möbius cube and then show that the algorithm is correct. We show correctness by associating each reduction with an actual edge on the Möbius cube; the circuit is correct iff the reduction made at each step is legal according to the connection rules and no node is visited twice.

Theorem 7: The Möbius cube has a Hamiltonian circuit.

Proof: We give an algorithm to generate a Hamiltonian circuit from any given node and then show it to be correct. Given an initial address **X**, we can use Pascal notation to write the algorithm as in (Figure 10), assuming that the array $X[n\text{-}1] \ldots X[0]$ contains components of the vector **X** and $X[n] = 0$ (i.e., a 0-Möbius cube) or $X[n] = 1$ (i.e., a 1-Möbius cube).

The algorithm will produce a path that follows legal transitions on the Möbius hypercube, because we state the transition rules in the algorithm. The last reduction is necessary to return to the starting address vector **X.**

```
for i:= 0 to pow(2, n) - 1 do begin
     k : = (n - 1) - log2( gcd(i, pow(2, n)));
     if X[k+1] = 0 then
          ZeroReduce(X, k);
     else if X[k+1] = 1 then
          OneReduce(X, k);
  end;
ZeroReduce(X, 0);
```

Figure 10. The algorithm for generating Hamiltonian circuits.

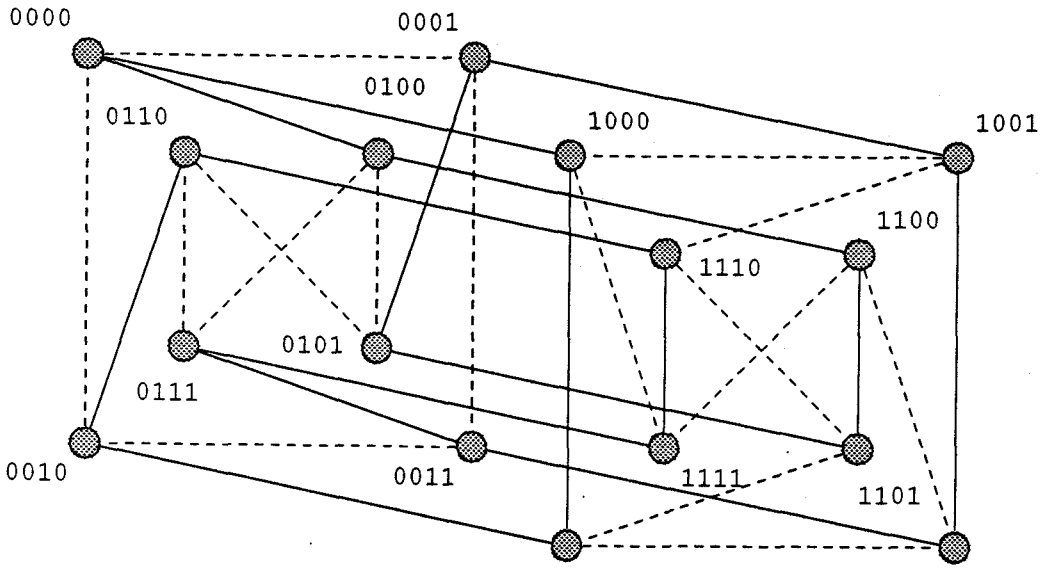

Figure 11. A Hamiltonian circuit for Möbius 4-cube.

Example: For the node **X** = 1111, the Hamiltonian circuit produced on the 0-Möbius cube is:

$$1111 \rightarrow 0111 \rightarrow 0011 \rightarrow 1011 \rightarrow 1001 \rightarrow 0001 \rightarrow 0101 \rightarrow 1101 \rightarrow$$
$$1100 \rightarrow 0100 \rightarrow 0000 \rightarrow 1000 \rightarrow 1010 \rightarrow 0010 \rightarrow 0110 \rightarrow 1110$$

The Hamiltonian circuit started at **X** = 1111 is shown in Figure 11.

We use a recursive argument to show that each of the of 2^n nodes in the circuit is unique. That is, the addresses of nodes produced in the sequence are unique if and only if the nodes in the first and second halves of the sequence are unique and any node

in the first half differs from all the nodes in the second half. Consider any block of nodes in the generated sequence from m to $m + 2^k$-1, where m is a multiple of 2^k. If $k = 0$, then there one node in the block, making it unique. If $k > 0$, then we can divide the block into two equal halves: from m to $m + 2^{k-1} - 1$, and from $m + 2^{k-1}$ to $m + 2^k - 1$. We assume that there are no duplicates within each half and show from this that there are no duplicates between m and $m + 2^k - 1$. Because:

$$k = (n - 1) - \log_2(\ \gcd(\ m + 2^{k-1},\ 2^n\ ))$$

our algorithm causes a reduction at component k to occur at step $m + 2^{k-1}$, but nowhere else in the sub-sequence from m to $m + 2^{k-1} - 1$.

Now consider any two nodes i_1 and i_2 generated by the algorithm, $m \le i_1 \le m + 2^{k-1}-1$ and $m + 2^{k-1} \le i_2 \le m + 2^k-1$, and the nodes associated with those steps, **X** and **Y**. By Lemma 4 below, we can assume that all the reductions on a particular component will either all be 0-reductions or all be 1-reductions. If the reductions at all components j for $n > j > k$, are 0-reductions, then $X_k \neq Y_k$ because the reduction at step $m + 2^{k-1}-1$ complemented component i. Hence $\mathbf{X} \neq \mathbf{Y}$ and the sequence from m to $m + 2^k-1$ is composed of unique nodes.

But if the reductions at one components j for $n > j > k$, are 1-reductions, then it is possible that $X_k = Y_k$ for some **X** and **Y**. Assume the lowest order such 1-reductions are at $j > k$, then we note that a 1-reduction at j will complement both X_j and X_k. Then either $X_j = Y_j$ and $X_k \neq Y_k$, or $X_j \neq Y_j$ and $X_k = Y_k$, so that $\mathbf{X} \neq \mathbf{Y}$. A further 1-reduction at any $j' > j$ will only invert both X_j and X_k, so that even then $\mathbf{X} \neq \mathbf{Y}$. Thus the sequence from m to $m + 2^k - 1$ contains no duplicates.

We must also show that that the first and last node of the sequence are adjacent. The last node in the sequence has an edge connecting to the starting node, because we have complemented all components X_i, $n > j > 0$, an even number of times, so that the only differing component between the first and last nodes is x_0. Hence the Möbius cube has a Hamiltonian circuit. ∴

Lemma 4: A reduction at any k, $n > k \ge 0$, in the algorithm given in Figure 10 will always be either a 0-reduction or a 1-reduction.

Proof: If an even number of reductions at every j, $k > j \geq 0$, occurs between each reduction at k, then component X_{k+1} will always be 1 or 0 when we complement X_k. Hence a reduction at k is either always a 1-reduction or a 0-reduction. From the algorithm, this condition is true if and only if there is an even number of steps i with $\gcd(i, 2^n) = 2^j$ between i_1 and i_2 with $\gcd(i_1, 2^n) = \gcd(i_2, 2^n) = 2^k$. We need to show that this is so:

Consider any two numbers i_1 and i_2, $2^n > i_2 > i_1$, with the $\gcd(i_1, 2^n) = \gcd(i_2, 2^n) = 2^k$. Then we can rewrite i_1 and i_2 as products:

$$i_1 = r_1 2^k,\ i_2 = r_2 2^k$$

We notice that:

$$\gcd(r_1, 2^n) = \gcd(r_2, 2^n) = 1$$

which implies that r_1 and r_2 are odd, and $r_2 - r_1$ is always even. Now consider all numbers i' between i_1 and i_2 such that i' divides $2^j, j < k$. There are:

$$\frac{r_2\, 2^k - r_1\, 2^k}{2^j} = \left(\frac{r_1 - r_2}{2}\right)\frac{2^{k+1}}{2^j} = \left(\frac{r_1 - r_2}{2}\right) 2^{k-j+1}$$

Such numbers i', and we can represent these i' by:

$$i' = r_1\, 2^k + s\, 2^k,\ \ 0 \leq s < \left(\frac{r_1 - r_2}{2}\right) 2^{k-j+1}$$

If s is even, then the $\gcd(i,2^n) > 2^{k-1}$, so s must be odd. There are exactly $(r_1 - r_2)\, 2^{k-j-1}$ such s, thus there must be an even number of i'. $\therefore$

Mesh networks

Obviously not all topologies that map to the hypercube will map as easily into the Möbius cube. Mesh networks, which we can embed onto the hypercube by Gray-encoding (Saad, Schultz 1988), seem to be impossible to embed onto the Möbius cube. We can embed the special case of a 2 by 2^{n-1} mesh by noting that the i-th node in a

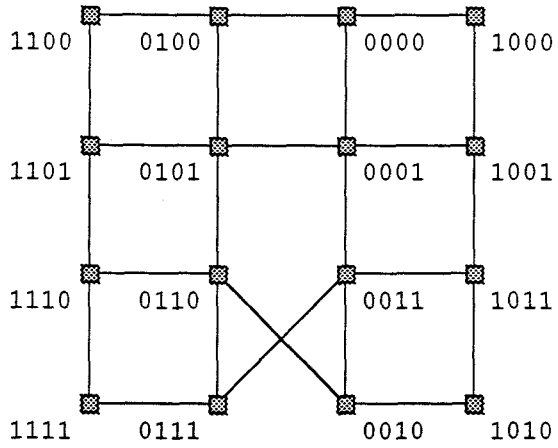

Figure 12. Approximating a 4 by 4 mesh on the Möbius 4-cube.

Hamiltonian circuit given by the algorithm above has the $(n\text{-}i\text{-}1)$-th node as a neighbor. The closest possible approximation to a 4 by 4 mesh on the Möbius cube of dimension 4 is shown in Figure 12 - two edges of the mesh end up being "crossed" when mapped to the Möbius cube. Toroidal mesh networks appear to map even less well.

Binomial trees

The binomial tree is a graph structure used in various algorithms for the hypercube topology. The useful feature of the binomial tree on the hypercube is that a broadcast algorithm can use the binomial tree as a "broadcast tree" to send messages to all nodes non-redundantly (i.e., each node receives exactly one message from the broadcasting node and no edge is used more than once). Many divide-and-conquer algorithms on the hypercube use the binomial tree for broadcasting and/or reducing values from/to a single processor. The binomial tree has a highly regular structure that makes it simple to define inductively:

Definition: (Brown 1978) We defined the class B_k of ordered binomial trees as follows:

1. Any tree of a single node is a B_0 tree.

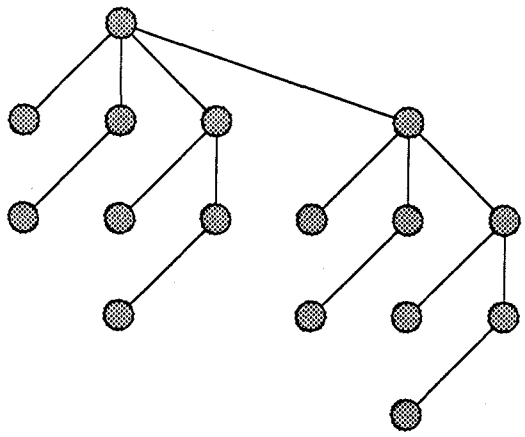

Figure 13. The binomial tree of order 4.

2. If X and Y are disjoint B_{k-1} trees for $k \geq 1$, then the tree obtained by adding an edge to make the root of Y become the leftmost offspring of the root of X is a B_k tree. All binomial trees of order k are isomorphic in the sense that they have the same topology.

Example: The binomial tree B_4 is in figure 13. The relationship between the binomial tree and the hypercube should be apparent.

We can trivially map binomial trees into the Möbius cube. As a result, most hypercube divide and conquer algorithms that use a binomial tree can also work on the Möbius cube without heavy modification. We show in the next theorem that any node in the Möbius cube can serve as the root of a binomial tree of order n. Because the binomial tree B_n has 2^n nodes and a depth of n, the runtime order of any non-redundant broadcast algorithm is also bounded from above by n steps from the binomial tree depth and below by $\lceil n/2 \rceil + 1$ steps from the graph diameter:

Theorem 8: Every node in the Möbius hypercube can be mapped to the root of at least one binomial tree B_n.

Proof: An inductive proof based on the expansibility of the Möbius cube. For $n = 1$, either node of the single edge (0, 1) is the root of a binomial tree of order 1. Assume also that we can map every node **X** to the root of an order n-1 binomial tree in a 0-Möbius or 1-Möbius cube of dimension n-1. We note then that we can divide a

```
if k = 0 or X[k+1] = 0 then
  ZeroReduce(X, k);
else if X[k] = 0
  then begin
    OneReduce(X, k);
    OneReduce(X, k-1);
  end
  else begin
    OneReduce(X, k-1);
    OneReduce(X, k);
  end
```

Figure 14. An algorithm for hypercube routing on the Möbius cube.

Möbius cube of dimension n into a 0-Möbius cube and a 1-Möbius cube, both of dimension n-1. Choose a node **X** in either Möbius sub-cube and **X**'s k-th neighbor $\mathbf{Y_k}$ in the other Möbius sub-cube. We can map the root of a binomial tree of order n-1 to **X**, map the root of another binomial tree of order n-1 to $\mathbf{Y_k}$ and join **X** and $\mathbf{Y_k}$ along the edge (**X**, $\mathbf{Y_k}$) to form a binomial tree of order n. ∴

Simulating hypercube programs

Because the hypercube and the Möbius cube of dimension n have the same computational resources, we could possibly write an algorithm for hypercube, then transfer it to the Möbius cube, replacing each of the hypercube's communication steps with a series of Möbius cube communication steps. If it is possible to emulate a hypercube routing step in a constant or linear (on n) number of steps on the Möbius cube, then we could directly run hypercube algorithms on the Möbius cube with little modification and only a constant or linear amount of "slowdown". Conversely, we could execute Möbius cube programs on the hypercube, by emulating the Möbius cube routing steps on the hypercube.

It is immediately apparent that if we map Möbius cube node addresses directly to hypercube addresses, the hypercube must take at worst $O(n)$ time to run a Möbius cube algorithm, because a single 1-reduction would take at worst n routing steps to emulate. This leads to a linear increase in the runtime order of any algorithm.

We are interested in emulating hypercube programs on the Möbius cube, because at present there are many more algorithms for the hypercube than for the Möbius cube. We assume a direct mapping of hypercube nodes onto the Möbius cube; this assumption simplifies translating the algorithm. For each hypercube routing step that a message takes from a node **X** to its i-th neighbor $\mathbf{Y_i}$, the Möbius cube performs a series of routing steps along legal Möbius cube edges to get the message to $\mathbf{Y_i}$. The number of routing steps on the Möbius cube must make for one hypercube routing step is never more than 2, because for any $i > 0$, $\mathbf{Z} + E_i + E_{i-1} = \mathbf{Z} + e_i$; in other words, two 1-reductions are equivalent to one 0-reduction . Also, we note that $\mathbf{Z} + E_0 = \mathbf{Z} + e_0$, that is, a 1-reduction on the 0-th component is equivalent to a 0-reduction. We can use these two observations to write a "emulation" algorithm, as in Figure 14.

If the original algorithm routes a message along an hypercube edge, the computer determines if there is a corresponding legal edge in the Möbius cube and sends the message along that. If there is no legal edge, the computer routes the message along two Möbius cube edges to reach the destination. The algorithm is correct because we can achieve an 0-reduction by doing the two 1-reductions on the components k and k-1 in proper order, guaranteeing that a 1-reduction on the component k-1 is legal when performed. Thus constant-order simulation time is sufficient to route a single n-cube message on the Möbius cube.

There are two other considerations we must face to guarantee a constant-order runtime. In any one time step, two adjacent nodes might send messages along the same edge in opposite directions - an *edge collision* - or that two or more messages arrive at the same node - a *vertex collision*.. Under realistic models of computation, either of these conditions might necessitate extra communication steps to disallow either type of collision.

First we use a model of computation that disallows edge collisions. We assume the algorithm disallows edge collisions on the hypercube; we must consider if it disallows edge collisions on the Möbius cube. The hypercube is disallowed from sending messages from **X** to **Y** and **Y** to **X** simultaneously along dimension i. If

$X_{i+1} = Y_{i+1} = 0$, then this condition is held on the Möbius cube, because this edge corresponds to one in the hypercube. For hypercube edges where no Möbius cube edge exists, 2 edge routings are necessary. We can guarantee that a message sent along dimension i on the hypercube does not collide with a message sent along dimension i-1 by handing all even-dimension and all odd-dimension hypercube routings separately, because a routing along dimension i requires at most 1-reductions on component i and i-1. But consider the situation where we route from $100x$ to $110x$ and from $101x'$ to $111x'$ simultaneously, where $x \in \{0,1\}^{i-2}$. The second step of both routings will be along the edge between $111x'$ and $110x$, causing an edge collision. One message must be delayed a step, hence 6 steps in all are sufficient to guarantee no edge collisions, giving an constant-order slowdown of the algorithm.

We then consider a model that disallows vertex collisions. The original algorithm must disallow vertex collisions on the hypercube; this disallows vertex collisions at the source or destination nodes of any message on the Möbius cube, so we need only consider the one intermediate node on two-step routings. We note that if a message is routed along hypercube edge i, the above algorithm will route a 2-step message through a node **X** iff $X_i = 1$. In the worst case, we have to route at most n-1 2-step messages through **X** = 111 ... 1. If we avoid vertex collisions by allowing only one message to arrive at **X** at each step, the routing simulation time must be at worst $O(n)$ steps, giving us a linear slowdown of the algorithm.

The above argument demonstrates only upper bounds on the emulation times. It may be possible, but unlikely, that we could improve on the emulation runtime bounds, either by using a different mapping of nodes between the topologies, or by using a different algorithm to emulate the edge routings.

9. General bounds on a logarithmic network

Can a logarithmic network have a smaller diameter than the Möbius cube? As the result of the diameter bounds we are able to get for the Möbius cubes, a good question would be to ask what the lower bound for any logarithmic graph would be. We will use a tree-based argument to obtain a lower bound on the diameter of logarithmic networks, and show by considering our tree that at most twice the lower bound is attainable. Our conclusion is that the Möbius cube has the smallest possible diameter for logarithmic networks only when n is small. For large n, the smallest possible diameter is approximately $n/\log n$, which will be much smaller than the approximately $n/2$ diameter of the Möbius cube.

Consider a network of 2^n nodes, where $n \in \mathbf{Z}^+$. For $n \geq 3$, the minimum diameter D must satisfy:

$$\frac{n\left[(n-1)^D-1\right]}{n-2} + 1 \geq 2^n$$

Because at distance 0 there is only one node, at distance 1 there are n distinct nodes and at distance i+1 the number of nodes is at most (n-1) times the nodes at distance i. Hence:

$$2^n \leq 1 + n + n(n-1) + n(n-1)^2 + \cdots + n(n-1)^{D-1} = 1 + \frac{n\left[(n-1)^D-1\right]}{n-2}$$

We rearrange the lower bound in terms of D_L, the smallest possible value of D:

$$D_L \geq \frac{n}{\log(n-1)} + \frac{\log\left(1 - \frac{2}{n} + \frac{2}{n2^n}\right)}{\log(n-1)} \geq \frac{n-1}{\log(n-1)}, \quad n > 3$$

Because D is an integer value, we can always round D_L up to fit the inequality. Any node is a distance at most D away from the the tree's root node. It is clear that the second term in the inequality above will reduce to zero for n large, thus

$D \geq n/log(n\text{-}1))$ should be a reasonable approximation, save when n is small or $n/log(n\text{-}1)$ is slightly greater than an integer.

This lower bound correctly shows that the optimum diameter for a logarithmic graph of order $n = 3$ is 2. For $n = 4$ or $n = 5$, the diameter is $D \geq 3$, one less than the Möbius cube diameter of 3. However, it states that the diameter for $n = 6$ is $D \geq 3$. This is, of course, only a lower bound; we could only determine that the Möbius cube had an optimal diameter for small values of n if we had the exact bounds.

For the tree then, a diameter of $D = 2D_L$ can be achieved. The network produced will not be a full tree; the bottom level of the tree will not be completely full. Indeed, there will be a number of nodes on the bottom level equal to *GAP*:

$$GAP = \frac{n\left[(n-1)^D - 1\right]}{n-2} - (2^n - 1)$$

$$= \frac{n(n-1)^D}{n\text{-}2} - \frac{n}{n-2} + \frac{n-2}{n-2} - 2^n$$

$$= \frac{n}{n\text{-}2}\left[(n-1)^D - 2\right] - 2^n$$

Because (if *GAP* is small) there are approximately n-1 edges still open for the network at the bottom two levels of the tree, we may connect any particular leaf in one sub-tree to leaf nodes in the n-1 other sub-trees from the root node. It should be possible to reduce D by 1 using the extra branch connections. Probably slightly better bounds can be done; the size of the smallest diameter grows as $O\left(\frac{n}{\log(n\text{-}1)}\right)$. In general, then, the diameter of the network must be greater than D_L.

The lower bound argument leaves open the possibility that logarithmic networks of even smaller diameter than the hypercube exist, although finding a simple topology to describe them may be a difficult task. Higher density networks than the Möbius cube have been achieved. Some examples are the "bubble sort", "pancake", and "n-star" graphs (Akers, Balakrishnan 1989), all of which connect $n!$ nodes of degree (n-1) and have a diameter of $O(n)$.

10. Conclusions

The Möbius cube has a number of properties comparable to those of the hypercube, including: the same number of vertices and edges; expansibility; a vertex degree that is logarithmic to the number of number of nodes; a diameter approximately half that of the hypercube, and an expected routing distance of approximately $n/3 + 1/9$, as compared to $n/2$ for the hypercube, and $3n/8$ for the twisted cube. To achieve the improved diameter, we sacrifice the symmetries and very simple routing algorithm of the hypercube. However, as we have shown, there is a relatively simple routing algorithm for the Möbius cube and this algorithm has $O(n)$ running time.

No parallel algorithms have yet been designed to take advantage of the Möbius cube's network topology. For now, the applications of this network are probably best suited to classes of problems that lack highly symmetric programming solutions - such problems generally cannot take advantage of the symmetries of the network they are run upon. One such example might be a program where each node is an asynchronous process that sends its results to other nodes not necessarily adjacent to it, as in semantic or functional networks; the smaller diameter of the Möbius cube makes it an attractive alternative to the hypercube in programming such problems.

Bibliography

Abraham, Seth, "Issues in the architecture of direct interconnection schemes for multiprocessors," PhD thesis, *CSRD Rpt. No. 977,* University of Illinois, 1990.

Abraham, Seth and Krishnan Padmanabhan, "An analysis of the twisted cube topology," *1989 International Conference on Parallel Processing*, Vol. 1, pp.116-120. Pennsylvania State Press.

Akers, Sheldon .B. and Balakrishnan Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE Transactions on Computers* 38: 4 (April 1989), pp.555-565.

Brown, M.R., "Implementation and analysis of binomial queue algorithms," *SIAM Journal on Computing*, 7: 8 (August 1978), pp. 298-319.

Feng, T.Y., "A survey of interconnection networks," *Computer* 14 (December 1981), pp.12-27.

Hilbers, Peter A. J., R.J. Marion Koopman and Jan L.A. Van de Snepscheut, "The twisted cube," *PARLE: Parallel Architecture and Languages Europe, Volume 1: Parallel Architectures*, pp. 152-8. deBakker, J., Numan, A. and Trelearen, P. (Eds.). Springer Verlag, Berlin, W. Germany, 1987.

Hillis, Daniel W., "The connection machine (computer architecture for the new wave)", *MIT AI Memo 646*, Sept. 1981.

Hillis, Daniel W. *The Connection Machine.* MIT Press, Cambridge Massachusetts, 1982.

Preparata, Franco P. and Jean Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Communications of the ACM* 25 : 5 (May 1981), pp.300-309.

Saad, Youcef and Martin J. Schultz, "Topological properties of hypercubes," *IEEE Transactions on Computers*, 37 : 7 (July 1988), pp. 867-872.

Seitz, Charles L., "Concurrent VLSI Architectures," *IEEE Transactions on Computers*, 33 : 12 (December 1984), pp. 1247-1264.