Order Number 9432333

A new methodology for design and analysis of interprocessor communication networks

Chen, Fu-Wei, Ph.D.

City University of New York, 1994

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

A

A NEW METHODOLOGY FOR DESIGN AND ANALYSIS OF INTERPROCESSOR COMMUNICATION NETWORKS

by

FU-WEI CHEN

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy,

The City University of New York

1994

This Manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

April 14, 1994
Date

Syed Ali Ghozati
Chair of Examining Committee
Professor Seyed-Ali Ghozati

April 14, 1994
Date

Stanley Habib
Executive Officer
Professor Stanley Habib

Professor Stanley Habib

Professor Theodore Brown

Professor Frank Hsu

THE CITY UNIVERSITY OF NEW YORK

ABSTRACT

A NEW METHODOLOGY FOR DESIGN AND ANALYSIS OF

INTERPROCESSOR COMMUNICATION NETWORKS

by

Fu-Wei Chen

Advisor: Professor Seyed-Ali Ghozati

It is well known that the performance of a parallel computer is often limited by the time spent communicating data from one processor to another. The problem of design, analysis, and control of interprocessor communication networks is considered in this thesis.

A novel approach based on a formal modeling of communication network by an algebraic structure, called Communication-Algebra (C-Algebra), is proposed. C-Algebra is used to investigate properties of networks and link them to their control structures. Two classes of communication networks, [N,K] cube and [N,K] PM2I, are modeled by the proposed algebra.

The following are the list of original contributions made in this thesis:

(1) Systematic enhancement of network's performance and its fault-tolerance,

(2) A new approach to establish parallel paths in the network,

(3) Proving that the [N,K] cube is maximally fault tolerant, i.e., the maximum number

of parallel paths between any pair of nodes equal to the degree of each node and the length of each parallel path is at most equal to the Hamming distance between source-destination nodes plus 2,

(4) Developing an adaptive distributed routing scheme which will successfully route messages between any pair of functional nodes in an [N,K] cube as long as these two nodes are connected,

(5) Establishing a lower bound on the number of parallel paths in the class of [N,K] PM2I networks,

(6) Developing several node-independent algorithms for embedding other topologies, such as rings, linear arrays, meshes and standard spanning trees in an [N,K] cube,

(7) Using Gray control sequences to establish Hamiltonian cycles in an [N,K] cube, and investigating properties of control vector space to numerate the number of distinct Gray control sequences in an [N,K] cube,

(8) Graph theoretical characterization of the class of [N,K] cubes to determine whether or not a given graph is isomorphic image of an [N,K] cube.

Acknowledgments

I would like to acknowledge the many people who have contributed to this thesis in so many ways. First, I wish to express my deep gratitude to Professor Seyed-Ali Ghozati, my Ph.D. thesis advisor, for interesting me in parallel processing systems, and for his patience, encouragement, suggestions, and continuous guidance throughout the research in this thesis. Next, I want to thank Professor Stanley Habib, the Executive Officer of the Ph.D. program in Computer Science and a member of my thesis committee, for his willingness to give his time, advice, and encouragement. I am also indebted to the other members of my thesis committee. I deeply appreciate Professor Theodore Brown's suggestions and corrections. I am also grateful to Professor Frank D. Hsu for his helpful suggestions.

Finally, I thank my father, Chien-Hsi, my mother, A-Man, my wife, Hsiu-Chuan, and children, Karen and Annie, for their love, support and patience.

Table of Contents

1. Introduction

The main purpose of a parallel processing system is to complete a large job faster than a single processor system by using several processors concurrently. Another advantage of using multiple processors is better reliability.

Two types of the most popular parallel processing systems are SIMD and MIMD. SIMD (single instruction stream-multiple data stream) machines typically consist of a set N processors, N memories, an interconnection network, and a control unit. The control unit broadcasts instructions to the processors, and all active ("turned on") processors execute the same instruction at the same time. Each processor executes instructions using data from a memory with which only it is associated. The interconnection network allows interprocessor communication. In an SIMD environment, all the processors operate in lockstep, and all active processors use the interconnection network simultaneously.

MIMD (multiple instruction stream-multiple data stream) machines also typically consist of N processors and N memories, but each processor can follow an independent instruction stream. As with SIMD architectures, there is a multiple data stream and an interconnection network.

The performance of a parallel processing system is effected by two major factors. One is f, the fraction of operations in a computation that must be performed sequentially. By Amdahl's law, the maximum speedup, S, achievable by a parallel computer with N processors is $S \leq \frac{1}{f+(1-f)/N} < \frac{1}{f}$.

Another factor is the communication cost caused by the overhead in interprocessor synchronization and its delay in communication network. Thus, the interprocessor communication network is very important for system performance.

We can classify the interprocessor communication networks into three basic types of networks according to the number of links: crossbar network, bus network and interconnection network. The crossbar (or fully connected) network provides direct connection from any source to any destination and minimal delay, but its network complexity and cost become prohibitive for large number of processors. On the other extreme, the bus network has the lowest cost, but its constant bandwidth limits the number of processors. The interconnection network whose cost and performance fall in between

these two extremes is considered more suitable for general parallel processing systems. To solve the problem of providing fast, reliable, and efficient communications at a reasonable cost in large parallel processing systems, many interconnection networks have been proposed, including the baseline [1], the Omega [2], the cube [4], and the Banyan [3].

Interconnection networks can be designed in multistage or single-stage forms. Multistage networks are typically designed using $\log_2 N$ stages of 2x2 switching elements to connect $N=2^n$ inputs to N outputs; each stage has 2^{n-1} switching elements. The minimum requirement of multistage networks is full connectivity, which means that any input terminal can be connected to any output terminal in one pass through the network. In general, there is only one path from any input to any output in a multistage network. Therefore, unless it is modified, multistage networks are not fault tolerant. Due to many attractive properties

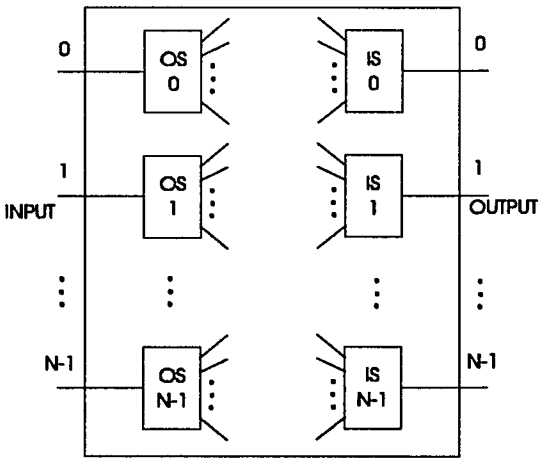

Fig. 1.1 Model of a single-stage network

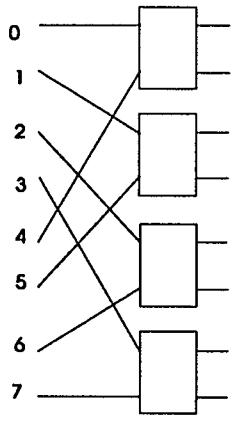

Fig. 1.2 Single-stage Omega network.

of single-stage networks, such as fault-tolerance and rich containment properties, my research is concentrated on this type of communication network. Conceptually, single-stage networks may be viewed as N Input Selectors(IS) and N Output Selectors(OS) [6]. Each IS is essential a 1-to-D demultiplexer and each OS is an M-to-1 multiplexer where $1 \leq D \leq N$ and $1 \leq M \leq N$, as shown in Fig. 1.1. However, other implementations are possible for single-stage networks, for example, the Shuffle-Exchange network [9] with 2^{n-1} 2x2 switching elements consists of one stage of an Omega network (shown in Fig. 1.2)[2]. In the Direct n-cube networks[13], each node consists of a processing element(PE) and a

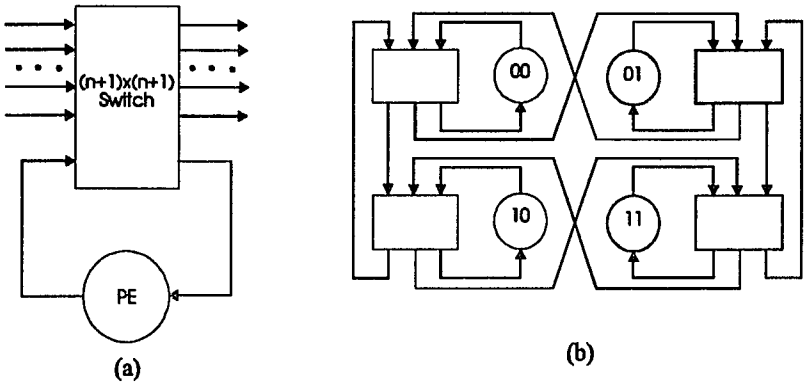

Fig. 1.3 (a)Implementation of a node in a n-cube. (b)Connections in a 2-cube network.

(n+1) x (n+1) crossbar switch as shown in Fig. 1.3. The performance and cost of a single-stage network depend on the actual implementation used [12], [13]. The single-stage network is also called a recirculating network. Data items may have to pass through the single stage several times before reaching their final destinations.

Besides the structure of interconnection networks, a great deal of research has been focused on the analysis of multiprocessor interconnection networks. Of particular interest has been the permuting properties of these networks - the ability to establish interconnections representable as permutations of the processors addresses. Part of the motivation stems from the fact that the communication requirements of several parallel algorithms may be expressed as permutations. Sorting and Fast Fourier Transform are two examples [9]. Another application of permutation network is the alignment of skewed operands when accessing from, or storing to, parallel memories [2]. The permuting properties of many such networks have been studied in detail, and topological equivalence between, and functional relationships among these networks have been established [10], [11]. It has been shown that single-stage network and blocking multistage networks cannot realize arbitrary permutations in a single pass.

Most previous research in interprocessor communication networks emphasis the topology design, and performance and permuting properties analysis. Little attention have been given to modeling "dynamic behavior" of communication networks, i.e., controlling of networks for creating paths and embedding other topologies. Hence, the main objective of this dissertation is to unify modeling and control of single-stage communication networks.

The frame of this research is based on one of my advisor's papers, "Modeling and Analysis of a Class of Interconnection Networks," 1993. We decide to extend his work to modeling

and control of the class of word-controlled symmetric communication networks and to present the modeling technique in more abstract form. To unify the modeling of all different types of communication networks, a mathematical structure called Communication-Algebra or C-Algebra is introduced. The main feature of the proposed algebra is its ability to enhance the network's performance and reliability, by systematically introducing more links, while keeping the number of nodes in the network unchanged. As a result, the diameter of the network reduces and it also becomes more fault tolerant.

Hypercube and PM2I (Plus-Minus 2i) networks are used, as two representative classes of single-stage networks, to demonstrate the application of C-Algebra for both modeling and enhancing the network. Many useful properties of enhanced Hypercube and PM2I networks, called [N,K] cube network and [N,K] PM2I network respectively, is derived using C-Algebra. These properties are related to control sequences instead of node-to-node mapping. Thus, it is shown that by using C-Algebra, communication networks can be designed and most importantly, some of their properties, such as containment properties, can be described in node-independent forms. The concept of node-independent property is central to my research. For instance, if a control sequence creates a ring in the network, from starting node i, then the same sequence produces the same length ring from any other nodes in the network. The same is true for other node independent properties. Thus we can find a set of control sequences to create different topologies or paths in the network, and use them without being concerned about the initiating nodes. Another feature of the

C-Algebra is its capability of investigating the existence of homomorphism between two different networks which aid in simulating one network by another.

Let me summarize the main features of Communication-Algebra:

1. C-Algebra is a design tool for networks modeling and enhancement - relation between the maximum links delay and the degree of enhancement can be established by this algebra
2. C-Algebra can be used to investigate network's properties and relating them to control sequences instead of node-to-node mapping
3. Node-independent containment properties of networks
4. Using C-Algebra to establish homomorphism between networks and using it as a simulation tool.

Besides design and modeling of existing networks, we have tried to design some new network topologies using C-Algebra. For example, networks with bitwise AND, OR or NAND operations. The diagrams of these networks are shown in Fig. 1.4., Fig. 1.5. and Fig. 1.6. respectively. These networks haven't been proven to be of any special use yet, but they show examples of the application of C-Algebra.

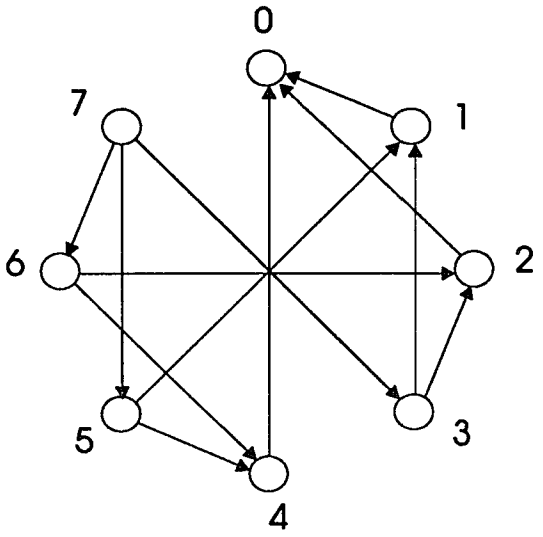

Fig. 1.4. [8,1] AND network

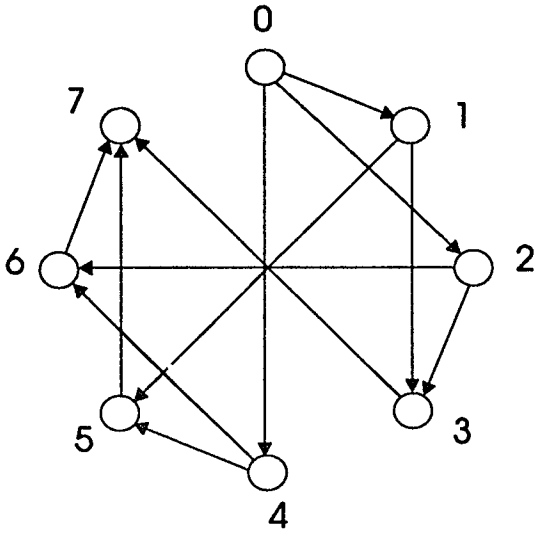

Fig. 1.5. [8,1] OR network

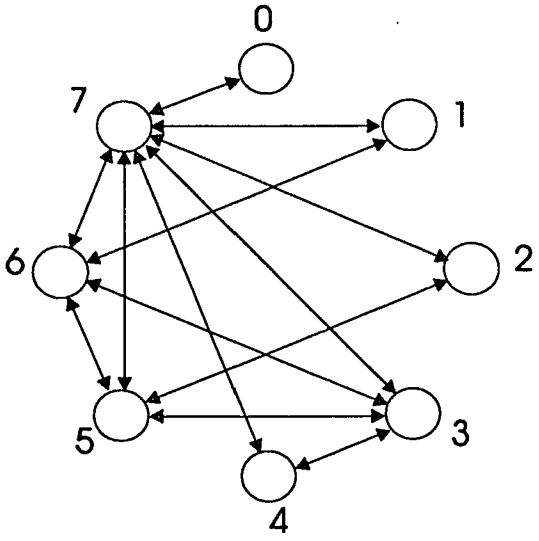

Fig. 1.6. [8,1] NAND network

The following paragraphs provide an overview of the dissertation. Chapter 1 is this introduction. Chapter 2 discusses the definition and properties of Communication-Algebra. Section 2.1 gives the definition of Communication-Algebra. Section 2.2 models the class of [N,K] cube networks. Section 2.3 models the class of [N,K] PM2I networks. Section 2.4 defines equivalent control sequences and investigate their properties. Section 2.5 shows node-independent property of networks and its importance.

Chapter 3 discusses the fault-tolerant networks. Section 3.1 gives the definition and measure of fault-tolerant networks. Section 3.1 investigates the number of disjoint paths (parallel paths) between any pair of nodes in [N,K] cube networks and presents an algorithm, called DP_G, to establish these paths. Section 3.2 develops an adaptive fault-tolerant routing algorithm, which incorporates Algorithm DP_G and depth-first search, will successfully route messages between any pair of functional nodes as long as there exists a path between. Section 3.4 investigates the number of disjoint paths in the [N,K] PM2I networks and obtains a lower bound which is always greater than or equal to the number of disjoint paths in an [N,K] cube networks with the same parameters, i.e., the same value of N and K.

Containment properties of [N,K] cube network are discussed in Chapter 4. Section 4.1 defines ring control sequences and discusses properties of ring which can be embedded into [N,K] cube. A simple algorithm which produces ring control sequences of length between

3 and 2n is given. Section 4.2 develops the theoretical foundation for investigating properties of Gray control sequences and numerates the number of distinct Gray control sequences in an [N,K] cube. Section 4.3 uses Gray control sequence to establish Hamiltonian cycle in an [N,K] cube. Section 4.4 presents a generalized algorithm which produces ring control sequences of any length which can be embedded into an [N,K] cube networks. Section 4.5 discusses how to establish linear arrays and meshes in an [N,K] cube. Section 4.6 discusses the problem of broadcasting in the [N,K] cubes and presents an algorithm for constructing a unique spanning tree.

Chapter 5 discusses the problem of cube identification. Section 5.1 gives notations and definitions which will be used through this chapter. Section 5.2 studies topological properties of [N,K] cubes. Section 5.3 uses structural properties of [N,K] cube, developed in the previous section, to solve the cube identification problem, i.e., to determine whether or not a given graph in isomorphic to an [N,K] cube. Section 5.4 gives some concluding remarks regarding this chapter.

Finally, Chapter 6 concludes this thesis and discusses several topics for continuing research.

2. Communication-Algebra

Many popular interconnection networks used in parallel computer systems can be modeled mathematically by few well-defined mappings, such as rotations, Boolean complement, Generalized complement, etc. For instance, in the case of multistage interconnection networks, basically each stage performs two types of permutations. One represents the switching elements' operations and the other one is to define the topology of the network between stages. Thus, for modeling each stage of the network two distinct mappings are required. In more complex networks, such as Plus-Minus 2i, each stage of the network can be modeled by a set of permutations, in which the number of permutations varies depending on the format of the control vectors (the notion of control vector will formally be defined in section 2.1). Formally, an interconnection network is defined by a set of interconnection functions [6]. Each interconnection function is a bijection (one-to-one and onto mapping) on the sets of input addresses and output addresses. The interconnection functions not only shows the permuting capabilities of the network but also describe the topology of the network. However, there is no intuitive way to enhance the network by introducing

additional links between nodes, especially for the networks with large number of nodes. This is true since the set of interconnection functions becomes very large. Moreover, the interconnection function does not carry any routing information. For the above reasons and to unify the modeling of all different types of communication networks, we present a new methodology for design and analysis of interprocessor communication networks using an algebraic structure called C-Algebra. C-Algebra can be used to describe the topology, permuting functions and control structure of the network while providing a systematic way to design and analyze the enhanced networks.

2.1 Definition of Communication-Algebra

In this section, first we introduce some notations that will be used throughout this thesis. Capital letters denote Boolean n-vectors, i.e., vectors with n components of 0 or 1. A Boolean n-vector is referred to as a **control vector** if it is used to map a node in the network to one of its immediate neighbors. $\mathcal{P}_n$ denotes the set of all control vectors of length n. Control vector P_{ik}, $i \in \{0, 1, ..., \lceil n/k \rceil - 1\}$ and $k \leq n$, is defined as an n-vector of all 0's except in positions ik to ik+k-1 which could be any polarity. This type of control vector, P_{ik}, plays an important role in defining the topology of the network. Combinations of k bits in positions ik to ik+k-1, is called a **k-digit**, in particular it is called **ith k-digit** of vector P_{ik} and P_j denotes the jth component of vector P. The $\lceil n/k \rceil - 1$ th k-digit covers bits

($\lceil n/k \rceil$-1)k to n-1. As an example, for n=6 and k=2 the control vectors P_{0k}, P_{1k}, P_{2k} are defined as: P_{0k}=(0000--), P_{1k}=(00--00) and P_{2k}=(--0000), where - denotes 0 or 1. An abstract definition of the Communication-algebra is presented below:

Definition 2.1: A Communication-algebra is a 3-tuple $\mathbb{C} = \langle \mathbb{N}, \mathcal{P}, \phi \rangle$, where

(i) $\mathbb{N} = \{0, 1, \ldots, 2^n-1\}$ is a nonempty finite set, called **node set.**

(ii) $\mathcal{P}$ is a nonempty set of control vectors $\subseteq \mathcal{P}_n$.

(iii) ϕ is a mapping, called **routing function**, which maps $\mathbb{N} \times \mathcal{P}$ into $\mathbb{N}$, and $\mathbf{I} \in \mathcal{P}$ denotes the identity element respect to ϕ such that $\forall S \in \mathbb{N}$, $\phi(S, \mathbf{I}) = S$.

Definition 2.2: Let $\mathcal{P}$ be the finite set of control vectors and k be a non-negative integer. P^k denotes the sequence obtained by concatenating k elements of $\mathcal{P}$, called **control sequence**, where $P^0 = \mathbf{I}$. The number of occurrence of elements of $\mathcal{P}$, k, is called the length of the sequence. The set of all control sequences defined over $\mathcal{P}$ is denoted by $\mathcal{P}^*$. $(P^k)^m$ denotes the concatenation of m copies of P^k and $\pi(P^k)$ is a control sequence obtained by a permutation on the control vectors forming P^k.

In order to establish a path between two arbitrary nodes in the network, the routing function ϕ may have to be executed several times. Therefore, it is necessary to extend the domain of routing function ϕ to include control sequences.

Definition 2.3: Let ϕ be a routing function defined on the set $\mathbf{N} \times \mathcal{P}_n$. Extension of ϕ, denoted by ϕ^m is defined recursively as follows:

$\phi^0(S, P^0) = \phi^0(S, \mathbf{I}) = S$

$\phi^m(S, P^1P^{m-1}) = \phi^{m-1}(\phi(S, P^1), P^{m-1}),\ \forall\ S \in \mathbb{N}, P^m \in \mathcal{P}^*$.

Definition 2.4: For any $R \subseteq \mathbb{N}$, the set of nodes reachable from any member of R is denoted by $\phi(R)$ and is defined by $\phi(R) = \{\phi^m(S, P^m) \mid S \in R, P^m \in \mathcal{P}^*\}$.

For instance, $\phi(R)$ may represent all the nodes reachable from R by applying all control sequences in $\mathcal{P}^*$. A few useful results concerning set $\phi(R)$ is presented in the following theorem.

Theorem 2.1: Let $R,Q \subseteq \mathbb{N}$ and $P \in \mathcal{P}^*$. Then

1. $R \subseteq \phi(R)$

2. $\phi(\phi(R)) = \phi(R)$

3. $D \in \phi(R)$ if and only if $\phi(\{D\}) \subseteq \phi(R)$

Proof: The proof of Theorem 1 is due to [Ghozati 93] and is repeated here.

(1) $\forall\ S \in R$, $S=\phi(S, \mathbf{I})$, which implies that $S \in \phi(R)$.

(2) By part (1) $\phi(R) \subseteq \phi(\phi(R))$ and for any $S \in \phi(\phi(R))$, $\exists\ i \in \phi(R)$ and control sequence $P \in \mathcal{P}^*$ such that $S=\phi(i, P)$. Moreover, $i \in \phi(R)$ implies that $\exists\ t \in R$ and $P' \in \mathcal{P}^*$ such that $i=\phi(t, P')$ or $S=\phi(\phi(t,P'),P)=\phi(t, P'P)$. Hence $\phi(\phi(R)) \subseteq \phi(R)$.

(3) Let $D \in \phi(R)$ and $Q \in \phi(\{D\})$. Then $\exists\ A^m$ and $B^k \in \mathcal{P}^*$ and $S \in R$, such that $\phi^k(S, B^k)=D$

and $\phi^m(D, A^m)=Q$, which implies that $\phi^m(\phi^k(S, B^k), A^m)=Q$, which in turn implies that $Q=\phi^{m+k}(S,B^kA^m)$ or $Q\in\phi(R)$. Thus $\phi(\{D\}) \subseteq \phi(R)$. Next, if $\phi(\{D\}) \subseteq \phi(R)$, then by part (1), $\{D\}\subseteq\phi(\{D\})\subseteq\phi(R)$, which implies $D\in\phi(R)$. ■

The communication-algebra can model any word-controlled communication network which maps physical address of source into physical address of destination node. For example, the class of [N,K] cube interconnection networks and the class of [N,K] PM2I interconnection networks can be modelled as in the following sections.

2.2 Modeling the Class of [N,K] Cube Networks

First, we define a routing function, called P-complement, which is used, as a manipulative tool to link nodes in the network and define the topology of [N,K] cube network.

Definition 2.5: The P-complement of a vector $X=(x_{n-1}...x_i...x_0)$ is obtained by complementing the components of X according to the value of the corresponding component of P; i.e., x_i is complemented if $p_i = 1$ and uncomplemented if $p_i = 0$. Thus, we define the P-complement as follows:

$X^P = X +_2 P, \forall P\in\mathcal{P}_n$, where $+_2$ denotes bitwise modulo 2 add operation.

Some useful properties of P-complement are listed below:

P1. $X +_2 \mathbf{0} = X$ and $X +_2 \mathbf{1} = X'$,

where $\mathbf{0}=(0,0,\ldots,0)$, $\mathbf{1}=(1,1,\ldots,1)$ and prime indicates Boolean complement.

P2. $P +_2 P = \mathbf{0}$ and $P +_2 P' = \mathbf{1}$.

P3. $X +_2 P = P +_2 X = P +_2 \mathbf{1} +_2 X' = P' +_2 X +_2 \mathbf{1}$

P4. $X +_2 P +_2 P = X$ and $X +_2 P +_2 P' = X'$

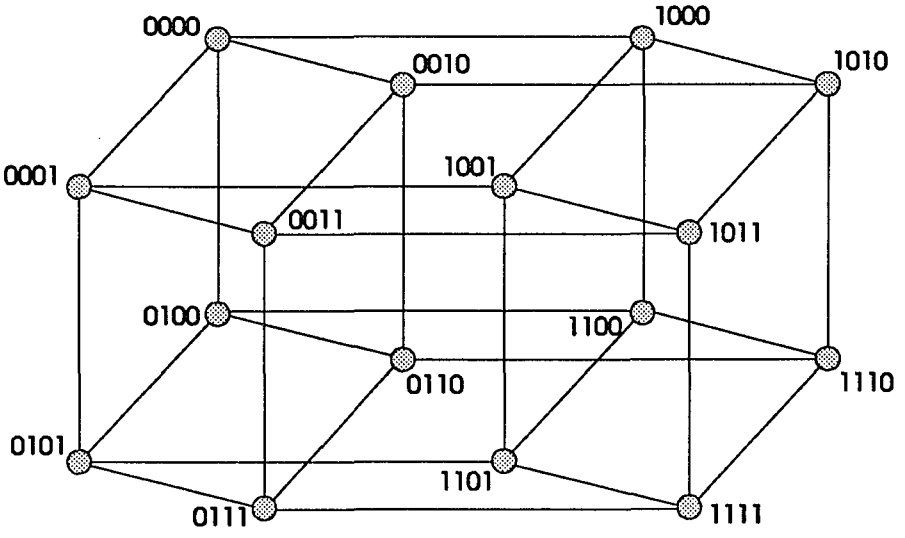

Fig. 2.1. 16-node cube Network

Now, the class of [N,K] cube interconnection network can formally be modelled as follows:

(i) $\mathbb{N} = \{0, 1, \ldots, 2^n-1\}$, where $N=2^n$.

(ii) $\mathcal{P} = \{P_{ik} \mid i \in \{0, 1, \ldots, \lceil n/k \rceil -1\}, k \langle n \}$, where $K=2^k$.

(iii) $\phi(S, P) = S +_2 P$, $\forall S \in \mathbb{N}, P \in \mathcal{P}$.

If we restrict k to one and $\mathcal{P}$={ **0**, P_0 , P_1 , . . .,P_{n-1} }, i.e., P_0=(0,0, . . .,1), P_{n-1}=(1,0,0, . . .,0), then, [N,K] cube becomes a cube network, as shown in Fig. 2.1. Otherwise, for k ≥ 2, it is a [N,K] cube network. Fig. 2.2. shows a [16,4] cube network.

Since $\phi(P, P) = \mathbf{0}$ and $\phi(P, P') = \mathbf{1}$, $\forall X \in N$, these two nodes are referred to as the fixed nodes of the class of [N,K] cube networks. Since each k-digit has K-1 non-zero control vectors except the $\lceil n/k \rceil$-1 th k-digit covers bits ($\lceil n/k \rceil$-1)k to n-1 and there exists $\lceil n/k \rceil$ k-digits in the network, we have the following proposition.

Proposition 2.1: The number of non-zero control vectors and the degree of each node in an [N,K] cube network are the same and is $\lfloor n/k \rfloor (K-1)+2^{n \bmod k}-1$.

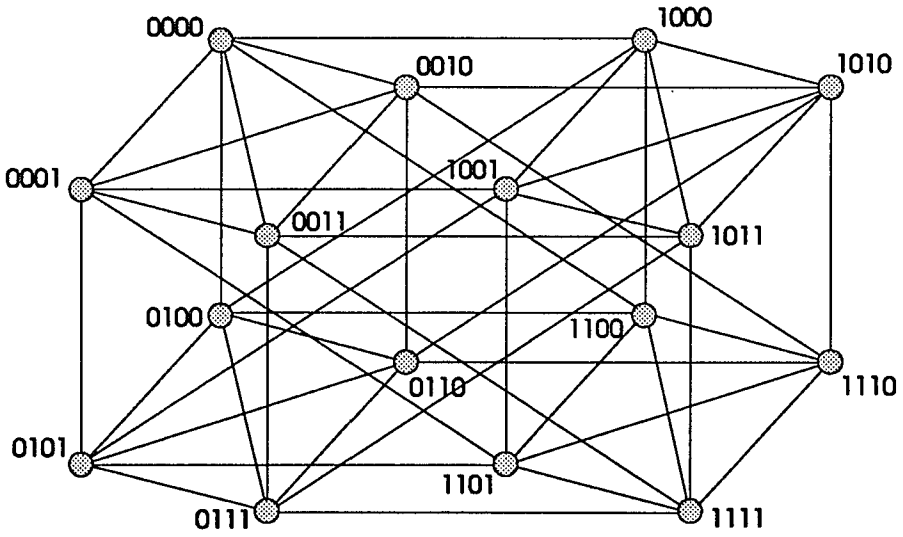

Fig. 2.2. [16,4] cube Network

Definition 2.6: $S+_2D=T$ is called the relative address of node S with respect to node D.

Definition 2.7: Let $H_k(S, D)$ be the number of non-zero k-digit in T. Then $H_k(S, D)$ is called the Hamming distance between the nodes S and D.

Example 2.1: In [64,4] cube network, the relative address of node 9 with respect to node 21 is $(001001) +_2 (010101) = (011100)$. The Hamming distance between node 9 and node 21 is $H_2(9, 21) = 2$.

To establish a path from node S to node D with $H_k(S, D)=\alpha$, one may choose a sequence of control vectors such that each vector reduces the Hamming distance by 1. Thus a control sequence of length no greater than $H_k(S, D)$ can be formed to connect S to D.

Proposition 2.2: The minimum distance between the nodes S and D in [N,K] cube networks is $H_k(S, D)$.

Proof: When a control vector is applied, Hamming distance to node D is decreased at most by one. Thus, there is no path of length smaller than Hamming distance between the nodes S and D. ■

Example 2.2: In [64,4] cube network, from Example 2.1, the Hamming distance from node 9 to node 21 is $H_2(9, 21) = 2$ and $T = (011100)$. We could choose

P^2=(010000)(001100) as the control sequence of length 2 to connect node 9 to node 21.

2.3 Modeling the Class of [N,K] PM2I Networks

The control vector, $P=(p_n , p_{n-1} , \ldots , p_0)$, is a Boolean (n+1) vector, where bit n is the sign bit; i.e., P is positive if p_n=0, P is negative if p_n=1. We denote $-P_{n-1}$ by (1, 1 , 0, . . ., 0) , P_{n-1} by (0, 1, 0, . . ., 0) and -P by (p_n ', p_{n-1} , . . ., p_0).

To model the class of PM2I communication networks by C-Algebra, the following routing function is defined and its properties are studied.

Definition 2.8: The routing function, P-addition, denoted by $+_N$, is defined as follows: $X +_N P = (X + P + iN) \bmod N$, where i is a non-negative integer such that $X+P+(i-1)N \angle 0 \le X+P+iN$, $\forall P \in \pm \mathcal{P}_n$ and N is the number of nodes in the network.

Some useful properties of P-addition are listed below:

P1. $X +_N \mathbf{0} = X$

P2. $P +_N (-P) = \mathbf{0}$

P3. $X +_N P +_N (-P) = X$

P4. $X +_N P = P +_N X$

Thus, the class of [N,K] PM2I interconnection networks can be modelled as follows:

(i) $\mathbb{N} = \{0, 1, \ldots, 2^n-1\}$, where $N=2^n$.

(ii) $\mathcal{P} = \{P_{ik}, -P_{ik} \mid i \in \{0, 1, \ldots, \lceil n/k \rceil-1\}, k < n\}$, where $K=2^k$.

(iii) $\phi(S, P) = S +_N P$, $\forall S \in \mathbb{N}, P \in \mathcal{P}$.

If we restrict k to one, it is a PM2I network. Otherwise, for $k \geq 2$, it is a [N,K] PM2I network.

Proposition 2.3: The number of non-zero control vectors, in an [N,K] PM2I network, is $2[\lfloor n/k \rfloor(K-1)+2^{n \bmod k}-1]$ and the degree of each node is $2\lfloor n/k \rfloor(K-1)+2^{n \bmod k}-1$.

Proof: Since each k-digit has 2(K-1) non-zero control vectors, total of non-zero control vectors are $2[\lfloor n/k \rfloor(K-1)+2^{n \bmod k}-1]$. However the degree of each node is $2\lfloor n/k \rfloor(K-1)+2^{n \bmod k}-1$ because $\forall P_{(\lceil n/k \rceil-1)k} \exists P'_{(\lceil n/k \rceil-1)k} \neq P_{(\lceil n/k \rceil-1)k}$ such that $\phi(S, P_{(\lceil n/k \rceil-1)k})=\phi(S, P'_{(\lceil n/k \rceil-1)k})$. Therefore two most significant k-digits of control vectors define the same link and only $2^{n \bmod k}-1$ links are used by $2(2^{n \bmod k}-1)$ most significant k-digit control vectors. ■

Definition 2.9: $D+_N(-S)=T$ is called the relative address of node S with respect to node D. $\bar{T}$ is two's complement of T.

Definition 2.10: Hamming distance from node S to node D in the [N,K] PM2I networks, , $H_k(S, D)$, is defined to be the minimum number of non-zero k-digit in T or $\bar{T}$.

Example 2.3: In [64,4] PM2I network, the relative address of node 9 with respect to node 21 is $T = (010101) +_N (-(001001)) = (0001100)$. $\bar{T} = (1110100)$ The Hamming distance from node 9 to node 21 is $H_2(9, 21) = 1$.

To reach node D from node S, similar to [N,K] cube, we could apply the control vectors obtained by setting all k-digit to zero except one k-digit to a non-zero k-digit in T one by one and setting the sign bit P_n to T_n if T has smaller number of non-zero k-digit than $\bar{T}$. Otherwise, we use $\bar{T}$ instead of T. The length of this path is equal to the Hamming distance from node S to node D.

Proposition 2.4: The minimum distance from node S to node D, in an [N,K] PM2I networks, is $H_k(S, D)$.

Proof: When a control vector is applied, Hamming distance to node D is decreased at most by one. Thus, there is no path of length smaller than Hamming distance between the nodes S and D. ■

Example 2.4: In [64,4] PM2I network, from Example 2.3, The Hamming distance from node 9 to node 21 is $H_2(9, 21) = 1$, which is determined by $T = (010101) +_N (-(001001)) = (0001100)$. Therefore, we could choose P^1=(0001100) as the control sequence of length 1 to connect node 9 to node 21.

2.4 Equivalent Control Sequences and their Properties

To facilitate comparison of control sequences and to study their relations, we define a mapping on the set $\mathcal{P}^*$ into set $\mathcal{P}$ as follows:

Definition 2.11: Let $P^\alpha = P_1P_2 \ldots P_\alpha \in \mathcal{P}^*$. Define function $\sigma : \mathcal{P}^* \rightarrow \mathcal{P}_n$ as : $\sigma(P^\alpha) = Q$ if and only if ϕ is one-to-one defined on $\mathcal{P}_n$ and $\exists Q \in \mathcal{P}_n$, such that

$$\phi^\alpha(S, P^\alpha) = \phi(S, Q). \qquad (2.1)$$

$\sigma(P^\alpha)$ is referred to as the **S-signature** of P^α (Note: Q may not belong to $\mathcal{P}$.).

Example 2.5: ([N,K] PM2I interconnection networks) Let $P^\alpha = P_1P_2 \ldots P_\alpha \in \mathcal{P}^*$. By solving eq.(2.1) one can obtain $\sigma(P^\alpha) = Q = P_1 +_N P_2 +_N \ldots +_N P_\alpha$ as the **S-signature** of P^α.

Example 2.6: ([N,K] cube interconnection networks) Let $P^\alpha = P_1P_2 \ldots P_\alpha \in \mathcal{P}^*$. Since $\phi^\alpha(S, P^\alpha) = S +_2 P_1 +_2 P_2 \ldots +_2 P_\alpha$, the **S-signature** of P^α is defined as the following:

$\sigma(P^\alpha) = P_1 +_2 P_2 +_2 \ldots +_2 P_\alpha$

The control sequences are means for setting up paths between different nodes in the network. Since two nodes can be connected by several different paths, the corresponding control sequences are related. The relation between control sequences

forced on $\mathcal{P}^*$ by the routing function ϕ is defined and studied next.

Definition 2.12: Let P^α and P^β represent two control sequences of length α and β respectively. P^α and P^β are said to be **S-equivalent**, $P^\alpha \equiv^S P^\beta$ if and only if $\phi^\alpha(S, P^\alpha) = \phi^\beta(S, P^\beta)$, $S \in \mathbb{N}, \forall P^\alpha, P^\beta \in \mathcal{P}^*$.

Lemma 2.1: S-equivalence defined on $\mathcal{P}^*$ is an equivalence relation.

Proof: It follows from the definition 2.12 that S-equivalent is reflexive, symmetric and transitive, thus it is an equivalence relation. ■

An equivalence relation on a set partitions the set into equivalence classes. Let $\mathcal{P}^*/\equiv$ denotes the set of equivalence classes determined on $\mathcal{P}^*$ by the routing function ϕ and source node address S. The members of a class are indistinguishable from one another by their action on ϕ. An equivalence class of $\mathcal{P}^*$ may be represented by any of its members and we shall denote such a class by $[\ P^\alpha\ ]$.

Theorem 2.2: The number of disjoint equivalent classes; i.e., cardinality of the set $\mathcal{P}^*/\equiv$, induced by ϕ on S and $\mathcal{P}^*$ in any interconnection network with N nodes is N.

Proof: Proof is by contradiction. Assume that there are N+1 disjoint classes in $\mathcal{P}^*/\equiv$. By definition 2.12, all the control sequences in a classes must connect node S to the same node. By Lemma 2.1; i.e., S-equivalent is an equivalent relation and partitions the

set $\mathcal{P}^*$ into equivalent classes, these disjoint classes of control sequences connect node S to N+1 different nodes in the network. But there are N nodes in the network. Thus, the number of disjoint equivalent classes in $\mathcal{P}^*/\equiv$ is no more than N. On the other hand, assume that there are N-1 disjoint equivalent classes in $\mathcal{P}^*/\equiv$. Because the basic requirement of interprocessor communication networks is that any source node can reach any other nodes; i.e., source node, S, can reach all of N-1 other nodes and itself by $\phi^0(S, \mathbf{I}) = S$, there are at least N disjoint equivalent classes in $\mathcal{P}^*/\equiv$. Thus, the number of disjoint equivalent classes in $\mathcal{P}^*/\equiv$ is exactly N. ■

Equivalency of control sequences can be linked directly to their signatures. This greatly simplifies identification of equivalent control sequences. In the next theorem, control sequences and their signatures are explored.

Theorem 2.3: Let P^α and P^β be defined over the set $\mathcal{P}$. Then, P^α and P^β are S-equivalent if and only if $\sigma(P^\alpha) = \sigma(P^\beta)$.

Proof: Let P^α and P^β be S-equivalent. By definition 2.12, $\phi^\alpha(S, P^\alpha) = \phi^\beta(S, P^\beta)$. But by definition 2.11, $\phi^\alpha(S, P^\alpha)=\phi(S, \sigma(P^\alpha))$, $\phi^\beta(S, P^\beta) = \phi(S, \sigma(P^\beta))$ and ϕ is one-to-one on $\mathcal{P}_n$, which implies that $\sigma(P^\alpha) = \sigma(P^\beta)$.

The necessary condition can be proven by a similar argument. ■

Corollary 2.1: Let $\pi(P^\alpha)$ denote a control sequence formed by a permutation on control vectors in P^α. Then P^α and $\pi(P^\alpha)$ are S-equivalent in [N,K] cube and [N,K] PM2I networks.

Proof: Since $\sigma(P^\alpha) = \sigma(\pi(P^\alpha))$, it then follows from theorem 2.3 that the two sequences are S-equivalent. ■

Corollary 2.2: In an [N,K] cube network, $(P^\alpha)^{2m} \equiv^S \mathbf{0}$, and $(P^\alpha)^{2m+1} \equiv^S P^\alpha$, $\forall P^\alpha \in \mathcal{P}^*$.

Proof: Follows from theorem 2.3 and can also be proven by induction on m, and using the following identies: $P^\alpha +_2 P^\alpha = \mathbf{0}$, and $P^\alpha +_2 P^\alpha +_2 P^\alpha = P^\alpha$. ■

2.5 Node-Independent Networks

Definition 2.7: A communication network is said to be **node independent** if $P^\alpha \equiv^S P^\beta$ implies $P^\alpha \equiv^D P^\beta$, $\forall D \in \mathbb{N}$. In such a network, we simply say that P^α and P^β are **equivalent** and $\sigma(P^\alpha)$ is the **signature** of P^α.

Next we prove that the class of [N,K] cube and [N,K] PM2I networks are node independent.

Theorem 2.4: The class of [N,K] cube networks is node independent.

Proof: Let $P^{\alpha} = A_1A_2 \ldots A_{\alpha}$ and $P^{\beta} = B_1B_2 \ldots B_{\beta}$, where A_i , $B_i \in \mathcal{P}$.

$\phi(S, P^{\alpha}) = \phi(S, P^{\beta})$ implies $S+_2A_1+_2A_2 +_2 \ldots +_2A_{\alpha} = S+_2B_1+_2B_2 +_2 \ldots +_2B_{\beta}$.

Add $(D +_2 S)$ to both side of the above equality and rearrange the terms to obtain

$\phi(D, P^{\alpha}) = \phi(D, P^{\beta})$, which implies that $P^{\alpha} =^{D} P^{\beta}$. ■

Theorem 2.5: The class of [N,K] PM2I networks are node independent.

Proof: Let $P^{\alpha} = A_1A_2 \ldots A_{\alpha}$ and $P^{\beta} = B_1B_2 \ldots B_{\beta}$, where A_i , $B_i \in \mathcal{P}$.

$\phi(S, P^{\alpha}) = \phi(S, P^{\beta})$ implies $S+_NA_1+_NA_2 +_N \ldots +_NA_{\alpha} = S+_NB_1+_NB_2 +_N \ldots +_NB_{\beta}$.

By adding $D+_N(-S)$ to both side of the above equality and rearranging the terms one can obtain $\phi(D, P^{\alpha}) = \phi(D, P^{\beta})$, which implies $P^{\alpha} =^{D} P^{\beta}$. ■

Theorem 2.6: In a node independent network, if $P^{\alpha} \equiv P^{\beta}$ and $P^{\alpha'} \equiv P^{\beta'}$, then $P^{\alpha}P^{\alpha'} \equiv P^{\beta}P^{\beta'}$. Where $P^{\alpha}P^{\alpha'}$ denotes concatenation of P^{α} and $P^{\alpha'}$.

Proof: $P^{\alpha} \equiv P^{\beta}$ implies

$$\forall S \in \mathbb{N}, \exists D \in \mathbb{N}, \phi(S, P^{\alpha}) = \phi(S, P^{\beta}) = D \tag{2.2}$$

, and $P^{\alpha'} \equiv P^{\beta'}$ implies

$$\phi(D, P^{\alpha'}) = \phi(D, P^{\beta'}) \tag{2.3}$$

Substituting D in (2.3) by equation (2.2), we have

$$\phi(\phi(S, P^{\alpha}), P^{\alpha'}) = \phi(\phi(S, P^{\beta}), P^{\beta'}).$$

By definition 2.3, the above equation implies

$$\phi(S, P^{\alpha}P^{\alpha'}) = \phi(S, P^{\beta}P^{\beta'})$$

which in turn implies $P^{\alpha}P^{\alpha'} = P^{\beta}P^{\beta'}$. ■

Node-independent property of a network enables us to develop node-independent algorithms, for embedding other topologies, such as rings linear arrays and meshes in the network. For instance, if a control sequence creates a ring in the network, from starting node i, then the same sequence produces the same length ring from any other nodes in the network. The same is true for other node independent properties. Thus we can find a set of control sequences to create different topologies or paths in the network, and use them without being concerned about the initiating nodes.

3. Fault Tolerant Networks

The reliability of interconnection networks are important to the overall system performance. In designing a fault-tolerant multiprocessor computer system, it is advantageous to use an interconnection network with more than one path between processor nodes. C-Algebra can be effectively used to increase the degree of reliability of the network in a systematic manner.

This chapter deals with the following two important topics:

1. Control strategy of communication networks that stays operational, i.e., creates a path between nodes, in spite of the link or node failures.
2. To define and evaluate reliability of communication networks..

Interconnection networks, as a major component of any parallel computer system, are being used in various areas, such as controlling spacecraft, monitoring hospital patients and other real time control applications. Failure of a single node or link may result in a disaster if it

is not fault tolerant. It is often impossible to perform manual maintenance on a complex interconnection network.

An Interconnection Network(IN) can be viewed as a set of switches(nodes) and links. In an IN, a failure is defined as an event in which the network does not perform its service in the manner specified. A fault is the physical cause of an error. Obviously, the link or node failure may causes some nodes not being able to communicate with each other, which in turn causes error in the computation, so failure is a physical cause of this error. Thus, from a system point of view, a node or link failure is a fault.

Links and nodes faults can be classified by their duration (permanent faults or transient faults), or by their values (determinate faults or indeterminate faults). One important determinate fault is called the link fault or stuck-type fault. It causes the link voltage to become fixed at a constant value.

Fault-tolerant networks are defined as the ability to create paths between two functional nodes regardless of other links or nodes failures. In all INs, new data paths can be established via alternate control sequences (i.e., equivalent control sequences). Fault tolerance in INs can be achieved by means of redundancy, which is inherent in single-stage INs. Path-level redundancy in INs is static. That is, it is permanent part of the network.

3.1 Definition and Measure of Fault-Tolerant Networks

In this section, we define strong and weak fault-tolerant networks and use Communication-Algebra to investigate their properties.

A network is said to be fully (or strongly) fault tolerant if for any pair of nodes in the network, there exist at least two paths connecting them together, or a network may only provide fault tolerant to some pair of nodes .

Definition 3.1: (a). A network is said to be **⟨S, D⟩ fault tolerant** if there exist at least two paths between nodes S and D.

(b). A network is said to be **strongly fault tolerant** if it is fault tolerant for every pair of nodes ⟨S, D⟩.

(c). A network which is ⟨S, D⟩ fault tolerant, but not strong fault tolerant, is referred to as **weak fault tolerant**.

We consider two types of faults which can occur in the networks.

1. A *link fault* is the failure of an individual link connecting two nodes.

2. A *node fault* is the failure of a node.

A node fault is equivalent to the set of link faults incident to the faulty node. This model is more conservative than stuck-at 0 or 1 models.

When faulty links or faulty nodes are known to exist and can be identified, it may be possible to avoid them by choosing an alternate path or node, i.e.,equivalent, control sequence. If a conflict occurs, the ability to dynamically use the alternate paths improve the throughput as well as fault tolerance.

The node-fault tolerance between a pair of nodes S and D or the node-fault tolerance of an undirected graph is measured by the vertex connectivity.

Definition 3.2: A pair of nodes S and D are said to have **vertex connectivity** $\xi(S, D)$ if $\xi(S, D)$ is the maximum number of faulty nodes that the network can tolerate and still remains $\langle S, D \rangle$ connected.

Definition 3.3: A graph G is said to have a **vertex connectivity** $\xi(G)$ if the graph G remains connected in presence of up to $\xi(G)$ node failures.

Proposition 3.1: $\xi(G) = \min\{\xi(S, D) | \forall S, D \in \mathbb{N} \text{ and } S \neq D\}$.

Proof: Let $\xi(G) = i, i \in \{0,1,...,N\}$.

By Definition 3.2, the graph G remains connected when an arbitrary set of less than i nodes are faulty. This implies that any pair of nodes in G remain connected when an arbitrary set of less than i nodes are faulty, which in turns implies that $\forall S, D \in \mathbb{N}$ and $S \neq D$: $\xi(S, D) \geq i \Rightarrow \min\{\xi(S, D) | \forall S, D \in \mathbb{N} \text{ and } S \neq D\} = i$.

Let i be the smallest member of the set $\{\xi(S, D) | \forall S, D \in \mathbf{N}$ and $S \neq D\}$.

By Definition 3.1, S and D remain connected when an arbitrary set of less than i nodes are faulty, which implies that the graph G remains connected when an arbitrary set of less than i nodes are faulty which in turn implies that $\xi(G) = i$. ■

Obviously, the vertex connectivity of a graph G cannot exceed its minimum node degree. Since if k divides n, then the degrees of each node in [N,K] cube networks and [N,K] PM2I networks are n/k(K-1) and (2(n/k)-1)(K-1) respectively, in the next proposition, we establish an upper bound on $\xi(G)$ for both cube and PM2I networks.

Proposition 3.2: (a). ξ([N,K] cube network) $\leq$ n/k(K-1) and

(b). ξ([N,K] PM2I network) $\leq$ (2(n/k)-1)(K-1).

Since the vertex connectivity of a graph G cannot exceed its minimum node degree, a graph with vertex connectivity equal to its minimum node degree is said to be maximally fault tolerant, and is defined next.

Definition 3.4: A graph is called **maximally fault tolerant** if vertex connectivity of the graph equals its minimum node degree.

In a fault-tolerant communication network, it is possible to set up several node-disjoint paths between any source-destination pair of nodes. In what follows, we defined node-disjoint control sequences and later we show that these sequences are capable of creating node-disjoint paths in the network. What is unique to this approach is the fact that node-disjoint control sequences are equivalent and thus can be applied to any source node in the network.

Definition 3.5: Two control sequences, P^{α} and P^{β} are said to be **disjoint**, and their corresponding path, **node-disjoint**, if and only if

$$\phi^{\alpha}(S, P^{\alpha}) = \phi^{\beta}(S, P^{\beta}) = D \text{ and } \phi^{i}(S, P^{i}) \neq \phi^{j}(S, P^{j}) \; \forall i<\alpha, j<\beta.$$

Our purpose in this chapter is to show that the class of [N,K] cube Networks have the vertex connectivity of n/k(K-1). Hence these networks are maximally fault tolerant. Then, we will present a simple routing algorithm to establish maximum number of node-disjoint paths between two arbitrary nodes simultaneously. We also present an adaptive distributed routing scheme to route messages between two fault-free nodes in faulty [N,K] cube networks as long as the two nodes remain connected.

3.2 Number of Disjoint Paths in the [N, K] Cube Networks

First, we present a theorem[Saad 88] to show that there exists n disjoint paths between any pair of nodes in an n-cube network. Later we will extend the theorem by using Communication-Algebra to find the number of disjoint paths in the [N,K] cube Networks.

Theorem 3.1: Let A, B be any two nodes of an n-cube and assume that $H_1(A, B) < n$. Then there exist n parallel paths between A and B. Moreover, the length of each path is at most $H_1(A, B)+2$.

Proof: See [Saad 88].

Lemma 3.1: Let S, D be any two nodes in an [N,K] cube network and $m=H_k(S, D) \le \lceil n/k \rceil$ be the number of non-zero k-digits in $T=S+_2D$. Then there exist m disjoint paths of length m between the nodes S and D.

Proof: $S+_2D$ can be decomposed into m non-zero control vectors $P^m=P_{j0k}P_{j1k} \ldots P_{j(m-1)k}$. Next, we construct the following control sequences:

$P(0)^m=P_{j0k}P_{j1k} \ldots P_{j(m-1)k}$,

$P(1)^m=P_{j1k}P_{j2k} \ldots P_{j(m-1)k}P_{j0k}$,

. . . ,

$P(i)^m=P_{jik}P_{j(i+1)k} \ldots P_{j(m-1)k}P_{j0k} \ldots P_{j(i-1)k}$,

. . . , and

$P(m-1)^m=P_{j(m-1)k}P_{j0k} \ldots P_{j(m-2)k}$

It is clear that $\sigma(P(i)^m)=\sigma(P^m)$, $\forall i : 0 \le i \le m-1$ which implies that all these control sequences are equivalent, i.e., connecting the nodes S and D. Since all P_{jik} modify different k-digit of the source node address, $\sigma(P(i)^l) \neq \sigma(P(j)^l)$, $\forall i \neq j$, $l<m$ which implies that they are disjoint. ■

Theorem 3.2: Let S, D be any two nodes in an [N,K] cube network, and $m=H_k(S, D) \le n/k$ be the number of non-zero k-digits in $T=S+_2D$, where n is assumed to be multiple of k. Then there exist $n/k(K-1)$ disjoint paths between S and D. Moreover, the length of each path is at most m+2.

Proof: Note that constructive proof given next yield exactly m paths of length m, m(K-2) of length m+1 and (n/k-m)(K-1) of length m+2. For convenience we assume without loss of generality that S and D differs in their m leading k-digits. Then the set of control vectors available to node S can be divided into the following 3 classes:

Class 1: $P_{ik} \in \{P_{j0k}, P_{j1k}, \ldots, P_{j(m-1)k}\}$. By Lemma 3.1, there are m disjoint paths of length m in this class.

Class 2: P_{ik}: $0 \le i \le m-1$, $P_{ik} \neq P_{jik}$. We construct the control sequences of length m+1 starting from P_{ik} as follows: $P(ik)^{m+1}=P_{ik}P_{j0k}P_{j1k} \ldots P_{j(i-1)k}P_{j(i+1)k} \ldots P_{j(m-1)k}(P_{ik}+_2P_{jik})$. Since $P_{ik}+_2(P_{ik}+_2P_{jik})=P_{jik}$, $\sigma(P(ik)^{m+1})=\sigma(P^m)$, $\forall i : 0 \le i \le m-1$. Therefore $P(ik)^{m+1}$ and P^m are equivalent. There are m(K-2) different P_{ik}'s in class 2.

Class 3: P_{ik}: $m \le i \le n/k-1$. We construct the control sequences of length m+2 starting

from P_{ik} as follows: $P(ik)^{m+2}=P_{ik}P_{j0k}P_{j1k}\ldots P_{j(m-1)k}P_{ik}$. Since $P_{ik}+_2P_{ik}=\mathbf{0}$, $\sigma(P(ik)^{m+2})=\sigma(P^m)$, $\forall i : m\le i\le n/k-1$. Therefore $P(ik)^{m+2}$ and P^m are equivalent. There are (n/k-m)(K-1) different P_{ik}'s in class 3.

In above 3 classes, all the control sequences start from different P_{ik}, then the rest of control vectors are never the same k-digit as P_{ik} or only the last control vector is the same k-digit as P_{ik}. Therefore they are disjoint until reaching the destination D. ■

Corollary 3.1: [N,K] cube network is maximally fault tolerant.

The following algorithm, called DP_G(S, D) (disjoint paths generator), based on the Theorem 3.2, is proposed to generate n/k(K-1) disjoint paths between the nodes S and D in [N,K] cube networks. The complexity of Algorithm DP_G is in O(n/k(K-1)).

Algorithm DP_G(S, D): /* Disjoint Paths Generator */

1. Find the relative address $T=S+_2D$, between the two nodes.
2. **for** each of m non-zero k-digits in T, construct P_{jik} such that the ith non-zero k-digit of T and P_{jik} are identical.
3. **for** each $P_{ik} \in \mathcal{P}$, construct the control sequence to connect node S and node D as follows:

   if $P_{ik} \in \{P_{j0k}, P_{j1k}, \ldots, P_{j(m-1)k}\}$ **then**

   send $P^m= P_{jik}P_{j(i+1)k}\ldots P_{j(m-1)}P_{j0k}P_{j1k}\ldots P_{j(i-1)k}$

else if the ith k-digit of T is non-zero **and** $P_{ik} \notin \{P_{j0k}, P_{j1k}, \ldots, P_{j(m-1)k}\}$ **then**

send $P^{m+1}=P_{ik}P_{j0k}P_{j1k} \ldots P_{j(i-1)k}P_{j(i+1)k} \ldots P_{j(m-1)k}(P_{ik}+P_{jik})$

else send $P^{m+2}=P_{ik}P_{j0k}P_{j1k} \ldots P_{j(m-1)k}P_{ik}$

Example 3.1: In an [16,4] cube network, there exist (4/2)(4-1)=6 disjoint paths between nodes 2 and 10. Bold solid lines in Fig. 3.1 show these 6 disjoint paths.

Since T(2, 10) = (0010) $+_2$(1010) = (1000), H_2(2, 10) = 1, there exist 1 path of length 1, 2 paths of length 2 and 3 paths of length 3, which can be established by the following control sequences:

$P(1)^1$=(1000),

$P(1)^2$=(0100)(1100),

$P(2)^2$=(1100)(0100),

$P(1)^3$=(0001)(1000)(0001),

$P(2)^3$=(0010)(1000)(0010), and

$P(3)^3$=(0011)(1000)(0011).

The nodes in each paths are as follows:

$P(1)^1$= 2→10,

$P(1)^2$= 2→6→10,

$P(2)^2$= 2→14→10,

$P(1)^3$= 2→3→11→10,

$P(2)^3$= 2→0→8→10, and $P(3)^3$= 2→1→9→10.

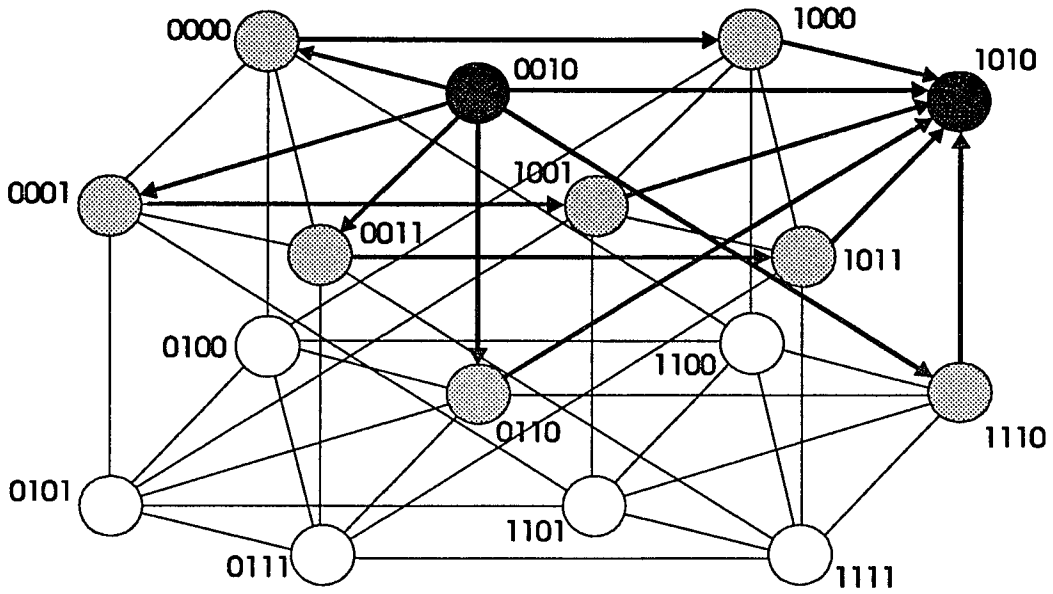

Fig. 3.1. disjoint paths between the nodes 2 and 10 in [16,4] cube network

Note that the above set of control sequences also establishes all the node-disjoint paths between any two nodes with relative address of T(S, D) = (1000). For example, between the nodes 11 and 3, the following 6 paths, as shown in Fig. 3.2, are established by the same above control sequences:

$P(1)^1$= 11→3,

$P(1)^2$= 11→15→3,

$P(2)^2$= 11→7→3,

$P(1)^3$= 11→10→2→3,

$P(2)^3$= 11→9→1→3, and

$P(3)^3$= 11→8→0→3.

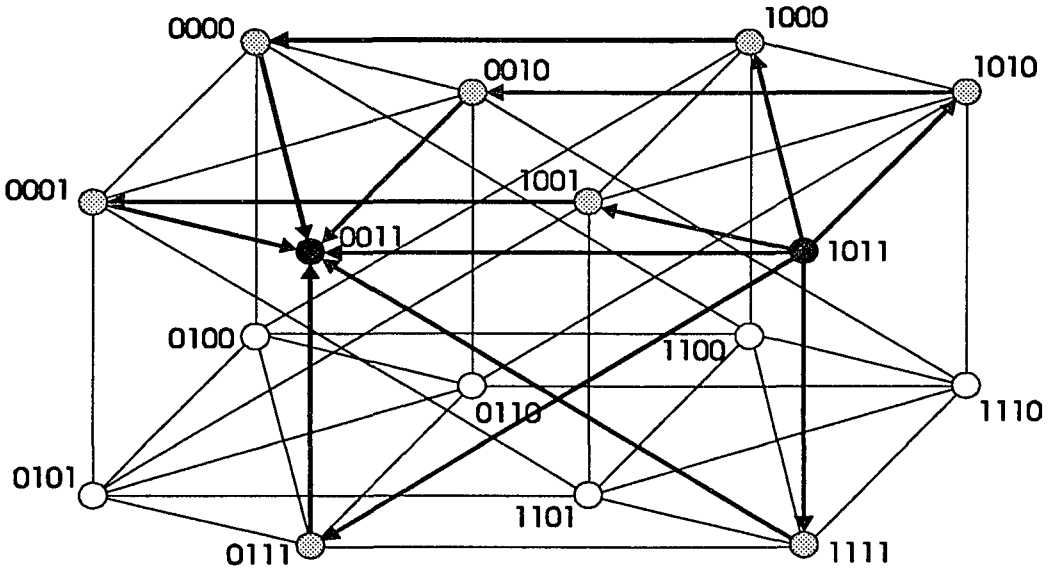

Fig. 3.2. disjoint paths between the nodes 11 and 3 in [16,4] cube network

Example 3.2: In an [64,4] cube network, there exist (6/2)(4-1)=9 disjoint paths between the nodes 0 and 9. Since $T(0, 9) = (000000) +_2(001001) = (001001)$, $H_2(0, 9) = 2$, there exist 2 paths of length 2, 4 paths of length 3 and 3 paths of length 4, which can be established by the following control sequences:

$P(1)^2$=(001000)(000001),

$P(2)^2$=(000001)(001000),

$P(1)^3$=(001100)(000001)(000100),

$P(2)^3$=(000100)(000001)(001100),

$P(3)^3$=(000010)(001000)(000011),

$P(4)^3$=(000011)(001000)(000010),

$P(1)^4$=(010000)(001000)(000001)(010000),

$P(2)^4$=(100000)(001000)(000001)(100000), and

$P(3)^4$=(110000)(001000)(000001)(110000).

The nodes in each paths are as follows:

$P(1)^2$= 0→8→9,

$P(2)^2$= 0→1→9,

$P(1)^3$= 0→12→13→9,

$P(2)^3$= 0→4→5→9,

$P(3)^3$= 0→2→10→9,

$P(4)^3$= 0→3→11→9,

$P(1)^4$= 0→16→24→25→9,

$P(2)^4$= 0→32→40→41→9, and

$P(3)^4$= 0→48→56→57→9.

3.3 Adaptive Fault-Tolerant Routing in the [N,K] Cube Networks

In this section, we develop an adaptive fault-tolerant routing algorithm, called AFTR, which requires every node to know only the condition of its own links. This algorithm, adapted from the algorithm DP_G and depth-first search, will successfully route messages between any pair of functional nodes as long as there exists a path between them.

Algorithm AFTR is described as follows: Each node, x, needs to keep track of the condition of its own links and update the number of faulty links(x), fault_No(x). To establish a path to the destination of a message, the required control sequence and its length m are appended to the message before transmission. We use depth-first search to deal with the problem of routing messages between connected pairs of non-faulty nodes in an [N,K] cube network with arbitrary number of faulty component. Therefore, N sets of used links(control vectors) in each node have to be added to the message to ensure that a node will not be visited twice except for backtracking. We also use a stack of used control vectors, P_stack, to guide the backtracking whenever it is forced to backtrack from a dead end. Before calling the algorithm AFTR, the source node needs to generate the control sequence, create a null P_stack, set all used_link(i).No = 0 and used_link(i).Link = $\varnothing$. Algorithm AFTR will attempt to route messages via shortest paths first. However, if all the

links(control vectors) in the received control sequence leading to the shortest paths are faulty, the algorithm will use a control vector which has the same non-zero k-digit of some control vector in the received control sequence to route the message via an alternative path which results in increasing the length of the path by 1. If all of the above links are not available, the algorithm tries an unused control vector to route the message which results in increasing the length of the path by 2.

Algorithm AFTR: /* Adaptive Fault-Tolerant Routing algorithm */

{ a node n receiving (m, P_{j0k} P_{j1k} . . . $P_{j(m-1)k}$, P_stack, used_link, message) }

1. **if** m = 0 **then** { the destination is reached! } **stop;**

2. **if** (used_link(n).No + fault_No) = $\lceil n/k \rceil (K-1)$ **then** {

   /* all the links are either faulty or used already */

   if P_stack is empty **then**

   { return to the source node, no connection is possible! }

   stop;

   else {

   /* backtracking */

   P_{ik} = pop(P_stack);

   send (m+1, P_{j0k} P_{j1k} . . . $P_{j(m-1)k}$ P_{ik}, P_stack, used_link, message)

   along the $\phi(n, P_{ik})$ link; } }

3. **for** i = 0 , m-1 **do {**

/* Try to send the message along a control vector in the received control sequence */

if (the P_{jik} link is not faulty and $P_{jik} \notin$ used_link(n).Link) **then** {

/* P_{jik} is not faulty and has not been used */

used_link(ϕ(n, P_{jik})).No = used_link(ϕ(n, P_{jik})).No +1;

used_link(n).No = used_link(n).No +1;

used_link(ϕ(n, P_{jik})).Link = used_link(ϕ(n, P_{jik})).Link $\cup$ P_{jik};

used_link(n).Link = used_link(n).Link $\cup$ P_{jik};

if used_link(ϕ(n, P_{jik})).No = 1 **then** {

/* node ϕ(n, P_{jik}) has never been visited before */

push(P_stack, P_{jik});

send (m-1, P_{j0k} P_{j1k} . . . $P_{j(i-1)k}P_{j(i+1)k}$ . . . $P_{j(m-1)k}$, P_stack, used_link, message) along the ϕ(n, P_{jik}) link;

stop; } } }

4. **for** i = 0 , m-1 **do** {

/* If the algorithm is not terminated yet, all control vectors in the received control sequence are faulty or used. Try to use a control vector which is the same k-digit of some control vector in the received control sequence*/

for $\forall P_{ik} \notin \{P_{j0k}, P_{j1k}, \ldots, P_{j(m-1)k}\}$ and ik = jik **do** {

/* P_{ik} is the same k-digit of P_{jik} and not equal to P_{jik} */

if (the P_{ik} link is not faulty and $P_{ik} \notin$ used_link(n).Link) **then** {

/* P_{ik} is not faulty and has not been used */

used_link($\phi(n, P_{ik})$).No = used_link($\phi(n, P_{ik})$).No +1;

used_link(n).No = used_link(n).No +1;

used_link($\phi(n, P_{ik})$).Link = used_link($\phi(n, P_{ik})$).Link $\cup$ P_{ik};

used_link(n).Link = used_link(n).Link $\cup$ P_{ik};

if used_link($\phi(n, P_{ik})$).No = 1 **then** {

/* node $\phi(n, P_{ik})$ has never been visited before */

push(P_stack, P_{ik});

send (m, P_{j0k} P_{j1k} . . . $P_{j(i-1)k}P_{j(i+1)k}$ . . . $P_{j(m-1)k}(P_{ik}+P_{jik})$

, P_stack, used_link, message) along the $\phi(n, P_{ik})$

link;

stop; }}}}

5. **for** i = m , $\lceil n/k \rceil$-1 **do** {

/* If the algorithm is not terminated yet, all tried control vectors are faulty or used. Now try to use a control vector left */

for $\forall P_{ik}$ **do** {

/* P_{ik} is not the same k-digit of any P_{jik} in message received */

if (the P_{ik} link is not faulty and $P_{ik} \notin$ used_link(n).Link) **then** {

/* P_{ik} is not faulty and has not been used */

used_link($\phi(n, P_{ik})$).No = used_link($\phi(n, P_{ik})$).No +1;

used_link(n).No = used_link(n).No +1;

used_link($\phi(n, P_{ik})$).Link = used_link($\phi(n, P_{ik})$).Link $\cup$ P_{ik};

used_link(n).Link = used_link(n).Link $\cup$ P_{ik};

if used_link($\phi(n, P_{ik})$).No = 1 **then** {

/* node $\phi(n, P_{ik})$ has never been visited before */

push(P_stack, P_{ik});

send (m+1, P_{j0k} P_{j1k} . . . $P_{j(m-1)k}$ P_{ik} , P_stack, used_link, message) along the $\phi(n, P_{ik})$ link;

stop; }}}}

6. **go to** step 2.

/* If the algorithm is not terminated yet, all control vectors of current node are either faulty or used already. backtrack to the previous node. */

3.4 Number of Disjoint Paths in the [N, K] PM2I Networks

In this section, we will find the lower bound on the number of disjoint paths in the [N,K] PM2I Networks. This bound is always greater than or equal to the number of disjoint paths in an [N,K] cube network with the same parameters value, i.e., the same N and K.

Lemma 3.2: Let S, D be any two nodes in an [N,K] PM2I network and $m=H_k(S, D) \leq \lceil n/k \rceil$ be the smaller number of non-zero k-digits in T=S+D or $\bar{T}$, where $\bar{T}$ is two's complement of T. Then there exists m disjoint paths of length m between the nodes S and D.

Proof: For convenience we assume without loss of generality that T has less non-zero k-digits than $\bar{T}$. T can be decomposed into m non-zero control vectors $P^m=P_{j0k}P_{j1k} \ldots P_{j(m-1)k}$. All m non-zero control vectors are positive if T is positive. Similarly, all m non-zero control vectors are negative if T is negative. The following m control sequences are disjoint and connect nodes S and D:

$P(0)^m=P_{j0k}P_{j1k} \ldots P_{j(m-1)k}$,

$P(1)^m=P_{j1k}P_{j2k} \ldots P_{j(m-1)k}P_{j0k}$,

. . . ,

$P(i)^m=P_{jik}P_{j(i+1)k} \ldots P_{j(m-1)k}P_{j0k} \ldots P_{j(i-1)k}$,

. . . , and

$P(m-1)^m=P_{j(m-1)k}P_{j0k} \ldots P_{j(m-2)k}$

It is clear that $\sigma(P(i)^m)=\sigma(P^m)$, $\forall i : 0 \leq i \leq m-1$, which implies that all these control sequences are equivalent, i.e., connecting the nodes S and D. Since each P_{jik} add different k-digit to the source node address, $\sigma(P(i)^l) \neq \sigma(P(j)^l)$, $\forall i \neq j$, $l<m$. Thus, the paths are disjoint. ■

Theorem 3.3: Let S, D be any two nodes in an [N,K] PM2I network and $m=H_k(S, D) \leq n/k$ be the smaller number of non-zero k-digits in T=S+D or $\bar{T}$, where n is assumed to be multiple of k. Then, there exists at least (2(n/k)-2m-1)(K-1)+m disjoint paths

between S and D if $P_{(n/k-1)k} \in P^m$; and at least (2(n/k)-m)(K-1)+m disjoint paths between S and D if $P_{(n/k-1)k} \notin P^m$. Moreover, the length of each path is at most m+2.

Proof: For convenience we assume without loss of generality that T has less non-zero k-digits than $\bar{T}$. T can be decomposed into m non-zero control vectors $P^m=P_{j0k}P_{j1k} \ldots P_{j(m-1)k}$. Note that constructive proof given next yield exactly m paths of length m and (2(n/k)-2m-1)(K-1) paths of length m+2 if $P_{(n/k-1)k} \in P^m$; and (2(n/k)-2m)(K-1) paths of length m+2 if $P_{(n/k-1)k} \notin P^m$. Then, the set of control vectors available to node S can be divided into the following 3 classes:

Class 1: $P_{ik} \in \{P_{j0k}, P_{j1k}, \ldots, P_{j(m-1)k}\}$. By Lemma 3.2, there are m disjoint paths of length m in this class.

Class 2: P_{ik}: $0 \le i \le m-1$, $P_{ik} \neq P_{jik}$. We construct the control sequences of length m+1 starting from P_{ik} as follows: $P(ik)^{m+1}=P_{ik}P_{j0k}P_{j1k} \ldots P_{j(i-1)k}P_{j(i+1)k} \ldots P_{j(m-1)k}(P_{jik}+_N(-P_{ik}))$. Since $P_{ik}+_N(P_{jik}+_N(-P_{ik})=P_{jik}$, $\sigma(P(ik)^{m+1})=\sigma(P^m)$, $\forall i : 0 \le i \le m-1$, $P(ik)^{m+1}$ and P^m are equivalent. However, $(P_{jik}+_N(-P_{ik}))$ may not belong to class 2, which implies that the last control vector of control sequences may be used by other control sequences in the other classes. Therefore we do not use this class of control vectors of node S.

Class 3: P_{ik}: $m \le i \le n/k-1$. We construct the control sequences of length m+2 starting from P_{ik} as follows: $P(ik)^{m+2}=P_{ik}P_{j0k}P_{j1k} \ldots P_{j(m-1)k}(-P_{ik})$. Since $P_{ik}+_N(-P_{ik})=\mathbf{0}$, $\sigma(P(ik)^{m+2})=\sigma(P^m)$, $\forall i : m \le i \le n/k-1$. Therefore $P(ik)^{m+2}$ and P^m are equivalent. There are (2(n/k)-2m-1)(K-1) P_{ik}'s in class 3 if $P_{(n/k-1)k} \in P^m$; (2(n/k)-2m)(K-1) P_{ik}'s in class 3 if $P_{(n/k-1)k} \notin P^m$.

In above class 1 and class 3, all the control sequences start from different P_{ik}, then the rest of control vectors are never the same k-digit as P_{ik} or only the last control vector is the same k-digit as P_{ik}. Therefore they are disjoint until reaching the destination D. ■

From Theorem 3.3, we know that when m=1 and $P_{(n/k-1)k} \notin P^m$, [N,K] PM2I network has the best lower bound of 2((n/k)-1)(K-1)+1. This bound is (K-2) less than the degree of node, disjoint paths between nodes S and D. In worst case, when m=$\lceil n/k \rceil$ and $P_{(n/k-1)k} \in P^m$, [N,K] PM2I networks have a lower bound of n/k disjoint paths between nodes S and D. This lower bound is much less than the actual value because even [N,K] cube networks, subgraphs of [N,K] PM2I networks, have n/k(K-1) disjoint paths between any pair of nodes. The next corollary concludes the lower bound on number of disjoint paths in [N,K] PM2I networks, we have obtained so far.

Corollary 3.2: Let S and D be any two nodes in an [N,K] PM2I network and $H_k(S, D)=m \leq n/k$ be the smaller number of non-zero k-digits in T=S+D or $\bar{T}$, where n is assumed to be multiple of k. Then, [N,K] PM2I network has a lower bound of max(n/k(K-1), (2(n/k)-2m)(K-1)+m) disjoint paths between S and D if $P_{(n/k-1)k} \in P^m$; or max(n/k(K-1), (2(n/k)-2m-1)(K-1)+m) disjoint paths between S and D if $P_{(n/k-1)k} \notin P^m$.

Lemma 3.3: Let S, D be any two nodes in a PM2I network, then $H_1(S, D)= m \leq \lceil n/2 \rceil$.

Proof: Assume m>$\lceil n/2 \rceil$ be the smaller number of non-zero digits in T=S+D or $\bar{T}$. Let

T have m non-zero digits. Then T's complement has n-m < $\lceil n/2 \rceil$ non-zero digits. But $\bar{T}$ is T's complement +1 which has at most n-m+1 ≤ $\lceil n/2 \rceil$ non-zero digits. This is a contradiction to the definition of Hamming distance. Therefore, $H_1(S, D) \leq \lceil n/2 \rceil$ in a PM2I network. ■

Theorem 3.4: Let S and D be any two nodes in a PM2I network and $H_1(S, D)=m$ be the smaller number of non-zero k-digits in T=S+D or $\bar{T}$. Then PM2I network has a lower bound of

max(2n-$\lceil n/2 \rceil$-1, 2n-m-1) disjoint paths between S and D if $P_{(n/k-1)k} \in P^m$;

or max(2n-$\lceil n/2 \rceil$, 2n-m) disjoint paths between S and D if $P_{(n/k-1)k} \notin P^m$.

Proof: 2n-m-1 or 2n-m can be obtained from Theorem 3.3 with k=1 and K=2. By Lemma 3.3, $m=H_1(S, D) \leq \lceil n/2 \rceil$. 2n-$\lceil n/2 \rceil$-1 or 2n-$\lceil n/2 \rceil$ can be obtained from Theorem 3.3 with k=1, K=2 and m=$\lceil n/2 \rceil$. ■

When n > 2, the lower bound on the number of disjoint paths in PM2I networks, max(2n-$\lceil n/2 \rceil$-1, 2n-m-1), is always greater than or equal to n(the number of disjoint paths in n-cube networks).

Theorem 3.5: Let S and D be any two nodes in a complete graph. Then there exist N-1 disjoint paths between the nodes S and D.

Proof: In a complete graph, every pair of nodes are directly connected and there are N-

2 paths of length 2 passing through each of the remaining N-2 nodes exactly once. Thus, there exists N-1 disjoint paths between any pair of nodes in a complete graph which is equal to the degree of node. ■

Corollary 3.3: Complete graph is maximally fault tolerant.

We have proved that [N,K] cube network and complete graph are maximally fault tolerant. We also have found the lower bound on number of disjoint paths between nodes S and D in [N,K] PM2I network which is greater than or equal to in [N,K] cube network with same N and K. However it needs more research to prove or disprove that [N,K] PM2I network is maximally fault tolerant.

4. Containment Properties of [N,K] Cube Network

In this Chapter, a special class of control sequences capable of producing paths with specific geometries, such as rings, linear arrays and meshes will be presented. These control sequences, in a node independent network, can produce the same topology from any starting node in the network. This feature of control sequences greatly simplifies the containment problem in communication networks and is one of the important features of Communication-Algebra.

This chapter is organized as follows:

First we investigate containment of rings and linear arrays in the cube type networks. Then algorithms of generating control sequences which establish meshes and spanning tree will be presented.

4.1 Ring Control Sequences

Before we investigate the properties of ring control sequences in an [N,K] cube network, we define cycle-free control sequences and ring control sequences. In what follows, notation $P^i \subseteq P^\alpha$ is used to indicate that P^i is a control sequence of length $i \leq \alpha$ such that $P^\alpha = P^i P^{\alpha-i}$, i.e., P^i is a subsequence of P^α.

Definition 4.1: A control sequence, P^α is said to be cycle-free if and only if $\phi(S,P^i) \neq \phi(S,P^{i+j}) : \forall i \in \{1, 2, ..., \alpha-1\}, \forall j \in \{1, 2, ..., \alpha-i\}$, where $P^i \subseteq P^\alpha$.

Definition 4.2: A control sequence, P^α, establishes a ring in an [N,K] cube network if and only if $\phi(S,P^\alpha) = S$ and $P^{\alpha-1}$ is cycle-free, where $P^{\alpha-1} \subseteq P^\alpha$.

Lemma 4.1: The path created by P^α is cycle-free if and only if $\sigma(P^{i+j}) \neq \sigma(P^i) : \forall i \in \{1, 2, ..., \alpha-1\}, \forall j \in \{1, 2, ..., \alpha-i\}$, where $P^i \subseteq P^\alpha$.

Proof: Let $P^\alpha = A_1 A_2 \ldots A_\alpha$.

Since P^α creates a cycle-free path, by Definition 4.1, $\phi^i(S,P^i) \neq \phi^{i+j}(S,P^{i+j}) : \forall i \in \{1, 2, ..., \alpha-1\}, \forall j \in \{1, 2, ..., \alpha-i\}$, which in turn implies that

$S +_2 A_1 +_2 A_2 +_2 \ldots +_2 A_i \neq S +_2 A_1 +_2 A_2 +_2 \ldots +_2 A_i +_2 \ldots +_2 A_j : \forall i \in \{1, 2, ..., \alpha-1\}, \forall j \in \{1, 2, ..., \alpha-i\}$.

By adding S to both side, we obtain:

$A_1 +_2 A_2 +_2 \ldots +_2 A_i \neq A_1 +_2 A_2 +_2 \ldots +_2 A_i +_2 \ldots +_2 A_j : \forall i \in \{1, 2, ..., \alpha\text{-}1\}, \forall j \in \{1, 2, ..., \alpha\text{-}i\}$.

Thus, $\sigma(P^{i+j}) \neq \sigma(P^i) : \forall i \in \{1, 2, ..., \alpha\text{-}1\}, \forall j \in \{1, 2, ..., \alpha\text{-}i\}$. ■

Theorem 4.1: P^α establishes a ring if and only if $\sigma(P^{i+j}) \neq \sigma(P^i) : \forall i \in \{1, 2, ..., (\alpha\text{-}1)\text{-}1\}, \forall j \in \{1, 2, ..., (\alpha\text{-}1)\text{-}i\}$ and $\sigma(P^\alpha) = \mathbf{0}$, where $P^i \subseteq P^\alpha$.

Proof: In order the path created by P^α to be closed, we should have

$\phi(S,P^\alpha) = S +_2 A_1 +_2 A_2 +_2 \ldots +_2 A_\alpha = S$.

Adding S to both side results in $A_1 +_2 A_2 +_2 \ldots +_2 A_\alpha = \mathbf{0}$, or $\sigma(P^\alpha) = \mathbf{0}$.

Since $P^{\alpha-1}$ is cycle-free,

$\sigma(P^{i+j}) \neq \sigma(P^i) : \forall i \in \{1, 2, ..., (\alpha\text{-}1)\text{-}1\}, \forall j \in \{1, 2, ..., (\alpha\text{-}1)\text{-}i\}$ follows from Lemma 4.1 ■

The next proposition imposes a constrain on the length of the ring embedded in the n-cube network. This constrain can be relaxed by enhancing the n-cube to an [N,K] cube, as will be discussed in this chapter.

Proposition 4.1: There are no cycles of odd length in an n-cube.

Proof: Let P^α be a control sequence which creates a cycle in an n-cube, then by Theorem 4.1, $\sigma(P^\alpha) = \mathbf{0}$. This implies that the number of control vectors of type i in P^α must be even for all i. Thus, the total number of control vectors in P^α is even, i.e., α is even. ■

However, it is possible to establish a ring of odd length in an [N,K] cube network as long as $K > 2$.

In chapter 3, Algorithm DP_G (disjoint paths generator), which can generate disjoint paths of length between 1 to n/k+1, was proposed. Since concatenation of two equivalent disjoint control sequences establishes a closed path in the network, concatenation of a control sequence and a reverse of its equivalent disjoint control sequence, P^{α}, and $R(P^{\alpha})$, as defined next, establishes a ring. Let R be a mapping, called vector reversal, defined on $\mathcal{P}^*$ by $R(P_{j0}P_{j1}, \ldots, P_{j(m-1)}) = P_{j(m-1)}P_{j(m-2)} \ldots P_{j0}$. By modifying Algorithm DR_G, the following algorithm, Ring_G(L), can generate ring control sequences with total length L which can vary between 3 to $2\log_2 N$.

Algorithm Ring_G(L): /* Ring control sequence of length $3 \le L \le 2n$ Generator */

1. $m = \lfloor L/2 \rfloor$.
2. choose m control vectors, $P_{j0}, P_{j1}, \ldots, P_{j(m-1)}$, out of the set $\{P_0, P_1, \ldots, P_{n-1}\}$.
3. $P(1)^m = P_{j0}P_{j1}, \ldots, P_{j(m-1)}$.
4. **if** L is even **then**

   $P(2)^m = P_{ji}P_{j(i+1)} \ldots P_{j(m-1)}P_{j0}P_{j1} \ldots P_{j(i-1)}$ for some $i \neq 0$

   $P^L = P(1)^m R(P(2)^m)$

   else

   find some P_i and P_{ji} such that $P_i \in \{P_0, P_1, \ldots, P_{n-1}\} - \{P_{j0}, P_{j1}, \ldots, P_{j(m-1)}\}$ and P_i and P_{ji} belong to the same k-digit.

$P(2)^{m+1}=P_iP_{j0k}P_{j1} \ldots P_{j(i-1)}P_{j(i+1)} \ldots P_{j(m-1)}(P_i+_2P_{ji})$

$P^L = P(1)^mR(P(2)^{m+1})$

end_if

Example 4.1: In a [16,4] cube network shown in figure 4.1, a ring of length 3 can be established by the following control sequences generated by Algorithm Ring_G(L):

$P(1)^1 = (1000)$,

$P(2)^2 = (0100)(1100)$,

Therefore, $P^3 = P(1)^1 R(P(2)^2) = (1000)(1100)(0100)$.

4.2 Control Vector Space of [N,K] cube

To utilize all the processors, we are interested in a control sequence capable of forming a ring which includes all the nodes in a cube network, referred to as, Hamiltonian cycle. This control sequence is referred to as Hamiltonian sequence. To find an algorithm which generates Hamiltonian sequences, we need to look more closely at the structure of the control vector space $\mathcal{P}^*$.

By Lemma 2.1, S-equivalence defined on $\mathcal{P}^*$ is an equivalence relation. An equivalence relation on a set partitions the set into equivalence classes. The set of equivalence classes

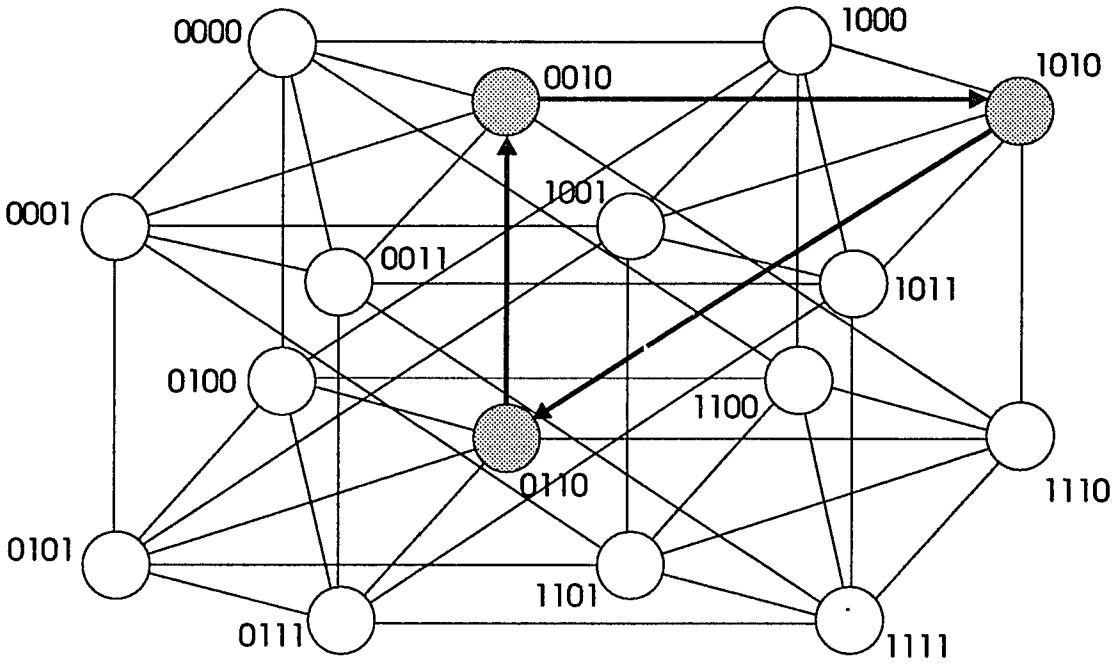

Fig. 4.1. ring of length 3 in [16,4] cube network

induced on $\mathcal{P}^*$ by the routing function ϕ and source node address S is denoted by $\mathcal{P}^*/\equiv$. The members of a class are indistinguishable from one another by their action on ϕ. An equivalence class of $\mathcal{P}^*$ may be represented by any of its members and we shall denote such a class by $[P^\alpha]$. In Theorem 2.2, it is shown that the number of disjoint equivalent classes; i.e., cardinality of the set $\mathcal{P}^*/\equiv$, induced by ϕ on S and $\mathcal{P}^*$, in any interconnection network with N nodes is N. The set of N control sequences such that each member belongs to different equivalent class can be formed by the generating set of control vectors. The generating set is usually referred to as a spanning set. In particular, it is desirable to find a "minimal" spanning set. To do this it is necessary to consider how the vectors in the set depend on each other. Consequently, we introduce the concepts of linear dependence and its definition. These concepts provide the keys to understanding the structure of vector space $\mathcal{P}^*$.

Definition 4.3: Let $P_{j1}, P_{j2}, \ldots, P_{jn}$ be control vectors in a vector space $\mathcal{P}^*$. The **linear combination** of $P_{j1}, P_{j2}, \ldots, P_{jn}$ is defined as: $c_1P_{j1} +_2 c_2P_{j2} +_2 \ldots +_2 c_nP_{jn}$, where $c_i \in \{0,1\} : 0 \le i \le n$. Since $S +_2 (2k)P_i = S +_2 0P_i$ and $S +_2 (2k+1)P_i = S +_2 P_i$, we restrict c_i's to be member of the set $\{0,1\}$.

Definition 4.4: The set $\{ P_{j1}, P_{j2}, \ldots, P_{jn} \}$ is called a **spanning set** of the vector space $\mathcal{P}^*$ if and only if every control sequence in $\mathcal{P}^*$ is equivalent to a linear combination of $P_{j1}, P_{j2}, \ldots, P_{jn}$. In this case, we say that $P_{j1}, P_{j2}, \ldots, P_{jn}$ span the vector space $\mathcal{P}^*$.

Proposition 4.2:

(i) If $P_{j1}, P_{j2}, \ldots, P_{jn}$ span a vector space $\mathcal{P}^*$ from any starting node and one of these control vectors is a linear combination of remaining n-1 control vectors, then these n-1 vectors span $\mathcal{P}^*$.

(ii) Given n control vectors $P_{j1}, P_{j2}, \ldots, P_{jn}$, it is possible to write one of the control vectors as a linear combination of the other n-1 control vectors if and only if there exist scalars $c_1, c_2, \ldots, c_n$, not all zero, such that:

$$c_1P_{j1} +_2 c_2P_{j2} +_2 \ldots +_2 c_nP_{jn} = \mathbf{0}, \text{ where } c_i \in \{0,1\} : 0 \le i \le n.$$

Proof: (i). Suppose P_{jn} can be written as a linear combination of $P_{j1}, P_{j2}, \ldots, P_{j(n-1)}$ as:

$$P_{jn} = c_1P_{j1} +_2 c_2P_{j2} +_2 \ldots +_2 c_{(n-1)}P_{j(n-1)}$$

Let $\sigma(P^\alpha)$ be the signature of any control sequence in $\mathcal{P}^*$, we can write

$$\sigma(P^\alpha) = a_1P_{j1} +_2 a_2P_{j2} +_2 \ldots +_2 a_nP_{jn}$$

$$= a_1P_{j1} +_2 a_2P_{j2} +_2 \ldots +_2 a_{(n-1)}P_{(n-1)} +_2 a_n(c_1P_{j1} +_2 c_2P_{j2} +_2 \ldots +_2 c_{(n-1)}P_{j(n-1)})$$

$$= (a_1 +_2 a_nc_1)P_{j1} +_2 (a_2 +_2 a_nc_2)P_{j2} +_2 \ldots +_2 (a_{(n-1)} +_2 a_nc_{(n-1)})P_{j(n-1)}$$

Since coefficients a_i's and c_i's $\in \{0,1\}$, $(a_i +_2 a_nc_i)$ is also $\in \{0,1\}$. Thus, any control sequence in $\mathcal{P}^*$ is equivalent to a linear combination of $P_{j1}, P_{j2}, \ldots, P_{j(n-1)}$ and hence, these n-1 vectors span $\mathcal{P}^*$.

(ii). Suppose that P_{jn} can be written as a linear combination of $P_{j1}, P_{j2}, \ldots P_{jn-1}$, i.e.,

$$P_{jn} = a_1P_{j1} +_2 a_2P_{j2} +_2 \ldots +_2 a_{(n-1)}P_{j(n-1)}$$

If we set $c_i = a_i$, for i = 1, . . ., n-1, and $c_n = 1$, then, it follows that

$$a_1P_{j1} +_2 a_2P_{j2} +_2 \ldots +_2 a_{(n-1)}P_{j(n-1)} +_2 a_1P_{j1} +_2 a_2P_{j2} +_2 \ldots +_2 a_{(n-1)}P_{j(n-1)} = \mathbf{0}$$

Conversely, if

$$c_1P_{j1} +_2 c_2P_{j2} +_2 \ldots +_2 c_nP_{jn} = \mathbf{0}$$

and at least one of the c_i's, say c_n, is nonzero, then:

$$P_{jn} = c_1P_{j1} +_2 c_2P_{j2} +_2 \ldots +_2 c_{(n-1)}P_{j(n-1)}$$ ■

Definition 4.5: The control vectors $P_{j1}, P_{j2}, \ldots, P_{jn} \in \mathcal{P}_n$ are said to be *linearly independent* if and only if

$c_1P_{j1} +_2 c_2P_{j2} +_2 \ldots +_2 c_nP_{jn} = \mathbf{0}$ implies that $c_1 = c_2 = \ldots = c_n = 0$,

where $c_i \in \{0,1\} : 0 \leq i \leq n$.

Definition 4.6: The control vectors $P_{j1}, P_{j2}, \ldots, P_{jn} \in \mathcal{P}_n$ are said to be *linearly dependent* if

there exist scalars $c_1, c_2, \ldots, c_n$, not all zero, such that

$c_1P_{j1} +_2 c_2P_{j2} +_2 \ldots +_2 c_nP_{jn} = \mathbf{0}$, where $c_i \in \{0,1\} : 0 \leq i \leq n$.

Example 4.2: The control vectors (0001), (0011) are linearly independent, since

if $c_1(0001) +_2 c_2(0011) = \mathbf{0} = (0000)$

then $c_2 = 0$ and $c_1 +_2 c_2 = 0$

and the only solution to this system is $c_1 = 0$, $c_2 = 0$.

Example 4.3: The control vectors (0001), (0011), (0010) are linearly dependent, since

$$(0001) +_2 (0011) +_2 (0010) = \mathbf{0}$$

In this case $c_1=1$, $c_2=1$, $c_3=1$.

Proposition 4.3: if $\{P_{j1}, P_{j2}, \ldots, P_{jn}\}$ is a minimal spanning set, then $P_{j1}, P_{j2}, \ldots, P_{jn}$ are linearly independent. Conversely, if $\{P_{j1}, P_{j2}, \ldots, P_{jn}\}$ are linearly independent and span the vector space $\mathcal{P}^*$, then $\{P_{j1}, P_{j2}, \ldots, P_{jn}\}$ is a minimal spanning set of $\mathcal{P}^*$.

Proof: It follows from Proposition 4.2(i) and Proposition 4.2(ii).

The elements of a minimal spanning set form the basic building blocks for the whole vector space $\mathcal{P}^*$. Thus, these elements are referred to as "basis" of the vector space.

Definition 4.7: The control vectors $P_{j1}, P_{j2}, \ldots, P_{jn}$ form a **basis** for a vector space $\mathcal{P}^*$ if and only if (i) $P_{j1}, P_{j2}, \ldots, P_{jn}$ are linearly independent, and

(ii) $P_{j1}, P_{j2}, \ldots, P_{jn}$ span $\mathcal{P}^*$.

Definition 4.8: Let $\mathcal{P}^*$ be a vector space. **Dimension** of $\mathcal{P}^*$ is defined to be the number of control vectors in its basis set.

To establish a Hamiltonian cycle or a spanning tree in an [N,K] cube network, we need to use a set of control vectors containing a minimal spanning set because in either case the control sequences used span all the equivalence classes of $\mathcal{P}^*$ to reach every node in the

network. In reliability or operational point of view, a minimal spanning set of control vectors available to each node ensures that the network is operational or functional. Conversely, it is not true because a Hamiltonian cycle only needs two control vectors for each node. However, if a minimal spanning set of control vectors is available to every node, then a Hamiltonian sequence can establish a Hamiltonian cycle from any starting node.

Theorem 4.2: In an [N,K] cube, $\{P_0, P_1, \ldots, P_{n-1}\}$ form a **basis** for a vector space $\mathcal{P}^*$.

Proof: (i). To prove $P_0, P_1, \ldots, P_{n-1}$ are linearly independent,

assume that:

$$c_0(0...01) +_2 c_1(0...010) +_2 \ldots +_2 c_{n-1}(10...0) = \mathbf{0} = (00...0)$$

then $c_0 = 0, c_1=0, \ldots, c_{n-1}=0$.

Therefore, $P_0, P_1, \ldots, P_{n-1}$ are linearly independent.

(ii). To prove $P_0, P_1, \ldots, P_{n-1}$ span $\mathcal{P}^*$, let $\sigma(P^\alpha)=(x_{n-1}x_{n-2}\ldots x_0)$ be the signature of any control sequence in $\mathcal{P}^*$. We can write

$$\sigma(P^\alpha) = (x_{n-1}x_{n-2}\ldots x_0)$$

$$= x_0(0...01) +_2 x_1(0...10) +_2 \ldots +_2 x_{n-1}(10...0)$$

If we set $c_i = x_i$ for i = 0, 1, ..., n-1, then is follows that

$$\sigma(P^\alpha) = c_0P_0 +_2 c_1P_1 +_2 \ldots +_2 c_{n-1}P_{n-1}$$

Thus, any control sequence is equivalent to a linear combination of $P_0, P_1, \ldots, P_{n-1}$. Hence, these vectors span $\mathcal{P}^*$. ■

Corollary 4.1: Dimension of vector space $\mathcal{P}^*$, in an [N,K] cube, is n.

Proof: It follows from Theorem 4.2 and Definition 4.8.

Corollary 4.2: Let P_{ik} span $\mathcal{P}_{ik}^*$. Then dimension of $\mathcal{P}_{ik}^*$ is k.

Proof: Let δ: $P_{ik} \rightarrow P_k$ be a bijection function defined as:

$\delta(x_{n-1} \ldots x_{ik+k-1} \ldots x_{ik} \ldots x_0) = (x_{ik+k-1} \ldots x_{ik})$

i.e., $\delta(P_{ik})$ is a control vector obtained from ith k-digit of P_{ik}.

Because P_{ik}'s contain all 0's except ith k-digit, function δ preserves the linear independence of vectors.

Therefore, $P_{ik}(1)$, $P_{ik}(2)$, ..., $P_{ik}(k)$ are linearly independent if and only if $\delta(P_{ik}(1))$, $\delta(P_{ik}(2))$, ..., $\delta(P_{ik}(k))$ are linearly independent.

Since δ is a bijection, dimension of P_{ik} and dimension of $\delta(P_{ik})$ are the same.

$\delta(P_{ik})$ is identical to the set of control vectors, P_k, which is the set of control vector of [K,K] cube. By theorem 4.2, dimension of vector space in a [K,K] cube is k.

Therefore, dimension of $\mathcal{P}_{ik}^*$ is k. ■

Since dimension of $\mathcal{P}_{ik}^*$ is k, each k-digit control vector set has bases containing k vectors. But the set of P_{ik} contains K-1 control vectors. Thus, there may exist many subsets of P_{ik} which forms a basis for $\mathcal{P}_{ik}^*$. The problem of how many possible different bases for $\mathcal{P}_{ik}^*$ is considered in the next theorem.

Theorem 4.3: The total number of bases, containing k control vectors, for vector space of type i, $\mathcal{P}_{ik}^{*}$, is $|\{basis\ (P_{ik})\}| = \binom{K-1}{2}\binom{K-1-\sum_{i=1}^{2}\binom{2}{i}}{1} \cdots \binom{K-1-\sum_{i=1}^{k-1}\binom{k-1}{i}}{1}$.

Proof: $\{P_{ik}\}$ has K-1 control vectors and its dimension is k. We choose k linearly independent vectors, as its basis, by the following algorithm:

Algorithm Basis($\{P_{ik}\}$,k) /* Input a set of vectors and its dimension; Output a basis set */

1. Choose any 2 vectors, $\{P_{j1}, P_{j2}\}$, from K-1 vectors. Clearly, these 2 vectors are linearly independent since they are different.
2. basis(P_{ik}) = $\{P_{j1}, P_{j2}\}$.
3. If |basis(P_{ik})| = k Then **Stop.**
4. Choose 1 vectors from K-1 vectors excluding the vectors which are equal to members of basis(P_{ik}) or their linear combinations.
5. basis(P_{ik}) = basis(P_{ik}) + {new chosen}
6. Go To Step 3.

In Step 4., if |basis(P_{ik})| = m, then

the number of vectors in $\{P_{ik}\}$ which are equal to members of basis(P_{ik}) is $m = \binom{m}{1}$, and

the number of vectors in $\{P_{ik}\}$ which are equal to linear combination of any 2 vectors in

basis(P_{ik}) is $\binom{m}{2}$, and

the number of vectors in $\{P_{ik}\}$ which are equal to the linear combination of any i vectors in basis(P_{ik}) is $\binom{m}{i}$.

Therefore, the number of vectors left to be chosen is $K-1-\sum_{i=1}^{m}\binom{m}{i}$.

The number of different ways of choosing 1 vector from the remaining vectors is $\binom{K-1-\sum_{i=1}^{m}\binom{m}{i}}{1}$.

Thus, $|\{basis\ (P_{ik})\}| = \binom{K-1}{2}\binom{K-1-\sum_{i=1}^{2}\binom{2}{i}}{1} \cdots \binom{K-1-\sum_{i=1}^{k-1}\binom{k-1}{i}}{1}$. ■

Example 4.4: If $\{P_{ik}\}$ = {(01),(11),(10)} then $\{P_{ik}\}$ has 3 bases set as follows:

{(01),(11)}, {(01),(10)}, {(11),(10)} and dimension of P_{ik} is k=2.

Example 4.5: If $\{P_k\} = \{(001),(011),(010),(110),(111),(101),(100)\}$ and k=3, then

$$|\{basis(P_{ik})\}| = \binom{7}{2}\binom{4}{1} = 84.$$

Corollary 4.3: If n is multiple of k, then the number of bases in an [N,K] cube network is

$|\{basis(\mathcal{P})\}| = |\{basis(P_k)\}|^{n/k}$.

Proof: By Corollary 4.2, each type of control vectors, P_k's, has bases containing k vectors. By Theorem 4.2, dimension of [N,K] cube is n = n/k x k. Since control vectors of different types are linearly independent, the number of bases in an [N,K] cube network is

$|\{basis(\mathcal{P})\}| = |\{basis(P_k)\}|^{n/k}$. ■

4.3 Hamiltonian Sequences

Gray code is a sequence of n-bit binary numbers such that any two successive numbers have only one different bit and so that all binary numbers having n bits are represented. If all the nodes in an **n-cube** are connected in such a way that binary representation of their addresses is a Gray code, then the connection forms a Hamiltonian cycle([Saad 88]). The control sequence which establishes a Gray code connection is called the **Gray control sequence.** For example, (001) (010) (001) (100) (001) (010) (001) (100) or $P_0P_1P_0P_2P_0P_1P_0P_2$ is a 3-bits Gray control sequence. This example is only one of many

possible such Gray control sequences. To obtain a different Gray code control sequence, in an n-cube network, one can start with any control vector and proceed to next control vector in any desired random fashion. By observing that all the control vectors in an n-cube network are linearly independent, the concept of two successive nodes having only one different bit can be extended to two successive nodes having difference of one basis vector.

Next, an algorithm, called Gray_S($\{P_{j0}, P_{j1}, \ldots, P_{j(n-1)}\}$,n), which produces Gray control sequences for rings of length 2^n constructed from n basis vectors, $\{P_{j0}, P_{j1}, \ldots, P_{j(n-1)}\}$, as follows, where $\{P_{j0}, P_{j1}, \ldots, P_{j(n-1)}\}$ are obtained from Algorithm Basis in the proof of Theorem 4.3:

Algorithm Gray_S($\{P_{j0}, P_{j1}, \ldots, P_{j(n-1)}\}$,n): /* Gray control sequence of length 2^n */

1. $P^1 = P_{j0}$.
2. For i=2 To n-1

   $P^i = P^{i-1} P_{j(i-1)} P^{i-1}$.
3. $P^n = P^{n-1} P_{j(n-1)} P^{n-1} P_{j(n-1)}$.

Example 4.6: Let {(0001),(0011),(0100),(1000)} be the basis obtained from Algorithm Basis. Then one possible Gray control sequence produced by Algorithm Gray_S is the following:

(0001)(0100)(0001)(0011)(0001)(0100)(0001)(1000)

(0001)(0100)(0001)(0011)(0001)(0100)(0001)(1000),

A Hamiltonian cycle starting from node 0 in a [16,4] cube network established by the above control sequence is shown in Fig. 4.2. The nodes in the Hamiltonian cycle are as follows:

$0 \rightarrow 1 \rightarrow 5 \rightarrow 4 \rightarrow 7 \rightarrow 6 \rightarrow 2 \rightarrow 3 \rightarrow 11 \rightarrow 10 \rightarrow 14 \rightarrow 15 \rightarrow 12 \rightarrow 13 \rightarrow 9 \rightarrow 8 \rightarrow 0.$

Theorem 4.4: There exists at least $|\{\text{basis}(\mathcal{P})\}|n! = |\{\text{basis}(P_{ik})\}|^{n/k}$ n! Gray control sequences in an [N,K] cube network.

Proof: In Algorithm Gray_S, vectors in the set $\{P_{j0}, P_{j1}, \ldots, P_{j(n-1)}\}$can be in any order. Thus, given a basis with n vectors, there are n! different Gray control sequences. By Corollary 4.3, the number of bases in an [N,K] cube network is $|\{\text{basis}(\mathcal{P})\}| = |\{\text{basis}(P_{ik})\}|^{n/k}$. Therefore, there exists at least $|\{\text{basis}(\mathcal{P})\}|n! = |\{\text{basis}(P_{ik})\}|^{n/k}$ n! Gray control sequences in an [N,K] cube network. ■

Corollary 4.4: There exists at least $|\{\text{basis}(\mathcal{P})\}|n! = |\{\text{basis}(P_{ik})\}|^{n/k}$ n! Hamiltonian cycles in an [N,K] cube network.

Proof: Since a Gray control sequence establishes a Hamiltonian cycle in an [N,K] cube network, it then follows from Theorem 4.4 that the number of Hamiltonian cycles is at least $|\{\text{basis}(\mathcal{P})\}|n! = |\{\text{basis}(P_{ik})\}|^{n/k}$ n!. ■

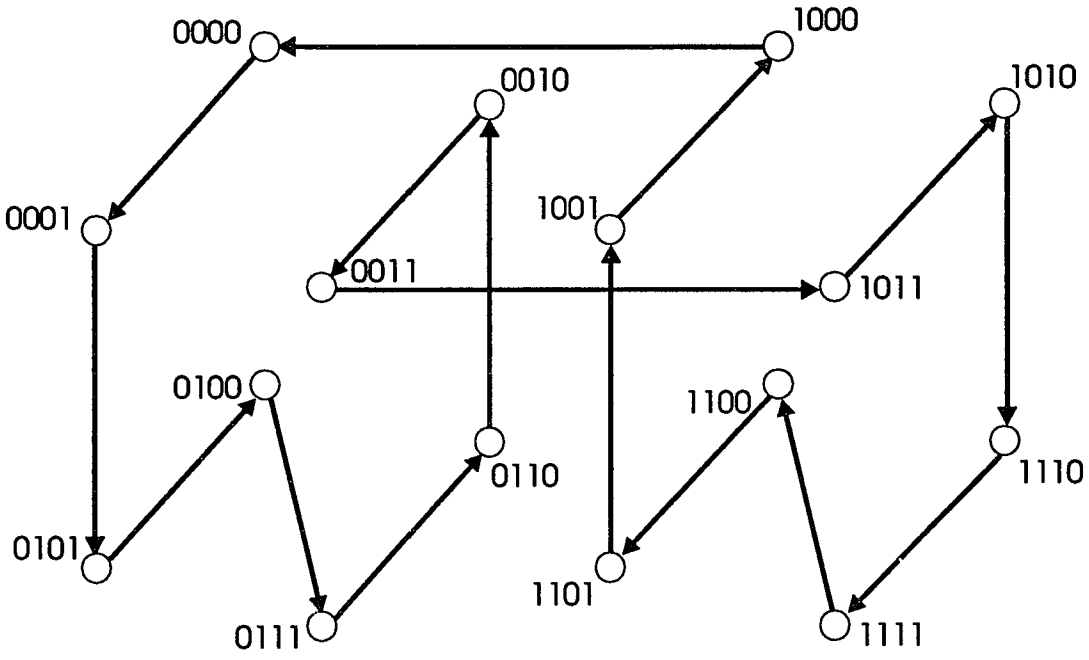

Fig. 4.2. Hamiltonian cycle in [16,4] cube network

4.4 Generalized Ring Sequences

In this section, we will investigate the class of rings topologies which can be embedded into [N,K] cube networks. Then, we will presented an algorithm which produces all ring control sequences, which establish rings of all possible length, in [N,K] networks. The following proposition [16], which shows a important containment property of n-cube, is restated next.

Proposition 4.2: A ring of length l can be mapped into an n-cube; where l is even and $4 \leq l \leq N=2^n$.

According to proposition 4.2, a ring of even length, $4 \leq l \leq 2^n$ can be embedded into an n-cube network. Since the length of any Gray control sequence is a power of 2, say 2^n, we use its partial sequence to generate a sequence of any even length, between 4 and 2^n, by the following algorithm:

Algorithm Gray($\{P_{j0}, P_{j1}, \ldots, P_{j(n-1)}\}$,$l$)**:**

/* Gray control sequence of even length between 4 and 2^n */

1. $n = \lfloor \log_2 l \rfloor$, $m = (l - 2)/2$.
2. $P^1 = P_{j0}$.
3. For i=2 To n-1

$P^i = P^{i-1}\ P_{j(i-1)}\ P^{i-1}$.

4. $P^l = P^{n-1}(m)\ P_{j(n-1)}\ P^{n-1}(m)\ P_{j(n-1)}$, where $P^{n-1}(m)$ is the first m vectors in P^{n-1}.

To show that odd length rings can also be embedded into an [N,K] cube network, with K>2, in section 4.1, we presented an algorithm, called Ring_G(L), which generate ring control sequences of length between 3 to 2n. Next, a generalized algorithm, which combines the modified Gray control sequence's ability of producing maximum length ring and Algorithm Ring_G(L)'s odd length sequence, will be presented to generate ring control sequences of length which can vary between 3 to 2^n=N.

Algorithm Ring($\{P_{j0}, P_{j1}, \ldots, P_{j(n-1)}\}$,*l*): /* Ring control sequence of length $3 \le l \le N$ Generator */

1. $n = \lfloor \log_2 l \rfloor$, $m = \lceil (l - 2)/2 \rceil$.
2. $P^1 = P(1)$.
3. For i=2 To n-1

   $P^i = P^{i-1}\ P(n-1)\ P^{i-1}$.
4. $P(i) = Last(P^{n-1}(m))$, where $P^{n-1}(m)$ is the first m vectors in P^{n-1}.
5. **if** *l* is odd **then**

   select P1, P2 $\in \{P_{j0}, P_{j1}, ..., P_{j(n-1)}\}$ such that P1 and P2 belong to the same k-digit.

   assign P(i)=P1 and P(n)=P2 .

   assign P(j) randomly from $\{P_{j0}, P_{j1}, ..., P_{j(n-1)}\}$-{P1,P2}: $j \ne i$ and $1 \le j \le n-1$.

$P^l = P^{n-1}(m)\ P(n)\ P^{n-1}(m-1)\ (P(i)+P(n))$.

else

assign P(j) randomly from the set $\{P_{j0}, P_{j1}, ..., P_{j(n-1)}\}$: $1 \le j \le n-1$.

$P^l = P^{n-1}(m)\ P(n)\ P^{n-1}(m)\ P(n)$.

end_if

Proposition 4.3: A ring of length l can be mapped into an [N,K] cube when K>2 and $3 \le l \le N=2^n$.

Example 4.7: Let {(0001),(0011),(0100),(1000)} be the basis obtained from Algorithm Basis. If Algorithm Ring assigns P(1)=(0001), P(2)=(0100), P(3)=(1000) and P(4)=(0011), then one possible ring control sequence produced is the following:

(0001)(0100)(0001)(1000)(0001)(0100)(0001)(0011)

(0001)(0100)(0001)(1000)(0001)(0100)(0010),

A ring of length N-1 starting from node 0 in a [16,4] cube network established by the above control sequence is shown in Fig. 4.3. The nodes in the ring are as follows:

$0 \to 1 \to 5 \to 4 \to 12 \to 13 \to 9 \to 8 \to 11 \to 10 \to 14 \to 15 \to 7 \to 6 \to 2 \to 0$.

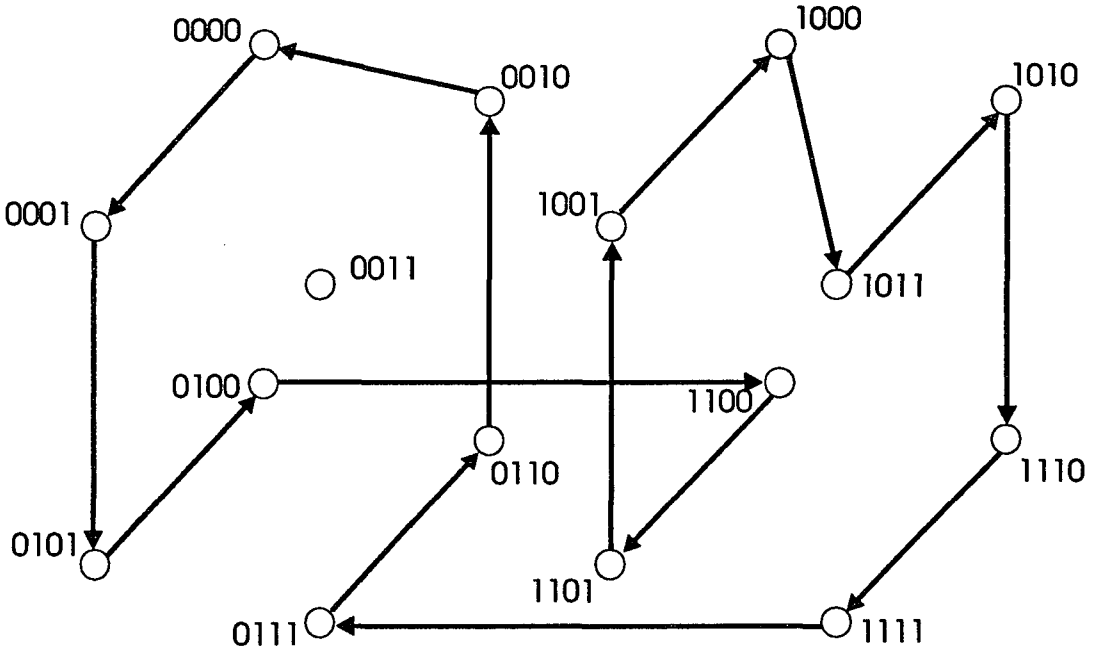

Fig. 4.3. ring of length 15 in [16,4] cube network

4.5 Containments of Linear Arrays and Meshes

There is no difficulty in embedding a linear array, instead of a ring, into an n-cube. It suffices to establish a linear array of arbitrary length $l \leq 2^n-1$, which contains $l+1$ nodes. Given a linear array of arbitrary length l, the smallest dimension n-cube into which it can be mapped is clearly the cube of dimension $n = \lceil \log_2 (l+1) \rceil$.

To map a 2-dimensional mesh or d-dimensional mesh, $d \leq n$, into [N,K] cube is straightforward. The basis set of control vectors are first partitioned into d subsets. Then, each subset is regarded as a basis set and used to generate Gray control sequence of 1-dimension. Thus, a d-dimensional mesh can be established by d Gray control sequences of 1-dimension.

Example 4.8: In a [64,4] cube network, let {(000001), (000011), (000100), (001000), (100000), (010000)} be the basis obtained from Algorithm Basis.

To embed a two dimensional 8 x 8 mesh, we can partition the basis into 2 subsets as follows:

basis(x) = {(000001), (000011), (000100)} and

basis(y) = {(001000), (100000), (010000)}.

The Gray control sequence built from basis(x) is the following:

X^8=(000001)(000011)(000001)(000100)(000001)(000011)(000001)(000100).

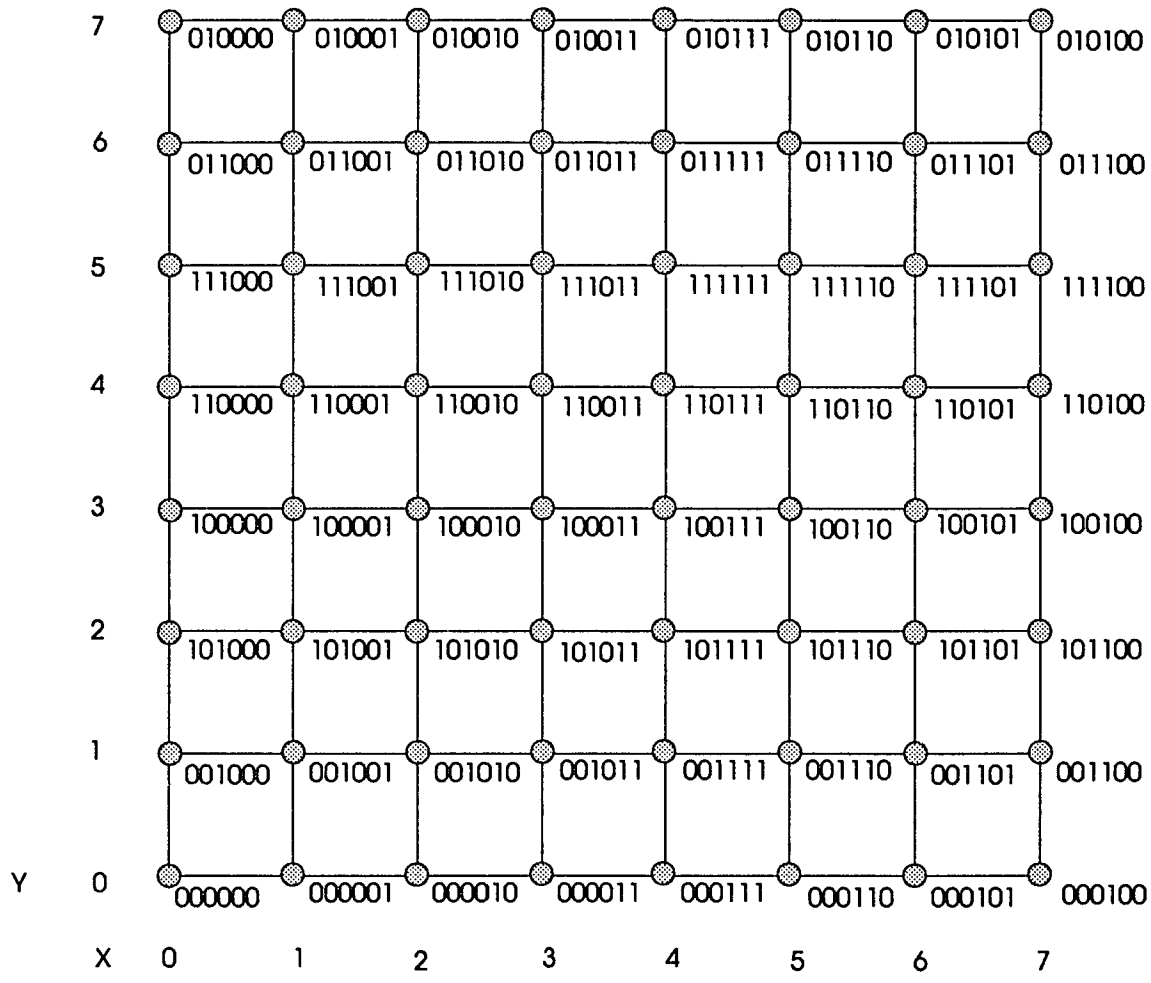

Fig. 4.4. two dimensional 8x8 mesh in [64,4] cube network.

The Gray control sequence built from basis(y) is the following:

Y^8=(001000)(100000)(001000)(010000)(001000)(100000)(001000)(010000).

Every node in the mesh with logical address (x, y) can be connected from original node by the control sequence $X^8(x)Y^8(y)$, where $X^8(x)$ is the first x vectors of X^8 and $Y^8(y)$ is the first y vectors of Y^8.

A two dimensional 8x8 mesh starting from node 0 in a [64,4] cube network established by the above control sequence is shown in Fig. 4.4.

4.6 Containment of Standard Spanning Trees (SST)

For distributed computation of iterative algorithms by a multiprocessor system, each processor needs to broadcast the updated values of variables to other processors in the network. The broadcasting problem has received substantial attention by many researchers. For special case of [N,2] cubes, the lower bound on the amount of time necessary for all nodes to complete broadcasting task is $(N-1)/\log_2 N$. The algorithm to achieve this bound [33], assumes that all nodes are perfectly synchronized, i.e., all processors start and end their computation at the same time.

In what follows, an algorithm for constructing a unique spanning tree, called SST, is presented. The k-digits in an n-vector , where k divides n, are labeled from left to right by

$P_{((n/k)-1)k}$, $P_{((n/k)-2)k}$, ..., P_{0k}. Let P_{0k} through $P_{(a-1)k}$ be the only k-digits different in a pair of nodes (x,y). Then, any shortest path connecting x to y requires control vectors $P_{0k} + P_{1k} +$ ... $+ P_{(a-1)k}$, i.e., $x+P_{0k}+ ... P_{(a-1)k}=y$. The path can be established regardless of the order in which control vectors are formed. To construct a unique spanning tree, we impose the conditions on the ordering of control vectors, namely the control vectors should be applies in increasing order. For instance, in an [16,4] cube, the shortest path between x=0000 and y=0110 is establish by applying control vector P_{0k}=0010 followed by P_{1k}=0100.

Proposition. A SST of an [N,K] cube, with $N=2^{\alpha}K$, has $2^{\alpha k}-1$ links and the number of paths of length α is $(2^k-1)^{\alpha}$.

Proof. Let L(n,k) denotes the number of links in an [N,K] cube. By adding one more k-digit in position m, the number of links increases by $L(n)(2^k-1)$. Thus L(n+k,k) satisfies the recursive equation $L(n+k,k)=2^kL(n,k)+2^k-1$ which has the solution $L(\alpha k,k)=2^{\alpha k}-1$ satisfying the initiation condition of $L(k,k)=2^k-1$. ■

By changing the order of applying control vectors, other spanning trees, isomorphic to SST, can be obtained.

5. Cube Identification

Design of any parallel processing or distributed system is greatly influenced by interconnection structure of the system. Among many possible topologies available, hypercubes are the most attractive, and have been the focus of extensive research in recent years [42],[43],[44],[46],[49],[51].

Some of the hypercube's attractive features are: a) other topologies such as tree, ring, and mesh can be embedded into a hypercube, b) has a short diameter, c) has a simple self-routing property, d) it has high degree of reliability due to alternate paths for each connection, and e) its structure can be enhanced, i.e., its diameter can be reduced and its reliability increased [16]. The properties of the underlying topology of an n-dimensional hypercube, known as the n-cube graph Q_n, has been studied in recent years [16], mainly because of the availability of hypercube multiprocessors [45],[53].

In this chapter, we study the enhanced hypercubes, or [N,K] cubes. The enhanced cube can

achieve considerable reduction in diameter and substantial improvement in mean internode distance and traffic density [15]. We also address the **cube identification** problem, i.e., to determine, whether or not, a given graph is an [N,K] cube.

Motivation for studying this problem are summarized below.

1) Embedding enhanced cube into other topologies or establishing equivalency between enhanced cube and other types of networks can be facilitated by having a theoretical characterization of the enhanced cube as a graph.

2) To alleviate data traffic to be concentrated on a few links, and to reduce communication latency for transferring large among of data, multiple-path routing have been suggested. For instance, consider communication between two processors involving repetitive packet transmission to a single destination. To minimize communication delay several node disjoint, or parallel, paths between the processors can be establish and transmit packets via different paths. The number of paths between source-destination pair of nodes can be increased by introducing additional links between cube nodes, which also improves network reliability. There exists considerable literature on routing algorithms in hypercube networks. The interested reader may find more information and references in [16]. In Chapter 3, we also presented several routing algorithms for [N,K] cube networks.

The rest of this chapter is organized as follows. In section 5.1, an interconnection topology, denoted by $Q_{(n,k)}$ and is called [N,K] cube, which is based on a simple modification of the

n-cube is defined. Our notations and terminologies which are used throughout this chapter are also given in this section. Section 5.2 is devoted to some useful topological properties of enhanced cubes. In section 5.3 [N,K] cube identification is discussed. Finally, concluding remarks appear in Section 5.4.

5.1 Notations and Definitions

Hypercube networks can be enhanced in a number of ways [47],[48],[52],[15]. In this chapter, we define the class of [N,K] enhanced hypercubes, or [N,K] cubes for short, and regard each network topology in this class as a graph, $Q_{(n,k)}$, in which each processor is represented by a node. The [N,K] cube consists of $N=2^n$ nodes, numbered from 0 to N-1. Each node has a binary identifier which is an n-vector, i.e., a vector with n components, coincides with the binary representation of the node. For each node x with binary identity $(x_{n-1}, ..., x_0)$, we define a *k-digit* of x as a group of k, $2^k=K$, consecutive bits, starting from least significant bit x_0. For instance, **ith k-digit** of x, $i \in \{0,1,..., \lceil n/k \rceil -1\}$, consists of bits $x_{ik+k-1} ... x_{ik}$. The $\lceil n/k \rceil -1$ th k-digit covers bits $x_{\lfloor n/k \rfloor k}$ to x_{n-1}. As an example, for n=5 and k=2, 2^{nd} k-digit of x is x_4, and 0th k-digit is x_1x_0.

Definition 5.1: The *Hamming distance*, $H_k(x,y)$, between two nodes x and y is defined as the number of k-digits in which their binary identifiers differ.

We will closely follow the graph theoretical terminology and notation of [50]; terms not defined here can be found in that book. Let G(N,A) represent a graph with node or vertex set V(G)=N and arc set A(G)=A. If an arc P=(x,y)$\in$A, then nodes x and y are said to be adjacent and x and y are the endpoints of arc P. Two arcs are said to be independent, or parallel, if they do not share an endpoint. For a node x$\in$N, L(x) represent the set containing x and all nodes adjacent to x, and its cardinality is denoted by |L(x)|. If G and G' are graphs, then G is isomorphic to a subgraph of G' if there is a one-to-one function h: V(G) $\rightarrow$ V(G') such that each arc (x,y)$\in$A(G) is mapped to an arc (h(x),h(y))$\in$A(G'). If G is a subgraph of G', we will write G$\subseteq$G'. The specific graph with which we will be concerned is $Q_{(n,k)}$ graph, describing the topology of [N,K] cube, as defined next.

Definition 5.2: A graph G with $N=2^n$ nodes such that each pair of nodes x and y with Hamming distance $H_k(x,y)=1$ are directly linked together is called [N,K] cube, where $K=2^k$, and is denoted by $Q_{(n,k)}$.

If x and y identifiers differ in ith k-digit, they will be referred to as *type i neighbors*. It is clear, from Definition 5.2, that $Q_n=Q_{(n,1)} \subseteq Q_{(n,k)}$. Note that $Q_{(n,n)}$ is a complete graph and $Q_{(n,1)}$ is isomorphic to Q_n. The $Q_{(3,2)}$, $Q_{(4,1)}$ and $Q_{(4,2)}$ are sketched in Fig.5.1., 5.2. and 5.3. respectively.

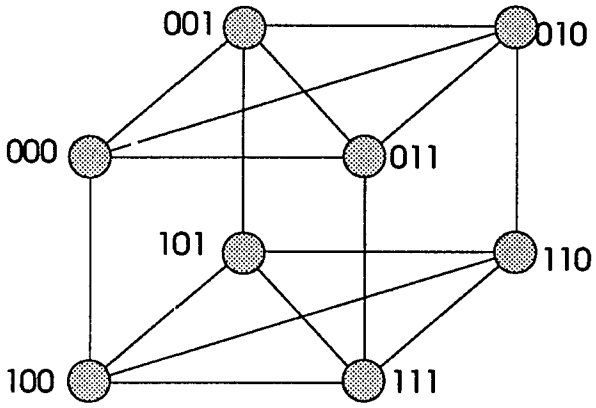

Fig. 5.1. [8,4] cube

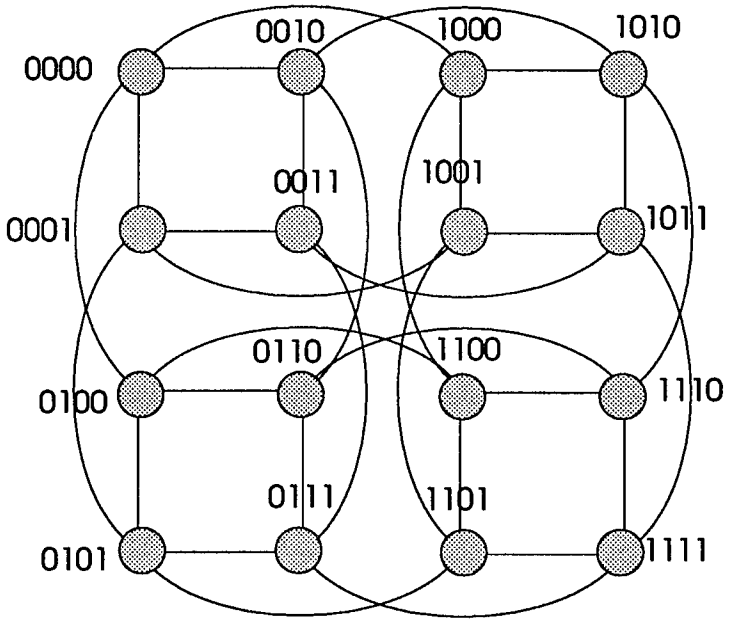

Fig. 5.2. [16,2] cube

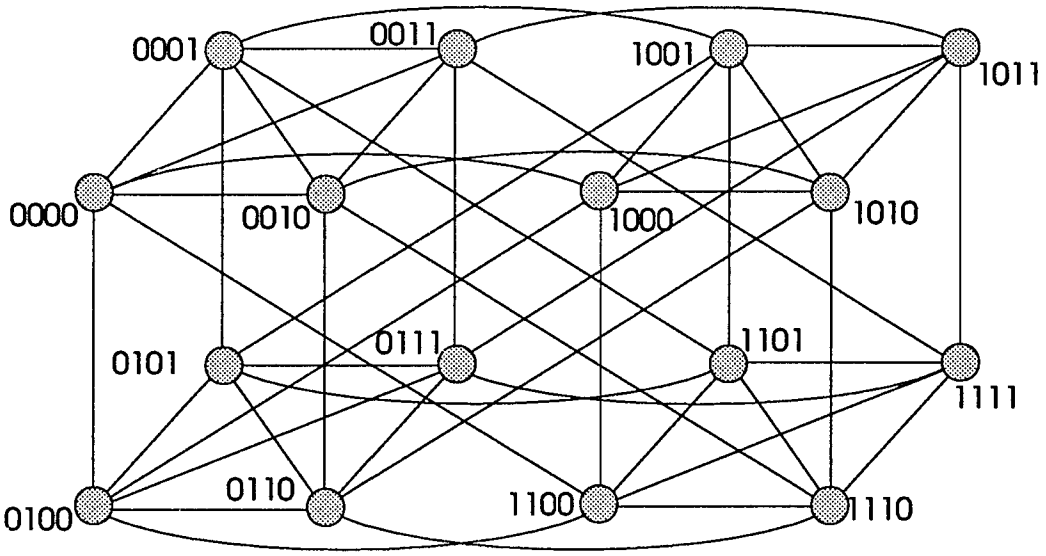

Fig. 5.3. [16,4] cube network

Any path from x to y contains at least as many links as the Hamming distance between x and y. The paths between x and y with exactly $H_k(x,y)$ links are characterized as *shortest.* The following notations are used throughout this chapter.

$\mathbb{N}$ and A denotes the node set and set of arcs of an [N,K] cube respectively.

$Z=\{0,1, ..., \lceil n/k \rceil -1\}$ and $J=\{0,1, ..., 2^k-1\}$

+ denotes modulo-2 addition and for two nodes x and y, $x+y=t$ is a vector $(x_{n-1}+y_{n-1}, ..., x_0+y_0)$. $t=x+y$ is referred to as relative distance of x and y.

For $i \in Z$ and $j \in J$, we denote by $P_i(j)$, an n-vector with all entries of 0 except for the ith k-digit, which is binary representation of j. We refer to this type of vectors as *control vector of type i.* As an example, in an [16,4] Hypercube $P_0(3)=(0011)$ and $P_1(2)=(1000)$. This terminology is appropriate, because vector $P_i(j)$ will be used to manipulate on ith k-digit of the source address and convert it to the ith k-digit of the destination address.

In the remainder of this chapter, to simplify notation, if the value assign to j is irrelevant to our discussion, $P_i(j)$ will be replaced by P_i.

For any arc $(x,y) \in A(Q_{(n,k)})$, x and y are neighbors of type i, for some $i \in Z$. Thus, control vector $P_i(j)$ exists such that $x+y=P_i(j)$, $j \in J$, or $y=x+P_i(j)=\phi(x,P_i(j))$. (1)

The mapping ϕ, defined by (1), is called **routing function.** Node y is said to be reachable from node x if there exist $P_i(j)$ such that $y=\phi(x,P_i(j))$. As an illustrative example, in a [16,4] cube node (1100) has the following type 0 neighbors.

$\phi((1100),P_0(1)) = (1100)+(0001) = (1101)$, $\phi((1100),P_0(2)) = (1100)+(0010) = (1110)$, and $\phi((1100),P_0(3)) = (1100)+(0011) = (1111)$. Nodes $P_i(j)$, for all $i \in Z$ and $j \in J$, are the only neighbors of node (0, ..., 0). In general, each node x has $x+P_i(j)$, for all possible i and j, as neighbors.

The main objectives of enhancing a hypercube is to a) improve its fault-tolerance and b) reduce its ***diameter***. From the above discussion, it is easily seen that the *diameter* of the [N,K] cube is $\lceil n/k \rceil$, where $\lceil a \rceil$ denotes the smallest integer than or equal to a. Thus, by a factor of k the diameter of $Q_{(n,k)}$ is shorter than that of Q_n.

By using the notion of control vector, one can formally define the [N,K] cube topology as:

$A(Q_{(n,k)})=A(Q_n)+\{(x,y) | x+y=P_i(j), \forall i \in Z, \forall j \in J \text{ and } j \neq 2^m, m=0, 1, ..., k-1\}$.

5.2 Topological Properties of [N,K] Cubes

In this section topological properties of [N,K] cube will be studied. These properties are used in the next section to solve the [N,K] cube identification problem. The orbit of a node , as defined below, plays an important role in formal description of [N,K] cubes.

Definition 5.3.: The *i-orbit* (or type i orbit) of a node x of an [N,K] cube is defined to be the set of nodes reachable from x by $P_i(j)$, $i \in Z$ and $\forall j \in J$, and is denoted by $O_i(x)$, i.e., $O_i(x)=\{y | \phi(x,P_i(j))=y \text{ for all } j \in J\}$.

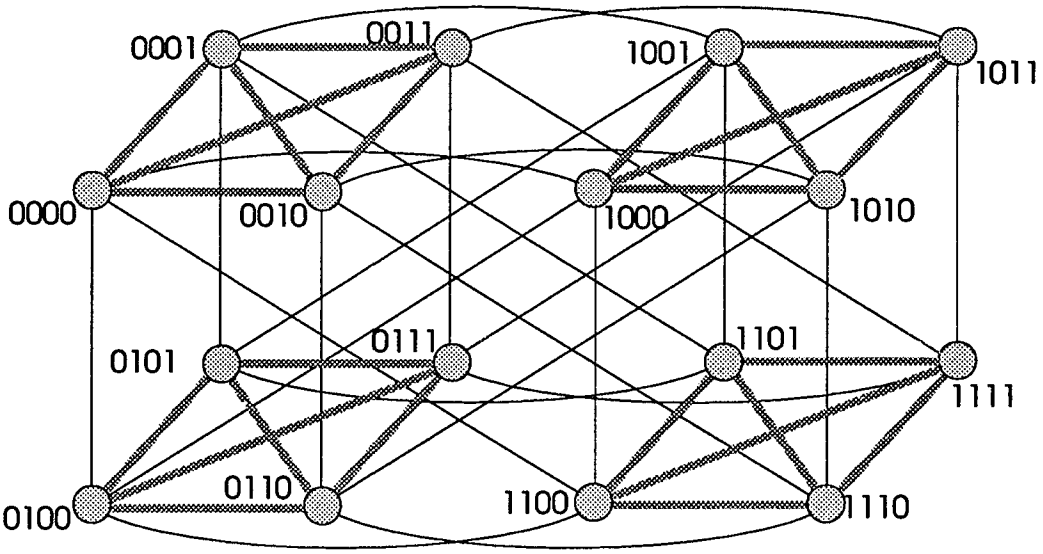

Fig. 5.4. 0 orbit of [16,4] cube network

The bold lines in Fig. 5.4. show 0 orbit in a $Q_{(4,2)}$. Some useful properties of $O_i(x)$ are listed in the following theorem.

Theorem 5.1: Let x and y be two nodes of an [N,K] cube. Then

a. $O_i(x)=O_i(y)$ if and only if $x+y=P_i(j)$, $j\in J$

b. $O_i(x)\cap O_i(y)=\varnothing$ if and only if $x+y\neq P_i(j)$,$\forall j\in J$

c. $\cup_{\forall i}(O_i(x)=\mathbb{N}$

d. There exist 2^{n-k} distinct orbits of type i and each orbit contains 2^k nodes.

e. Each node of the set $L(x)-O_i(x)$ belongs to a different orbit of type i.

f. For all $x,y \in O_i(x)$, there exist $z\in L(x)-O_i(x)$ and $w\in L(y)-O_i(x)$ such that $z\in O_i(w)$.

Outline of the Proof:

(a) For a fixed node x; let $L(x)=\{x+P_i(j)|\ i\in Z,\forall j\in J\}$. It can be seen that $O_i(x)=L(x)\cap L(y)$, $y\in L(x)$ and $x+y=P_i$. Thus $O_i(y)=L(y)\cap L(x)=O_i(x)$.

(b) Assume that there exist node $z\in\mathbb{N}$ such that $z\in O_i(x)$ and $z\in O_i(y)$. This implies that $x+z=P_i(j)$ and $y+z=P_i(j')$ for some j and $j'\in J$, which in turn implies that $x+y=P_i(J'')$, which is a contradiction.

(c) follows from the fact that $\forall x\in\mathbb{N}$, x is covered by one of the i-orbits.

(d) Follows from definition of orbit.

(e) For all y_1 and $y_2 \in L(x)-O_i(x)$, $H_k(x,y_1)=H_k(x,y_2)=1$ (since they are member of L(x)), and since ith k-digit of x, y_1, and y_2 are the same (otherwise, they would be members of $O_i(x)$), y_1 and y_2 each belong to a different i-orbits.

(f) Since x and y are in the same orbit of type i, their remaining k-digits are the same. By (e), z and w member of $L(x)$-$O_i(x)$ and $L(y)$-$O_i(x)$ respectively, can be found such that $H_k(x,z)=H_k(y,w)=1$ and z and w have identical k-digits except the ith one. Thus $z+w=P_i(j)$, for some $j\in J$, and they belong to the same i-orbit. ■

The main implication of Theorem 5.1 is that the set $\mathbb{N}$ can be partitioned into 2^{n-k} blocks and each block contains one of the orbits of type i. To explore this property of orbits further, we next define two relations on $\mathbb{N}$ and $A(Q_{(n,k)})$.

Definition 5.4: Let $\mathbb{N}$ denote the node set of an [N,K] cube. Define relation R_i and S on $\mathbb{N}$ and $\mathbb{N}^2$ respectively as:

xR_iy if and only if $O_i(x)=O_i(y)$, $\forall x,y \in \mathbb{N}$. (2)

$(x,y)S(z,w)$ if and only if x and $z\in O_i(x)$ implies that y and $w \in O_i(y)$, for all $i\in Z$ and all x and y such that $x+y\neq P_i(j)$, $j\in\{0,1, \ldots ,K-1\}$. (3)

Lemma 5.1: Relation R_i and S defined by (2) and (3) are equivalence relations.

As an illustrative example, in Fig. 5.4., there are four 0-orbits; $O_0(0) = \{0,1,2,4\}$, $O_0(4) = \{4,5,6,7\}$, $O_0(8) = \{8,9,10,11\}$. and $O_0(12) = \{12,13,14,15\}$. The class of S-equiv{(4,8),(5,9),(6,10),(7,11)}, $[(8,12)]_S = \{(8,12),(9,13),(10,14),(11,15)\}$, and $[(12,0)]_S =$ {(4,8),(5,9),(6,10),(7,11)}, $[(8,12)]_S = \{(8,12),(9,13),(10,14),(11,15)\}$, and $[(12,0)]_S =$

{(12,0),(13,1),(14,2),(15,3)}.

From the above discussion, it is clear that each orbit contains a set of nodes which are completely connected, and the set of arcs connecting two orbits are equivalent. Thus it is possible, by mapping each orbit into a single node and replacing all equivalent arcs by a single arc, to find a subgraph of G which is isomorphic to an [N/K,K] cube. Such a mapping, δ, is defined next.

Definition 5.5: Function δ_i, is defined on the node set of an $Q_{(n,k)}$ as

$$\delta_i(x_{n-1}..x_{ik+k},x_{ik+k-1}... x_{ik},x_{ik-1}, ... x_0) = (x_{n-1} ... x_{ik+k}\ x_{ik-1} ... x_0).\ i\in Z, \text{ and } j\in A \qquad (4)$$

i.e., $\delta_i(x)$ is a node obtained by removing ith k-digit of binary representation of node x.

Some useful properties of mapping δ_i are listed below.

Theorem 5.2: Let δ_i be the mapping defined by equation (4), then

a. $\delta_i(y)=\delta_i(x)$ for all $y\in O_i(x)$

b. $\forall x,y\in N$, if $H_k(x,y)=1$ and $O_i(x)\cap O_i(y)=\phi$, then $H_k[\delta_i(x),\delta_i(y)]=1$

c. Graph $\delta(G)=G'=(N',A')$ defined as $N'=\{\delta_i(x)|x\in N\}$ and $(\delta(x),\delta(y))\in A'$ if $(x,y)\in A$ and $O_i(x)\cap O_i(y)=\phi$ is an [N/K,K] cube.

Outline of the Proof.

(a). Since $O_i(x)$ contains nodes with identical k-digits, except in ith position, $\delta(x)=\delta(y)$.

(b). Since $O_i(x)$ and $O_i(y)$ are disjoint and $H_k(x,y)=1$, jth k-digit of x and y, $j\neq i$, must be different, implying that the Hamming distance of $\delta_i(x)$ and $\delta_i(y)$ is 1.

(c). Since there are 2^k nodes in each orbit and corresponding to each orbit there is a single node in $\mathbb{N}'$, the cardinality of $\mathbb{N}'$ is 2^{n-k}. By (b) each pair of nodes, $\delta(x)$ and $\delta(y)$, of $\mathbb{N}'$ with Hamming distance of 1 are connected. Thus G' is an [N/K,K] cube. ■

5.3 [N,K] Cube Identification

In this section, we will use structural properties of the [N,K] cube, developed in the previous section, to solve the cube identification problem, i.e., to determine whether or not a given graph $G=(\mathbb{N},A)$ of N nodes is isomorphic to a $Q_{(n,k)}$. The following notations are used throughout this section.

For a given graph $G=(\mathbb{N},A)$, let L(x), O(x), and O(x,y) denote the following sets.

L(x): Set containing x and all of its neighbors.

O(x): A set of nodes, including x, forming a complete subgraph of G. O(x) is referred to as **orbit of x.**

O(x,y): The orbit of x which contains node y.

Proposition 5.1: $O(x,y) = L(x) \cap L(y)$ and is unique.

Proof: $\forall z \in O(x,y)$, $z \in L(x)$ and $z \in L(y)$, thus $z \in L(x) \cap L(y)$. Now let $z \in L(x) \cap L(z)$, then links (z,x) and (z,y) exist, implying that $z \in O(x,y)$. Therefore, $O(x,y)=L(x) \cap L(y)$. Uniqueness of O(x,y) can be simply proven by contradiction. ■

Theorem 5.3: Let G=(N,A) be a connected graph of N nodes, then G is an [N,K] cube, or $Q_{(n,k)}$, if and only if

a. G has $N=2^n$ nodes, where n is assumed to be multiple of $k=\log_2 K$.

b. Every node has degree $n(2^k-1)/k$

c. For any node x in N, O(x) contains 2^k nodes

d. If G is not a complete graph, then for all $(x,y) \in A$ and for all $z \in L(x)-O(x,y)$, there exist $w \in L(y)-O(x,y)$, such that $O(x,z) \cap O(y,w)=\phi$ and O(x,z) and O(y,w) are connected in one-to-one fashion.

Outline of the Proof:

By Theorem 5.1, conditions a-d are necessary for a graph to be an [N,K] cube.

Sufficiency can be proven by induction on a=n/k. The case of a=1, i.e., [K,K] cube is trivial. Assume that any network of 2^{n-k} nodes (n/k=a-1) satisfying conditions a-d is an [N/K,K] cube. We show that a graph of 2^n nodes, satisfying the above conditions, can be partitioned into 2^k identical subgraphs, each having similar properties as the original graph.

Let function δ be defined on the set N as $\delta(z)=x$ for all $z \in O(x)$.

Note that since by Theorem 5.1 $O(z)=O(x)$ if and only if $z \in O(x)$, δ is well defined. Based on δ, G is mapped into graph $\delta(G)= G'=(N',A')$ as follows.

i. Start with any pair of nodes x and z and form $O(x,z)=L(x) \cap L(z)$.

ii. Map O(x,z) into x and assign x to N'.

iii. For each $y \in L(x)-O(x,z)$ find $w \in L(z)-O(x,z)$ such that $(y,w) \in A$ (by condition d such a node exists). Map O(y,w) into y and assign (y,w) to A'.

iv. Replace x and z by y and w respectively and go to step (ii) and continue till all the nodes in the graph G are covered.

Note that since by (b) each orbit contains 2^k nodes, mapping δ can be defined initially on O(x,z) in 2^k different ways. We show that the 2^k subgraphs, one for each choice of mapping, are node disjoint. This implies that Graph G can be partitioned into 2^k node disjoint subgraphs.

Suppose $G_1'=(N_1',A_1')$ and $G_2'=(N_2',A_2')$ are two subgraphs obtained by defining $\delta(z)=x_1$ and $\delta(z)=x_2$, for all $z \in O(x_1,x_2)$ and $x_1 \neq x_2$, respectively. if $O(y_1,y_2)$ is the orbit connected to $O(x_1,x_2)$ and $(x_1,y_1) \in A$, then $y_1 \in N_1'$ and by property d, a node such as $y_2 \neq y_1$, is connected to x_2 and therefore $y_2 \in N_2'$. With the same argument all nodes added to N_1' and N_2', during construction of the two subgraph, are different and as a result the G_1' and G_2' are node disjoint. With the similar reasoning, one can conclude that all 2^k subgraphs constructed by mapping δ are node disjoint.

Next, we prove that G' satisfies conditions a-d, with N replaced by $N/2^k=N/K$, and thus by induction hypothesis is an [N/K,K] cube. Each node of graph G belongs to an orbit which

by (c) contains 2^k nodes. Since each orbit is mapped into one node of $\mathbb{N}'$, $\mathbb{N}'$ contain $N/2^k = 2^{n-k}$ elements, thus property a holds for G'. To show that G' has property b, first we note that, as a consequence of property d, nodes in L(x)-O(x,y) belongs to different orbits (otherwise, one-to-one connection of orbits would be violated). And since each orbit is mapped into a distinct node in G'(that is 2^k-1 links are removed) the degree of x as a node in G' is 2^k-1 less than the degree of the same node in G.

As a result each node of G' has degree of $n(2^k-1)/k -(2^k-1)=(n-k)(2^k-1)/k$.

To show that each node in G' form an orbit with 2^k nodes, consider nodes x and y of G' with $(x,y)\in A'$. Since $(x,y)\in A$, then by (c) O(x,y) contains 2^k elements and by (d) each node in O(x,y) is in different orbits of graph G, thus each one is mapped into a node which is linked to the node x of G'. This implies that each orbit of G' contains 2^k elements.

Finally we show that property (d) holds for G'. Let $(x,y)\in A'$, then by (c) O(x) and O(y) of G' exist and each contains 2^k members. If disjoint O(x) and O(y) can not be found, then G' must be a complete graph. If O(x) and O(y) are node disjoint, then each link between the two orbits is also a link in the graph G connecting two orbits together (not necessarily the orbits chosen to split G) and thus by (d) they are connected in one-to-one fashion.

Next, by labeling nodes of G', we show how these 2^k subcubes can be connected to form the original graph G satisfying definition of [N,K] cube. Since G' is a [N/K,K] cube, label the nodes of G' conforming to the cube requirement. Repeat this step for other 2^k-1 subgraph G' with the condition that those nodes which are part of the same orbit have identical labels. Next label each orbit, which connects nodes of different subgraphs with the

same label, starting from 0 to 2^k-1 such that the nodes of two connected orbits have the same orbit identification (by property d this is possible). Finally, form the concatenation of orbit identification with subgraph label and assign it as the label of each node in G. Clearly, any two nodes of G which are connected are either in the same orbit or in two connected orbits implying that their corresponding labels differ in one k-digit, which in turn implies that G is an [N,K] graph. ■

As a simple example, graph G1 and G2, shown in Fig. 5.5. and Fig. 5.6., satisfy conditions stated in Theorem 5.3 and isomorphic to $Q_{(4,2)}$ and $Q_{(4,1)}$ respectively. Graph G3 in Fig. 5.7. violates property d and as a result is not a cube network.

Next, we will demonstrate that the structural characteristics of an [N,K] cube, as described in the Theorem 5.3., is an extension of the work reported in [16], by proving that the two results converges for the special case of k=1. Conditions (a) and (b) of Theorem 5.3. when k=1, are identical to (1) and (2) in Theorem 2.1 of [16]. Condition (c) is not necessary to be tested on a Q_n, since the largest complete subgraph of Q_n consists of two nodes. As shown in the proof of the Theorem 5.3., if a graph satisfies condition (d), automorphism δ can be defined on that graph. Thus, for any two adjacent nodes x and y in the graph, the nodes adjacent to x are linked in a one-to-one fashion to those adjacent to y, which is condition (4) stated in [16].

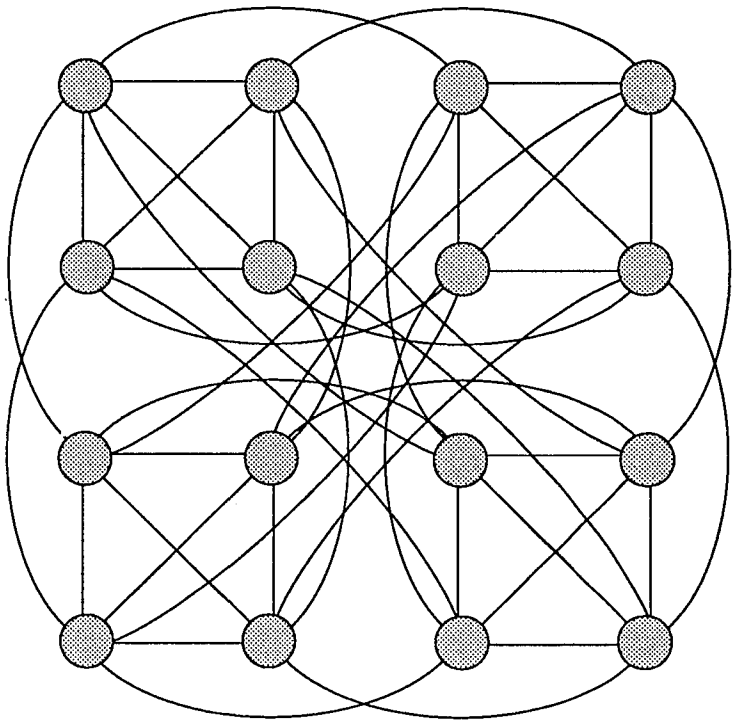

Fig. 5.5. Graph G1

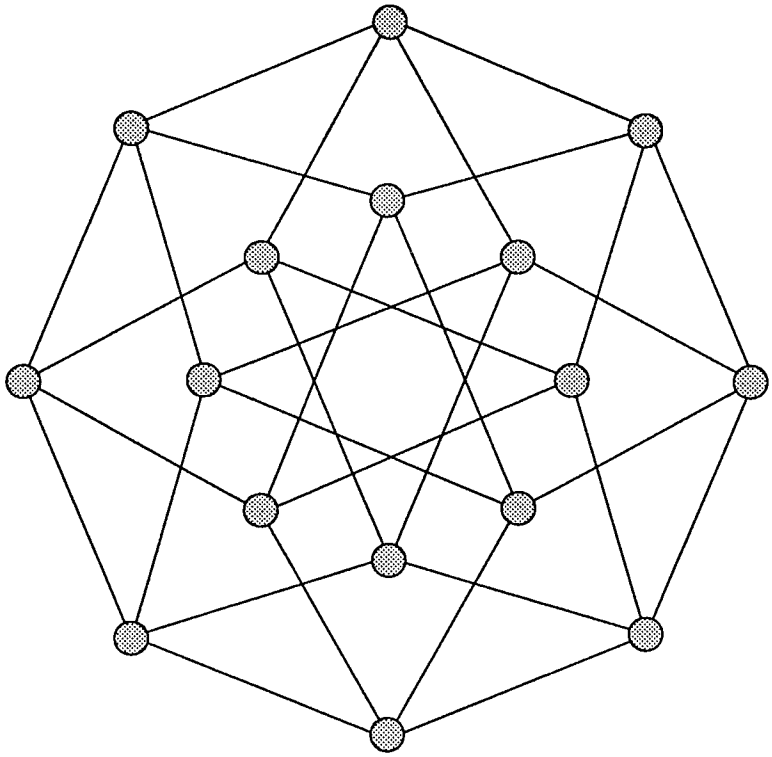

Fig.5.6. Graph G2

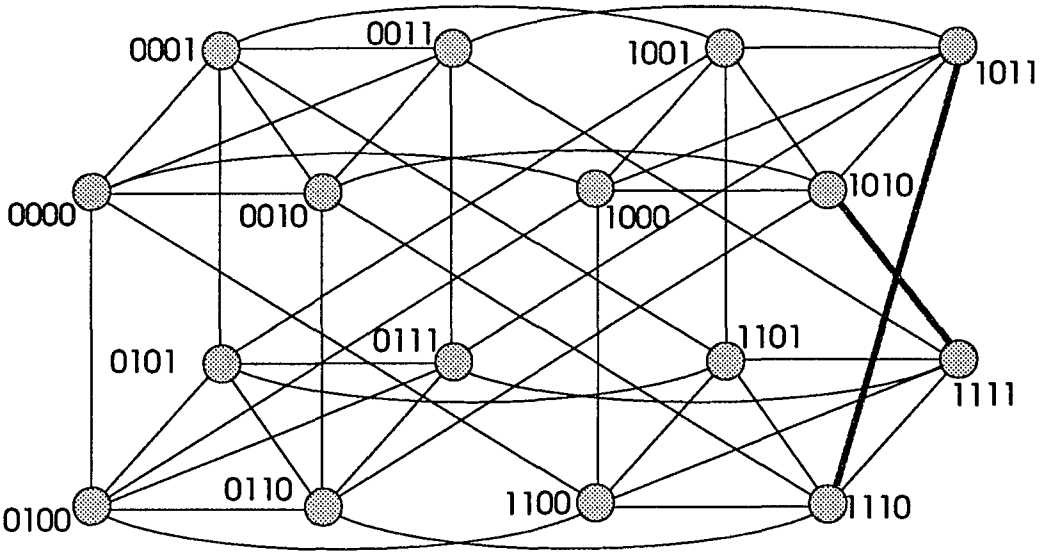

Fig. 5.7. Graph G3

Labeling the nodes of a graph which satisfies properties a-d of Theorem 5.3 is not unique. The next theorem shows in how many different ways the nodes can be labeled to conform to the cube labeling requirement.

Theorem 5.4: There are $(n/k)!.[(2^k)!]^{n/k}$ different ways in which the 2^n nodes of an [N,K] cube can be labeled.

Proof: Let M(n,k) denote the number of ways that the nodes of an [N,K] cube can be labeled. According to the Theorem 5.3, by fixing a k-digit of any node identifier, the cube can be decomposed into 2^k subcubes and there are C(n/k,1) different ways of selecting a single k-digit. The number of different ways that each $[N/2^k,K]$ subcube can be labeled is M(n-k,k) and in a group of 2^k subcubes, obtained from fixing a k-digit ,one can distinguished them from each other by appending a k-digit to each subcube within the group and there are $(2^k)!$ ways of doing that. Thus, one can conclude that M(n,k) must satisfy the recursive equation $M(n,k)=(n/k)\ (2^k)!M(n-k,k)$. Solving this recursive equation results in $M(n,k)= (n/k)!.[(2^k)!]^{n/k}$. ■

Note that if 2^k nodes in an [N,K] cube are directly linked together, their identifier differ in exactly one k-digit.

5.4 Concluding Remarks

This chapter has focused on the problem of enhanced hypercube identification, which was addressed in Theorem 5.3. The class of [N,K] cubes are defined and their topological properties were studied. In summary, the [N,K] cube, $Q_{(n,k)}$, has the following properties. $Q_{(n,k)}$ consists of 2^k disjoint $Q_{(n/k,k)}$ subgraphs and its diameter is $\lceil n/k \rceil$. There are $(n/k)![(2^k)!]^{n/k}$ different ways in which the nodes of an [N,K] cube can be labeled, and Q_n is a subgraph of $Q_{(n,k)}$. The results presented in this chapter is an attempt to response to the following two graph theoretic questions.

1) How to enhanced the topology of Q_n in order to reduce its diameter by a factor of k.

2) How to characterize the enhanced topology by a set of structural properties, which enables us to determine whether or not a given graph is its isomorphic image.

6. Conclusion and Future Research

In this thesis, we have introduced a new mathematical tool, called Communication-Algebra, for modeling, analysis, and control of interprocessor communication networks. In chapter 2, Communication-Algebra, was defined and used to investigate many useful properties of the networks and to link them to their control structures. Two classes of networks, [N,K] cube and [N,K] PM2I, were modeled, and some of their properties were studied. One application of C-Algebra, enhancement of communication networks, was also discussed in this chapter.

Chapter 3 is devoted to a discussion on network's fault-tolerance. The class of [N,K] cube is shown to be maximally fault tolerant, i.e., the maximum number of parallel paths between any two nodes is the same as the degree of each node, n/k(K-1), and the length of each parallel path is at most equal to the Hamming distance plus 2. The Algorithm DP_G is developed to generate these parallel paths. By combining Algorithm DP_G and depth-first search we proposed an adaptive fault-tolerant routing algorithm, called AFTR, which will

successfully route messages between any pair of functional nodes in an [N,K] cube as long as there exists a path between them. More research on Algorithm AFTR is needed to evaluate its performance, i.e., average path length generated, in term of probability of a link failure. The lower bound on the number of parallel paths in an [N,K] PM2I are obtained in Corollary 3.2. However, more research is needed to investigate the degree of fault-tolerance of an [N,K] PM2I.

In chapter 4, we have proposed several node-independent algorithms for embedding other topologies, such as rings, linear arrays, meshes and standard spanning trees in the [N,K] cubes. We used Gray control sequences to establish Hamiltonian cycles in an [N,K] cube, and developed the theoretical foundation for investigating properties of Gray control sequences and numerated the number of distinct Gray control sequences in an [N,K] cube.

In chapter 5, The class of [N,K] cubes' topology properties along with a formal characterization of [N,K] cube as a graph are presented. These properties were used to determine whether or not a given graph is isomorphic to an [N,K] cube.

Homomorphism between networks is one of the topics left for future research. For example, using C-algebra to investigate the existence of homomorphism between networks which result in simulating one network by others. Homomorphism is formally defined as follows:

Definition 6.1: Let $\mathbb{C} = \langle \mathbb{N}, \mathcal{P}, \phi \rangle$ and $\mathbb{C}' = \langle \mathbb{N}', \mathcal{P}', \phi' \rangle$ be interconnection networks. A **homomorphism** from $\mathbb{C}$ onto $\mathbb{C}'$ is a pair of maps, $\mathcal{H} = \langle h_N , h_{\mathcal{P}} \rangle$,

$h_N : \mathbb{N}' \rightarrow \mathbb{N}_1$, where $\mathbb{N}_1$ is a subset of $\mathbb{N}$ and h_N is one-to-one and onto,

$h_{\mathcal{P}} : \mathcal{P}' \rightarrow \mathcal{P}_1$, where $\mathcal{P}_1$ is a subset of $\mathcal{P}$,

such that $h_N (\phi'(S', P')) = \phi(h_N(S'), h_{\mathcal{P}}(P'))$, $\forall S' \in \mathbb{N}'$, $\forall P' \in \mathcal{P}'$.

If $\mathbb{C}'$ is homomorphic image of $\mathbb{C}$, then the network $\mathbb{C}$ simulates the interconnection network $\mathbb{C}'$. Since $\mathbb{N}_1$ is a subset of $\mathbb{N}$, it is possible to decompose the interconnection network $\mathbb{C}$ into several subnetworks, $\mathbb{C}_1 , \mathbb{C}_2 , \ldots , \mathbb{C}_m$, such that $\mathbb{N}_i \cap \mathbb{N}_j = \varnothing$, $\forall 1 \leq i, j \leq m$ and $i \neq j$. The theory underlying the partitioning of permutation networks has been explored [6]. However, node-independent algorithms to perform dynamically partitioning the networks is yet to be discovered.

In summary, this thesis demonstrates many applications of C-Algebra in modeling the classes of [N,K] cube and [N,K] PM2I. The methodology presented in this thesis can be applied to other types of networks such as the class of n-stars and mesh connected networks.

Bibliography

[1] Feng, T.Y., "A Survey of Interconnection Networks," IEEE Computer, pp. 12-27, Dec. 1981.

[2] Lawrie, D.H., "Access and Alignment of Data in an Array Processor," IEEE Trans. Comput. vol. C-24, pp. 1145-1155, Dec. 1975.

[3] Goke, L.R. and Lipoviski, G.J., "Banyan networks for partitioning multiprocessor systems, " Proc. First Annual Computer Architecture conf., pp. 21-28, Dec. 1973.

[4] Siegel, H.J., "Analysis techniques for SIMD machine interconnection networks and the effect of processor address masks," IEEE Trans. Comput. vol. C-26, pp.153-161, Feb. 1977.

[5] Siegel, H.J., "A Model of SIMD Machines and a Comparison of Various Interconnection Networks," IEEE Trans. Comput. vol. C-28, pp. 907-917, Dec. 1979.

[6] Siegel, H.J., "The Theory Underlying the Partition of Permutation Networks," IEEE Trans. Comput. vol. C-29, pp.791-801, Sept. 1980.

[7] Siegel, H.J. and McMillen, R.J., "Using the Augmented Data Manipulator Network in PASM," IEEE Computer, pp.25-33, Feb.1981.

[8] Siegel, H.J., Interconnection Networks for Large-scale Parallel Processing. Lexington, MA: Lexington Books, 1986.

[9] Yalamanchil, S. and Aggarwal, J.K., "A Characterization and Analysis of Parallel Processor Interconnection Networks," IEEE Trans. Comput. vol. C-36, pp.680-691, June 1987.

[10] Wu, C.L. and Feng, T.Y., "On a Class of Multistage Interconnection Network," IEEE Trans. Comput. Vol. C-29, pp.694-702, Aug. 1980.

[11] Wu, C.L. and Feng, T.Y., "The Reverse-Exchange Interconnection Network," IEEE Trans. Comput. vol. C-29, sept. 1980.

[12] Patel, J.H., "Performance of Processor-Memory Interconnection for Multiprocessors," IEEE Trans. Comput. vol. C-30, pp. 771-780, Oct. 1981.

[13] Abraham, S. and Padmanabhan, K., "Performance of the Direct Binary n-Cube Network for Multiprocessors," IEEE Trans. Comput. vol. C-38, pp.1000-1011, July 1989.

[14] Ghozati, S.A., "Modeling and Analysis of a Class of Interconnection Networks", summitted, IEEE , 1993.

[15] Tzeng, N.F. and Wei, S., "Enhanced Hypercubes," IEEE Trans. Comput., vol. C-40, pp.284-294, March 1991.

[16] Saad, Y. and Schultz, M.H., "Topological Properties of Hypercubes," IEEE Trans. Comput., vol C-37, pp.867-872, July 1988.

[17] Adams, G.B. and Siegel, H.J., "On the Number of Permutations Performable by the Augmented Data Manipulator Network," IEEE Trans. Comput.., vol C-31, pp.270-277, April 1982.

[18] Parker, D.S. and Raghavendra C.S., "The Gamma Network," IEEE Trans. Comput., vol C-33, pp.367-373, April 1984.

[19] McMillen, R.J. and Siegel, H.J., "Routing Schemes for the Augmented Data Manipulator Network in an MIMD System," IEEE Trans. Comput., vol C-31, pp.1202-1214, Dec. 1982.

[20] Rau, D., Fortes, J.A.B. and Siegel, H.J., "Destination Tag Routing Techniques Based on a State Model for the IADM Network," IEEE Trans. Comput., vol. C-41, pp.274-285, March 1992.

[21] Katseff, H.P., "Incomplete Hypercubes," IEEE Trans. Comput., vol. C-37, pp.604-608, May 1988.

[22] Bhuyan, L.N. and Agrawal, D.P., "Design and Performance of Generalized Interconnection Networks," IEEE Trans. Comput., vol. C-32, pp.1081-1090, Dec. 1983.

[23] Gottlieb, A., Grishman, R., Kruskal, C.P., McAuliffe, K.P., Rudolph, L. and Snir, M., "The NYU Ultracomputer-Designing an MIMD Shared Memory Parallel Computer," IEEE Trans. Comput., vol. C-32, pp.175-189, Feb. 1983.

[24] Padmanabhan, K. and Lawrie, D.H., "A Class of Redundant Path Multistage Interconnection Networks," IEEE Trans. Comput., vol C-32, pp.1099-1108, Dec. 1983.

[25] Kumar, V.P. and Reibman, A.L., "Failure Dependent Performance Analysis of a Fault-Tolerant Multistage Interconnection Network," IEEE Trans. Comput., vol C-38, pp.1703-1713, Dec. 1989.

[26] Wu, C.L. and Lee, M., "Performance Analysis of Multistage Interconnection Network Configurations and Operations," IEEE Trans. Comput., vol C-41, pp.18-27, Jan. 1992.

[27] Chen, M.S. and Shin, K.G., "Adaptive Fault-Tolerant Routing in Hypercube Multicomputer," IEEE Trans. Comput., vol C-39, pp.1406-1416, Dec. 1990.

[28] Agrawal, D., "Graph Theoretical Analysis and Design of Multistage Interconnection Networks," IEEE Trans. Comput., vol C-32, pp.637-648, July 1983.

[29] Jeng, M. and Siegel, "Design and Analysis of Dynamic redundancy Networks," IEEE Trans. Comput., vol C-37, pp.1019-1029, Sept. 1988.

[30] Stone, H.S., High-Performance computer Architecture, second ed. Reading, Massachusetts: Addison-Wesley Publishing Company, 1990.

[31] Hennessy, J.L. and Patterson, D.A., Computer Architecture: A Quantitative Approach. San Mateo, California: Morgan Kaufmann Publishers, Inc., 1990.

[32] Hwang, K., Advanced Computer Architecture: Parallelism, Scalability, Programmability. New York: McGraw-Hill, Inc., 1993.

[33] Bertsekas, D.P. and Tsitsiklis, J.N., Parallel and Distributed Computation: Numerical Methods. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1989.

[34] Siegel, H.J. and McMillen, R.J., "The Multistage Cube: A Versatile Interconnection Network," IEEE Computer, pp.65-76, Dec. 1981.

[35] Barnes, G.H. and Lundstrom, S.F., "Design and Validation of a Connection Network for Many-Processor Multiprocessor Systems," IEEE Computer, pp.31-41, Dec. 1981.

[36] Blake, J.T. and Trivedi, K.S., "Multistage Interconnection Network Reliability," IEEE Trans. Comput., vol C-38, pp.1600-1604, Nov. 1989.

[37] Esfahanian, A.H., "Generalized Measures of Fault Tolerant with Application to N-Cube Networks," IEEE Trans. Comput., vol C-38, pp.1586-1591, Nov. 1989.

[38] Latifi, Shahram, "Combinatorial Analysis of the Fault-Diameter of the n-cube," IEEE Trans. Comput., vol C-42, pp.27-33, Jan. 1993.

[39] Even, Shimon, Graph Algorithms. Rockville, Maryland: Computer Science Press, Inc., 1979.

[40] Bavel, Zamir, Introduction to the Theory of Automata. Reston, Virginia: Reston Publishing Company, Inc., 1983.

[41] Leon, Steven J., Linear Algebra with Applications, New York: Macmillan Publishing Co., Inc., 1980.

[42] Abraham, S. and Padmanabhan, K. "Reliability of the Hypercube," Proc. Int. Conf. on Parallel Processing, vol. 1, pp. 90-94, Aug. 1988.

[43] Amawy, A.E. and Latifi, S. "Properties and Performance of Folded Hypercubes," IEEE Trans. Parallel Distributed Systems 2, 1, pp. 31-42, Jan. 1991.

[44] Bhuyan, L.N. and Agrawal, D.P. "Generalized hypercube and hyperbus structures for a computer network," IEEE Trans. Comput., vol. C-33, pp. 323-333, 1984.

[45] Chan,T.F. and Saad, Y. "Multigrid algorithms on the hypercube multiprocessors," IEEE Trans. Comput., vol. C-35, no. 11,pp. 969-977, 1986.

[46] Chen, H.L. and Tzeng, N.F. "Efficient Resource Placement in Hypercubes Using Multiple-Adjacency Codes," IEEE Trans. Comput., vol. 43, no. 1, pp. 23-33, Jan. 1994.

[47] Chen, H.L. and Tzeng, N.F. "Enhanced Incomplete Hypercubes," Proc. International Conference on Parallel Processing, IEEE Computer Society, Silver Spring, MD, Vol. I, pp. 270-277, Aug. 1989.

[48] Esfahanian, A. Ni, L.M. and Sagan, B.E. "On enhancing hypercube multiprocessors ,"in Proc. 1988 Int. Conf. Parallel Processing, pp. 86- 89, Aug. 1988.

[49] Grunwald, D.C. and Reed, D.A. "Benchmarking hypercube hardware and software," Tech. Rep. UIUCDCS-R-86-1303, Dept. Comput. Sci., Univ. of Illinois at Urbana-Champaign, 1986.

[50] Harary, F., Graph Theory, Reading, MA: Addison-Wesley,1972.

[51] Hayes, J.P., Mudge, T.N., Stout, Q.F., Colley, S. and Palmer, J., "Architecture of a hypercube supercomputer," in Proc. 1986 Int. Conf. Parallel Processing, pp. 653-660, Aug. 1986.

[52] Hsu, W. T., Yew, P.C. and Zhu, C.Q. "An enhanced scheme for hypercube interconnection networks," in Proc. 1987 Int. Conf. Parallel Processing, pp. 820-823, Aug. 1987.

[53] Lan, Y. Esfahanian, A. and Ni, L. "Multicast in hypercube multiprocessors," J. Parallel Distributed Comput., pp. 30-41, Jan. 1990.

[54] Chen, F. W. and Ghozati, S. A., "Communication Algebra for Design and Analysis of Interconnection Networks," AMSE, Advances in Modelling and Analysis, B, vol 31, No. 1, pp.55-64, 1994.

[55] Ghozati, S. A. and Chen, F. W., "A Graph Approach to Cube Identification," International Journal of Computer Science and Electrical Engineering (Submitted).