Working Paper

Some Theoretical Results of Hypercube for Parallel Architecture

Motoyasu Nagata

WP-92-18
February 1992

IIASA International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Some Theoretical Results of Hypercube for Parallel Architecture

Motoyasu Nagata

WP-92-18
February 1992

International Institute for Applied Systems Analysis □ A-2361 Laxenburg □ Austria
Telephone: +43 2236 715210 □ Telex: 079 137 iiasa a □ Telefax: +43 2236 71313

Foreword

This paper surveys some theoretical results of the hypercube for design of VLSI architecture. The parallel computer including the hypercube multiprocessor will become a leading technology that supports efficient computation for large uncertain systems.

Some Theoretical Results of Hypercube for Parallel Architecture

Motoyasu Nagata

1 Introduction

The parallel computer is one of the remarkable developments of methodology and technology in computer science in recent years. Due to multiprocessor structure of the computer architecture, this computer has a capability to execute multiple instructions or multiple data simultaneously. The parallel computer not only provides support for efficient computation of mathematical, economical, industrial, and ecological problems but also aims new computer architecture beyond the traditional von Neumann type. Parallel processing will become a leading technology in the 1990s.

The technologies of the parallel computer mainly depends on the development of the VLSI. The parallel computer with hypercube interconnection is regarded as the most promising technology due to its topological structure and its surprising computing power [1] [4] [12] [26], and [31]. Examples of VLSI chips for the hypercube are iPSC/1, iPSC/2, and NCube/+4. On the other hand, methodologies of the hypercube have been developed based on graph theory, data structure and combinatorial optimization, taking into consideration computer architecture [5], and [12]. There are many theoretical problems in the design of the hypercube computer, for example, topological structure, embedding into hypercube, communication synchronization, and fault tolerance.

The purposes of the hypercube computer are parallel processing, parallel communication, and synchronization in all processors which are interconnected by hypercube topology. Fundamental hardware configuration of the hypercube processor, which is sometimes called the node, consists of the CPU, local memory, buffer, and interfaces [20]. Every node is connected to adjacent nodes via interfaces. Data structure for task or data is decomposed into pieces and allocated to the processor of the hypercube. Processor allocation is desirable if adjacent data of the data structure are mapped on adjacent nodes of the hypercube. Next, parallel internode communication is necessary for decomposing the unit

of the task or for transmitting data to another node. After data have been transmitted to all allocated nodes, parallel processing is carried out. Synchronization is necessary for parallel processing and communication. Processing results are also gathered by parallel internode communication. This paper surveys some theoretical results in VLSI architecture for hypercube.

2 Topological Structure of Hypercube

This section treats topological structure of the hypercube from viewpoints of addressing and graph.

Definition 2.1 [24] The n-dimensional hypercube Q_n has 2^n nodes. Addresses of these nodes are from 0 to $2^n - 1$. Any two nodes are adjacent if and only if two binary addresses differ by one and only one bit.

Definition 2.2 Let $a_n a_{n-1}.....a_1$, $b_n b_{n-1}.....b_1$ be binary addresses of two nodes a and b of n-dimensional hypercube Q_n. Nodes a and b are adjacent if the Hamming distance between two binary addresses $H(a,b) = \sum_{i=1}^{n} |a_i - b_i|$ is one.

Definition 2.3 [24] Addresses of the n-dimensional hypercube Q_n are recursively constructed as follows:

(1) Addresses of two nodes of one-dimensional hypercube Q_1 are 0 and 1.

(2) Let $a_{n-1}.....a_1$ be binary address of any node of $(n-1)$-dimensional hypercube Q_{n-1}. For two Q_{n-1}s, concatenate 0 and 1 to the leftmost bit positions of two nodes with the same address $a_{n-1}.....a_1$, and connect these two nodes.

Remark 2.4 The two nodes addressed by $0a_{n-1}.....a_1$, $1a_{n-1}.....a_1$ of Q_n in Definition 2.3 are adjacent.

The hypercube is recursively defined using a graph.

Definition 2.5 [8] Let $G = (V, E)$ be a graph where V is a set of nodes and E is a set of edges. Let $G_p = (V_p, E_p)$ be a product of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, denoted by $G_1 \times G_2$, where set of nodes $V_p = V_1 \times V_2$. Two nodes $u = (u_1, u_2), and v = (v_1, v_2)$ are adjacent in G_p if and only if $u_1 = v_1 (u_2 = v_2)$ and $u_2 (u_1)$ is adjacent to $v_2 (v_1)$.

Definition 2.6 The graph is called a complete graph if there exists only one edge that connects any two nodes.

Definition 2.7 [8] The graph of n-dimensional hypercube Q_n is recursively constructed as follows:

(1) Q_0 is a trivial graph with one node.

(2) $Q_n = K_2 \times Q_{n-1}$, where K_2 is a complete graph which consists of two nodes.

Some lemmas about topological properties of the hypercube have been proposed from these definitions.

Lemma 2.8 [24] There are n different ways for decomposition of Q_n into two Q_{n-1}s.

Proof By Definition 2.3, there are n concatenations of 0 (or 1) to address Q_{n-1}, that is, $0a_{n-1}.....a_1$, $a_{n-1}0a_{n-2}.....a_1$, $a_{n-1}.....a_10$.□

Lemma 2.9 [24] There are $n!2^n$ different ways of addressing Q_n.

Proof The result is trivial for $n = 1$. Assume that Lemma 2.9 holds for $n - 1$. Since there are $(n-1)!2^{n-1}$ ways of addressing Q_{n-1} and there are n concatenations of 0 (or 1) to any address of the first (or second) Q_{n-1}, the number of ways of addressing Q_n is $n \times (n-1)!2^{n-1} + n \times (n-1)!2^{n-1} = n!2^n$.□

Lemma 2.10 [24] Let A and B be any adjacent node of Q_n. Then the nodes adjacent to A and the nodes adjacent to B are conneced in one-to-one fashion.

Proof Let addresses of A and B be $0a_{n-1}.....a_1$, $1a_{n-1}.....a_1$ without loss of generality. Since the address of any node that is adjacent to A (or B) is obtained by reversing one and only one bit of the address of A (or B), $0a_{n-1}...\overline{a}_i...a_1$ and $1a_{n-1}...\overline{a}_i...a_1$ are adjacent for $1 \leq i \leq n$. □

Lemma 2.11 [24] There are no cycles of odd length in Q_n.

Proof Consider a cycle A_1, A_2, ..., A_m of Q_n, where $A_1 = A_m$. The length of the cycle, $m - 1$, is the sum of bit reversing and its reversing again for some a_i $(1 \leq i \leq n)$.□

Definition 2.12 The graph is called a connected graph if there exists a path that connects any two nodes of the graph. Maximum distance between two nodes of the graph is called the diameter.

Lemma 2.13 [24] Q_n is a connected graph of diameter n.

Proof Consider a node A with address 0.....0 of Q_n and a node B whose address has k 1s and $(n-k)$ 0s. By Definition 2.3 there exists a path with k edges that connects A and B, then Q_n is a connected graph and $k \leq n$.□

Lemma 2.14 [29] Q_n has $n2^{n-1}$ edges that connect 2^n nodes.

Proof Since any node has n adjacent nodes, the number of edges is $n2^n \div 2 = n2^{n-1}$.□

Definition 2.15 The number of edges connected to the node of the graph is called degree of the node.

Theorem 2.16 [24] A graph $G = (V, E)$ is Q_n if and only if

(1) V has 2^n vertices.

(2) Every vertex has degree n.

(3) G is connected.

(4) For any two adjacent nodes A and B, the nodes adjacent to A and the nodes adjacent to B are connected in one-to-one fashion.

Lemma 2.17 [24] The minimum distance between any two nodes A and B of Q_n is the Hamming distance $H(A, B)$.

Proof Assume, without loss of generality, that two addresses of nodes A and B differ in k leading bits. Then one path from A to B is the following:

$$a_n a_{n-1} a_{n-2} \ldots\ldots a_{n-k+1} a_{n-k} \ldots\ldots a_1 = A$$

$$b_n a_{n-1} a_{n-2} \ldots\ldots a_{n-k+1} a_{n-k} \ldots\ldots a_1$$

$$b_n b_{n-1} a_{n-2} \ldots\ldots\ldots a_{n-k+1} a_{n-k} \ldots\ldots\ldots a_1$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$b_n b_{n-1} b_{n-2} \ldots\ldots b_{n-k+1} a_{n-k} \ldots\ldots a_1 = B.$$

Then the result holds.□

Lemma 2.18 [24] Let A and B be any two nodes of Q_n and assume that $H(A, B) < n$. Then there are $H(A, B)$ parallel paths of length $H(A, B)$ between the nodes A and B.

Proof Assume that the addresses of nodes A and B are the same as those in the proof of Lemma 2.17. We construct parallel paths from A to B with length $H(A, B)$. The i-th path can be constructed as follows:

$$a_n \ldots\ldots\ldots a_{n-i+2} a_{n-i+1} a_{n-i} \ldots\ldots\ldots a_{n-k+1} a_{n-k} \ldots\ldots\ldots a_1$$

$$a_n \ldots\ldots\ldots a_{n-i+2} b_{n-i+1} a_{n-i} \ldots\ldots\ldots a_{n-k+1} a_{n-k} \ldots\ldots\ldots a_1$$

$$a_n \ldots\ldots\ldots a_{n-i+2} b_{n-i+1} b_{n-i} \ldots\ldots\ldots a_{n-k+1} a_{n-k} \ldots\ldots\ldots a_1$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$b_n a_{n-1} a_{n-2} \ldots a_{n-i+2} b_{n-i+1} b_{n-i} \ldots b_{n-k+1} a_{n-k} \ldots a_1$$

$$b_n b_{n-1} a_{n-2} \ldots a_{n-i+2} b_{n-i+1} b_{n-i} \ldots b_{n-k+1} a_{n-k} \ldots a_1$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$b_n b_{n-1} \ldots\ldots\ldots\ldots\ldots b_{n-k+2} b_{n-k+1} a_{n-k} \ldots\ldots\ldots\ldots\ldots a_1$$

Then there exists $H(A,B)$ parallel paths.□

Lemma 2.19 [24] Let A and B be any two nodes of Q_n and assume that $H(A,B) < n$. Then there are n parallel paths between the nodes A and B. Moreover, the length of each path is at most $H(A,B)+2$.

Proof Assume that addresses of nodes A and B are the same as those in the proof of Lemma 2.17. We have already proved that there exist k parallel paths with length $H(A,B)$ in Lemma 2.18. We prove the existence of $(n-k)$ parallel paths with length $H(A,B)+2$ besides k parallel paths with length $H(A,B)$. At first reverse a_i of address of node A, next take a procedure of Lemma 2.17, and finally reverse a_i again. We apply these procedures for a_i, $2 \leq i \leq n-k$. Then the result holds.□

Definition 2.20 [9] [10] The subcube with m-dimension ($m < n$) is a subset of Q_n that satisfies four properties of Theorem 2.16. The subcube is addressed using notations $0, 1, *$, where $*$ is don't care symbol that stands for 0 or 1. The dimension of the subcube is represented by number of $*$s.

Definition 2.21 [9] The distance between two subcubes a and b is represented by Hamming distance:

$$H(a,b) = \sum |a_i - b_i|,$$

where $|a_i - b_i| = 1$ if $(a_i, b_i) = (1,0)$ or $(0,1)$.

Example 2.22 Consider a subcube of Q_3 with nodes $000, 001, 011, 010$. This subcube is with address $0**$ and has two dimensions. The distance between two subcubes $00**$ and $011*$ is one due to $H(00**, 011*) = 1$.

Lemma 2.23 [9] The m-dimensional subcube q of Q_n is adjacent to, at most, $(n-m)2^m$ subcubes in Q_n.

Proof Assume that the address of a subcube q is $00.....0**.....*$ that is represented by $(n-m)$ 0s and m $*$s. If the addresses of subcubes Q_ms adjacent to a subcube q are

$$10.....0**.....*$$

$$01.....0**.....*$$

$$.....$$

$$00.....1**.....*,$$

then there exist $(n-m)2^m$ Q_0s that are adjacent to subcube q.□

3 Gray Code

The Gray code consists of binary codes such that adjacent codes differ by one bit. The Gray code plays an important role when the data structure is allocated to processors of the hypercube.

Definition 3.1 [12] The n-dimensional binary-reflected Gray code (BRGC) G_n is recursively constructed as follows:

$$G_1 = (0, 1)$$

$$G_n = (0G_{n-1}, 1\overline{G}_{n-1}),$$

where $0G_{n-1}$ is a concatenation of 0 and G_{n-1} and $\overline{G}_{n-1}$ is a backward-sorted code of G_{n-1}.

Example 3.2 Two and three-dimensional BRGCs are shown

$$G_2 = (0G_1, 1\overline{G}_1) = (00, 01, 11, 10)$$

$$G_3 = (0G_2, 1\overline{G}_2) = (000, 001, 011, 010, 110, 111, 101, 100).$$

Remark 3.3 We can intuitively understand that there exists a mirror between G_{n-1} and $\overline{G_{n-1}}$.

The relationship between the BRGC and its ordering is formulated by the following famous theorem discovered by Gray.

Theorem 3.4 [12] [15] Let binary representation of the n-dimensional BRGC G_n and its ordering be $g_{n-1}.....g_0$ and $b_{n-1}.....b_1$, respectively. Then the relationship between the BRGC and its ordering is formulated by the following equations:

$$g_i = b_i + b_{i+1}, \ \mathrm{mod}2 \ , i < n-1$$

$$g_{n-1} = b_{n-1}$$

The equivalent inverse relation also holds.

$$b_i = g_i + b_{i+1}, \ \mathrm{mod}2 \ , i < n-1$$

$$b_{n-1} = g_{n-1}$$

.

Furthermore, b_i is represented by g_j, where $i \leq j \leq n-1$

$$b_i = g_i + ... + g_{n-1}, \text{ mod2}$$

.

4 Embedding Data Structure into Hypercube

Definition 4.1 [16] Let $d(i,j)$ be the distance between two nodes i and j. Let $d(G_i, G_j)$ be the distance between two addresses G_i and G_j that are encoded using the BRGC. Then $maxd(G_i, G_{i+1})$ is called dilation, where G_i and G_{i+1} are adjacent BRGCs.

Definition 4.2 Consider an embedding of a graph $G = (V, E)$ into another graph $G' = (V', E')$. Then $n(V')/n(V)$ is called expansion, where $n(V)$ is the number of elements of node set V.

4.1 Loop

Lemma 4.3 [16] The loop of length $|L| = 2^{n-1} + 2k$, $k = \{1, 2, ..., 2^{n-2}\}$ can be embedded into an n-dimensional hypercube Q_n with dilation 1 by BRGC.

Proof Assume that nodes $\{0, 1, ..., 2^{n-1} + k - 1\}$ can be embedded using the BRGC of the loop node index. And assume that loop nodes $\{2^{n-1}+k, 2^{n-1}+k+1, ..., 2^{n-1}+2k-1\}$ can be embedded corresponding to the BRGCs of $\{2^n - k, 2^n - k + 1, ..., 2^n - 1\}$. Then the Gray code of $2^{n-1}+k-1$ is $(1G_{n-1}(2^{n-1}-k))$ and Gray code of $2^n - k$ is $(1G_{n-1}(k-1))$, where $G_{n-1}(2^{n-1} - k)$ is the $(2^{n-1} - k)$-th code of $(n-1)$-dimensional BRGC. If

$$G_{n-1}(2^{n-1} - k) = (1G_{n-2}(k-1))$$

$$G_{n-1}(k-1) = (0G_{n-2}(k-1)),$$

then

$$1G_{n-1}(2^{n-1} - k) = (11G_{n-2}(k-1))$$

$$1G_{n-1}(k-1) = (10G_{n-2}(k-1)).$$

Thus both codes are adjacent.□

Lemma 4.4 [16] The loop of length $|L| = 2^{n-1} + 2k + 1$, $k = \{1, 2, ..., 2^{n-2} - 1\}$ can be embedded into an n-dimensional hypercube Q_n with at least one edge of length 2.

Lemma 4.5 [16] The BRGCs that encode i and $i + 2^k \bmod 2^n$ differ in 2 bits.

Proof For two integers i and j whose binary representations are

$$i_n i_{n-1} .....i_1$$

$$j_n j_{n-1} .....j_1,$$

two BRGCs that correspond to i and j are assumed as follows:

$$g_n g_{n-1} .....g_1$$

$$h_n h_{n-1} .....h_1.$$

Then

$$i_m = j_m, m = \{1, 2, ..., k\}$$

$$j_m = \bar{i}_m, m = \{k+1, ..., s\},$$

where carry stops propagation at bit position s. From Gray's theorem, the result is obtained

$$h_m = g_m, m = \{1, 2, ..., k-1, k+1, ..., s-1\}$$

$$h_k = \overline{g}_k, h_s = \overline{g}_s,$$

where the overline stands for the complement of the bit.□

4.2 Mesh

Johnsson presented an example for the naive embedding of a multidimensional array. We generalize his example by the following algorithm that guarantees adjacencies with wraparound.

Algorithm 4.6: Embedding of a $2^n \times 2^n$ mesh in Q_{2n} by BRGC

Suppose the matrix A^{2^i} stands for $2^i \times 2^i$ mesh whose nodes are addressed using a binary-reflected Gray code (BRGC).

Step 1 For embedding of a 2×2 mesh, allocate two-dimensional BRGC $G_2(i)$ $(i = 0, \cdots, 3)$, 00, 01, 11, 10 to four nodes of this mesh in a clockwise manner, that is,

$$A^{2^1} = \begin{pmatrix} 00 & 01 \\ 10 & 11 \end{pmatrix}.$$

Step 2 Embedding of a $2^{i+1} \times 2^{i+1}$ mesh can be done using the following recursive formula

$$A^{2^{i+1}} = \begin{pmatrix} 00A^{2^i} & 01(A^{2^i}_{\pi/2})^T \\ 10(A^{2^i}_{3\pi/2})^T & 11A^{2^i}_{\pi} \end{pmatrix}.$$

where $A^{2^i}_{\pi/2}$ indicates rotation of A^{2^i} by $\pi/2$ radian and $(A^{2^i}_{\pi/2})^T$ means transpose of $A^{2^i}_{\pi/2}$. Furthermore, $00A^{2^i}$ stands for concatenation of 00 to the leftmost bits of all nodes of A^{2^i}.

Example 4.7

$$A^{2^2} = \begin{pmatrix} 0000 & 0001 & 0101 & 0100 \\ 0010 & 0011 & 0111 & 0110 \\ 1010 & 1011 & 1111 & 1110 \\ 1000 & 1001 & 1101 & 1100 \end{pmatrix}.$$

The adjacency with wraparound is guaranteed, for example, $(0000, 0100)$, $(0000, 1000)$. Furthermore, the adjacency in an ordinary sense is also guaranteed, for example, $(0011, 0111)$, $(0011, 1011)$. We also show A^{2^3}..

$$A^{2^3} = \begin{pmatrix} 000000 & 000001 & 000101 & 000100 & 010100 & 010101 & 010001 & 010000 \\ 000010 & 000011 & 000111 & 000110 & 010110 & 010111 & 010011 & 010010 \\ 001010 & 001011 & 001111 & 001110 & 011110 & 011111 & 011011 & 011010 \\ 001000 & 001001 & 001101 & 001100 & 011100 & 011101 & 011001 & 011000 \\ 101000 & 101001 & 101101 & 101100 & 111100 & 111101 & 111001 & 111000 \\ 101010 & 101011 & 101111 & 101110 & 111110 & 111111 & 111011 & 111010 \\ 100010 & 100011 & 100111 & 100110 & 110110 & 110111 & 110011 & 110010 \\ 100000 & 100001 & 100101 & 100100 & 110100 & 110101 & 110001 & 110000 \end{pmatrix}.$$

Remark 4.8 We can intuitively interpret the algorithm about embedding of the multidimensional array from the viewpoint of the mirror. Images of A^{2^i} reflected in two mirrors are $(A^{2^i}_{\pi/2})^T$ and $(A^{2^i}_{3\pi/2})^T$, respectively. Furthermore, images of $(A^{2^i}_{\pi/2})^T$ and $(A^{2^i}_{3\pi/2})^T$ reflected in mirrors result in $A^{2^i}_{\pi}$. Thus, the two-dimensional Gray code can be generated using Algorithm 4.6.

Lemma 4.9 [16] The naive embedding of the multidimensional array into the hypercube can be done such that an $N_1 \times N_2 \times \cdots\cdots \times N_r$ mesh can be embedded into an $N = ([\log_2 N_1] + [\log_2 N_2] + \cdots\cdots + [\log_2 N_r])$-dimensional hypercube Q_N by assigning

$[\log_2 N_i]$ cube dimensions to dimension i of the mesh. Here $[\cdot]$ stands for a notation such that $[x] = \sup x \in Z$, $x \in R$, where Z and R are integer and real spaces.

Lemma 4.10 [16] The naive embedding of the multidimensional arrays is efficient for $N_i = 2^{n_i}$, but for $2^{n_i} < N_i < 2^{n_i+1}$ there exists the expansion

$$e = \frac{2([\log_2 N_1] + \cdots\cdots + [\log_2 N_r])}{N_1 \times \cdots\cdots \times N_r}.$$

Modified mesh embedding methods are found in the literature [6], [7], [14], [19], and [27].

4.3 Graph

Definition 4.11 [9] Relaxed-squashed (RS) embedding is a node-to-subcube distance preserving mapping from source graph to the hypercube.

Definition 4.12 [9] The dimension of the minimal cube required for the RS embedding of a source graph $G = (V, E)$ is called the weak cubical dimension $wd(G)$ of the graph.

Definition 4.13 For graph $G = (V, E)$, an induced subgraph $ind_G(V_S)$ of G with a node set $V_S \subseteq V$ is the maximal subgraph with the node set V_S.

Lemma 4.14 [9] Let G be a graph with n nodes. Then the following inequality holds: $[\log_2 n] \leq wd(G) \leq n - 1$.

Proof Graph G is a subgraph of the complete graph K_n. If $wd(K_n) = n - 1$, then $wd(G) \leq n - 1$ holds. By Definition 4.12, $[\log_2 n] \leq wd(G)$ is obvious.□

Lemma 4.15 [9] Let $G = (V, E)$ be a connected graph and let $G_S = (V_S, E_S)$ be a subgraph of G. Suppose that the induced subgraph $ind_G(V_S)$ can be RS embedded into m-dimensional hypercube Q_m, and the removal of all edges in E_S from G results in $|V_S|$ disjoint graphs, $G_i = (V_i, E_i)$, $1 \leq i \leq |V_S|$. Then $wd(G) \leq \max_{1 \leq i \leq |V_S|} wd(G_i) + m$.

Proof Let v_i, $1 \leq i \leq |V_S| = k$ be the nodes in $V_S \cap V_i$ of $G_S \cap G_i$. The notation $address_{G_S}(v_i)$ represents the encoding of v_i in V_S so that G_S can be RS embedded into Q_m. The notation $address_{G_i}(v)$ represents the encoding of $v \in V_i$ for the RS embedding of G_i into $Q_{wd(G_i)}$, where $address^j_{G_i}(v)$ is the j-th bit of $address_{G_i}(v)$. The RS embedding generates $address_G(w)$ for each w in G by the following procedures:

Algorithm 4.16: RS Embedding of single graph

Step 1 For each $w \in V_i$, $1 \leq i \leq k = |V_S|$,

$address^j_G(w) \longleftarrow address^j_{G_S}(v_i)$, where $1 \leq j \leq m$.

Step 2 For each $w \in V_i$, $1 \leq i \leq k = |V_S|$,

if $address^{j}_{G_S \cap G_i}(v_i) = 1$

then $address^{j}_{G}(w) \longleftarrow \overline{address^{j-m}_{G_i}(v)}$

(overline indicates complement of binary code)

else $address^{j}_{G}(w) \longleftarrow address^{j-m}_{G_i}(v)$

where $m + 1 \leq j \leq wd(G_i) + m$.

Step 3 For each $w \in V_i$, $1 \leq i \leq k = |V_S|$,

$address^{j}_{G}(w) \longleftarrow *$ (don't care symbol),

where $wd(G_i) + m + 1 \leq j \leq \max_{1 \leq i \leq k} wd(G_i) + m$.

As shown in the Algorithm 4.16, the weak cubical dimension $wd(G)$ that is the dimension of the minimum cube required for the RS embedding of the graph $G = (V, E)$ is smaller than $\max_{1 \leq i \leq |V_S|} wd(G_i) + m$.□

4.4 Tree

4.4.1 Embedding with expansion 1

Theorem 4.17 [16] An embedding of a complete binary tree of n-height in a Q_n, by labeling the tree nodes in inorder and embedding the tree by a binary encoding of the node indices, yields an embedding in which a parent node and its left descendant are at distance 1, the parent and its right descendant are at distance 2, and the right and left descendants are at distance 1 from each other.

Theorem 4.18 [16] An embedding of a complete binary tree of n-height in a Q_n, by labeling the tree nodes in inorder and embedding the tree by a BRGC encoding of the node indices, yields an embedding in which a leaf node is at distance 1 from its parent node and all other nodes are at distance 2 from their respective parent node. Left and right descendants of a node are always at distance 2 from each other.

4.4.2 Embedding with dilation 1

Wu [30] presents the embeddability of binary tree of n-height into $n + 1$-dimendional hypercube using a definition of the free-free neighbor, which is outlined in Proof 1 of Theorem 4.20. Johnsson [16] presents alternative proof of the same theorem. We reformulate his proof.

Definition 4.19 [30] Let f_p be a map from binary tree of height p into hypercube Q_{p+1}. The free-free neighbor property is defined by if $R = f_p$(binary tree of height p) has a free

neighbor R_1 and R_1 has a free neighbor R_2, then $\{R_1, R_2\}$ is not a subset of f_p(nodes of binary tree of height p).

Theorem 4.20 [16] [30] The binary tree of height n can be embedded into an $(n+1)$-dimensional hypercube Q_{n+1} with dilation 1.

Proof 1 The following inductive algorithm proves the result.

Algorithm 4.21: Tree embedding [30]

Step 1 Left subtree of height $(n-1)$ is embedded by f_{n-1} into $0Q_n$ of Q_{n+1}. A node $0L = f_{n-1}$(root of left subtree) has a free neighbor $0L_1$, which has a free neighbor $0L_2$.

Step 2 Right subtree of height $(n-1)$ is also embedded by g_{n-1} into $1Q_n$ of Q_{n+1}. A node $1R = g_{n-1}$(root of right subtree) has a free neighbor $1R_1$, which has a free neighbor $1R_2$.

Step 3 A hypercube Q_{n+1} can be constructed by combining $0Q_n$ and $1Q_n$ in such a way that $0L_1$ is a neighbor of $1R$ and $0L_2$ is a neighbor of $1R_1$. The node $0L_1$ of Q_{n+1} corresponds to an embedded node from the root of the tree.□

Proof 2 The addressing for embedding complete binary tree of height n into an $(n+1)$-dimensional hypercube Q_{n+1} can be realized by the following procedure:

Algorithm 4.22 Addressing of tree embedding

Step 1 Allocate root node of the tree of height 1 to an address 00 in Q_2. For embedding of the tree of height 2 into Q_3, allocate left-leaf and right-leaf nodes to 010 (in subcube $01*$ in Q_3) and 100 (in subcube $10*$ in Q_3), respectively. The root address 00 in Q_2 can be readdressed into 000 in Q_3.

Step 2 For embedding the tree of $(n+1)$ height into $(n+2)$-dimensional hypercube Q_{n+2} using embedded addresses about the n-height tree into Q_{n+1}, if leaf-tree node of the $(n+1)$-height tree is left child of the leaf node of the n-height tree, allocate these left children to nodes in a subcube $0Q_{n+1}$ of Q_{n+2} such that nodes in a subcube 01 ($0*\cdots*1*\cdots*$ in Q_{n+1}) are mapped on a subcube 11 (in Q_{n+1}), nodes in a subcube 11 (in Q_{n+1}) are mapped on subcube 10 (in Q_{n+1}), and nodes in a subcube 10 (in Q_{n+1}) are mapped on the same subcube 10 (in Q_{n+1}) without conflicting with previously allocated addresses.

Step 3 If the leaf-tree node of the $(n+1)$-height tree is right child of the leaf node of the n-height tree, allocate these right children to nodes in a subcube $1Q_{n+1}$ of Q_{n+2} such that nodes in a subcube 01 (in Q_{n+1}) are mapped on 01 (in Q_{n+1}), nodes in 11 on 11, and nodes in 10 on 10.

Step 4 Concatenate 0 to the leftmost bit of addresses of all nodes except for embedded

leaf nodes in order to embed these nodes into a subcube $0Q_{n+1}$ of Q_{n+2}.

Lemma 4.23 Assume that L_n represent leaf nodes in the n-height tree. By Algorithm 4.22, leaf nodes $(L_n/2, L_n/4, L_n/4)$ can be respectively allocated to subcubes $(01, 11, 10)$ in Q_{n+1} for every natural number n.

Proof We prove by induction. Using Step 1 of Algorithm 4.21, $(L_n/2, L_n/4, L_n/4)$ are mapped from subcubes $(01, 11, 10)$ of Q_{n+1} on subcubes $(11, 10, 10)$ of Q_{n+1} in $0Q_{n+1}$. Similarly using Step 2 of Algorithm 4.21, $(L_n/2, L_n/4, L_n/4)$ are mapped from subcubes $(01, 11, 10)$ of Q_{n+1} on subcubes $(01, 11, 10)$ of Q_{n+1} in $1Q_{n+1}$. Since, in $0Q_{n+1}$, allocated leaf nodes are $(L_n/2, L_n/4, L_n/4)$ in subcubes $(011, 010, 010)$ of $0Q_{n+1}$, and $(L_n/2, L_n/4, L_n/4)$ in subcubes $(101, 111, 110)$ of $1Q_{n+1}$, then allocated leaf nodes are $(L_{n+1}/2, L_{n+1}/4, L_{n+1}/4) = (L_n/2+L_n/4+L_n/4, L_n/4+L_n/4, L_n/2)$ in subcubes $(01*, 11*, 10*)$, that is, $(01, 11, 10)$, of Q_{n+2}.□

Lemma 4.24 Algorithm 4.22 enables addressing without conflict with previously addressed nodes.

Proof Suppose that ♣s represent 2^n leaf nodes of $(n+1)$-height tree, ◇s represent 2^{n-1} leaf nodes of n-height tree, ♡s represent 2^{n-2} leaf nodes of $(n-1)$-height tree, and ♠s represent 2^{n-3} leaf nodes of $(n-2)$-height tree. At Step 2 of Algorithm 4.22, consider four subcubes 00, 01, 11, 10 in $0Q_{n+1}$. Assume that n-height tree is embedded into Q_{n+1} without conflict. For mapping from 01 to 11 in $0Q_{n+1}$, since there is no conflict between ◇s and ♡s in 01, then ♣s in 11 (mapped from ◇s in 01) and ◇s in 11 (from ♡ in 01) do not conflict. Next consider mapping from 11 to 10 in $0Q_{n+1}$. Locations of ◇s in 11 are equivalent to locations of ♡s in 01, and equivalent to locations of ♠s in 00. Since ♡s and ♠s do not conflict in 00, then ◇s in 10 (from ♡s in 00) and ♣s in 10 (from ◇s in 11) do not conflict. Next consider mapping from 10 to 10 in $0Q_{n+1}$. Locations of previously allocated ◇s in 10 and previously allocated ♣s in 10 are locations of ♡s and ♠s in 00, respectively. Since in 00 mapping is possible from ♡s to other nodes without conflict with ♠s (see Lemma 4.25), then in 10 mapping is possible from ◇s to other nodes without conflict with previously allocated ♣s. For mapping from 01 in $0Q_{n+1}$ to 01 in $1Q_{n+1}$, mapping from ◇s to ♣s is obviously without conflict. Mappings from 11 (or 10) in $0Q_{n+1}$ to 11 (or 10) in $1Q_{n+1}$ is similarly without conflict.□

Lemma 4.25 In a subcube 00 of Lemma 4.24, mapping is possible from ♡s to other nodes without conflict with ♠s.

Proof Divide a subcube 00 into four smaller subcubes 00, 01, 11, 10. By mapping rules

$10 \rightarrow 00$, $11 \rightarrow 10$ or 11, mapping is possible from ♡s to other nodes without conflict with ♠s.

From Lemma 4.23 and Lemma 4.24, Proof 2 of Theorem 4.20 is completed.□

5 Concluding Remarks

This paper surveys topological structure of the hypercube and embedding of several data structures into the hypercube. Topics in complexity of tree embedding [28] [29], odd-even shift [13] [23], communication [17], synchronization [21] [25], relevant interconnection [1] [31], fault tolerance [11], and applications [2] [3] [18] of the hypercube can be found in the respective literature.

References

[1] F. Annexstein, M. Baumslag, and A. L. Rosenberg, Group Action Graphs and Parallel Architectures, *SIAM J. Computing*, **19**, (3), 544-569, 1990.

[2] C. K. Baru and O. Frieder, Database Operations in a Cube-Connected Multiprocessor System, *IEEE Transaction on Computers*, **C-38**, (6), 920-927, 1989.

[3] C. K. Baru and P. Goel, Squashed Embedding of E-R Schemas in Hypercubes, *Journal of Parallel and Distributed Computing* **8**, 340-348, 1990.

[4] J. C. Bermond and C. Peyrat, **De Bruijn and Kautz Networks: A Competer for the Hypercube and Distributed Computers**, Edited by A. Andre and J. P. Verjus, Elsevier Science Publishers B. V. (North Holland), 1989.

[5] L. N. Bhuyan and D. P. Agrawal, Generalized Hypercube and Hyperbus Structures for a Computer Network, *IEEE Transactions on Computers*, **C-33**, **4**, 323-333, 1984.

[6] M. Y. Chan and F. Y. L. Chin, On Embedding Rectangular Grids in Hypercubes, *IEEE Transactions on Computers*, **37**, (10), 1285-1288, 1988.

[7] T. F. Chan and Y. Saad, Multigrid Algorithms on the Hypercube Multiprocessor, *IEEE Transaction on Computers*, **C-35**, (11), 969-977, 1986.

[8] M. S. Chen and K. Shin, Processor Allocation in an N-Cube Multiprocessor Using Gray Codes, *IEEE Transaction on Computers*, **C-36**, (12), 1396-1407, 1987.

[9] M. S. Chen and K. Shin, On Relaxed Squashed Embedding of Graphs into a Hypercube, *SIAM J. Computing*, **8**, (6), 1226-1244, 1989.

[10] M. S. Chen and K. Shin, Subcube Allocation and Task Migration in Hypercube Multiprocessors, *IEEE Transaction on Computers*, **39**, (9), 1146-1155, 1990.

[11] M. S. Chen and K. Shin, Depth-First Search Approach for Fault Tolerant Routing in Hypercube Multicomputers, *IEEE Transaction on Parallel and Distributed Systems*, **1**, (2), 152-159, 1990.

[12] W. J. Dally, **A VLSI Architecture for Concurrent Data Structures**, Kluwer Academic Publishers, 1987.

[13] Z. Fang, X. Li, and L. M. Ni, Parallel Algorithms for Image Template Matching on Hypercube SIMD Computers, *Proceedings IEEE Computer Society Workshop*

on Computer Architecture for Pattern Analysis and Image Database Management, 1985.

[14] C. T. Ho and S. L. Johnsson, Embedding Meshes in Boolean Cubes by Graph Decomposition, *Journal of Parallel and Distributed Computing*, **8**, 325-339, 1990.

[15] H. J. Gray and P. V. Lavonian, An Analog-to-Digital Converter for Serial Computing Machines, *Proceedings of the I.R.E.*, **41**, (10), 1462-1465, October, 1953.

[16] S. L. Johnsson, Communication Efficient Basic Linear Algebra Computations on Hypercube Architectures, *Journal of Parallel and Distributed Computing*, **4**, 133-172, 1987.

[17] S. L. Johnsson and C. T. Ho, Optimum Broadcasting and Personalized Communication in Hypercubes, *IEEE Transaction on Computers*, **C-38**, (9), 1249-1267, 1989.

[18] A. J. Laub, Hypercube Implementation of Some Parallel Algorithms in Control, *NATO ASI Series*, Vol. F47: **Advanced Computing Concepts and Techniques in Control Engineering**, Edited by M. J. Denham and A. J. Laub, Springer-Verlag Berlin, Heiderberg, 1988.

[19] S. Matic, Emulation of Hypercube Architecture on Nearest Neighbor Mesh-Connected Processing Elements, *IEEE Transaction on Computers*, **39**, (5), 698-700, 1990.

[20] M. Nagata, S. Fukuda, and K. Kihara, Design and Implementation of a Multiprocessor with a Hypercube Interconnection Network, *Proceedings of IECON'89*, 1989.

[21] D. Peleg and J. Ullman, An Optimal Synchronizer for the Hypercube, *SIAM J. Computing*, **15**, (4), 740-747, 1989.

[22] F. P. Preparata and J. Vuillemin, The Cube-Connected Cycles: A Versatile Network for Parallel Computation, *Communication of the ACM*, **24**, (5), 300-309, 1981.

[23] S. Ranka and S. Sahni, Odd Even Shifts in SIMD Hypercubes, **1**, (1), 1990.

[24] Y. Saad and M. H. Schultz, Topological Properties of Hypercubes, *IEEE Transaction on Computers*, **37**, (7), 867-872, 1988.

[25] Y. Saad and M. H. Schultz, Data Communication in Hypercubes, *Journal of Parallel and Distributed Computing*, **6**, 115-135, 1989.

[26] M. R. Samatham and D. K. Pradhan, The De Bruijn Multiprocessor Network: A Versatile Parallel Processing and Sorting Network for VLSI, *IEEE Transaction on Computers*, **38**, (4), 567-581, 1989.

[27] D. S. Scott and J. Brandenburg, Minimal Mesh Embeddings in Binary Hypercubes, *IEEE Transaction on Computers*, **37**, (10), 1284-1285, 1988.

[28] A. Wagner, Embedding Arbitrary Binary Trees in a Hypercube, *Journal of Parallel and Distributed Computing*, **7**, 503-520, 1989.

[29] A. Wagner and D. G. Corneil, Embedding Trees in a Hypercube is NP-Complete, *SIAM J. Computing*, **19**, (4), 570-590, 1990.

[30] A. Y. Wu, Embedding of Tree Networks into Hypercubes, *Journal of Parallel and Distributed Computing*, **2**, 238-249, 1985.

[31] A. S. Youssef and B. Narahari, The Banyan-Hypercube Networks, *IEEE Transaction on Parallel and Distributed Systems*, **1**, (2), 160-169, 1990.