[6] M.-B. Lin and A. Yavuz Oruç, "The design of a network-based arithmetic processor," Tech. Rep. UMIACS-TR-91-141, Univ. of Maryland, College Park, MD, Oct. 1991.
[7] M.-B. Lin, "Unified algebraic computations on permutation networks," Ph.D. dissertation, EE Dept., Univ. of Maryland, College Park, 1992.
[8] M. Maresca and H. Li, "Connection autonomy in SIMD computers: A VLSI implementation," *J. Parallel Distrib. Computing*, vol. 7, pp. 302–320, 1989.
[9] R. Miller, V. K. Prasanna Kumar, D. Reisis, and Q. F. Stout, "Data movement operations and applications on reconfigurable VLSI arrays," in *Int. Conf. Parallel Processing*, St. Charles, IL, vol. I, Aug. 1988, pp. 205–208.
[10] A. Yavuz Oruç, V. G. J. Peris, and M. Yaman Oruç, "Parallel modular arithmetic on a permutation network," in *Int. Conf. Parallel Processing*, St. Charles, IL, vol. 1, Aug. 1991, pp. 706–707.
[11] S. J. Piestrak, "Design of residue generators and multi-operand modular adders using carry-save adders," in *IEEE 10th Comput. Arith. Symp.*, 1991, pp. 100–107.
[12] W. Shen and A. Yavuz Oruç, "Mapping algebraic formulas onto mesh connected processor networks," *Inform. Sci. Syst. Conf.*, Princeton Univ., NJ, pp. 535–538, 1986.
[13] S. P. Smith and H. C. Torng, "Design of a fast inner product processor," in *Proc. IEEE 7th Comput. Arith. Symp.*, 1985, pp. 38–43.
[14] E. E. Swartzlander, Jr., B. K. Gilbert and I. S. Reed, "Inner product computers," *IEEE Trans. Comput.*, vol. C-27, pp. 21–31, Jan. 1978.
[15] B. F. Wang, G. H. Chen, and F. C. Lin, "Constant time sorting on a processing array with a reconfigurable bus system," *Inform. Processing Lett.*, pp. 187–192, 1990.
[16] B. F. Wang and G. H. Chen, "Constant time algorithms for the transitive closure and some related graph problems on processor arrays with reconfigurable bus systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 1, pp. 500–507, Oct. 1990.

Structural and Tree Embedding Aspects of Incomplete Hypercubes

Nian-Feng Tzeng and Hsing-Lung Chen

***Abstract*—Since the hypercube is not incrementally scalable, a variant hypercube topology with more flexibility in the system size, called an *incomplete hypercube*, is examined. An incomplete hypercube may also result from a complete hypercube which operates in a degraded manner after some nodes fail. Elementary properties, including diameter, mean internode distance, and traffic density, of incomplete hypercubes with size $2^n + 2^k$, $0 \leq k \leq n$, are derived. Interestingly, traffic density over links in such an incomplete hypercube is found to be bounded by 2 (messages per link per unit time), despite its structural nonhomogeneity. Thus, cube links can easily be constructed so as to avoid any single point of congestion, guaranteeing good performance. The minimum incomplete hypercubes able to embed binary trees with node adjacencies preserved are determined.**

***Index Terms*— Hypercubes, incomplete hypercubes, message routing, network topology, structural properties, tree embeddings.**

Manuscript received August 26, 1992; revised May 13, 1993 and December 16, 1993. this work was supported in part by the NSF under Grants MIP-9201308 and CCR-9300075 and by the State of Louisiana under Contract LEQSF(92-94)-RD-A-32.

N.-F. Tzeng is with the Center for Advanced Computer Studies, University of Southwestern Louisiana, P.O. Box 70504–4433, Lafayette, LA 70504–4330 USA; e-mail: tzeng@cacs.usl.edu.

H.-L. Chen is with the Department of Electronic Engineering, National Taiwan Institute of Technology, Taipei, Taiwan, R.O.C.

IEEE Log Number 9404364.

I. Introduction

Unlike a complete hypercube, an *incomplete hypercube* allows for the construction of a system with size not necessary a power of 2. It may also result from a complete hypercube after some nodes become faulty and the system is reconfigured, as discussed in [7]. Simple and deadlock-free algorithms for routing and for broadcasting messages in the incomplete hypercube have been developed in [5]. In this brief contribution, we deal with the incomplete hypercube comprising two complete hypercubes, one of size 2^n and the other of size 2^k $(0 \leq k \leq n)$. A related structure introduced recently is the Fibonacci cube, which consists of two smaller Fibonacci cubes of unequal sizes and is a subgraph of a hypercube [2].

Here we are interested in finding whether or not an incomplete hypercube exhibits any heavy traffic link or node that may become a vulnerable point with respect to performance and reliability. Structural properties of incomplete hypercubes, including diameter, mean internode distance, and traffic density, are obtained. The highest traffic density over links in incomplete hypercubes is bounded by 2, despite its nonhomogeneity. With bounded traffic density, incomplete hypercubes are clearly superior to other nonhomogeneous topologies, such as trees and stars, where points of congestion are likely to exist and serious performance degradation may result because in such a topology, the highest traffic density is proportional to its size. If one wants to put a complete hypercube into operation even after some nodes become faulty and the system is reconfigured into an incomplete hypercube, cube links can easily be so designed that every link is still below half saturated in the presence of faults to prevent any traffic bottleneck from occurring.

In the second part of this work, embedding binary trees in the incomplete hypercube is pursued and compared to that in its complete counterpart. It is illustrated that the incomplete hypercube is capable of better supporting binary trees than a compatible complete hypercube. The embedding results also reveal that a complete hypercube still can effectively support binary trees even after cube links/nodes fail, provided that the operating portion of the injured hypercube is no smaller than the respective incomplete hypercubes able to embed those binary trees.

II. Notations and Background

An n-dimensional complete hypercube, denoted by H_n, comprises 2^n nodes, each with n bidirectional links connecting to n immediate neighbors. An incomplete hypercube of interest IH_k^n comprises two complete cubes, H_n and H_k, which respectively have 2^n and 2^k nodes, where $0 \leq k \leq n$. Nodes in IH_k^n are labeled from 0 to $2^n + 2^k - 1$ by an $(n+1)$-bit binary representation in such a way that any two nodes connected by a link differ in their labels by exactly one bit, and that nodes in H_n are numbered from 0 to $2^n - 1$, while nodes in H_k are from 2^n to $2^n + 2^k - 1$. Fig. 1 shows an incomplete hypercube with 12 nodes, IH_2^3. A d-dimensional subcube in IH_k^n contains 2^d nodes and is represented by a string of $n+1$ symbols over $\{0, 1, *\}$ such that there are exactly d $*$'s (which denote *don't care*). The link between two neighboring nodes A and B is referred to as λ_B^A. A *path* from node A to node B is denoted by Λ_B^A, and we are interested in the *shortest paths* only. A link is assumed to have *link number* i if it connects two nodes whose addresses differ in the ith bit position (starting with the least significant bit as bit 0).

As opposed to those in a complete hypercube, nodes in an incomplete hypercube no longer play an identical role. For instance, every node in subcube $(00**)$ of IH_2^3 shown in Fig. 1 has four links

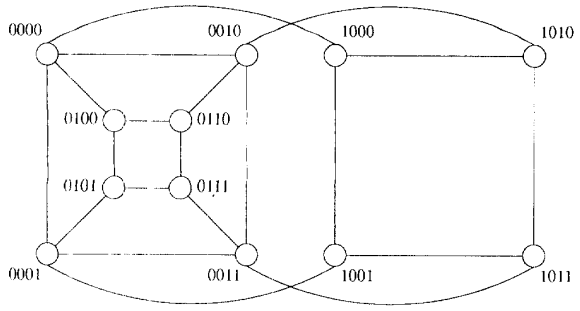

Fig. 1. An incomplete hypercube with 12 nodes, IH_2^3.

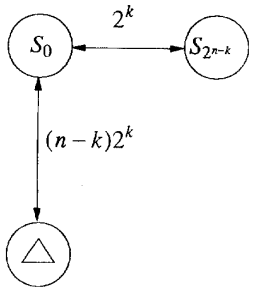

Fig. 2. An abstract structure of IH_k^n with $n-k \geq 1$.

(while the other nodes have three links each) and is connected to a corresponding node in subcube (10**); communication between nodes in subcube (01**) and nodes in subcube (10**) must go through nodes in subcube (00**). Nodes in subcube (00**) are particular and are referred to as *pivot* nodes. Generally, IH_k^n has 2^k pivot nodes, which lie in subcube $(0^{n-k+1}*^k)$, where $*^k$ represents k consecutive $*$'s. There are three different types of links in IH_k^n, i.e., links for connecting nodes inside the constituent complete cube H_n (called L^n); links for connecting nodes inside the constituent complete cube H_k (called L^k); and links for connecting nodes between pivot nodes and nodes in H_k (called L^c). It is obvious that, in the incomplete hypercube IH_k^n, L^n, L^k and L^c consist respectively of $n2^{n-1}$, $k2^{k-1}$ and 2^k links. A link $\in L^n$ and a link $\in L^k$ are the *corresponding links* of each other if they have the same link number.

Consider the collection of subcubes of IH_k^n:$\Phi = \{S_i \mid S_i = (x_n x_{n-1} \cdots x_k *^k)$ such that the value of $x_n x_{n-1} \cdots x_k = i,\ 0 \leq i \leq 2^{n-k}\}$. Each subcube S_i contains 2^k nodes, whose addresses are from $i2^k$ to $(i+1)\ 2^k - 1$; each node in IH_k^n belongs to one and only one subcube. As an instance, the collection of subcubes of IH_2^3 shown in Fig. 1 is $\Phi = \{S_0 = (00**), S_1 = (01**), S_2 = (10**)\}$. Notice that node X belongs to S_0 if and only if X is a pivot node in IH_k^n. Let the subset of Φ, $\bigcup_{i=1}^{2^{n-k}-1} S_i$, be denoted as Δ. Then, Δ consists of all such nodes in $(0*^n)$ that have no direct links to any node in $(10^{n-k}*^k)$.

An abstract structure of IH_k^n with $n-k \geq 1$ is sketched in Fig. 2, where arrows indicate links between subcubes with a number next to an arrow denoting the amount of actual links present. For the case of $n = k$, the circle containing Δ and the arrow incident to it are removed altogether, yielding a simpler figure, which shows the case of H_{n+1}. There are $(n-k)2^k$ links between S_0 and Δ; whereas only 2^k links (which constitute L^c) exist between S_0 and $S_{2^{n-k}}$. Traffic density over a link $\in L^c$ is thus expected to be higher.

Traffic density analyzed subsequently is under the following routing process. Every issued message in incomplete hypercubes carries the relative address of its source node and its destination node as the routing tag. The routing tag is checked leftwards, starting with the least significant bit consistently. A message is routed through link i only if bit i is the least significant nonzero bit in the tag *and* link i exists. This routing is similar to that introduced in [5], and it is deadlock-free as lower (existing) dimension links are always demanded before higher dimension ones [3].

A complete binary tree of height n, denoted by T_n, is composed of a root node and two T_{n-1}'s. The quality of embeddings can be measured by the ratio of the embedded network size to the embedding hypercube size, given that adjacent nodes in the embedded network are mapped to neighboring hypercube nodes (which corresponds to the case of unit dilation in [9]). The ratio reflects the *utilization* of hypercube nodes. It has been shown by Wu [9] that the smallest degree of complete hypercubes possible to embed T_n with adjacency preserved is $n+1$, yielding utilization slightly less than 50%. Later, H_n is shown [8] to be capable of housing a "stretched" binary tree with height n (which is a T_n with a two-degree node inserted between the root node and one of its two T_{n-1}'s). The embedding of a "stretched" binary tree in H_n does not preserve node adjacencies of T_n, the "original" binary tree. Embedding binary trees in faulty hypercubes has been treated recently [4]. It is found in [4] that T_n can be embedded in H_{n+1} with no more than $n - \lceil \log_2(n+1) \rceil$ failed nodes/links.

III. Structural Analysis of Incomplete Hypercubes

Our analysis is carried out for the case of *uniform* message distributions, i.e., the average rate at which node A sends messages to node B is the same for all nodes A and B, where $A \neq B$.

A. Diameter

The diameter of IH_k^n can be obtained by finding the pairs of nodes which are the furthest from each other. For any two nodes in IH_k^n, the number of differing bits between their addresses is less than or equal to $n+1$. Since the pair of nodes (01^n) and (10^n) always exist and their addresses differ in exactly $n+1$ bits, the diameter of IH_k^n is clearly $n+1$.

B. Mean Internode Distance

The mean number of traversals of messages in a topology, called the *mean internode distance* (denoted by T_{mean}), indicates the transmission latency of a "typical" message and is a fundamental property of a topology. The mean traversal of messages issued from a node in an incomplete hypercube is not fixed. For example, in IH_2^3 given in Fig. 1, the mean traversal of messages issued from node (0110) is larger than that issued from node (0000). Due to this asymmetry, we calculate T_{mean} by making use of the fact that T_{mean} of a topology is equivalent to the average value of mean traversal of messages issued from all nodes in the topology [1]. The derivation of T_{mean} is carried out by examining the mean traversal of messages issued from each cube node, under the uniform message distribution. Let $T_{\text{mean}}(X)$ denote the mean traversal of messages issued from node X. Then, T_{mean} of IH_k^n with size N is expressed by

$$T_{\text{mean}} = \frac{1}{N} \sum_{X \in IH_k^n} T_{\text{mean}}(X). \tag{1}$$

For any incomplete hypercube IH_k^n, we have the following lemma, which enables us to calculate T_{mean} efficiently. (Due to the space limitation, consult [6] for proofs of Lemma 1 and Lemma 2 below.)

Lemma 1: $T_{\text{mean}}(X) = T_{\text{mean}}(Y)$ for all nodes $X, Y \in S_i$, where $S_i \in \Phi$.

The above lemma describes that the mean traversal of messages issued from any node in the same subcube ($\in \Phi$) is the same. Let T^i_{mean} represent $T_{\text{mean}}(X \in S_i)$ for simplicity. Then, we have Lemma 2, which characterizes the mean traversal of messages issued by a node in any subcube.

Lemma 2: For any subcube $S_i \in \Phi$,

$$T^i_{\text{mean}} = \left(\sum_{l=1}^{n}\binom{n}{l} \times l + \sum_{l=0}^{k}\binom{k}{l} \times (1+|i|+l)\right) \times p, \quad \text{for } 0 \le i < 2^{n-k},$$

$$\text{and } T^i_{\text{mean}} = \left(\sum_{l=0}^{n}\binom{n}{l} \times (1+l) + \sum_{l=1}^{k}\binom{k}{l} \times l\right) \times p, \quad \text{for } i = 2^{n-k},$$

where $|i|$ is the number of bit ones in the binary representation of i and $p = \frac{1}{2^n+2^k-1}$, the probability of a message terminating at any node (other than the issuing node) under a uniform message distribution.

From Lemma 1, Lemma 2, and (1), we get the next theorem, whose proof is provided in [6].

Theorem 1: T_{mean} of IH^n_k is given by

$$T_{\text{mean}} = \frac{2^k \times \left(2^n(2+n+n2^{n-k-1}) + k2^{k-1}\right)}{(2^n+2^k-1)(2^n+2^k)}. \tag{2}$$

Note that when $k = n$, (2) is reduced to $T_{\text{mean}} = \frac{(n+1)2^n}{2^{n+1}-1}$, which denotes T_{mean} of H_{n+1} [6]. This is expected because when $k = n$, IH^n_k becomes H_{n+1}. It is found from (2) that for a fixed n, T_{mean} virtually stays unchanged when k is small, since at that time the number of nodes in the constituent complete hypercube of $(10^{n-k}*^k) = H_k$ is much less than 2^n and the majority of communication takes place inside H_n, giving rise to T_{mean} near $n/2$. As k is big enough (roughly, $\ge n-5$), T_{mean} starts to grow noticeably and approaches $(n+1)/2$ as $k = n$. Like in a complete hypercube, T_{mean} of an incomplete hypercube grows only at a logarithmic rate of the network size, a desirable property.

C. Traffic Density

The average number of messages over a link per unit time is referred to as *traffic density* (denoted by TD), which is an important property of topologies. Traffic density is estimated under the situation that messages are generated by all nodes with a rate of 1.0. A message involves on an average T_{mean} link traversals in IH^n_k, and totally there are $(n2^{n-1}+k2^{k-1}+2^k)$ links serving $N = (2^n+2^k)$ nodes, each of which issues one message at the beginning of every unit time; so the average traffic density over all links in the system, TD_a, is $\text{TD}_a = \frac{N \times T_{\text{mean}}}{n2^{n-1}+k2^{k-1}+2^k}$.

In fact, traffic density over each link in an incomplete hypercube varies because the number of links connected to a node is not fixed. To obtain actual traffic density over an individual link requires a detailed traffic pattern analysis, which is closely related to the routing algorithm employed. Here, traffic density with respect to the routing algorithm described in Section II is examined.

Traffic density over a link is derived by estimating the average number of messages over the link during a unit time. It comprises all *traffic density components* due to messages sent from a node A to another node B during a unit time such that Λ^A_B contains the link. Let the traffic density component over link λ^C_D due to messages sent from A to B be denoted by $\text{TDC}\langle\lambda^C_D, \Lambda^A_B\rangle$. Traffic density over link λ^C_D equals $\sum_{\lambda^C_D \subset \Lambda^A_B} \text{TDC}\langle\lambda^C_D, \Lambda^A_B\rangle$, where the summation is taken over all possible pairs A and B such that $\lambda^C_D \subset \Lambda^A_B$.

Traffic density over links in IH^n_k is widely dispersed. We investigate in the following the highest traffic density (TD_h) over links in an incomplete hypercube. Finding out the highest traffic density is important because the worst case communication scenario tends to rely upon the highest traffic density. The subsequent two lemmas serve as the basis to arrive at TD_h (for a proof of them, please refer [6]). Lemma 3 describes the equality of traffic density components over a link due to messages among nodes.

Lemma 3: For an arbitrary node $A_1 = (10^{n-k}a_{k-1}a_{k-2}\cdots a_0) \in S_{2^{n-k}}$, and the pivot node $A_2 = (0^{n+1-k}a_{k-1}a_{k-2}\cdots a_0)$, we have a) for all $X \in (0*^n)$ and $\lambda^C_D \in L^n$, $\text{TDC}\langle\lambda^C_D, \Lambda^X_{A_1}\rangle = \text{TDC}\langle\lambda^C_D, \Lambda^X_{A_2}\rangle$, b) for all $Z = (0z_{n-1}\cdots z_k z_{k-1}\ z_{k-2}\cdots z_0) \in \Delta$ and $\lambda^C_D \in L^n$, $\text{TDC}\langle\lambda^C_D, \Lambda^{A_1}_Z\rangle = \text{TDC}\langle\lambda^C_D, \Lambda^J_Z\rangle$, where $J = (0^{n+1-k}z_{k-1}\ z_{k-2}\cdots z_0) \in S_0$, and c) for all $X \in (0*^n)$ and $\lambda^E_F \in L^k$, $\text{TDC}\langle\lambda^E_F, \Lambda^{A_1}_X\rangle = \text{TDC}\ \langle\lambda^C_D, \Lambda^{A_2}_X\rangle$, where λ^C_D is the corresponding link of λ^E_F.

Result a) above implies that if a message from X to A_1 traverses a link inside constituent cube H_n, any message from X to A_2 must traverse the same link; whereas b) indicates that if a message from A_1 to $Z(\in \Delta)$ passes through a link inside constituent cube H_n, any message from J to Z must pass through the same link. Result c) reveals that if a message from A_1 to X traverses a link inside constituent cube H_k, any message from A_2 to X must traverse its corresponding link. These results characterize how messages travel over various links in IH^n_k. Lemma 4 below specifies that the total traffic components over an L^k link contributed by messages from H_k to H_n are the same as those over its corresponding link contributed by messages internal to H_n.

Lemma 4: For any link $\lambda^E_F \in L^k$ and its corresponding link λ^C_D, we have

$$\sum_{X_1 \in S_{2^{n-k}}, Y_1 \in (0*^n)} \text{TDC}\langle\lambda^E_F, \Lambda^{X_1}_{Y_1}\rangle = \sum_{X_2, Y_2 \in (0*^n)} \text{TDC}\langle\lambda^C_D, \Lambda^{X_2}_{Y_2}\rangle.$$

We now need to evaluate the expression of $\sum_{X_2,Y_2\in(0*^n)} \text{TDC}\langle\lambda^C_D, \Lambda^{X_2}_{Y_2}\rangle$ for any $\lambda^C_D \in L^n$. Actually, it is the same as traffic density over a link in H_n, except that each node now takes on an average 2^n+2^k-1 (instead of 2^n-1) time units to send a message to each of the other nodes, or equivalently, the probability of an issued message terminating at each of the other nodes is $\frac{1}{2^n+2^k-1}$. From TD over a link in H_n being $\frac{2^n}{2^n-1}$ [6], we have traffic density over link $\lambda^C_D \in L^n$ equal to $\frac{2^n}{2^n+2^k-1}$. This result together with Lemmas 3 and 4 leads to the next important theorem, whose proof can be found in [6].

Theorem 2: For IH^n_k, traffic density over each link in L^n or L^k does not exceed $\frac{2^{n+1}}{2^n+2^k-1}$.

In the following theorem, we prove that traffic density on each link $\in L^c$ is precisely $\frac{2^{n+1}}{2^n+2^k-1}$. Since traffic density on each link $\in L^n$ or L^k is shown to be no higher than $\frac{2^{n+1}}{2^n+2^k-1}$, the highest traffic density in IH^n_k is thereby $\frac{2^{n+1}}{2^n+2^k-1}$.

Theorem 3: For IH^n_k, traffic density over each link in L^c is $\frac{2^{n+1}}{2^n+2^k-1}$, which equals TD_h.

Proof: Traffic density over a link $\lambda^P_Q \in L^c$ is given by $\sum_{X_1\in S_{2^{n-k}}, Y_1\in(0*^n)} \text{TDC}\ \langle\lambda^P_Q, \Lambda^{X_1}_{Y_1}\rangle + \sum_{X_2\in(0*^n), Y_2\in S_{2^{n-k}}} \text{TDC}\langle\lambda^P_Q, \Lambda^{X_2}_{Y_2}\rangle$. Based on the facts that 1) there are 2^k nodes in $S_{2^{n-k}}$ and 2^n nodes in $(0*^n)$, 2) the probability of a message issued by a node in $S_{2^{n-k}}$ to a node in $(0*^n)$ is $\frac{1}{2^n+2^k-1}$, 3) a message sent from a node in $S_{2^{n-k}}$ to a node in $(0*^n)$ must pass through one of the links in L^c (see Fig. 2), and 4) all the messages sent from $S_{2^{n-k}}$ to $(0*^n)$ are through the 2^k links in L^c equally likely due to symmetry, the first term is equal to $\frac{2^k \times 2^n}{2^k \times (2^n+2^k-1)} = \frac{2^n}{2^n+2^k-1}$. Similarly, the second term equals

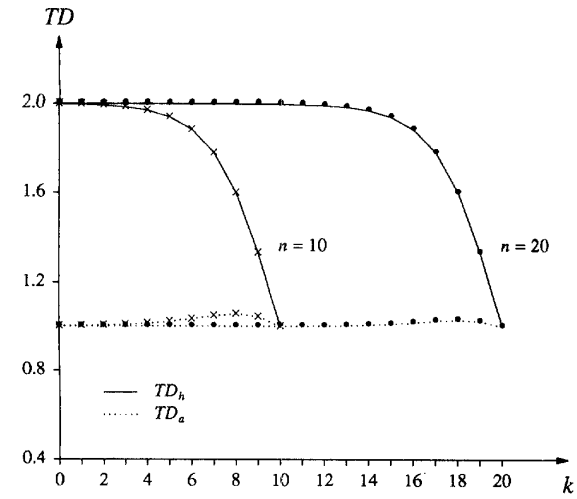

Fig. 3. Traffic density (TD) versus k in IH_k^n under a uniform message distribution.

$\frac{2^n}{2^n+2^k-1}$. From Theorem 2 and the preceding statement, it is clear that the highest traffic density (TD_h) in IH_k^n equals $\frac{2^{n+1}}{2^n+2^k-1}$. □

When $k = n$, TD_h and TD_a agree as expected because IH_k^n then becomes H_{n+1}, where each link has the same traffic density, whose value is given by $\frac{2^{n+1}}{2^{n+1}-1}$ [6]. From this theorem, we know that traffic density over any link in the incomplete hypercube is no larger than 2, irrespective of the system size and despite the fact that the number of links between S_0 and $S_{2^{n-k}}$ is confined to 2^k (see Fig. 2). This is an interesting property. Compared with other asymmetric structures, such as trees or stars, in which certain points (links or nodes) have traffic density as high as $O(N)$ for a size N system, the incomplete hypercube clearly is more applicable in practice because a structure with very high traffic density over certain points is likely to suffer from degraded performance and reliability. It is much easier to design a link with enough bandwidth for incomplete hypercubes than for other asymmetric structures so as to keep every link below half saturated and yield a short queueing time (the queueing knee is near half saturation). If we want to maintain good performance of a complete hypercube even after some links/nodes become faulty, cube links can easily be so designed that every link is still below half saturated after faults arise.

Traffic density is plotted as a function of k for $n = 10$ and 20 in Fig. 3, where both TD_h and TD_a are given. It can be seen that as k is small, the number of messages passing through a link with the densest traffic during a unit time is about 2. When k is large enough (roughly, $\geq n-5$), TD_h starts to improve rapidly, and as $k = n$, it eventually reaches the point where every link has the same traffic density, which is close to 1. TD_a, however, always stays pretty much around 1, indicating that the majority of links has only one message traversed each during a unit time.

IV. Tree Embedding in Incomplete Hypercubes

The minimum incomplete hypercube able to embed a given binary tree with adjacency preserved is pursued here. Although the number of nodes in IH_k^n exceeds that in T_n for any $k = 0, 1, \cdots, n-1$, it is discovered that IH_k^n cannot embed T_n for $k < n-1$ when n is larger than 4, as stated in the next lemma.

Lemma 5: A binary tree T_n, $n > 4$, cannot be embedded in IH_k^n with adjacency preserved for $k < n-1$.

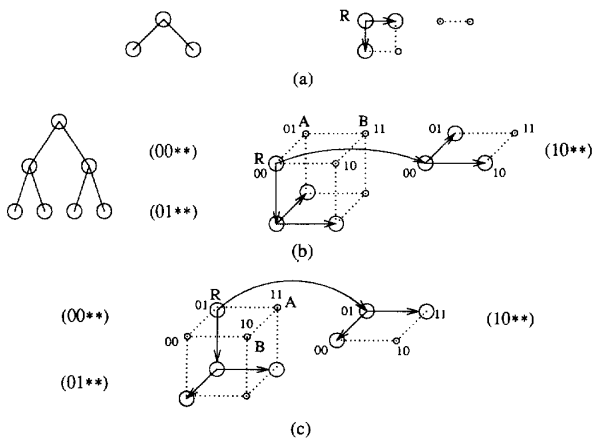

Fig. 4. The embedding of (a) T_2 in IH_1^2 and (b) T_3 in IH_2^3. (c) The result of applying $RT_{1,0}$ to (b). (Small circles denote free nodes and tree links are mapped to solid arrows. Some unmapped cube links are left out for clearness.)

A proof of this lemma is provided in [10]. What remains to consider is the embedding of T_n into IH_{n-1}^n. Indeed, there is a systematic procedure that enables us to inductively realize the embedding. The subsequent statements help to formulate the procedure. Let $f_i: T_i \rightarrow IH_{i-1}^i$ be an embedding of T_i into IH_{i-1}^i. A *free* node in IH_{i-1}^i under embedding f_i is one to which no tree node is mapped (or equivalently, which is not the image of any tree node). Embedding f_i is said to have the "free-free neighbor" property if the image of the root of T_i, R, has a free neighbor A and A has a free neighbor B, namely, neither A nor B is an image of any node $\in T_i$. (The "free-free neighbor" property was first introduced by Wu [9].) The embedding shown in Figs. 4(b) and 4(c) have the "free-free neighbor" property. This property allows us to construct a larger sized embedding from smaller ones, as will be seen shortly.

A *rotational transformation* $\mathrm{RT}_{i,j}$ rotates an incomplete hypercube along (i,j)-plane by 90 degrees, namely, the address of every node $(x_{n-1}x_{n-2}\cdots x_{i+1}x_ix_{i-1}\cdots x_{j+1}x_jx_{j-1}\cdots x_0)$ is transformed into

$$\mathrm{RT}_{i,j}(x_{n-1}x_{n-2}\cdots x_{i+1}x_ix_{i-1}\cdots x_{j+1}x_jx_{j-1}\cdots x_0) = x_{n-1}x_{n-2}\cdots x_{i+1}z_ix_{i-1}\cdots x_{j+1}z_jx_{j-1}\cdots x_0 \quad (3)$$

with $\langle x_i\ x_j,\ z_i\ z_j\rangle$ satisfying $\langle 0\ 0,\ 0\ 1\rangle$, $\langle 0\ 1,\ 1\ 1\rangle$, $\langle 1\ 1,\ 1\ 0\rangle$, and $\langle 1\ 0,\ 0\ 0\rangle$ (or equivalently, nodes with $(x_i\ x_j) = (0\ 0)$, $(0\ 1)$, $(1\ 1)$, and $(1\ 0)$ are rotated to nodes with $(z_i\ z_j) = (0\ 1)$, $(1\ 1)$, $(1\ 0)$, and $(0\ 0)$, respectively). Similarly, $\mathrm{RT}_{j,i}$, a rotational transformation on an incomplete hypercube along (j,i)-plane, can be defined by (3) with $\langle x_i\ x_j,\ z_i\ z_j\rangle$ satisfying $\langle 0\ 0,\ 1\ 0\rangle$, $\langle 1\ 0,\ 1\ 1\rangle$, $\langle 1\ 1,\ 0\ 1\rangle$, and $\langle 0\ 1,\ 0\ 0\rangle$. Fig. 4(c) illustrates the result of $\mathrm{RT}_{1,0}$ on Fig. 4(b). Since IH_k^n consists of subcubes $(0x^n)$ and $(10^{n-k}*^k)$, the set of allowable rotational transformations on IH_k^n can easily be shown as $\{\mathrm{RT}_{i,j} \mid i,j \leq k-1\}$. Notice that $\mathrm{RT}_{i,j}$ on complete hypercubes exists for an arbitrary pair of i and j; while on incomplete hypercubes, only for restricted i and j. For example, $\mathrm{RT}_{3,2}$ on Fig. 4(b) is not permitted because IH_2^3 contains no nodes with addresses $11**$ which are necessary for the transformation to exist.

Like the construction of a tree, the embedding of a tree is accomplished by building up from the embeddings of smaller subtrees. To realize f_n (i.e., the embedding of T_n in IH_{n-1}^n), we start with two copies of f_{n-1}'s. The free neighbor of an arbitrary one of the two f_{n-1}'s is chosen as the root image of f_n, with the incomplete hypercube where the other f_{n-1} resides being rotated 90 degrees so as to preserve node adjacencies. It is clear that rotational transformation

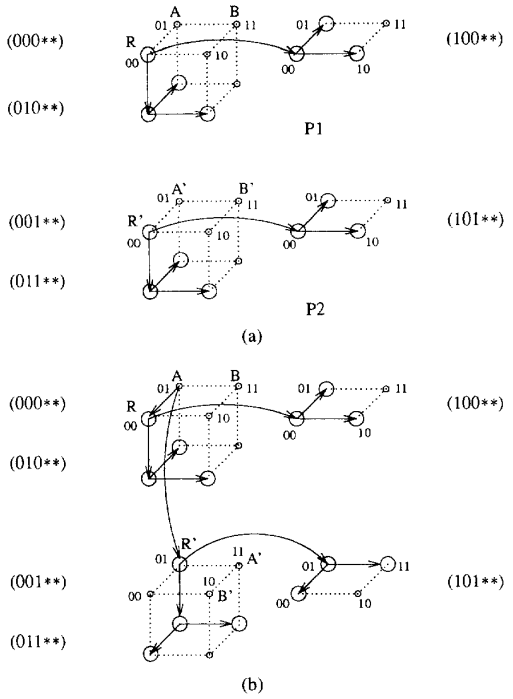

Fig. 5. Creating the embedding of T_4 in IH_3^4 after (a) step (S2) and (b) Step (S5) of Procedure E.

$\mathrm{RT}_{i,j}$ defined above satisfies this requirement. To guarantee that f_n still possesses the "free-free neighbor" property, the rotation should be carried out along the plane determined by the locations of the root image R and its "free-free neighbors" A and B in f_{n-1}'s; specifically, if link μ connects R and A and link ν connects A and B, then, the transformation $\mathrm{RT}_{\mu,\nu}$ is used. This transformation always exists as long as both μ and ν are less than or equal to $n-3$ (because the incomplete hypercube in question is IH_{n-2}^{n-1}). Apparently, there are many selections for μ and ν with a given n. In the following constructive procedure, we choose $\mu = n-3$ and $\nu = n-4$. The use of other allowable transformations can be explained similarly.

Procedure E (to construct f_n from a pair of f_{n-1}'s)

(S1) Duplicate IH_{n-2}^{n-1} and the embedding f_{n-1} in it. This duplicate part is called $P2$; whereas the original part is $P1$.

(S2) The address of node $x_{n-1}x_{n-2}x_{n-3}x_{n-4}\cdots x_0 \in P1$ is relabeled as $x_{n-1}x_{n-2}\ 0\ x_{n-3}x_{n-4}\cdots x_0$ (notice that the address of a node in the resulting structure involves one additional bit); whereas that of node $x_{n-1}x_{n-2}x_{n-3}x_{n-4}\cdots x_0 \in P2$, as $x_{n-1}x_{n-2}\ 1\ x_{n-3}x_{n-4}\cdots x_0$.

(S3) Apply $\mathrm{RT}_{n-3,n-4}$ to nodes in $P2$.

(S4) Connect a node $\in P1$ and a corresponding node $\in P2$ whose addresses differ only in one bit.

(S5) Suppose that A is the free neighbor of the image of the root node in $P1$, and B is the free neighbor of A. Select A as the root node image of the resultant embedding and make the connections to the images of its two children.

Fig. 5 illustrates how f_4 is constructed from two f_3's, each of which is identical to that given in Fig. 4(b). The configuration together with node addresses after step (S2) is depicted in Fig. 5(a). Fig. 5(b) gives the embedding of T_4 in IH_3^4 after step (S5) is performed. Notice that $P2$ after step (S3) is precisely the structure depicted in Fig. 4(c), and that Fig. 5(b) has certain connections intentionally left out for clarity. Procedure E indeed provides us with a systematic way to construct the embedding of T_n in IH_{n-1}^n. In general, this embedding is always possible, as described below.

Theorem 4: A binary tree T_n with $n > 0$ can be embedded in an incomplete hypercube IH_{n-1}^n such that the adjacencies of nodes $\in T_n$ are preserved.

Proof: (By induction). Figs. 4(a) and 4(b) demonstrate how T_2 and T_3 are embedded respectively in IH_1^2 and IH_2^3 with adjacency preserved. The embedding of T_4 in IH_3^4 is sketched in Fig. 5(b), where the root node is $R = (00001)$, which has the "free-free neighbors" $A = (00011)$ and $B = (00111)$.

Suppose that T_{n-1}, $n > 4$, can be embedded in IH_{n-2}^{n-1} with adjacency preserved. According to Procedure E, IH_{n-2}^{n-1} and the embedding in it are first duplicated (this part is called $P2$, in contrast to the original part, $P1$). Let the root node image $R = (00001^{n-4})$ in $P1$ have a free neighbor $A = (00011^{n-4})$, which has a free neighbor $B = (00111^{n-4})$. Similarly, let the root node image R' in $P2$ have free-free neighbors A' and B'. The addresses of these nodes after relabeling become: $R = (000001^{n-4})$, $A = (000011^{n-4})$, $B = (000111^{n-4})$, $R' = (001001^{n-4})$, $A' = (001011^{n-4})$, and $B' = (001111^{n-4})$. After applying $\mathrm{RT}_{n-3,n-4}$ to $P2$, the addresses of nodes R', A', and B' are 001011^{n-4}, 001111^{n-4}, and 001101^{n-4}, respectively. Thus, node A is adjacent to both nodes R and R', and is chosen as the root node image of the new embedding f_n, the embedding of T_n in IH_{n-1}^n. Embedding f_n maintains adjacencies and also has the "free-free neighbor" property: the root node image (00001^{n-3}) has a free neighboring node (00011^{n-3}), which in turn has a free neighboring node (00111^{n-3}). This completes the proof. □

Theorem 4 indicates that an adjacency-preserved embedding of binary trees can always be attained better in incomplete hypercubes IH_{n-1}^n than in complete hypercubes H_{n+1}, improving utilization from roughly 50% to 67%. Equivalently, it also shows that T_n can be embedded in H_{n+1} with up to one quarter of nodes failed, as long as the H_{n+1} operating portion contains IH_{n-1}^n.

V. Conclusion

In this brief contribution, we have investigated the structural and tree embedding aspects of incomplete hypercubes. Structural properties, including diameter, mean internode distance, and traffic density, of the incomplete hypercube with size $2^n + 2^k, 0 \le k \le n$, are analyzed. Our analytic results indicate that the mean internode distance in such an incomplete hypercube increases only at a logarithmic rate of the system size. More importantly, traffic density over links under the uniform message distribution has been shown to be bounded by 2, irrespective of the system size and despite its structural nonhomogeneity, when messages are issued by all the nodes with a rate of 1.0 at the beginning of every unit time. Compared to other asymmetric topologies, such as stars and trees, the incomplete hypercube is much less likely to cause any single point of congestion, avoiding any potential degradation in performance and reliability. Cube links can easily be so constructed that every link is below half saturated in order to assure a short mean message queueing time, maintaining good performance.

We have presented that a complete binary tree with height n, T_n, can be embedded in incomplete hypercube IH_{n-1}^n with adjacencies preserved, yielding improved utilization. This result also reveals that H_{n+1} still can effectively support T_n in the presence of failed cube links/nodes, as long as the operating portion of H_{n+1} contains IH_{n-1}^n. In addition, it is found in [10] that for an absolute majority of cases, a better grid embedding can be achieved in IH_k^n than in H_{n+1}. With virtually every advantage of complete hypercubes plus a better embedding ability but without a rigid restriction on the system size, the incomplete hypercube appears to be an attractive and practical topology.

ACKNOWLEDGMENT

The authors thank S. Wei for his contribution at the initial phase of this work and P.-J. Chuang for his assistance in preparing figures.

REFERENCES

[1] D. A. Reed and R. M. Fujimoto, *Multicomputer Networks: Message-Based Parallel Processing*. Cambridge, MA: The MIT Press, 1987.
[2] W.-J. Hsu, "Fibonacci cubes—A new interconnection topology," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, pp. 3–12, Jan. 1993.
[3] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. C-36, pp. 547–553, May 1987.
[4] M. Y. Chan and S.-J. Lee, "Fault-tolerant embedding of complete binary trees in hypercubes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, pp. 277–288, Mar. 1993.
[5] H. P. Katseff, "Incomplete hypercubes," *IEEE Trans. Comput.*, vol. 37, pp. 604–608, May 1988.
[6] N.-F. Tzeng, "Structural properties of incomplete hypercube computers," in *Proc. 10th IEEE Int. Conf. Distrib. Computing Syst.*, May 1990, pp. 262–269.
[7] H.-L. Chen and N.-F. Tzeng, "Distributed identification of all maximal incomplete subcubes in a faulty hypercube," in *Proc. 8th Int. Parallel Processing Symp.*, Apr. 1994, pp. 723–728.
[8] S. R. Deshpande and R. M. Jenevein, "Scalability of binary tree on a hypercube," in *Proc. 1986 Int. Conf. Parallel Processing*, Aug. 1986, pp. 661–668.
[9] A. Y. Wu, "Embedding of tree networks into hypercubes," *J. Parallel Distrib. Computing*, vol. 2, pp. 238–249, 1985.
[10] N.-F. Tzeng, P.-J. Chuang, and H.-L. Chen, "Embeddings in incomplete hypercubes," in *Proc. 1990 Int. Conf. Parallel Processing*, vol. III, Aug. 1990, pp. 335–339.

Hierarchical Classification of Permutation Classes in Multistage Interconnection Networks

Nabanita Das, Bhargab B. Bhattacharya, and Jayasree Dattagupta

***Abstract*— This brief contribution explores a new hierarchy among different permutation classes, that has many applications in multistage interconnection networks. The well-known LC (linear-complement) class is shown to be merely a subset of the closure set of the BP (bit-permute) class, known as the BPCL (bit-permute-closure) class; the closure is obtained by applying certain group-transformation rules on the BP-permutations. It indicates that for every permutation P of the LC class, there exists a permutation P^* in the BP class, such that the conflict graphs of P and P^* are isomorphic, for n-stage MIN's. This obviates the practice of treating the LC class as a special case; the existing algorithm for optimal routing of BPC class in an n-stage MIN, can take care of optimal routing of the LC class as well. Finally, the relationships of BPCL with other classes of permutations, e.g., LIE (linear-input-equivalence), BPIE (bit-permute-input-equivalence), BPOE (bit-permute-output-equivalence) are also exposed. Apart from lending better understanding and an integral view of the universe of permutations, these results are found to be useful in accelerating routability in n-stage MIN's as well as in $(2n-1)$-stage Benes and shuffle-exchange networks.**

***Index Terms*— Multistage interconnection networks (MIN), BP(bit-permute) permutations, linear permutations, baseline network, Benes network, conflict graph, optimal routing.**

I. INTRODUCTION

Design and analysis of multistage interconnection networks (MIN) play a crucial role in large scale parallel processing systems. Various topologies of such networks have been reported in the literature for use in SIMD and MIMD computers. An $N \times N$ unique-path, full-access MIN, e.g., the baseline or omega, has $n(=\log_2 N)$ stages and $(N.n/2)$ binary switches [1]–[6]. One fundamental criterion in the selection and design of a MIN is its permutation capability [7], i.e., the ability to establish simultaneous one-to-one and onto communications among different modules, usually represented as a permutation.

Routing an arbitrary permutation P through a unique-path full-access MIN, may require the usage of common links leading to a conflict, and therefore, may not be realizable in a single pass. The problem of *optimal routing* is to determine the minimum number of passes required to realize P. Equivalently, one needs to partition P into minimum number of subsets, such that transmissions included in each subset are conflict-free, and hence routable in a single pass. The conflict information is usually represented by a graph, called the conflict graph $G(V, E)$ [8], that consists of N vertices each representing a transmission of P; two vertices are adjacent if and only if the corresponding transmissions are conflicting, i.e., they demand a common link in the network. The optimal routing problem can then be mapped to the well-known graph-coloring problem [8], which for an arbitrary permutation, is NP-hard [9]. A heuristic algorithm of complexity $O(N^3)$ was reported in [10] to tackle an arbitrary permutation. However, the BPC (bit-permute-complement) class of permutations, is optimally routable in omega/delta network [8]; the time complexity of the algorithm turns out to be linear in the number of switches. Later, it has been shown that the same algorithm can

Manuscript received October 13, 1992; revised August 16, 1993.
The authors are with the Electronics Unit, Indian Statistical Institute, 203, Barrackpore Trunk Road, Calcutta 700 035, India; e-mail: bhargab@isical.ernet.in.
IEEE Log Number 9404363.