Embedding Hierarchical Networks Into The Hypercube

Mounir Hamdi
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
Email: hamdi@cs.ust.hk

Abstract

The embedding of one interconnection network into another is a very important issue in the design and analysis of parallel algorithms. Through such embeddings the algorithms originally developed for one architecture can be directly mapped to another architecture. This paper describes novel methods for the embedding of hierarchical interconnection networks in the hypercube to minimize the dilation and the expansion costs, and mathematically proves their *optimality*. To the best of our knowledge, this is the first result on embedding hierarchical networks into the hypercube. Thus, this embedding has significant practical importance in enhancing the capabilities of the hypercube since hierarchically constructed networks have proven to be very cost-effective in a wide range of applications, and are considered as the future generation topologies for massively parallel computer systems.

Keywords: Embedding, Hypercube, Hierarchical Networks, Dilation.

1. Introduction

The hypercube is one of the most versatile and efficient interconnection networks yet discovered for parallel computation. One of the biggest reasons for the popularity of the hypercube is its ability to efficiently embed many parallel architectures. There is a lot of motivation behind embedding other parallel architectures. First, efficient parallel algorithms may exist for some architecture which suit the needs of these algorithms perfectly, and we may wish to implement these algorithms on the hypercube. Second, the proof of embedding for an architecture is also a proof for all algorithms to be implemented in the hypercube architecture, with a level of efficiency determined only by the cost associated with the embedding. Further, since the embedded architecture is usually easier to understand and visualize, it is often easier to design algorithms for the simpler architecture. In this sense, the embedded architecture can be considered an abstraction from the hypercube, where the irrelevant connections are masked out. Finally, embedded architectures can be considered as parallel data structures for parallel architectures. The embedding method shows the way to implement these data structures on the hypercube parallel computer.

For example, most of the matrix algorithms developed for the hypercube use a mesh connected abstraction even though they may take advantage of other hypercube connections. As a result, algorithm development is simplified by focusing on the connections in the mesh architecture. It is well known that multidimensional meshes of suitable dimensions can be embedded in the hypercube without loss of adjacency properties [1]. Therefore a hypercube architecture can simulate the mesh architecture with constant overhead. For the above reasons a lot of research has been carried on embedding different topologies into the hypercube. The complete two-rooted binary tree of size 2^n has been shown to be embeddable in the n-dimensional hypercube while preserving the adjacency properties [8]. Hence, a hypercube architecture can also provide a binary tree abstraction when it is more suitable for an algorithm. Also, it has been shown that the pyramid architecture can be efficiently embedded into the hypercube [7]. Thus, a hypercube can provide a multi-resolution abstraction when it is desired for an algorithm.

Despite the popularity and the efficiency of the hypercube, it is relatively expensive to built such a parallel architecture because of the big number of links connected to each of its nodes. For this reason, many researchers have proposed hierarchical interconnection networks as an alternative to the hypercube [2, 3, 4, 5, 6]. Most of these interconnection networks are constructed by connecting a number of *clusters*, where a cluster is a hypercube of small dimension, with another interconnection network. These interconnection networks are shown to require less hardware than the hypercube, yet their performance approaches that of the hypercube. In this paper, we present an *optimal* embedding of these hierarchical interconnection networks into the hypercube. All of the proposed hierarchical interconnection networks are closely related to each other. However, in this paper, we will concentrate on the hierarchical interconnection networks proposed in [2, 3, 4]. But the results found in this paper can be easily extended to the other hierarchical interconnection networks. These hierarchical interconnection networks are

0-7803-2428-5/95 $4.00 © 1995 IEEE

shown to lend themselves to efficient parallel computations ranging from numerical algorithms to image processing algorithms and, thus, can be a good abstraction for the hypercube.

This paper is organized as follows. Section 2 presents the notations used in this paper and introduces the hierarchical networks. Section 3 presents the optimal embedding of the hierarchical networks into the hypercube. Finally, in Section 4, we give some concluding remarks.

2. Notations And Terminologies

The problem of embedding some guest graph $G = \{V_G, E_G\}$ into a host graph $H = \{V_H, E_H\}$ is to find a one-to-one function g: $V_G \longrightarrow V_H$. Dilation cost of embedding is max $\{d(f(u), f(v)); (u, v) \in E_G\}$, i.e., the longest distance between the images of the adjacent vertices of G. The embedding is adjacency preserving if dilation of f is equal to 1. Expansion cost is defined as the ratio $|V_H|$ / $|V_G|$. If dilation cost is minimum and expansion cost is minimum, then the embedding is *optimum.* Thus, the purpose of the embedding would be to minimize the dilation cost and the expansion cost. Minimizing dilation cost leads to the minimization of the degradation in time performance emulating the guest architecture, and minimizing expansion cost leads to the minimization of the hardware needed by the host architecture to emulate the guest architecture.

The guest graph, G, to be embedded into the host graph, H, (i.e. hypercube) is a hierarchical hypercube network, denoted HHN. A HHN is constructed by systematically connecting a number of clusters together, where a cluster is hypercube graph (network). A HHN is characterized by the number of nodes (vertices) in the hypercube cluster, N_Q. It is constructed by fully interconnecting N_Q hypercube clusters, creating a fully interconnected graph of hypercube clusters, with a total of N_Q^2 vertices. Each vertex in a HHN is labelled by an n-bit binary string ij (in our embedding it will be denoted by the pair (i, j)), where the $n/2$ most significant bits, i, is the label of the hypercube cluster that this vertex belong to, and the least significant $n/2$ bits, j, are vertex labels within the cluster. The edges between these hypercube clusters are formed by connecting vertex ij to vertex ji for all i and j, with $i \neq j$. These edges will be termed *diagonal* connections in our embedding. The rest of the edges will be termed *horizontal* connections in our embedding. Fig. 1 shows a HHN where the hypercube cluster has four vertices.

The binary n-cube (hypercube) which is the host graph, H, is a special case of a family of r-ary m-cubes. It has n dimensions and r nodes on each dimension. Let N be the total number of nodes. Consider

$$N = r^m = 2^n \text{ or } m = \log_2 N.$$

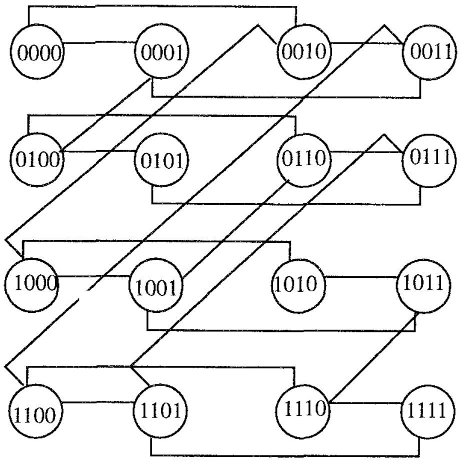

Fig. 1. A labelled HHN graph with 16 vertices.

Denote by $(e_0, e_1, \dots, e_{m-1})$, $0 \le e_i < r$, the nodes of a r-ary m-cube. Two nodes e and e' are adjacent if

$$e = (e_0, e_1, \dots, \dots, e_i, \dots, e_{m-1})$$

$$e' = (e_0, e_1, \dots, \dots, e_i \pm \bmod r, \dots, e_{m-1})$$

Consider the nodes of H

$$0, 1, 2, \dots, 2^{n-1}$$

in their binary representation

$$00 \dots 00, 00 \dots 01, \dots, 11 \dots 10, 11 \dots 11$$

This is shown graphically in Fig. 2 for a hypercube of size 16.

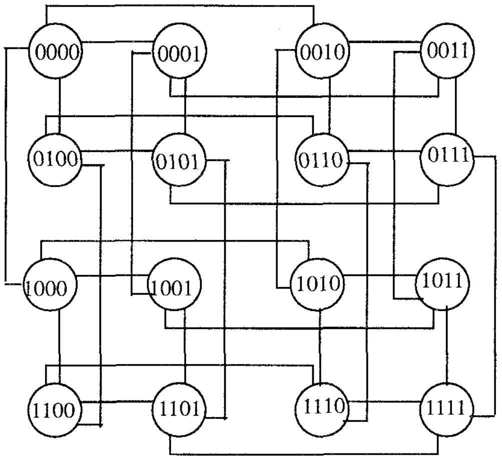

Fig. 2. A labelled hypercube graph with 16 vertices.

3. Embedding of HHN Into the Hypercube

A straight forward way of embedding a HHN into the hypercube is by mapping the nodes of the HHN to the nodes hypercube where their indices are the same, as suggested by some researchers [3, 9]. However, the dilation cost would be $\log_2(N)$ for a network of size N (e.g. nodes 0011 and 1100 in Fig. 1 are directly connected in a HHN, but are at a distance $\log_2(16) = 4$ from each other on the hypercube of Fig. 2). Thus, the dilation cost would be 4 in this case). Hence, the purpose of this paper is to present novel techniques employing recursive schemes that enable the embedding of a HHN into a hypercube of the same size with dilation cost 2 and expansion cost 1.

Let us arrange these numbers in an m-dimensional matrix $r \times r \times \ldots \times r$, denoted by $T(r, m)$. The elements of $T(r, m)$ are binary numbers of $m \log r$ bits each. Consider the case of $m = 2$ and, for simplicity, write $T(r)$ instead of $T(r, 2)$. The nodes of the binary n-cube are laid out in a square $r \times r$ matrix denoted by

$$T(r) = (t_{ij}) = \begin{bmatrix} t_{00} & t_{01} & \cdots & t_{0,r-1} \\ t_{10} & t_{11} & \cdots & t_{1,r-1} \\ \cdots & \cdots & \cdots & \cdots \\ t_{r-1,0} & t_{r-1,1} & \cdots & t_{r-1,r-1} \end{bmatrix}$$

where $0 \le i, j < r$, $r = 2^{n/2}$ and each t_{ij} is a binary number of $n = m \log r$ bits. To embed the graph G into H, we use the following mapping g:

$$node\ (i, j)\ of\ \ G \longrightarrow node\ \ t_{ij}\ \ of\ \ H$$

Denote by $00T(r) = (t'_{ij})$ where $t'_{ij} = 00t_{ij}$, that is 00 followed by the bits of t_{ij}. In other words, $00T(r)$ is the matrix $T(r)$ followed by 00 in front of their elements.

$$00T(r) = \begin{bmatrix} 00t_{00} & \cdots & 00t_{0,r-1} \\ \cdots & \cdots & \cdots \\ 00t_{r-1,0} & \cdots & 00t_{r-1,r-1} \end{bmatrix}$$

Analogously we have $01T(r)$, $10T(r)$ and $11T(r)$. The bits that precede $T(r)$ are said to be concatenated to $T(r)$.

Definition 1: We define $T(r)$ by means of the following recurrence:

$$T(2) = \begin{bmatrix} 10 & 01 \\ 00 & 11 \end{bmatrix}$$

$$T(r) = \begin{bmatrix} 10T(\frac{r}{2}) & 01T(\frac{r}{2}) \\ 00T^T(\frac{r}{2}) & 11T^T(\frac{r}{2}) \end{bmatrix}$$

where $T^T(r)$ is the transpose of $T(r)$. Fig. 3 is obtained directly from the definition of $T(r)$. As can be seen the maximum Hamming distance between any two directly connected nodes is equal to 2.

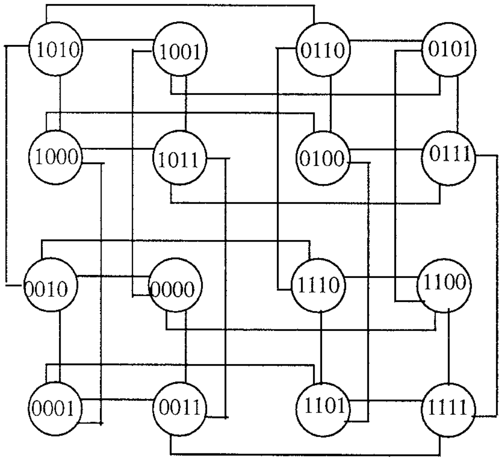

Fig. 3. Layout of the binary hypercube by $T(4)$.

Theorem 2: Let G be an HHN and let H be a binary n-cube of the same size. With the definition of $T(r)$ and using the mapping

$$node\ (i, j)\ \ of\ \ G \longrightarrow node\ \ t_{ij}\ \ of\ \ H$$

we have an embedding of G into H of dilation 2, that is, adjacent nodes of G are mapped to nodes which are at most 2 links away in H.

Proof: By induction on r.
For $r = 2$,

$$T(2) = \begin{bmatrix} t_{00} & t_{01} \\ t_{10} & t_{11} \end{bmatrix} = \begin{bmatrix} 10 & 01 \\ 00 & 11 \end{bmatrix}$$

The pairs of elements of $T(2)$, horizontal and diagonal (as in the HHN connections), have a maximum dilation of 2 which is between the nodes labeled 10 and 01, and the nodes labeled 00 and 11.

Assume valid the theorem for $r/2$. It is not difficult to show that the theorem holds for $T(r)$. By the induction hypothesis, the maximum Hamming distance (representing dilation in this case) horizontally between $10T(r/2)$ and $01T(r/2)$ is obviously equal to 2 since the only difference between them is in their most significant 2 bits. Analogously, the Hamming distance horizontally between $00T^T(r/2)$ and $11T^T(r/2)$ is also equal to 2. Further, the Hamming distance between the directly connected nodes in $01T(r/2)$ and $00T^T(r/2)$ (i.e. diagonal connections) is 1 since it is exactly equivalent to horizontal connection between $01T(r/2)$ and $00T(r/2)$.

Lastly, we should also prove that the directly connected nodes in $T^T(r/2)$ have a maximum Hamming Distance of 2. We also prove this by induction on r.

For $r = 2$.

$$T^T(2) = \begin{bmatrix} 10 & 00 \\ 01 & 11 \end{bmatrix}$$

The pairs of elements of $T^T(2)$ have a maximum Hamming distance of 1 between the directly connected nodes (which is horizontally). That is between nodes 10 and 00 and between nodes 01 and 11. Assume the induction is valid for $T^T(r/2)$, and let us prove it is valid for $T^T(r)$.

$$T^T(r) = \begin{bmatrix} 10T^T(\frac{r}{2}) & 00T(\frac{r}{2}) \\ 01T^T(\frac{r}{2}) & 11T(\frac{r}{2}) \end{bmatrix}$$

The connections between the nodes directly connected (i.e. diagonally) in the quadrant $00T(r/2)$ and $01T^T(r/2)$ have a Hamming distance of 1 since the differences between any two directly connected nodes is in the most significant 2 bits (i.e. either 00 or 01). We should also prove that the Hamming distance between the nodes directly connected (i.e. horizontally) between the quadrants $10T^T(r/2)$ and $00T(r/2)$ and between $01T^T(r/2)$ and $11T(r/2)$ is less than or equal to 2. In other words, we should prove that the Hamming distance between $T^T(r/2)$ and $T(r/2)$ is 1 which is clearly the case. Thus, the nodes directly connected in $T^T(r/2)$ have a Hamming distance of at most 2. ■

Consequently, a HHN can be embedded into a binary hypercube with a maximum dilation of 2 which is the *optimal* case because any embedding of the HHN into the hypercube must have a maximum dilation greater that 1 since it contains odd cycles [10, 11]. Moreover, since the number of nodes in the hypercube and the HHN are the same, the expansion cost is also optimal. This leads to the following corollary.

Corollary 3: The embedding of the HHN into the binary hypercube has maximum dilation equal to 2, and expansion cost equal to 1, and is optimal.

4. Conclusion

In this paper, we considered the problem of embedding hierarchical networks into the binary hypercube. A natural embedding method where nodes of the same indices are embedded on each other would lead to a dilation of $\log(N)$ for a network of size N. In this paper, however, we presented a more efficient embedding method. The proposed embedding is based on a recursive function. Once the nodes of the binary hypercube are laid out according to the given recursion, then it structures itself optimally in embedding the HHN which contains horizontal and diagonal connections. We obtained a dilation of 2 and an expansion cost of 1. Thus, any algorithm designed for the HHN can be executed on a binary hypercube of the same size with at most twice the degradation in time performance. This is one more affirmation of the usefulness of the rich architectural characteristics of the hypercube. Thus, we believe that as hardware cost of massively parallel systems goes down, we will witness the popularity of the hypercube more than ever before.

References

[1] M. Y. Chan and F. Y. L. Chin, "On Embedding Rectangular Grids in Hypercubes," *IEEE Transactions on Computers*, pp. 1285-1288, 1988.

[2] S. P. Dandamudi and D. L. Eager, "Hierarchical Interconnection Networks for Multicomputer Systems," *IEEE Transactions on Computers*, pp. 786-797, 1990.

[3] S. P. Dandamudi and D. L. Eager, "On Hypercube-based Hierarchical Interconnection Networks," *Journal Parallel and Distributed Computing*, pp. 283-289, 1991.

[4] M. Hamdi and R. W. Hall, "An Efficient Class of Interconnection Networks for Parallel Computing," *The Computer Journal*, Vol. 37, No. 2, 1994.

[5] J. M. Kumar and L. M. Patnaik, "Extended Hypercube: A Hierarchical Interconnection Network of Hypercubes," *IEEE Transactions on Parallel and Distributed Systems*, pp. 45-57, 1992.

[6] S. Laksmivarahan and S. K. Dhall, "A New Hierarchy of Hypercube Interconnection Schemes for Parallel Computers," *Journal of Supercomputing*, pp. 81-107, 1988.

[7] T. H. Lai and W. White, "Mapping Pyramid Algorithms Into Hypercubes," *Journal of Parallel and Distributed Computing*, pp. 42-54, 1990.

[8] A. Y. Wu, "Embedding of Tree Networks Into Hypercubes," *Journal of Parallel and Distributing Computing*, pp. 238-249, 1985.

[9] A. Zubair and S. N. Gupta, "Embeddings on a Boolean Cube," *BIT*, pp. 245-256, 1990.

[10] M. Hamdi and R. W. Hall, "Embedding Wavelet Transforms Onto Parallel Architectures," In *International Conference On Image Processing: Theory And Applications*, pp. 243-246, 1993.

[11] M. C. Ip, K. Y. Ng, K. L. Pun, M. Hamdi, and I. Ahmad, "Embedding Pyramids Into 3D Meshes," In *International Conference on Parallel and Distributed Systems*, 1993.