Efficient Embeddings Into The Hypercube Using Matrix Transformations

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

S. W. Song
Department of Computer Science
University of Sao Paulo
C.P. 20570 - Sao Paulo
SP - 01452-990, Brazil

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, based on matrix transformations, for efficiently embedding different networks into the hypercube (binary n-cube). First, we use this method to embed r-ary m-cubes into a binary n-cube of the same size with dilation 1. While our method has the same dilation as traditional methods using reflected Gray code, it has the additional property of making the layout of the binary n-cube more suitable for divide-and-conquer algorithms. Second, we use our matrix transformation method to optimally embed hierarchical interconnection networks into the binary n-cube which we would not achieve using reflected Gray code embedding. Thus, this embedding method has significant practical importance in enhancing the capabilities of the hypercube..

Keywords: Embedding, hypercube, hierarchical networks, dilation, *r*-ary *m*-cube.

1. Introduction

The hypercube (binary *n*-cube) 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 [3], [4]. 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 [16], [19]. 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 [10], [12], [15]. Thus, a hypercube can provide a multi-resolution abstraction when it is desired for an algorithm.

It is known that the binary n-cube can optimally embed an r-ary m-cube, with the same number of nodes [3], [4], [16], [18], [20]. These construction methods are based on the *reflected Gray code*. Using a different construction, we show similar results that an r-ary m-cube can be optimally embedded in the binary n-cube with the same number of nodes. In this case, we fold the hypercube recursively so that the final layout contains the r-ary m-cube. This layout has the following useful property: Once the binary n-cube has been laid out according to this method as an r-ary m-cube (with $2^n = r^m$), the sub-cubes formed by the nodes with the same first km bits, $1 \le k < \log r$, form smaller $(r/2^k)$-ary m-cubes. This property makes it suitable to be used in divide-and-conquer algorithms [1], [16]. Thus our embedding method further constitutes a useful tool for designing parallel algorithms for the hypercube. Our embedding method is based on the matricial visualization of the hypercube, together with a series of properties derived from the hypercube's recursive definition. This visualization method has been successfully used for the design of many hypercube algorithms [1], [2], [7], [16], [17].

In addition to efficiently embedding an r-ary m-cube, we use our matrix transformation method to efficiently embed hierarchical networks onto the binary hypercube. Many researchers have proposed hierarchical interconnection networks as an alternative to the hypercube [5], [6], [8], [9], [11], [13], [14]. Most of these interconnection networks are constructed by connecting a number of *clusters*, where a cluster is a hypercube of small dimension, with another 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 using a similar method adopted for embedding the r-ary m-cubes. 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 [8], [9]. But our embedding method can be easily extended to other hierarchical interconnection networks where their embedding on the hypercube is not documented. These hierarchical interconnection networks are 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 guest networks to be embedded into the binary n-cube. Section 3 and Section 4 present the optimal embedding of r-ary m-cubes and hierarchical networks into the hypercube, using our embedding method, respectively. Finally, in Section 5, 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: $V_G \rightarrow V_H$. The 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 binary n-cube (hypercube) which is the host graph, H, is a special case of a family of r-ary m-cubes, cubes with m dimensions and r nodes on each dimension. Let N be the total number of nodes. Consider

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

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

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

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

For simplicity, we use also, $i, j, \ldots$ instead of $e_0, e_1, \ldots$, for smaller dimensions. In the r-ary 2-cube, for example, node (i, j) is adjacent to nodes $(i, j \pm 1 \bmod r)$ and $(i \pm 1 \bmod r, j)$. Figure 1 shows an 8-ary-2-cube.

Besides embedding k-ary m-cubes into the binary n-cube, we will use the matrix transformation method to embed hierarchical networks into the binary n-cube. The guest graph, G, in this case is a hierarchical hypercube network, denoted HHN. An 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

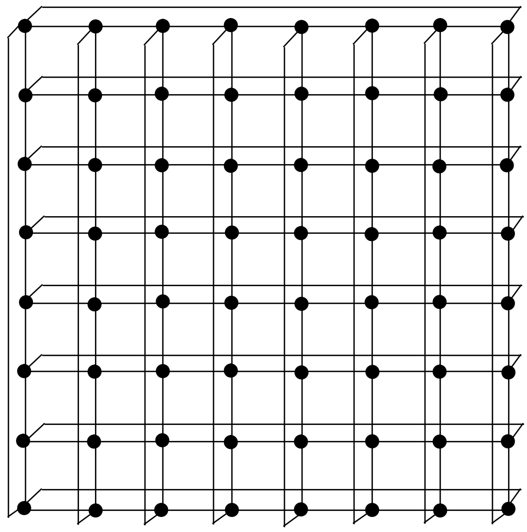

Figure 1. An 8-ary 2-cube.

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 an 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 belongs 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. For more details about HNN, the reader is referred to [8], [9]. Figure 2 shows an HHN where the hypercube cluster has eight vertices.

3. Embedding *r*-ary *m*-cube By Recursive Folding

Let $N = r^m = 2^n$ and r a power of 2 ($r = 2^s$). Consider as the guest graph G, the r-ary m-cube, and the host graph H, the binary n-cube. Consider the nodes of H

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

and their binary representation as

$$00 \ldots 00,\ 00 \ldots 01,\ \ldots,\ 11 \ldots 10,\ 11 \ldots 11$$

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

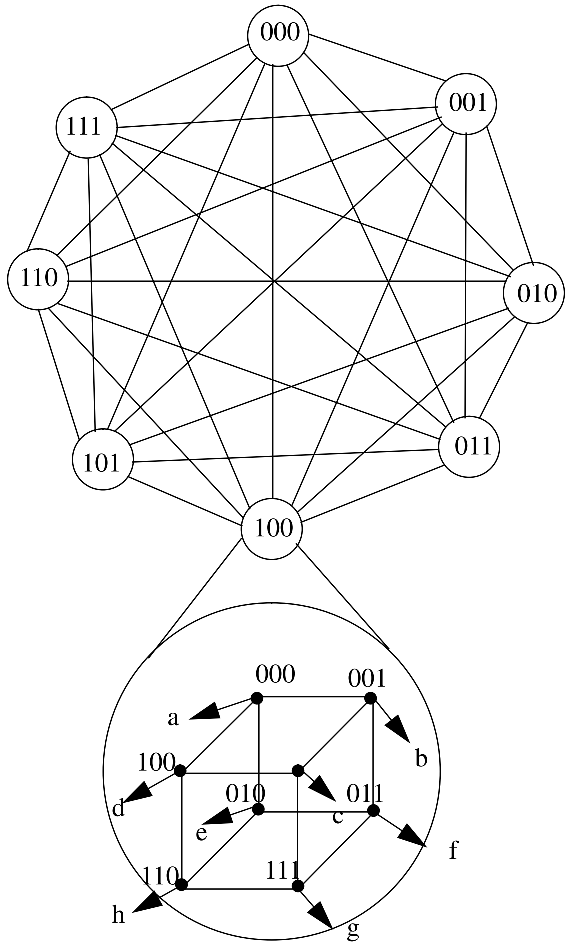

Figure 2. A hierarchical hypercube network (HNN)of size 64.

$T(r, 2)$. The general case will be handled later on.

3.1. Case *m* = 2

The nodes of the binary n-cube are laid out in a square matrix $r \times r$ denoted by

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

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:

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

Notation 1: 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.

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

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

Notation 2: Let

$$T(r) = \begin{bmatrix} t_{00} & t_{01} & \dots & t_{0,\,r-1} \\ t_{10} & t_{11} & \dots & t_{1,\,r-1} \\ \dots & \dots & \dots & \dots \\ t_{r-1,\,0} & t_{r-1,\,1} & \dots & t_{r-1,\,r-1} \end{bmatrix} = \begin{bmatrix} l_0 \\ l_1 \\ \dots \\ l_{r-1} \end{bmatrix}$$

$$= \begin{bmatrix} c_0 & c_1 & \dots & c_{r-1} \end{bmatrix}$$

where

$$l_i = \begin{bmatrix} t_{i0} & t_{i1} & \dots & t_{i,\,r-1} \end{bmatrix} \qquad and \qquad c_j = \begin{bmatrix} t_{0j} \\ t_{1j} \\ \dots \\ t_{r-1,\,j} \end{bmatrix}$$

We use the following notation:

$$T^{N}(r) = T(r)$$

$$T^{LR}(r) = \begin{bmatrix} c_{r-1} & \dots & c_1 & c_0 \end{bmatrix}$$

$$T^{UD}(r) = \begin{bmatrix} l_{r-1} \\ \dots \\ l_1 \\ l_0 \end{bmatrix}$$

$$T^{T}(r) = \begin{bmatrix} T(r) \end{bmatrix}^{T}$$

The superscripts N, LR, UD, and T of the matrices denote normal, left-right, up-down, and transpose, respectively.

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

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

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

Theorem 1: Let G be the r-ary 2-cube, with $r = 2^s$ and let H be the binary n-cube, with $n = 2\log r$. With the definition of $T(r)$ and using the following mapping:

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

we have an embedding of G in H of dilation 1, that is, adjacent nodes of G are mapped to adjacent nodes of H.

Proof: By induction on r.

For $r = 2$,

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

The pairs of elements of $T(2)$, horizontal and vertical, are adjacent because they differ in exactly one bit.

Assume the theorem is valid for the case $r/2$. It is not difficult to show that the theorem holds for $T(r)$. By the induction hypothesis, we have in $T(r/2)$ or $T^N(r/2)$,

t_{ij} and $t_{i\pm 1 mod\, r,\, j}$ are adjacent and

t_{ij} and $t_{i,\, j\pm 1 mod\, r}$ are adjacent

This is also true for $T^{LR}(r/2)$, $T^{UD}(r/2)$ and $T^{T}(r/2)$, since the inversion of rows or columns does not alter the adjacency characteristics.

Thus, each one of $T^N(r/2)$, $T^{LR}(r/2)$, $T^{UD}(r/2)$ and

$T^T(r/2)$ forms an $(r/2)$-ary 2-cube. Consider

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

$$= \begin{bmatrix} 1st\ \text{quadrant} & 2nd\ \text{quadrant} \\ 3rd\ \text{quadrant} & 4th\ \text{quadrant} \end{bmatrix}$$

Each of the quadrants constitutes a sub-cube, whose nodes possess the same two most significant bits. These sub-cubes form $(r/2)$-ary 2-cubes. Hence, it suffices to show that the four quadrants are connected through the boundaries (including the top, bottom, left and right boundaries).

Consider the last row of the first quadrant and the first row of the third quadrant. The corresponding elements of these rows differ only in the first bit, since, by definition, the last row of $T^N(r/2)$ is identical to the first row of $T^{UD}(r/2)$. Thus, they are adjacent. Similarly, this holds for the other boundaries. ■

We thus have a method of arranging the nodes of the binary n-cube in the form of an r-ary 2-cube. Figure 3 is obtained directly from the definition of $T(r)$ and it shows $T(8)$. At the center of each quadrant, we represent the two most significant bits of the elements belonging to the quadrant. Each quadrant, in turn, is divided into four quadrants, whose centers contain the two next bits of the elements belonging to them, and so on. In Figure 4, we show the layout of the nodes of a binary 6-cube according to $T(8)$.

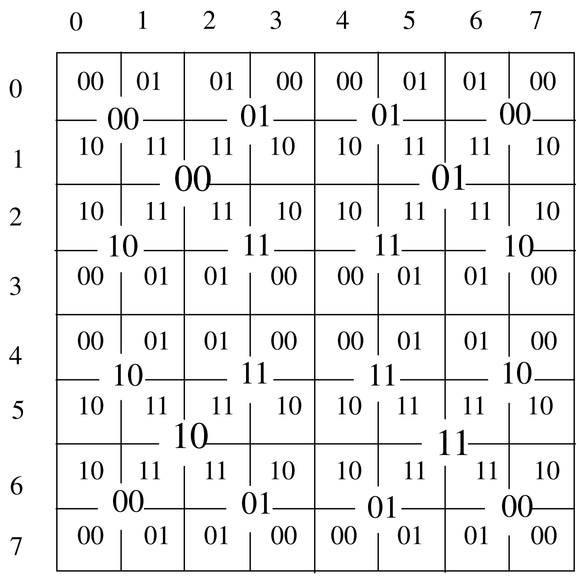

Figure 3. Layout of the nodes of the binary 6-cube by T(8).

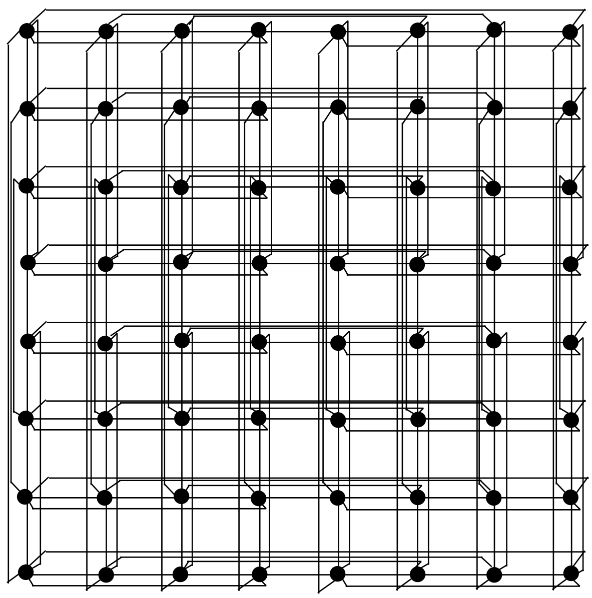

Figure 4. Layout of the binary 6-cube by T(8).

3.2 The general Case: $m \geq 2$

The embedding that we have shown for $m = 2$ can be generalized to any case $m \geq 2$. It is instructive to study the case $m = 3$, which can still be illustrated adequately.

Consider a binary n-cube and a r-ary 3-cube, $r = 2^s$ and $n = 3 \log r$. As before, we have the similar definition of $T(r, 3)$, as shown in Figure 5. $T(r, 3) = (t_{ijk})$ is a 3-dimensional $r \times r \times r$ matrix of binary numbers of 3 log$r$ bits each.

$T(r, 3)$ is obtained from two copies of $T(r, 2)$, one of which preceded by 0 and the other by 1. The additional dimension is represented by an additional superscript, denoted by F or B (Front or Back), that represents the direction of the new index k of t_{ijk}, according to the bit concatenated.

This construction can thus be extended to any m. Instead of using the more suggestive superscripts, N, LR, UD, FB, etc., that cannot be extended for higher dimensions, we can use a different notation in which each superscript corresponds to a dimension and represents a direction considered normal and another its inverse. This

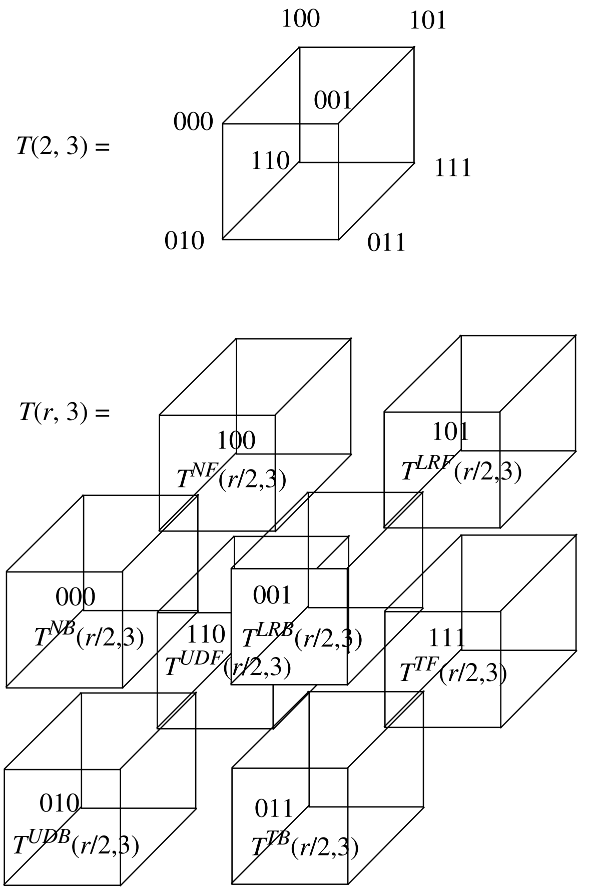

Figure 5. Definition of $T(r, m)$ for the case $m = 3$.

leads to the following theorem.

Theorem 2: Let G be an r-ary m-cube, with $r = 2^s$ and let H be a binary n-cube, with $n = m \log r$. With $T(r, m)$ as constructed above and using the following mapping:

$$node\left(e_0, \ldots, e_{m-1}\right) of\ G \rightarrow node\ t_{e_0, \ldots, e_{m-1}}\ of\ H$$

we will have an embedding of G in H of dilation 1, with adjacent nodes of G mapped to adjacent nodes of H.

This leads to the following corollary.

Corollary 3: If the nodes of a binary n-cube are laid out according to $T(r, m)$, then all sub-cubes formed by the nodes with the first km bits, for $1 \leq k < \log r$, are laid out as $(r/2^k)$-ary m-cubes.

4. Embedding of HHN Into the Hypercube

A straightforward way of embedding a HHN into the hypercube is by mapping the nodes of an HHN to the nodes of the hypercube where their indices are the same, as suggested by some researchers [6], [14]. However, the dilation cost would be $\log_2(N)$ for a network of size N (e.g. nodes 000111 and 111000 in Figure 2 are directly connected in an HHN, but are at a distance $\log_2(64) = 6$ from each other on a hypercube of the same layout. Thus, the dilation cost would be 6 in this case). Hence, we use our embedding method using matrix transformations to enable the embedding of a HHN into a hypercube of the same size with dilation cost of 2 and expansion cost of 1.

In this embedding, we consider as the guest graph G, the HNN, and the host graph H, the binary n-cube. Then, we represent the nodes of G by the pair (i, j) where i is the cluster number and the j is the label of the node within the cluster. Thus, an HNN of size 2^n can be viewed as composed of $2^{n/2}$ rows where each row is a hypercube of size $2^{n/2}$ as shown in Figure 6 where $n = 6$. We denote the links within these rows as horizontal links. Further, these rows are fully interconnected together such that node (i, j) (labeled as ij in the figure) is connected to node (j, i). We denote these inter-row links as *diagonal* links. We can also view an HNN of size 2^n using the following matrix form.

$$T(r) = \left[t_{ij}\right]$$

$$= \begin{bmatrix} 0, 0 & 0, 1 & \ldots & 0, 2^{n/2} - 1 \\ 1, 0 & 1, 1 & \ldots & 1, 2^{n/2} - 1 \\ \ldots & \ldots & \ldots & \ldots \\ 2^{n/2} - 1, 0 & 2^{n/2} - 1, 1 & \ldots & 2^{n/2} - 1, 2^{n/2} - 1 \end{bmatrix}$$

$$HNN = \begin{bmatrix} 1st\ \text{quadrant} & 2nd\ \text{quadrant} \\ 3rd\ \text{quadrant} & 4th\ \text{quadrant} \end{bmatrix}$$

Then, the 1st quadrant and the 4rd quadrant each are an HNN of size 2^{n-2}. Further, each node (i, j) of the 1st quadrant is directly connected to node $(i + 2^{n-2/2}, j)$ of the 2nd quadrant. Also, each node (i, j) of the 4th quadrant is directly connected to node $(i - 2^{n-2/2}, j)$ of the 3rd quadrant. In addition, each node (i, j) of the 2nd quadrant is directly connected to node (j, i) of the 3rd quadrant.

Next, we consider the nodes of H:

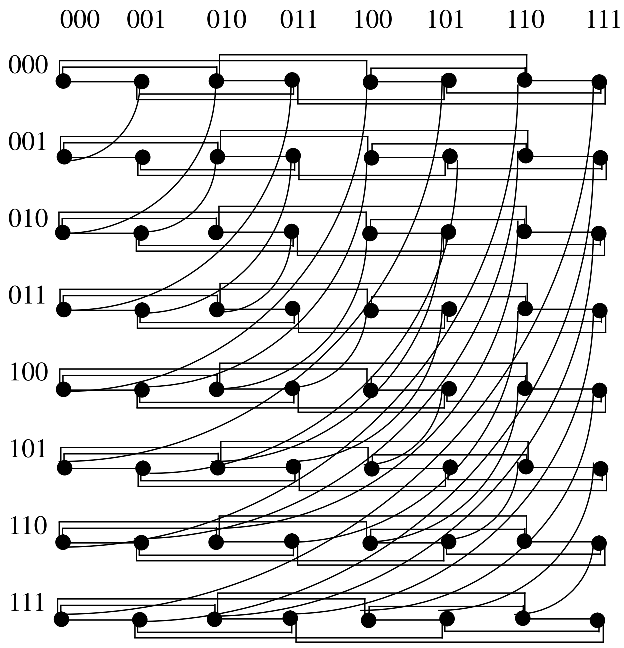

Figure 6. A hierarchical hypercube network (HNN) of size 64.

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

and their binary representation as

$$00 \ldots 00,\ 00 \ldots 01,\ \ldots,\ 11 \ldots 10,\ 11 \ldots 11$$

Let us arrange these numbers in a 2-dimensional matrix $r \times r$, denoted by $W(r, 2)$. The elements of $W(r, 2)$ are binary numbers of 2 log$r$ bits each. For simplicity, write $W(r)$ instead of $W(r, 2)$. The nodes of the binary n-cube are laid out in a square $r \times r$ matrix denoted by

$$W(r) = \left(w_{ij}\right) = \begin{bmatrix} w_{00} & w_{01} & \cdots & w_{0,\,r-1} \\ w_{10} & w_{11} & \cdots & w_{1,\,r-1} \\ \cdots & \cdots & \cdots & \cdots \\ w_{r-1,\,0} & w_{r-1,\,1} & \cdots & w_{r-1,\,r-1} \end{bmatrix}$$

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

$$node\,(i, j)\ of\ G \rightarrow node\ w_{ij}\ of\ H$$

Similar to the case of the r-ary m-cube, we denote by $00W(r) = (w'_{ij})$ where $w'_{ij} = 00w_{ij}$, that is 00 followed by the bits of w_{ij}. Analogously we have $01W(r)$, $10W(r)$ and $11W(r)$. The bits that precede $W(r)$ are said to be concatenated to $W(r)$.

Definition 4: We define $W(r)$ by means of the following recurrence:

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

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

where $W^T(r)$ is the transpose of $W(r)$.

Theorem 5: Let G be an HHN and let H be a binary n-cube of the same size. With the definition of $W(r)$ and using the mapping

$$node\,(i, j)\ \ of\ \ G \rightarrow node\ \ w_{ij}\ \ of\ \ H$$

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 $r = 2$,

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

The pairs of elements of $W(2)$, 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 the theorem is valid for $r/2$. It is not difficult to show that the theorem holds for $W(r)$. The 1st quadrant of $W(r)$, $10W(r/2)$, is used to embed an HNN of size 2^{n-2}. However, by the induction hypothesis, we know that this embedding has a maximum dilation of 2. Analogously, the 4th quadrant of $W(r)$, $11W^{\mathrm{T}}(r/2)$, is also used to embed an HNN of size 2^{n-2}. However, from the topological definition of the binary n-cube, $W(r/2)$ has the same structure as $W^{\mathrm{T}}(r/2)$ because it simply transforms the rows into columns and the columns into rows. Since the rows and the columns are both hypercube connected, the transformation does not change the structure. This is illustrated in Figure 7 for a binary 4-cube. Thus, by the induction hypothesis, we know that this embedding has a maximum dilation of 2.

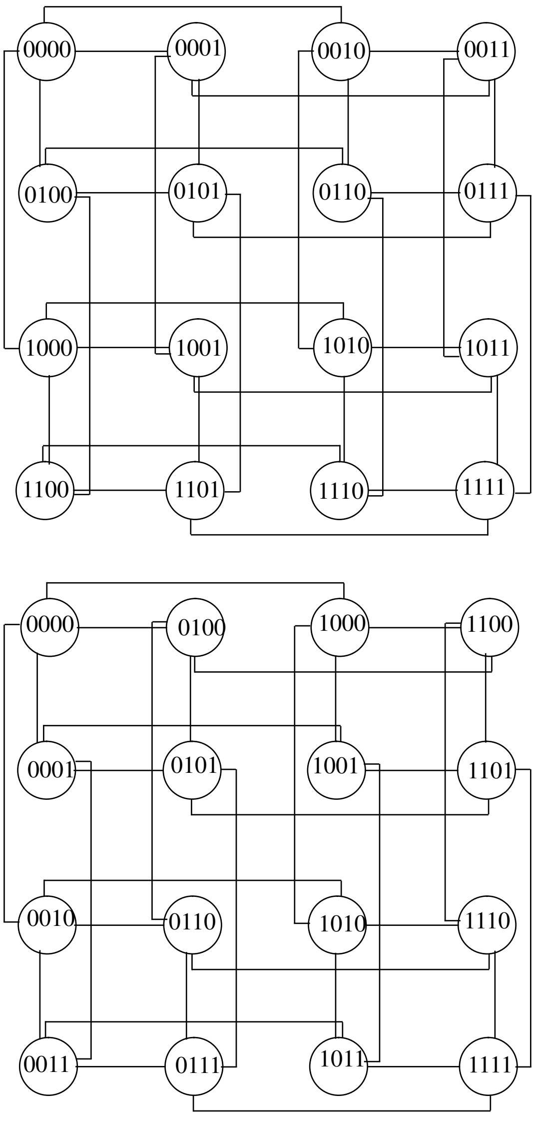

Figure 7. The structure of binary 4-cube and its transpose.

We also know that each node (i, j) of the 1st quadrant is directly connected to node $(i + 2^{n\text{-}2/2}, j)$ of the 2nd quadrant in an HNN. However, each node (i, j) of $01W(r/2)$ is two links away from node $(i + 2^{n\text{-}2/2}, j)$ of $10W(r/2)$ since they differ by only their two most significant bits. Thus, the maximum dilation in this case is 2. The same argument applies for the connection between the 3rd quadrant, $00W^{\mathrm{T}}(r/2)$, and the 4th quadrant, $11W^{\mathrm{T}}(r/2)$.

Lastly, we should find the maximum dilation between each node (i, j) of the 2nd quadrant, $01W(r/2)$, and each node (j, i) of the 3rd quadrant, $00W^{\mathrm{T}}(r/2)$, since these nodes are directly connected by diagonal links in an HNN. This is equivalent to finding the maximum distance between each node (i, j) of $01W(r/2)$ and each node (i, j) of $00W(r/2)$. This maximum distance is just one. Hence, $W(r)$ can embed an HNN with a maximum dilation of 2. ■

Thus, we have found a method of arranging the nodes of the binary n-cube to optimally embed an HNN. Figure 8 is obtained directly from the definition of $W(r)$ and it shows $W(8)$. At the center of each quadrant, we represent the first two bits of the elements belonging to the quadrant. Each quadrant, in turn, is divided into four quadrants, whose centers contain the next two bits of the elements belonging to them, and so on.

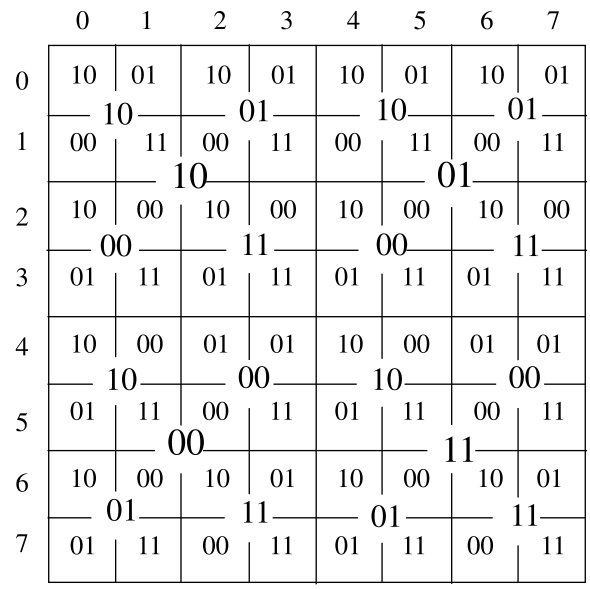

Figure 8. Layout of the nodes of the binary 6-cube by $W(8)$.

Consequently, a HHN can be embedded into a binary hypercube with a maximum dilation of 2. This is the *optimal* case because any embedding of the HHN into the hypercube must have a maximum dilation greater than 1 since it contains odd cycles [15], [18]. 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 6: The embedding of an HHN into the binary hypercube has maximum dilation equal to 2 and expansion cost equal to 1, and is optimal.

5. Conclusion

In this paper, we considered the problem of embedding *r*-ary *m*-cubes and hierarchical networks into the binary *n*-cube. Our proposed method of embedding *r*-ary *m*-cubes is based on recursive folding which is different from the traditional methods using reflected Gray code. This method has the new property that once the nodes of the binary *n*-cube are laid out according to our matrix transformations, then all its sub-cubes formed by the nodes of the same initial *km* bits also form smaller *m*-cubes. This property does not hold using the traditional reflected Gray code method. This can be very helpful in designing divide-and-conquer algorithms and parallel task placement. Thus, it constitutes a useful tool for the design of parallel algorithms for the hypercube.

We also used the embedding method with matrix transformation to optimally embed a class of hierarchical networks into the hypercube. We obtained a dilation of 2 and an expansion cost of 1. 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*. Thus, using our embedding method, 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 hypercube's rich architectural characteristics.

References

[1] S. G. Akl. *The Design And Analysis of Parallel Algorithms*. Prentice Hall, 1991.

[2] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

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

[4] M. Y. Chan, "Embedding of Grids Into Optimal Hypercubes," *SIAM J. Comp*., pp. 834-864, 1991.

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

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

[7] E. Dekel, D. Nassimi, and S. Sahni, "Parallel Matrix and Graph Algorithms," *SIAM J. Comp*., pp. 657-675, 1981.

[8] K. Ghose and K. R. Desai, "The HCN: A Versatile Interconnection Network Based on Cubes," *International Conference on Parallel Processing*, pp. 426-435, 1991.

[9] M. Hamdi and R. W. Hall, "An Efficient Class of Interconnection Networks for Parallel Computing," *The Computer Journal,* Vol. 37, No. 2, pp. 206-218, 1994.

[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] K. Hwang and J. Ghosh, "Hypernet: A Communication Efficient Architecture for Constructing Massively Parallel Computers," *IEEE Trans. Comp*., pp. 1450-1466, 1987.

[12] 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.

[13] 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.

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

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

[16] F. T. Leighton. I*ntroduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufman, 1993.

[17] D. Nassimi and S. Sahni, "Parallel Permutation and Sorting Algorithms and a New Generalized Connection Network," *Journal of the ACM*, pp. 642-667, 1982.

[18] Y. Saad and M. H. Schultz, "Topological Properties of Hypercubes," *IEEE Trans. Comp*., pp. 867-872, 1988.

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

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