© European Journal of Scientific Research, Vol 10, No 2, 2005

EMBEDDING LINEAR ARRAY NETWORK INTO THE TREE-HYPERCUBE NETWORK

Qatawneh Mohammed
Department of Computer science
Faculty of Science and Information Technology
Al-Zaytoonah University of Jordan
Jordan-Amman

ABSTRACT

Graph embedding or graph mapping is an important problem in interconnection networks. A good mapping is said to exist when adjacent processors in the guest network are mapped to reasonably close processors in the host network (i.e. small dilation) and when the paths between adjacent processors in the guest network are chosen in such a way that the congestion at each host node and across each host edge is moderately small (i.e. small node- and edge-congestion). In the case of mapping guest networks onto smaller hosts, the processors of the host have to be assigned to about the same number of processes from the guest (i.e. small load-factor). In this paper an approach for embedding linear array onto Tree-Hypercubes networks is proposed.

INTRODUCTION

Many network issues such as network cost, communication cost, routing, partitioning and embedding have received much attention in recent years. An important feature of an interconnection network is its ability to efficiently simulate programs written for other architectures. This paper is to investigate embedding a linear array onto Tree-Hypercube Network. Tree–Hypercubes TH(s,d) network combines the advantages of both tree and hypercubes and avoids their shortcomings [1][2]. A TH(s,d) network is constructed by taking a full tree of degree s (s is a power of 2) and of depth d. Levels of the tree are numbered 0,.., d. Each level i has s^i nodes representing processing elements as a hypercube. Thus, nodes at level i constitute (i log s)-cube. At each level nodes are labeled in binary from 0 to s^k-1, where k= level i. Each node in TH is identified by a pair (L, X) of its level number L and its cube address X. The total number of nodes in TH(s,d) is N= $(s^{d+1}-1)/(s-1)$ Figures 1 and 2 show two Tree-Hypercubes TH(2,2) and TH(2,3), respectively.

PRELIMINARIES

Definition 1

An embedding of a graph G=(V,E) into a graph G' = (V', E') is a function Φ from V to V'.

Definition 2

Let Φ be the function that embeds graph G = (V,E) onto graph G' = (V',E'). The dilation of the embedding is defined as follows: dil(Φ) = max {dist(Φ(u),Φ(v)) | (u,v) €E} where dist(a,b) is the distance between vertices a and b in G'.

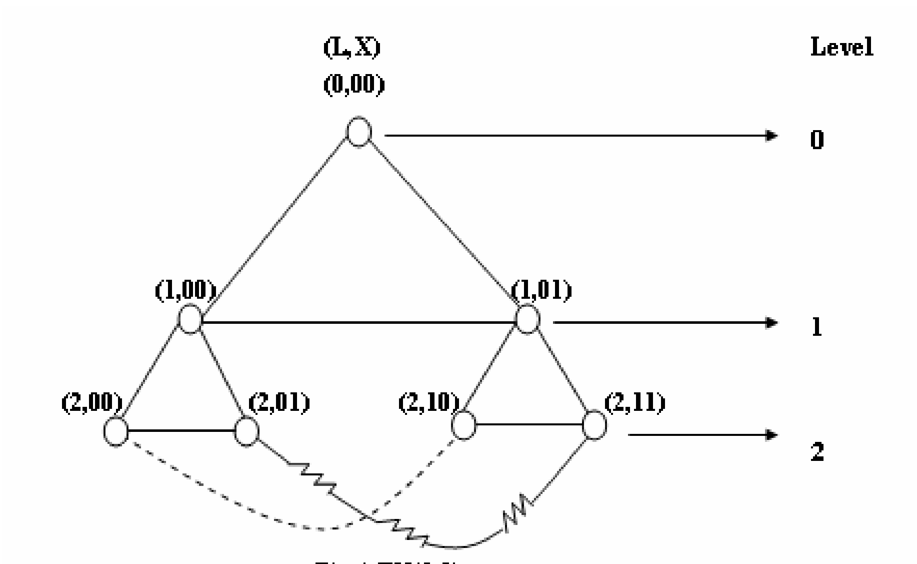

Fig. 1 TH(2,2)

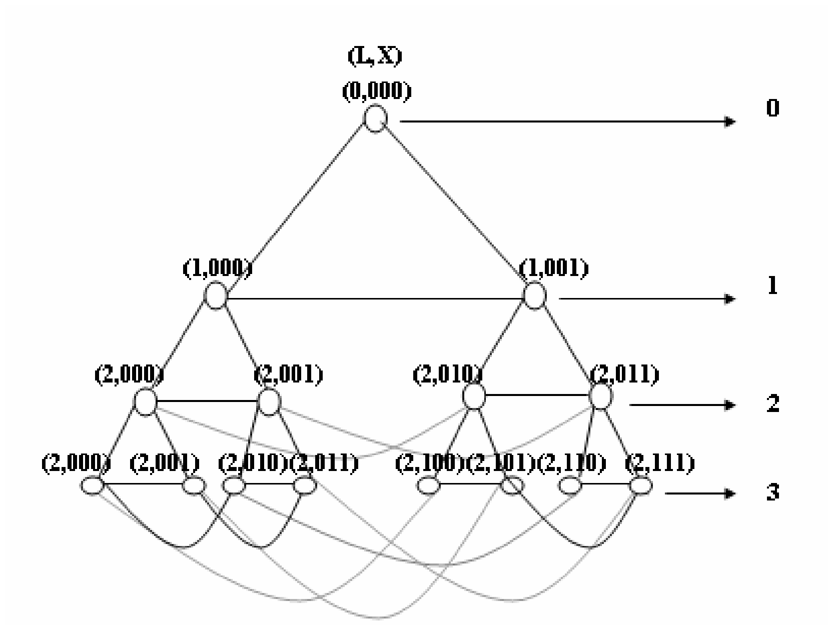

Fig. 2 TH(2,3)

EMBEDDING LINEAR ARRAY ONTO TREE-HYPERCUBES NETWORKS.

Arrays are one of the most natural (data or process) structures for many applications. Linear array of 2^{d+1}-1 processors (where d=0,1,2,,3,…,n) can be embedded into an Tree-Hypercubes TH(s,d) by mapping processor G onto processor G'(L,X) of the Tree-Hypercubes by two steps:

First step: As hypercube (at each level of tree-hypercubes, because each level i in tree-hypercubes has s^i nodes representing processing elements as a hypercube) nodes either in natural order or in Binary-Reflected Gray Code (BRGC) order. It has been noted that many algorithms use the rotation (or cyclic shift) in a natural way as a primitive step [3] [4] [5]. Hence, efficient implementation of rotations on the hypercube network is vital to the performance of hypercube computers. Routing algorithms for rotations in the natural order embedding have been studied in [7] and those for rotations in the BRGC embedding have been studied in [3] [6] [9]. The BRGC embedding has the property that any two adjacent numbers are mapped to neighboring nodes that can communicate directly through a link. Therefore, it uses only one link to route a message from any node to its destination node when the rotation distance is one. In this case, it is superior to the natural order embedding.

An embedding of a linear array onto each level of tree-hypercube is a mapping of its linear array elements to hypercube nodes such that each linear array element is mapped to a distinct node. The natural order embedding maps linear array element *i* to the node whose binary string is the binary encoding of *i*. On the other hand, the binary-reflected Gray code embedding is based on the *n*-bit Binary-Reflected Gray Code (BRGC) [7][8], denoted by *Gn*. It is a sequence of 2*n*-bit binary strings defined recursively as follows:

- $G_1=\{0,1\}$,
- $G_n=\{0G_{n-1},1G_{n-1}^R\}$, where
  - $0G_i$ and $1G_i$ denotes prefixing each element of G_i with bit *0* and *1*, respectively,
  - G_i^R is sequence G_i in reversed order.

As an example, *G*2 = [00, 01, 11, 10] and *G*3= [000, 001, 011, 010, 110, 111, 101, 100]. Let *Gn* (*i*) represents the (*i*+1)th *n*-bit string in *Gn*. The BRGC embedding of an *n*-dimensional hypercube maps linear array element *i* to the node whose binary string is *Gn*(*i*). It is easy to verify that any two adjacent strings in *Gn* differ in exactly one bit. Hence, two adjacent numbers will be mapped onto neighboring nodes that can communicate directly through a link. Note that, according to the definition, *G*(*i*) is undefined if *i* ≥*N*. To simplify notations, we may use *G*(*i*) in place of *G*(*i* mod *N*) for short when *i* ≥*N*. Let the binary encoding of *i*, 0 ≤*i* ≤2*n*-1, be (*inin*-1. . . *i*1*i*0) and the BRGC encoding of *i* be *Gn*(*i*) = (*gngn*-1. . .*g*2*g*1). As noted in [15], the BRGC encoding can also be defined as *gj* = (*ij*+*ij*-1) mod 2 and, conversely, *ij*= (–*nk*=*j*+1*gk*) mod 2. According to the encoding, we can derive the following lemma and corollary.

Lemma 1: For an *n*-dimensional BRGC hypercube and positive integers *m*, *k* and *r*, where 1 ≤*k* ≤*n*-1and 0 ≤*r* ≤2*k*-1, the following equations hold.*Gn*(2*k*(2*m*)+*r*) = *Gn*-*k*(2*m*)*Gk*(*r*)*Gn*(2*k*(2*m*+1)+*r*) = *Gn*-*k*(2*m*+1)*Gk*(2*k*-1-*r*)

Corollary 1: For an *n*-dimensional BRGC hypercube and any positive integer *m*, the following equations hold.$Gn(4m)$ = Gn-1($2m$)0Gn($4m$+1) = Gn-1($2m$)1Gn($4m$+2) = Gn-1($2m$+1)1Gn($4m$+3) = Gn-1($2m$+1)0.

Second step: As tree nodes as follow:

```
For i = 0      (node at level 0)
    Connect the root node (L,X) = (0,00) to the children node by adding 1 to the L and
    shift X left one bit {0+1,00)= (1,00)
      For i=1 to d-1
        Connect the end node of linear array at this level {in this case the end node is
        (1,01)}to the children node by adding 1 to the L and shift X left one bit,
          (1+1,10)=(2,10).
```

Figures 3 and 4 show an embedding linear array onto TH(2,2) and TH(2,3)

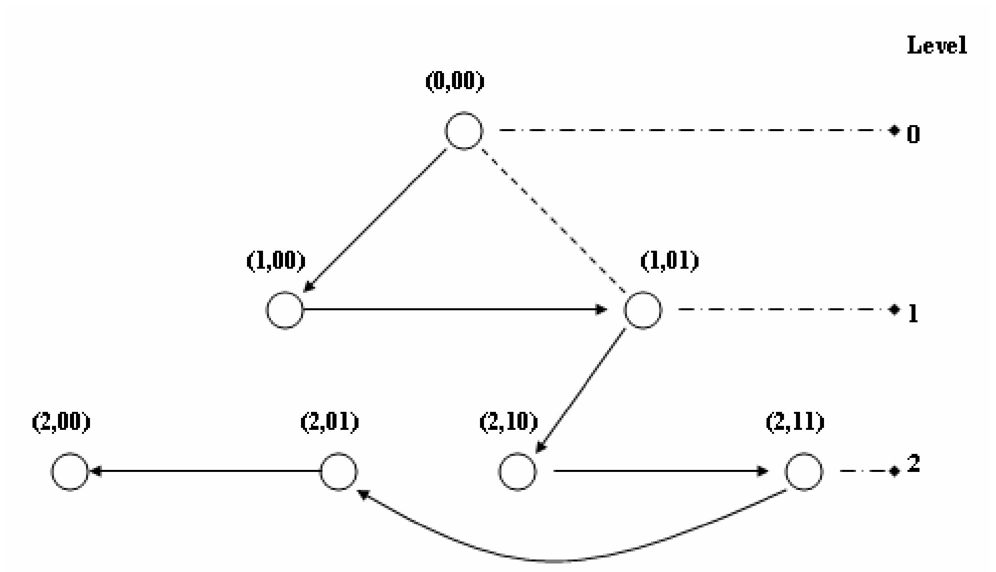

Fig. 3 Embedding linear array onto TH(2,2)

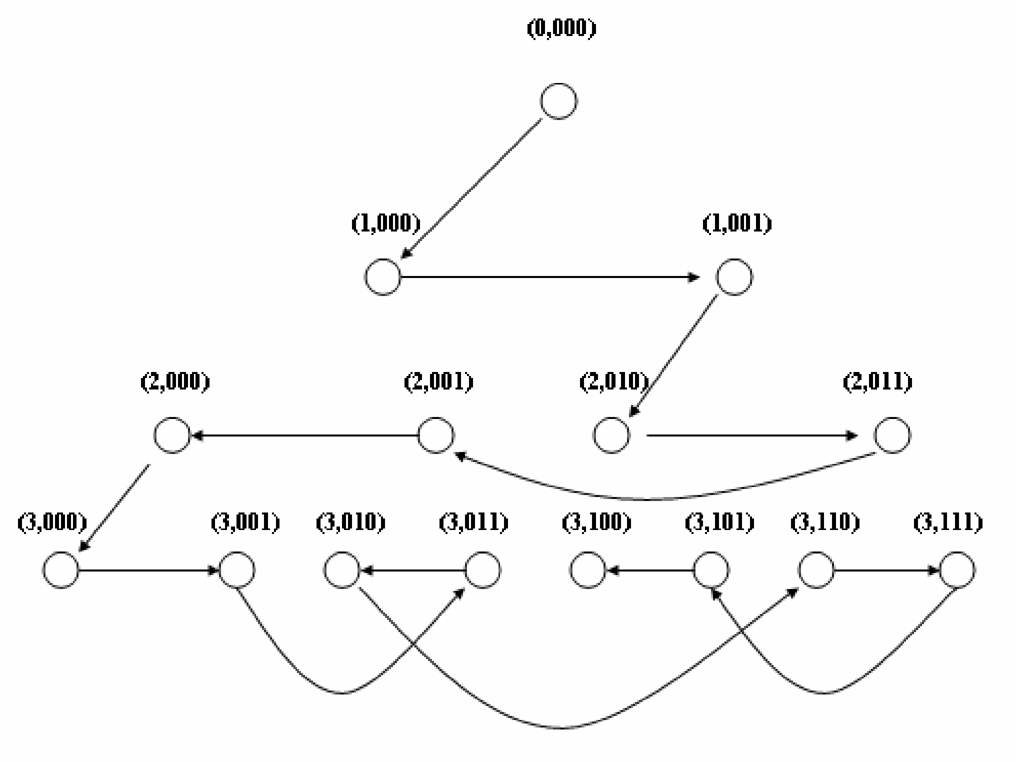

Fig. 4 Embedding linear array onto TH(2,3)

CONCLUSION

In this paper, we have proposed a new and easy approach for embedding a linear array onto tree-hypercubes networks, using the both tree and hypercubes edges. Tree-hypercubes network which combine the advantages of tree and hypercubes, make it a good candidate for general applications and also for the development of many numerical algorithms.

REFERENCES

[1] M. Al- Omari and H. Abu-Salem, ”Tree-Hypercubes: A Multiprocessor Interconnection Topology,”Abhath Al-Yarmouk, Vol.6,No.2,pp.9-24,1997.

[2]Al-Qatawneh M.Sulieman, “ Adaptive Fault-Tolerant Routing in Tree-Hypercube Multicomputer, Proceedings of the second Conference on Computers and their Applications SCCA, pp.260-265, 1998.

[3] S. L. Johnson, ”Communication efficient basic linear algebra computations on hypercube architectures„„*Journal of Parallel and Distributed Computing*, No. 4, pp. 133 ’172, 1987.

[4] Z. M. Kedem, ”Optimal allocation of Area for single-chip computations„„*SIAM Journal of Computing*, pp. 730 ’ 743, August 1985.

[5] D. Nassimi and S. Sahni, ”An optimal routing algorithm for mesh-connected parallel computers„„*Journal of the Association for Computing Machinery*, Vol. 27, No. 1, pp. 6’29, Jan. 1980.

[6] P. Ouyang and K. V. Palem, ”Very efficient cyclic shifts on hypercubes„„*Proceedings of Symposium on Parallel and Distributed*

Processing, pp. 556 ’ 563, 1991.

[7] S. Ranka and S. Sahni, *Hypercube Algorithms: With Applications to Image Processing and Pattern Recognition*, Springer, New York, 1990.

[8] E. M. Reingold, J. Nievergelt and N. Deo, *Combinatorial Algorithms: Theory and Practice*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[9] C.-M. Wang, ”A new routing algorithm for cyclic shifts on BRGC hypercubes„„*Information Processing Letters*, Vol. 49, No. 4, pp. 165 ’ 169, Feb. 1994.