Embedding Hex-Cells into Tree-Hypercube Networks

Awad Ibrahim Qatawneh [1]
[1] **Computer and Information Department, Jazan University**
Jazan, Saudi Arabia

Abstract

Graph embedding or graph mapping is an important aspect for interconnection networks used for communication between processors in parallel systems. Some parallel algorithms use communication structures which can be represented by hex-cells. In order to run these algorithms on a tree-hypercube multiprocessor system, without changing the current topology and the running application, their communication graphs need to be embedded into tree- hypercube. In this research, we have designed an algorithm for embedding hex-cells of n nodes into tree-hypercube TH(2,d) where d >= 2. The embedding has dilation one, congestion one, and expansion 1.1. In the algorithm an embedding of irregular shape of hex-cells into tree-hypercube TH(2,d) where d >= 2 is performed.

Keywords: Graph Embedding, Hex-Cells, Tree-Hypercube networks, Parallel Systems, Dilation, Congestion, Expansion.

1. Introduction

Parallel computing is a form of computing in which many instructions are carried out simultaneously [12]. Parallel computing operates on the principle that large problems can almost always be divided into smaller ones, which may be carried out concurrently ("in parallel"). Parallel computing exists in several different forms: bit-level parallelism, instruction level parallelism, data parallelism, and task parallelism. It has been used for many years, mainly in high performance computing, but interest in it has become greater in recent years due to physical constraints preventing frequency scaling. Parallel computing has recently become the dominant paradigm in computer architecture, mainly in the form of multi core processors [4].

Interconnection networks provide mechanisms for data transfer between processing nodes or between processors and memory modules; they are classified as static and dynamic. Static interconnection networks consist of pointї to-point communication links among processing nodes, they are also referred to as direct networks. Dynamic networks on the other hand, are built using switches and communication links. Communication links are connected to one another dynamically by the switches to establish paths among processing nodes and memory banks. Dynamic networks are also referred to as indirect networks [20].

One of the primary design considerations for a massive multiprocessor system is the topology of the interconnection network used for communication between processors. Some of the important interconnection networks which have been studied so far are hypercube [18], meshes, trees, mesh-of-trees [27], tree-hypercube [19], hex-cell [24], and pyramids. tree-hypercube is one of the popular general purpose networks due to its small degree, short diameter, symmetry, optimal fault-tolerant properties, embedding of other networks, and their ability of implementing fast algorithms [13] [19].

One of the important properties of any general interconnection network is that it should be able to embed other interconnection networks with low overheads. This attribute makes it a good candidate for a topology underlying a general-purpose parallel machine [7]. The problem of mapping other interconnection topologies into tree-hypercube network has not received much attention from researchers [26].

Topology embedding is an important aspect in parallel computing for performing mapping techniques between network topologies. Such that each processing element (node or vertex) and edge in the a graph G = (V,E) that represents the guest topology is mapped into nodes and edges of graph G' = (V',E') which represents the host graph. This mapping has two advantages; the first one is that it reduces the amount of time processes spend interacting with each other, and the second is that it reduces the total amount of time some processes are idle while the others are engaged in performing some tasks. This improves the performance of communication between processing nodes, and gets over the problem of congesting communication. Another main advantage for embedding, is that if a graph G is mapped into G', then G' can simulate the behavior of G with less overhead [12] of executing tasks in parallel.

In our research, we will discuss the problem of embedding hex-cell static networks into tree-hypercube network.

2. Related Work

2.1 Embedding Topologies into Hypercubes

In this section, we present some embeddings of different topologies into the hypercube networks; these different topologies include arbitrary binary trees, balanced binary trees, ring, and mapping of meshes on star graphs. In [17] embeddings in hypercubes [1], stated that one important aspect of efficient use of a hypercube computer to solve a given problem is the assignment of subtasks to processors in such a way that the communication overhead is low. The subtasks and their inter-communication requirements can be modeled by a graph, and the assignment of subtasks to processors viewed as an embedding of the task graph into the graph of the hypercube network. We survey the known results concerning such embeddings, including expansion/dilation tradeoffs for general graphs, embeddings of meshes and trees, packings of multiple copies of a graph, the complexity of finding good embeddings, and critical graphs which are minimal with respect to some property.

In [14] the embedding of hierarchical hypercube networks [5] [11] into the hypercube was presented. And states that 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 architecture. This paper describes a new embedding method, based on matrix transformations, for optimally embedding hierarchical hypercube networks (HHNs) into the hypercube (binary n-cube). Thus, this embedding method has practical importance in enhancing the capabilities and extending the usefulness of the hypercube, since hierarchical hypercube networks have proven to be very cost-effective for a wide range of applications**.** On mapping n-dimensional meshes on a star graph of degree n an algorithm developed in[22], the mapping has expansion 1 and dilation 3. This shows that an *n*-degree star graph can efficiently simulate an *n*-dimensional mesh. The mapping in this research is based on performing a sequence of exchanges along each dimension i of the mesh. On the other hand, embedding binary trees into hypercubes [9], states that hypercubes are known to be able to simulate other structures such as grids and binary trees. In this paper, an embedding of arbitrary binary trees into a hypercube with expansion 1 and dilation no more than 2 is proposed, and embeddings with expansion 2 and dilation 1.

2.2 Embedding Topologies into Tree-Hypercube Networks

Here, we introduce the mapping of Linear Array network into tree-hypercube [21], and its extension which is the mapping of a ring into a tree-hypercube [3]. The embedding of Linear Array network into tree-hypercube is presented in [21]. This research shows that graph embedding exists 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 way that the congestion of each host node and across each host edge is moderately small (i.e. small congestion). An approach for embedding linear array into tree-hypercube is proposed in that paper. 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,é ,n) can be embedded into a tree-hypercube TH (s,d) by mapping processor G onto processor Gô(L,X) of the tree-hypercube by two steps: In the first step, the mapping is performed as a hypercube (at each level of the tree-hypercube, because each level i in the tree-hypercube has s^i nodes representing processing elements as a hypercube) nodes that are represented in BRGC order. 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. An embedding of a linear array into 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. In the second step of mapping, the mapping is performed as tree nodes [21].

An algorithm for mapping a ring into tree-hypercube has been proposed in [3]. In that paper, the authors revealed that TH (s,d) has the ability to implement algorithms that use the communication pattern of linear array ring. They also showed that the tree-hypercube which combines the advantages of both trees and hypercubes, can emulate the topology of a ring, and also other topologies such as linear arrays, trees, binary arrays [15] [16] [29], meshes, and the embedding of fault-free cycles in crossed cubes with conditional link faults. In [8], embedding of binary trees into hypercubes is described, the method is based on an iterative embedding of binary trees into their graphs, and presents a class of binary trees, that allow a dilation two embedding into their optimal hypercubes. Then, concludes that there exist an embedding for an arbitrary binary tree [2]. An embedding of combinational circuits in hypercubes was considered in [23], such that the nodes of the scheme go into vertices of the hypercube and bundles of arcs go into similar bundles or the so-called transition trees of the hypercube having no common internal vertices. This embedding was considered since that in the recent years, it has become popular to realize Boolean functions by combinational circuits. In many cases, the further use of the scheme constructed requires its geometric realization, i.e., a certain embedding in one or another specific geometric structure. The role of such structure is often played by the unit *n*-dimensional cube[23]. In [25], the embedding of fault-free cycles in crossed cubes with conditional link faults, the researcher states that the crossed cube, which is a variation of the hypercube, possesses some properties that are superior to those of the hypercube. It is shown that with the assumption of each node incident with at least two fault-free links, an *n*-dimensional crossed cube with up to 2*n*-5 link faults can embed, with dilation one, fault-free cycles of lengths ranging from 4 to 2*n*. The assumption is meaningful, for its occurrence probability is very close to 1, and the result is optimal with respect to the number of link faults tolerated [6] ,[28].

3. Embedding Hex-Cells into Tree-Hypercubes Networks

Hex-Cell is a new interconnection topology, which is suitable for large parallel computers, and can connect massive number of nodes with 3 links per node [24]. It has the ability to efficiently simulate programs written for architectures such as linear arrays, rings, and meshes. The hex-cell topology can easily embed these structures into it. Hex-Cell can be embedded into tree-hypercube by mapping nodes of the hex-cell into nodes of the tree-hypercube, and edges of the hex-cell onto edges of the tree-hypercube; without having additional number of edges (i.e. dilation of one) and with lower expansion and congestion as possible. Since the smaller dilation of mapping, the shorter communication delay that the host graph (tree-hypercube) simulates the guest graph (hex-cell), and the smaller the expansion of mapping, the more efficient the processor utilization that the host graph (Tree-Hypercube) simulates the guest graph (hex-cell) [10]. In the next section we present the algorithm for mapping hex-cells into tree-hypercube TH (2,d).

3.1 The Proposed Algorithm for Embedding Hex-Cells into Tree-Hypercube TH(2,d).

We have designed an algorithm for mapping hex-cells of n nodes into tree-hypercube TH(2,d); such that every hex-cell with six nodes and six edges is mapped onto nodes and edges of the tree-hypercube TH(2,d) where d >= 2. The mapping has dilation one, congestion one, and expansion 1.1,[30] Algorithm 3.1 for embedding hex-cells into tree-hypercube TH(2,d), where d >= 2.

```
int  d,   // depth of Tree-Hypercube
if ( d >= 2 )
{
   For  i = 1  to  (d - 1)        // each level of the tree-hypercube
      For  j = 1  to  $2^i$  step 2     // each node at that level of the tree-hypercube.

            Node (i ,  j-1) in TH = Node 1 in HC.
            Node (i ,  j) in TH = Node 2 in HC.
            Node (i+1, 2*(j-1) + 3) in TH = Node 3 in HC
            Node (i+1, 2*(j-1) +2) in TH= Node 4 in HC.
            Node (i+1, 2*(j-1)) in TH =  Node 5 in HC.
            Node (i+1, 2*(j-1) + 1) in TH = Node 6 in HC.
            Connect node 1 with node 6 through a link.
      Next j
   Next i
}
```

3.2 Examples on the Embedding Algorithm 3.1

Example 1: embedding 1 Hex-Cell (Fig. 2) into Tree-Hypercube TH(2,2); since d=2, with respect to the algorithm nodes of the Hex-Cell will be mapped as the following,[30], Fig. 1 illustrates the example:

node (1,0) in TH = Node 1 in HC
node (1,1) in TH = Node 2 in HC
node (2,3) in TH = Node 3 in HC
node (2,2) in TH = Node 4 in HC
node (2,0) in TH = Node 5 in HC
node (2,1) in TH = Node 6 in HC
Connect Node 1 with node 6 through a link.

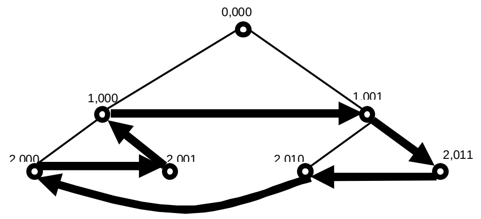

Fig. 1 Embedding One Hex-Cell into Tree-Hypercube TH(2, 2).

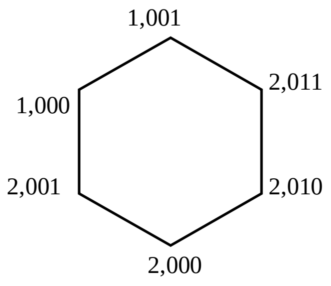

Fig. 2: 1 Hex-Cell.

Example 2: embedding 3 Hex-Cells (Fig. 4) into Tree-Hypercube TH(2,3); since d=3, with respect to the algorithm nodes of the Hex-Cell will be mapped as the following, Fig. 3 illustrates the example:

At First Iteration (i = 1):
Node (1,0) in TH = Node 1 in HC
Node (1,1) in TH = Node 2 in HC
Node (2,3) in TH = Node 3 in HC
Node (2,2) in TH = Node 4 in HC
Node (2,0) in TH = Node 5 in HC
Node (2,1) in TH = Node 6 in HC
Connect Node 1 with node 6 through a link.
At Second Iteration (i = 2):
Node (2,0) in TH = Node 1 in HC
Node (2,1) in TH = Node 2 in HC
Node (3,3) in TH = Node 3 in HC
Node (3,2) in TH = Node 4 in HC
Node (3,0) in TH = Node 5 in HC
Node (3,1) in HC = Node 6 in HC
Connect Node 1 with node 6 through a link.

Node (2,2) in TH = Node 1 in HC
Node (2,3) in TH = Node 2 in HC
Node (3,7) in TH = Node 3 in HC
Node (3,6) in TH = Node 4 in HC
Node (3,4) in TH = Node 5 in HC
Node (3,5) in TH = Node 6 in HC
Connect Node 1 with node 6 through a link.

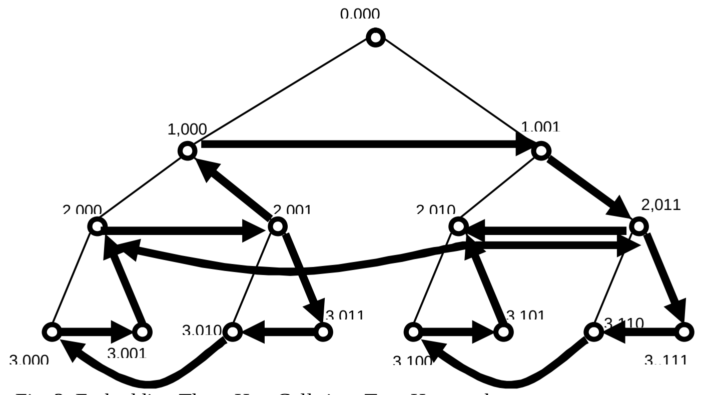

Fig. 3 Embedding Three Hex-Cells into Tree-Hypercube TH (2,3)

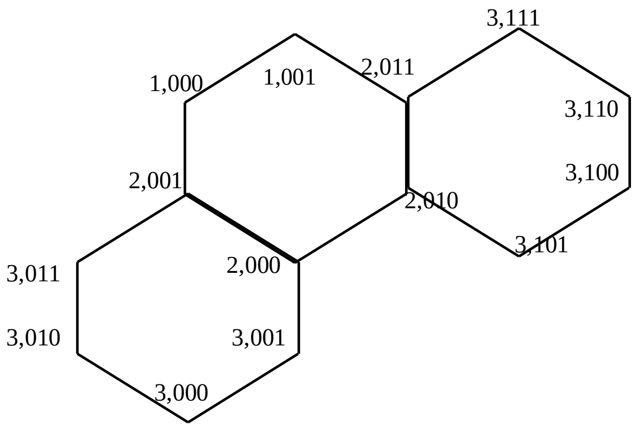

Fig. 4: 3 Hex-Cells.

Example 3: embedding of seven hex-cells (Fig. 6) into Tree-Hypercube TH(2,4); since d=4, with respect to the algorithm nodes of the Hex-Cell will be mapped as the following, Fig. 5 illustrates the example:

At First Iteration (i = 1):
Node (1,0) in TH = Node 1 in HC
Node (1,1) in TH = Node 2 in HC
Node (2,3) in TH = Node 3 in HC
Node (2,2) in TH = Node 4 in HC
Node (2,0) in TH = Node 5 in HC
Node (2,1) in TH = Node 6 in HC
Connect Node 1 with node 6 through a link.

At Second Iteration (i = 2):
Node (2,0) in TH = Node 1 in HC
Node (2,1) in TH = Node 2 in HC
Node (3,3) in TH = Node 3 in HC
Node (3,2) in TH = Node 4 in HC
Node (3,0) in TH = Node 5 in HC
Node (3,1) in HC = Node 6 in HC
Connect Node 1 with node 6 through a link.

Node (2,2) in TH = Node 1 in HC
Node (2,3) in TH = Node 2 in HC
Node (3,7) in TH = Node 3 in HC
Node (3,6) in TH = Node 4 in HC
Node (3,4) in TH = Node 5 in HC
Node (3,5) in TH = Node 6 in HC
Connect Node 1 with node 6 through a link.

At Third Iteration (i = 3)
Node (3,0) in TH = Node 1 in HC
Node (3,1) in TH = Node 2 in HC
Node (4,3) in TH = Node 3 in HC
Node (4,2) in TH = Node 4 in HC
Node (4,0) in TH = Node 5 in HC
Node (4,1) in TH = Node 6 in HC
Connect Node 1 with node 6 through a link.

Node (3,2) in TH = Node 1 in HC
Node (3,3) in TH = Node 2 in HC
Node (4,7) in TH = Node 3 in HC
Node (4,6) in TH = Node 4 in HC
Node (4,4) in TH = Node 5 in HC
Node (4,5) in TH = Node 6 in HC
Connect Node 1 with node 6 through a link.

Node (3,4) in TH = Node 1 in HC
Node (3,5) in TH = Node 2 in HC
Node (4,11) in TH = Node 3 in HC
Node (4,10) in TH = Node 4 in HC
Node (4,8) in TH = Node 5 in HC
Node (4,9) in TH = Node 6 in HC
Connect Node 1 with node 6 through a link.

Node (3,6) in TH = Node 1 in HC
Node (3,7) in TH = Node 2 in HC
Node (4,15) in TH = Node 3 in HC
Node (4,14) in TH = Node 4 in HC
Node (4,12) in TH = Node 5 in HC
Node (4,11) in TH = Node 6 in HC
Connect Node 1 with node 6 through a link.

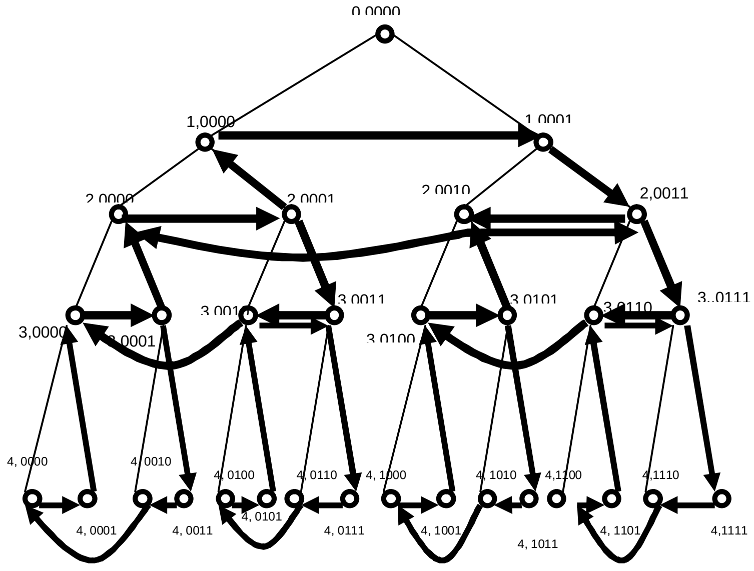

Fig 5:Embedding of seven Hex-Cells into Tree-Hypercube TH(2,4)

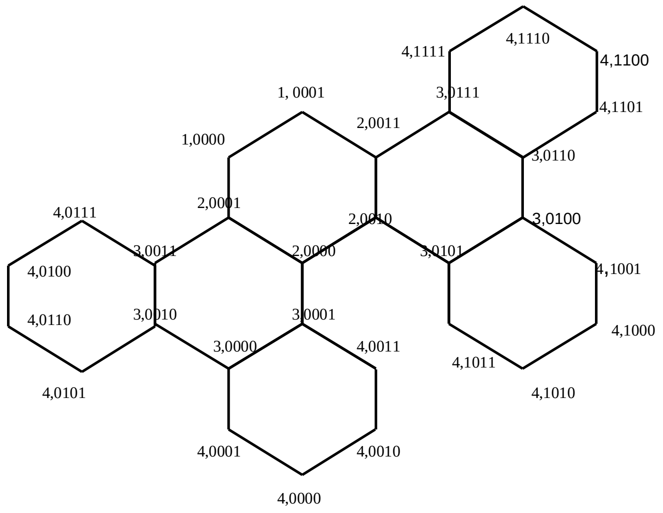

Fig. 6: Seven Hex-Cells

4. Conclusion and Future Work

As a result the embedding algorithm 3.1 [30] performs an embedding of irregular shape of hex-cells into tree-hypercube TH(2,d) where d >= 2 is performed. Embedding of regular shape of hex-cells into tree-hypercube can't be performed, requiring the structure of the tree-hypercube. And so as not to have additional number of links and not to increase the cost. The embedding of hex-cells into tree-hypercube has dilation 1, congestion 1, and expansion 1.1 [30].

For future work, we suggest embedding of hex-cells into other topologies, such as meshes and binary trees, and with lower dilation and expansion as possible. Furthermore, we will consider the embedding of hex-cell with respect to their depth, not depending on the structure of the host topology. In future work, in order to embed the regular shape of hex-cells, we suggest the embedding of hex-cells into hierarchal tree-hypercube, and also by using the required additional number of links.

References

[1] E. Abuelrub, and S. Bettayeb, "Embedding Rings into Faulty Twisted Hypercubes", Computing and Informatics Journal, Vol. 16, No. 4, 1997.

[2] M. Aderhold, andJ. Slack , "Embedding Balanced Binary Trees into the Hypercube", Journal of Information Science and Engineering, Vol. 14, No. 5, pp. 740-752, 1997.

[3] W. Almobaideen,M. Qatawneh , A. Sliet, I. Salah , and S. Al-Sharaeh, "Efficient Mapping Scheme of Ring Topology onto Tree- Hypercubes". Journal of Applied Sciences, Vol. 7, No. 18, pp. 2666-26770, 2007.

[4] K. Asanovic,"The Landscape of Parallel Computing: A Research View", University of California, Berkeley, Technical Report No. UCB/EECS-183, 2006.

[5] D. Bein, W.Bein , and S. Latifti, "An Optimal Embedding of Honeycomb Networks into Hypercubes", Parallel Processing Letters, Vol. 14, No. 3-4, pp 367 375, 2004.

[6] S. Bettayeb , B. Girou , and M. Sudborough , "Embedding star networks into Hypercubes Computers", IEEE Transactions, Vol. 45, No. 2, pp. 186-194, 1996.

[7] H. Y. Chang , and R. J. Chen , "Embedding cycles in IEH graphs", Information Processing Letters, Vol. 64, No. 4, pp. 23-27, 1997.

[8] W. K..Chen , and M. F. Stallmann , "On Embedding Binary Trees into Hypercubes″, Journal of Parallel Distributed Computing, Vol. 24, No. 3, pp. 132-138, 1995.

[9] T. Dvorak , "Dense Sets and Embedding Binary Trees into Hypercubes", Discrete Applied Mathematics Journal, Vol. 155, No. 4, pp. 506-514, 2007.

[10] J.Fan , and X. Jia , "Embedding Meshes into Crossed Cubes", Information Sciences Journal, Vol. 177, No. 15, pp. 3151-3160, 2007.

[11] J. Fan ,X. Jia , and L. Xiaola "Embedding of Cycles in Twisted Cubes with Edge-Pancyclic", Algorithmica Journal, Vol. 51, No. 3, pp. 264-282, 2007.

[12] A. Grama , A. Gupta , G. Karypis , and V. Kumar , ñIntroduction to Parallel Computingò, Addison Wesley, pp. 12-72, 2003.

[13] H. Hamdi, and S. W. Song , ñMatrix Transformations: An Efficient Method for Embedding Networks into the Hypercubeò, IEEE Transactions on Parallel and Distributed Systems, Vol. 8, No. 6, pp. 897-902,1997.

[14] M.Hamdi, and S. W. Song , "Embedding Hierarchical Hypercube Networks into the Hypercube", IEEE Transactions on Parallel and Distributed Systems, Vol. 8, No. 9, pp. 876-883, 1997.

[15] V.Heun, and E. W. Mayr , "Eembedding of Arbitrary Binary Tree into its Optimal Hypercube ", Journal of Algorithms, Vol. 20, No. 4, pp. 431-445, 1996.

[16] C. H. Huang , J. Y. HSIAO, and C. T. LEE R , "Embedding Incomplete Binary Trees into Incomplete Hypercubes", IEE proceedings on Computers and digital techniques, Vol. 145, No.6, pp. 377-384, 1998.

[17] L.Marilynn , and F. Stout , "Embeddings in Hypercubes", Mathematical and Computational ModellingJournal, Vol.11, No. 3, pp. 222-227, 1988.

[18]M.Omari , "Tree-Hypercube: Hierarchal, Partitionable Multiprocessor Interconnection Network", PHD Dissertation, Illinois Institute of Technology, USA, 1992.

[19] M. Omari, andH. Abu-Salem , "Tree-Hypercube: A Multiprocessor Interconnection Topology", Abhath Al-Yarmouk, Vol. 6, No. 2, pp. 9-24, 1998.

[20] L.Peterson , and B. Dave , "Computer Networks, A System Approach", Morgan Kaufmann, Third Edition, 2003.

[21] M. Qatawneh , ñEmbedding Linear Array onto Tree-Hypercube Networkò, European Journal of Scientific Research, Vol. 10, No. 2, pp. 72-77, 2005.

[22] R. Sanjay, J. Wang and N.Yeh. "Embedding Meshes on the Star Graph", IEEE Proceeding Conference on Super Computers, Vol. 12, No.3 , pp. 476- 485, 1990.

[23] O. Sedelev ,"Realization of Boolean functions Combinational Circuits Embedded in the Hypercub Moscow University Computational Mathematics Cybernetics Journal, Vol. 32, No. 1, pp. 44-50, 2008.

[24] A.Sharieh ,M. Qatawneh , W. Almobaideen , and A. Sleit , "Hex- Cell: Modeling, Topological Properties and Routing Algorithm", European Journal of Scientific Research, Vol.22, No. 3,pp. 457-468, 2008.

[25] J. Sheng , H. Hung , and G. Huey , "Embedding Fault-Free Cycles in Crossed Cubes with Conditional Link Faults", Journal of Supercomputing,Vol. 46, No. 2, pp 143-152, 2008.

[26] C. Tsai , "Cycles Embedding in Hypercubes with Node Failures", Information Processing Letters, Vol. 102, No. 6, pp. 242-246, 2007.

[27] K. H. Wang , and F. A. Briggs , ñComputer Architecture and Parallel Processingò, Mc-Graw Hill, 1994.

[28] S. C. Wang ,Y. R. Leu, and S. Y. Keu , "Distributed Fault Tolerant Embedding of Several Topologies into Hypercubes", Journal of Information Science and Engineering, Vol. 20, No.4, pp. 707-732, 2004.

[29] X.Yang ,Y. Tang , and J. Cao , " Embedding Torus in Hexagonal Honeycomb Torus", Computers and Digital Techniques Journal, Vol. 2, No. 2, pp. 86-93, 2008.

[30] A. I. Qatawneh, ñ Embedding Hex-Cells into Tree Hypercube Networks ñ , MSc Thesis, Computer Science Department, Faculty of Information Technology, Middle East University for Graduate Studies, Amman, Jordan, 2009.

Awad Qatawneh is a Full Time Lecturer at The Computer and Information Department at Jazan University, Saudi Arabia. He earned his Bachelorôs degree in Computer Science in 2004 from Mutah University, Jordan, and his Masters in Computer Science in 2009 from Middle East University for Graduate Studies, Jordan.