International Journal of Information Technology and Knowledge Management
July-December 2009, Volume 2, No. 2, pp. 361-364

A CASE STUDY OF FAULT TOLERANT ROUTING MECHANISM FOR TORUS EMBEDDED HYPERCUBE INTERCONNECTION NETWORK FOR PARALLEL ARCHITECTURE

N. Gopalakrishna Kini[*], M. Sathish Kumar[**] & Mruthyunjaya H. S.[***]

This paper describes a fault-tolerant routing mechanism to facilitate data routing in Torus embedded hypercube interconnection network subject to node failures in parallel computing. It is shown that by only using feasible paths routing can be substantially simplified. Though there are algorithms for fault tolerant communication in torus and hypercube networks, there exits no efficient algorithm for the embedded architecture. We present an algorithm to provide an efficient fault tolerant routing mechanism for a (2, 2, 8)-Torus embedded hypercube interconnection network.

Keywords: Hypercube Network, Torus Network, Embedded Network, Torus Embedded Hypercube Network, Fault Tolerance, Fault List, Routing.

1. Introduction

It is quite evident that fault tolerance in highly parallel computers is important for achieving reliable and high performance computing [1]-[3]. Fault tolerance is the ability of an interconnection network to continue operating in presence of single or multiple faulty nodes [4]. As the network size scales up [11]-[13] the probability of processor failure also increases. It is therefore essential to design fault tolerant routing algorithms that allow to route messages between non-faulty nodes in the presence of faulty processors.

The hypercube network's node degree grows logarithmically with number of vertices making it difficult to build scalable architectures [7]-[9]. On the other hand, the torus network supports the scalability as all the nodes are having constant node degree. A better network performance is achieved by embedding the torus and hypercube networks that will give rise to a Torus embedded hypercube network. Such a combination results in a system which can be implemented with small node degree and a reduction in hardware cost per node. Also, a constant node degree results in a system that is scalable without having to modify the individual nodes [4]-[6].

Torus embedded hypercube network, when one or more nodes fail, a large number of available links enable the fault free nodes to continue communicating with other nodes. We consider a (2, 2, 8)-torus embedded hypercube interconnection network for demonstrating our algorithm. Our algorithm gives the optimum path even in the presence of single/multiple faulty nodes in the interconnection network.

2. Architectural Properties of Torus Embedded Hypercube Network

In this section, a brief discussion on embedding of the torus and the hypercube networks is done to obtain the torus embedded hypercube network and more details on this can be found in [4]-[6], [11], [12].

Let $l \times m$ be the size of several concurrent torus networks and N be the number of nodes connected in the hypercube. Nodes with identical positions in the torus networks will form a group of N number of nodes and hence the resultant torus embedded hypercube network having a size of (l, m, N). The nodes in the network can be addressed with three components; row number i and column number j of torus appended with the address of node k of hypercube. Hence, a (l, m, N)–torus embedded hypercube network will have $l \times m \times N$ number of nodes and a node will be addressed as (i, j, k) where $0 \le i < l$, $0 \le j < m$ *and* $0 \le k < N$.

Combining the data routing functions of torus and hypercube will provide with the routing functions of the torus embedded hypercube [2], [11] as in (1)-(5).

$$T_{h1}(i, j, k) = (i, (j + 1) \bmod m, k) \quad (1)$$

$$T_{h2}(i, j, k) = (i, (m + j - 1) \bmod m, k) \quad (2)$$

$$T_{h3}(i, j, k) = ((i + 1) \bmod l, j, k) \quad (3)$$

$$T_{h4}(i, j, k) = ((l + i - 1) \bmod l, j, k) \quad (4)$$

[*] Dept. of Computer Science and Engineering, Manipal Institute of Technology (Manipal University), Manipal, Karnataka, INDIA *E-mail: ng.kini@manipal.edu*

[**] School of EECS, Seoul National University, Seoul, SOUTH KOREA. *E-mail: mskuin@yahoo.com*

[***] Dept. of Electronics and Communication, Manipal Institute of Technology (Manipal University), Manipal, Karnataka, INDIA *E-mail: mruthyu.hs@manipal.edu*

$$T_{C_d}(k_{n-1}...k_d k_{d-1}...k_0) = (k_{n-1}...k_{d+1}\bar{k}_d k_{d-1}...k_0) \quad (5)$$

for d = 0, 1, …, n–1 where k_j for (j = 0 to n–1) is the binary representation of node address k and $n = log_2(N)$ where N is the total number of nodes in the hypercube.

The address of individual node is represented by n-bit binary vector. A link will exist between two nodes where the addresses of these two nodes differ exactly by one bit [7]. For a (2, 2, 8)-torus embedded hypercube network, a node with a five bit address has its left most bit representing row number, the next bit representing column number and the remaining least significant bits representing the address of a node in the hypercube as shown in Figure 1.

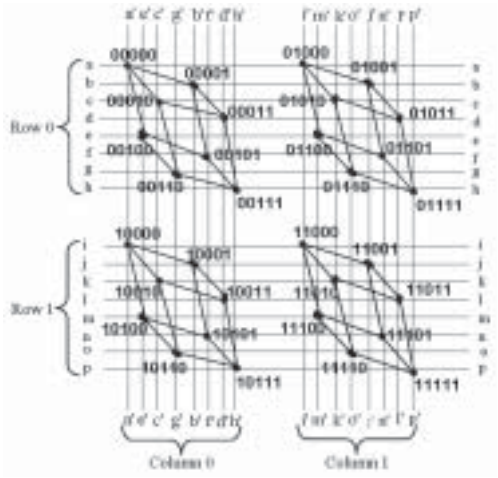

Figure 1: A (2, 2, 8)–Torus Embedded Hypercube Network

The ring connections of row/column of each torus are not shown in Figure1 for simplicity. A wraparound connection is done along each row/column if they have same label as a completion of (2, 2, 8)-torus embedded hypercube network.

3. FAULT TOLERANCE IN TORUS EMBEDDED HYPERCUBE NETWORK

The algorithm developed is explained in this section. This algorithm technique enable to formally prove that even with multiple faulty nodes, the torus embedded hypercube interconnection network can still route the data successfully.

Each node in torus embedded hypercube network is labeled with an n-dimensional binary vector ($x_1 x_2 ... x_{n-1} x_n$), where $x_i \in (0, 1)$ for $1 \le i \le n$. The neighboring nodes of a node Z is different from Z in one bit with reference to their binary representations. In the binary vector the initial bits $x_1 ... x_i$ and $x_{i+1} ... x_j$ are representing the row number and column number respectively where $i < j < n$. Note that, for the (2, 2, 8) network i = 1, j = 2 and n = 5 are considered.

In the initial phase of the algorithm, $x_{j+1} ... x_n$ bits are taken into account by considering $x_1 ... x_i x_{i+1} ... x_j$ bits as don't care combinations. The source node presently referred to as the current node which would like to communicate with the destination will compare each bits of its address $x_{j+1} ... x_n$ starting from left most bit with the corresponding bits of its destination node. If the bits are found to be same we leave current node address as it is and move on to the very next consecutive bit which is at the right of it. If the bits are different then that bit in the current node is changed as that of corresponding bit of destination node to derive a new node.

Now this new node will become the current node for further comparison. This is done till all the bits $x_{j+1} ... x_n$ are processed. We assume that each processor has a *fault list* for recording the faulty/nonfaulty status of its neighboring nodes. The new node derived, if found in the *fault list*, will be ignored; otherwise will be considered as current node and the process of comparison will be continued from thereon. As new nodes are encountered which is not in the *fault list*, they will get added to the list of *routing path*. With this a data routing path is discovered within the cube by ignoring all the faulty nodes with respect to current node [8]-[10].

In the second phase, $x_1 ... x_i x_{i+1} ... x_j$ are taken into account by considering $x_{j+1} ... x_n$ as don't care combinations. The current node which is obtained from the previous phase will now continue this process of comparison of bits of its address starting from left with the corresponding bits of its destination node. If the bits are found to be same we leave current node address as it is and go ahead with the very next consecutive bit.

If the bits are different then that bit in the current node is changed as that of corresponding bit of destination node to derive a new node. Now this new node will become the current node for further comparisons. This is done till all the bits in $x_1 ... x_i x_{i+1} ... x_j$ are processed. Again the new node derived if found in the *fault list* will then be ignored; otherwise considered as current node and the process of comparison will continue. As new nodes are encountered which are not in the *fault list*, they will be added to the list of *routing path*. With this a data routing path is discovered within the concurrent torus by ignoring all the faulty nodes with respect to current node.

4. RESULTS AND DISCUSSION

Table 1 shows the result of the algorithm that has provided the data routing paths in (2, 2, 8)–torus embedded hypercube network. A routing between seven cases of random source and destination nodes are considered for demonstration. An efficient optimal routing path is established between them with reference to the available links as in Figure 1 and

equations (1)-(5). If the network do not have any faulty nodes the algorithm generates dedicated path between any random source and destination nodes as shown in Table 1. It is also proved to be the optimal paths.

Table 1
Results of Fault Free Data Routing Path

No.	*Source Node*	*Destination Node*	*Intermediate Nodes Crossed with Optimal Path*
1	00000	00111	00000→00100→00110→00111→
2	00000	01000	00000→01000
3	00000	01111	00000→00100→00110→00111→ 01111
4	00000	10101	00000→00100→00101→10101→
5	00000	11111	00000→00100→00110→00111→ 10111→11111
6	11111	00000	11111→11011→11001→11000→ 01000→00000
7	01101	00000	01101→01001→01000→00000

In Table 2 we have considered a case with a source node of address 00000 and a destination node of address 01111. In this case the network is analyzed to arrive at fault-tolerant communication. The first row shows the data routing without any faulty nodes. The remaining rows show data routing when there are single/multiple faulty nodes in the network.

In presence of faulty nodes algorithm will find an alternative path between source and destination nodes. Alternative paths found are proved to be the optimal routing in presence of those specific faulty nodes.

Table 2
Source Node 00000 Communicating with Destination 01111

No.	*Introducing Faulty Node / Nodes*	*Intermediate Nodes Crossed with Optimal Path*
1	No Faulty Nodes	00000→00100→00110→00111→ 01111
2	00100	00000→00010→00011→01011→ 01111
3	00110	00000→00100→00101→01101→ 01111
4	00111	00000→00100→00110→01110→ 01111
5	00100, 00010	00000→00001→01001→01101→ 01111
6	00100, 00010, 00001	00000→01000→01100→01110→ 01111
7	00100, 00010, 00001 01000	00000→10000→11000→11100→ 11110→11111→01111
8	00100, 00010, 00001 01000, 11000	00000→10000→10100→10110→ 10111→00111→01111
9	00100, 00010, 00001 01000, 11000m 10000	Source Node get Isolated from the destination as there is no way to establish a communication path.

5. CONCLUSION AND FUTURE WORK

We have developed a data routing algorithm for a (2, 2, 8)–torus embedded hypercube network. The analysis and functioning of the algorithm has been provided with the situation wherein the network is fault free. Also the case is considered to support our algorithm with source node of address 00000 and a destination node of address 01111 communicating each other by considering all combinations of faulty nodes. A node in torus embedded hypercube network can tolerate not more than (n-1) faulty nodes, where n is the node degree and also a non-faulty node will get disconnected when all its n neighbors become faulty. Though the algorithm is for a specific (2, 2, 8) network, we feel that the algorithm can be generalized with some limited modifications.

REFERENCES

[1] K. Hwang, "*Advanced Computer Architecture:* Parallelism, Scalability, Programmability," New York McGraw-Hill, (1993).

[2] Hesham El-Rewini and Mostafa Abd-El-Barr, "Advanced Computer Architecture and Parallel Processing," John Wiley & Sons, Inc., Hoboken, New Jersey, (2005).

[3] J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach," 3rd ed., Morgan Kaufmann, (2005).

[4] Ahmed Louri and Hongki Sung, "An Optical Multi-Mesh Hypercube: A Scalable Optical Interconnection Network for Massively Parallel Computing," *Journal of Lightwave Technology,* **12**(4), (1994) 704-716.

[5] Ahmed Louri, "Optical Interconnection Networks for Scalable High-Performance Parallel Computing Systems," Optical Interconnects Workshop for High Performance Computing Oak Ridge, Tennessee, (1999).

[6] Ahmed Louri and Hongki Sung, "A scalable optical hypercube-based interconnection network for massively parallel computing," Applied optics, **33**(11) (1994).

[7] Kim Jong-Seok, Lee Hyeong-Ok and Heo Yeong-Nam, "Embedding Among HCN (n, n), HFN (n, n) and Hypercube," Proceedings of Eighth International Conference on Parallel and Distributed Systems (ICPADS), 533–540, (2001).

[8] Tze Chiang Lee and John P. Hayes, "A Fault-Tolerant Communication Scheme for Hypercube Computers," *IEEE Transactions on Computers,* **41**(10) 1242-1256, (1992).

[9] Jianer Chen, Iyad A. Kanj and Guojun Wang, "Hypercube Network Fault Tolerance: A Probabilistic Approach," Proceedings of the IEEE International Conference on Parallel Processing (ICPP'02), (2002).

[10] Shih-Chang Wang and Sy-Yen Kuo, "Fault Tolerance in Hyperbus and Hypercube Multiprocessors Using Partitioning Scheme," IEEE International Conference on Parallel and Distributed Systems, 340-347, (1994).

[11] N. Gopalakrishna Kini, M. Sathish Kumar and Mruthyunjaya H. S., "Analysis and Comparison of Torus

Embedded Hypercube Scalable Interconnection Network for Parallel Architecture," *International Journal of Computer Science and Network Security,* **9**(1), 242-247, (2009).

[12] N. Gopalakrishna Kini, M. Sathish Kumar and Mruthyunjaya H. S., "A Torus Embedded Hypercube Scalable Interconnection Network for Parallel Architecture," *IEEE Explore Conference Publications,* 2009, *URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?isnumber=4808969 & arnumber=4809127.*

[13] N. Gopalakrishna Kini, M. Sathish Kumar and Mruthyunjaya H. S., "Design and Comparison of Torus Embedded Hypercube with Mesh Embedded Hypercube Interconnection Network," International Journal of Information Technology and Knowledge Management, **2**(1) 87-90 (2009).