Proceedings of the World Congress on Engineering 2007 Vol I
WCE 2007, July 2 - 4, 2007, London, U.K.

Data Communication and Parallel Computing on Twisted Hypercubes

E. Abuelrub, Department of Computer Science, Zarqa Private University, Jordan

***Abstract*- Massively parallel distributed-memory architectures are receiving increasing attention to meet the increasing demand on processing power. Many topologies have been proposed for interconnecting the processors of distributed computing systems. The hypercube topology has drawn considerable attention due to many of its attractive properties. The appealing properties of the hypercube topology such as vertex and edge symmetry, recursive structure, logarithmic diameter, maximally fault-tolerance, simple routing and broadcasting, and the ability to simulate other interconnection networks with minimum overhead have made it an excellent candidate for many parallel processing applications. Many variations of the hypercube topology have been reported in the literature, mainly to add to the computational power of the hypercube. One of the attractive versions of the hypercube that was introduced to enhance the performance is the twisted hypercube. A twisted hypercube has the same structural complexities of the hypercube. It preserves the attractive properties of the hypercube and improves on the communication time by reducing the diameter by a factor of two. This paper presents the basic communication and some of the basic operations usually needed in parallel computing on the twisted hypercube interconnection network.**

***Index Terms*- hypercube, interconnection network, parallel prefix, routing, twisted hypercubes.**

I. INTRODUCTION

Recent advances of VLSI and computer networking technologies have made it attractive to build massive parallel machines. In the last decade, we have witnessed a tremendous surge in the availability of very fast and inexpensive hardware. These have been possible by using novel interconnections between processors and memories such as the hypercube topology. Parallel architectures based on the hypercube topology have gained widespread acceptance in parallel computing due to many of its attractive features. The hypercube offers a rich interconnection topology with high communication bandwidth, low diameter, maximum fault-tolerance, and a recursive structure that is suited naturally to divide and conquer applications.

The hypercube has been the topic of many recent researches [1]-[8]. Various researchers have done extensive work in showing the parallel computational power of the hypercube structure in many directions. In one direction, many researchers have shown the capability of the hypercube to simulate other networks such as rings, trees, grids and other interconnection networks with minimum overhead. In another direction, researchers have shown the power of the hypercube in solving many computational problems in parallel such as sorting, merging, parallel prefix, and other problems. In a third direction, researchers have shown the robustness and fault-tolerance of the hypercube, focusing on the hypercube's ability to simulate, compute, and reconfigure itself in the presence of faults.

Many researchers have proposed modifications on the hypercube structure to improve its computational power. Bhuyan and Agrawal [9] proposed a generalized hypercube structure that is suited to many applications. Preparata and Vuillemin [10] introduced the cube-connected cycles in which the degree of the diameter was reduced to a fixed constant. El-Amaway and Latifi [11] proposed the folded hypercube to reduce the diameter and the traffic congestion with little hardware overhead. Youssef and Narahari [12] proposed the banyan-hypercube network to reduce the communication overhead. Zheng at el. [13] proposed the star-hypercube hybrid interconnection network to combine the advantageous features and properties of both stars and hypercubes. Twisted hypercubes proved to contain the attractive properties of the hypercube and better communication capabilities. In parallel architectures, the communication cost dominates the computation cost. The over all performance of the parallel machine depends heavily on the underlying interconnection network. In a twisted hypercube, the diameter of the network is reduced by a factor of two over that of the hypercube. Many of the hypercube attractive features such as partitioning, routing, fault-tolerance, and embedding are incorporated into the twisted hypercube and new gains are achieved in diameter, average distance, and embedding efficiency [14]-[20].

The remainder of this paper is organized as follows. In section 2, we establish few preliminary definitions. In section 3, we present the data communication on the twisted hypercube. Section 4 presents some basic parallel computation operations. Finally, section 5 concludes the paper and discusses some future possible work.

II. PRELIMINARIES AND NOTATION

In this paper, we use undirected graphs to model interconnection networks. Let G = (V, E) be a finite undirected graph, where V and E are the vertex and edge sets of G, respectively. Each vertex represents a processor and each edge a communication link between processors. A *hypercube* of dimension n, denoted Q_n, is an undirected graph consisting of 2^n vertices. Each vertex corresponds to an n-bit binary string, labeled from 0 to 2^n-1, and such that there is an edge between any two vertices if and only if the binary representation of their labels differ by exactly one bit position. Each vertex is incident to n other vertices, one for each bit position. The edges of the hypercube can be naturally partitioned according to the dimensions that they traverse. An edge is called a *dimension i edge* if it connects two vertices that differ in the i^{th} bit position.

Let G be any undirected labeled graph, then G^b is obtained from G by prefixing every vertex label with b. Two binary strings $x = x_1x_0$ and $y = y_1y_0$, each of length two, are *pair-related* if and only if $(x, y) \in \{(00, 00), (10, 10), (01, 11), (11, 01)\}$. Now, we define a *twisted hypercube* of dimension n, denoted TQ_n, as an undirected graph consisting of 2^n vertices labeled from 0 to 2^n-1 and defined recursively as following:

a. TQ_1 is the complete graph on two vertices with labels 0 and 1.

b. For n > 1, TQ_n consists of two copies of TQ_{n-1} one prefixed by 0, TQ^0_{n-1}, and the other by 1, TQ^1_{n-1}. Two vertices $u = 0u_{n-2}...u_0 \in TQ^0_{n-1}$ and $v = 1v_{n-2}...v_0 \in TQ^1_{n-1}$ are adjacent if and only if

1. $u_{n-2} = v_{n-2}$, if n is even, and
2. For $0 \le i \le \lfloor (n-1)/2 \rfloor$, $u_{2i+1}\, u_{2i}$ and $v_{2i+1}\, v_{2i}$ are pair-related.

Figure 1 shows a twisted hypercube for dimension 3. The most important topological properties of the twisted hypercube including the following:

1. *Size*: A twisted hypercube of dimension n, TQ_n, consists of 2^n nodes.
2. *Degree*: A twisted hypercube of dimension n, TQ_n, has a degree n.
3. *Diameter*: A twisted hypercube of dimension n, TQ_n, has a diameter $\lceil (n+1)/2 \rceil$.
4. *Connectivity*: Let n be the dimension of the twisted hypercube, then the bisection width of the twisted hypercube is 2^{n-1}.
5. *Number of node-disjoint paths*: Let u and v be two nodes in TQ_n, then the number of node-disjoint paths is n.

III. DATA COMMUNICATION

One of the most important components of an interconnection network is its communication mechanism. In a parallel machine, communications become a bottleneck due to a great amount of time that is spent in interchanging information between different processors. It is very important to get the right data to the right place within a reasonable time. In parallel architectures, the communication cost dominates the computation cost. The over all performance of the parallel machine depends heavily on the underlying interconnection network. Data communication is considered the most essential attractive property for a parallel machine and usually one of the main topics addressed by researchers when proposing new topologies [5], [11], [12], [21]-[23],

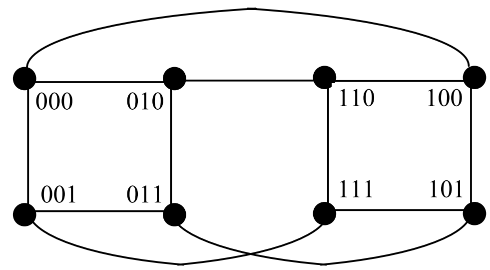

Figure 1. A twisted hypercube of dimension 3.

A. Data Routing

It is essential for a parallel architecture to support a mechanism which allows any two processors to exchange data. This may be achieved by finding the shortest path from the source processor to the destination processor. In this section, we introduce a shortest path algorithm that takes an advantage of the hierarchical nature of the twisted hypercube.

In order to find a route between two vertices of a twisted hypercube, we make extensive use of TQ_3. Suppose $u = u_2u_1u_0$ and $v = v_2v_1v_0$ are nonadjacent vertices of TQ_3 with $u_2 \ne v_2$. Since the diameter of TQ_3 is two, there is a vertex w which is a common neighbor of u and v. For example, node 010 and node 101 are connected through node 011. Therefore, node 011 is considered as a common neighbor of node 010 and node 101. Note that you might have two common neighbors in some cases. The following algorithm finds the shortest route between two nodes.

ROUTE (u, v)
Where u is the source and v is the destination.

STEP 1: Locate the leftmost differing bit between u and v, say bit u_k.

STEP 2: Group the bits to the right of the differing bit, bit k, into pairs, starting from the right.

STEP 3: Start at the leftmost differing bit position, bit k, and scan u and v from left to right, comparing pairs of bits from u with the corresponding bits of v, stopping at the first pair which is not pair-related, say the pair $u_{j+1}u_j$.

STEP 4: Locate the common neighbor between $u_ku_{j+1}u_j$ and the corresponding three bits in v using TQ_3, say xyz.

STEP 5: Construct the intermediate vertex w_i between u and v from u as follows:
a. Replace the three bits u_k, u_{j+1}, and u_j by x, y, and z, respectively.
b. Use the pair-related relation to replace all the pairs to the right of the j^{th} bit.

STEP 6: Repeat the previous steps, such that the source node is w_i until you reach node v.

For example, suppose the shortest route from u = 101001000110 to v = 101111101101 is desired. After step one of the routing algorithm, the result is 101**0**01000110. The differing bit is the 8^{th} bit. After step

ISBN:978-988-98671-5-7

two, the result is 101**0** 01 00 01 10. After step three, the result is 101**0** 01 **00** 01 10. The first pair in u that is not pair-related with the corresponding pair in v is $u_{j+1}u_j = 00$. After step four, xyz = 010, which is the common neighbor between 000 and 110 in TQ_3. After step five, w_1 = 101001101110. After repeating the same process, we get w_2 = 101001100110 and w_3 = 101001100111. Hence, the shortest route from u to v is through the nodes 101001101110, 101001100110, and 101001100111.

The routing algorithm can be easily proved by induction on the length of the path between the source node, node u, and the destination node, node v. It is important to note that each step of the routing algorithm will reduce the difference between the source and the destination by at least two bits, and hence the length of the path is at most half of the dimension.

B. Broadcasting

Broadcasting is the most essential communication operation in an interconnection network. In this operation, data that is initially in a single processor (source) is to be transmitted to all other processors in the network. The height of the broadcast tree of a network is at most its diameter. Since the twisted hypercube reduces the diameter by a factor of two, the height of its broadcast tree is also reduced by a factor of two. The broadcast tree of any network can be easily found by running a breadth-first algorithm. A Bridth-first spanning tree T of TQ is a spanning tree for which every path from a node to the root of T is a shortest path in TQ. A breadth-first spanning tree can be constructed easily by computing all shortest paths from node 0 to all other nodes in TQ using our *Route Algorithm*. The breadth first spanning tree constructed by the *Route Algorithm* represents the broadcast tree of the network [23]. Figure 2 shows the breadth-first spanning tree which is equivalent to the broadcast tree of a twisted hypercube for n=3, while Figure 3 shows the actual broadcasting on TQ_4.

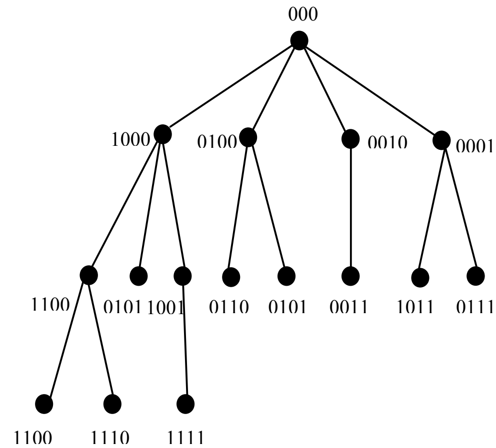

Figure 2. Breadth-first spanning tree for TQ_4.

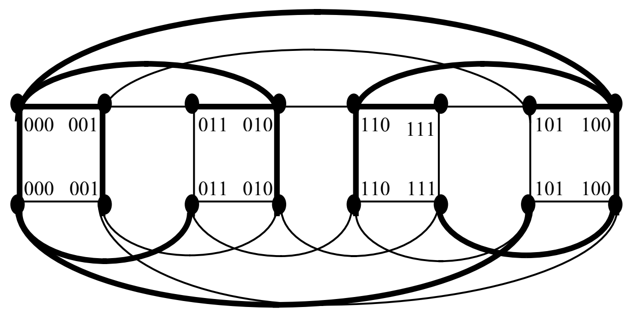

Figure 3. Broadcasting in TQ_4.

IV. BASIC COMPUTING OPERATIONS

This section demonstrates the ability of the twisted hypercube to perform many of the basic operations that are needed in designing parallel algorithms. These operations usually appear as sub problems in solving other major problems [6], [14], [17], [24].

A. Associative Computations

Associative operations are used frequently and appear as sub problems in solving other problems. They include addition, multiplication, finding the smallest, finding the largest, and others. Let + be the addition operation on some domain X. For a given tuple $\{x_0, x_1, ..., x_{k-1}\} \in X$, the addition operation is to compute the summation $y_0 = x_0 + x_1 + ... + x_{k-1}$.

We assume that each processor P_i, $0 \le i \le 2^n-1$, contains the value x_i. The computation is considered to be complete when the final summation y_0 is at processor 0. The symbol $\overset{j}{\Leftarrow}$ denotes a data transfer from a processor to an adjacent processor by a link through dimension j. The function BIT(j) returns the j^{th} bit of the node's label. The following algorithm performs the addition operation.

ADDITION (X)
begin
  for all P_i, $0 \le i \le 2^n-1$, *do*
    $y_i \leftarrow x_i$
  for $j \leftarrow n$ *to 1 do*
    for all P_i, $0 \le i \le 2^j-1$, *do*
      if BIT (j) = 1
      then $temp_k \overset{j}{\Leftarrow} y_i$, *where* P_k *is a neighbor through dimension j.*
      if BIT (j) = 0
      then $y_i \leftarrow y_i + temp_i$
    end for
  end for
end ADDITION

Algorithm ADDITION takes n communication steps which is the same time that takes to run the same procedure in a hypercube machine.

B. Parallel Prefix

The parallel prefix operation is a very important operation that appears frequently in designing parallel algorithms. It was first introduced by Ladner and Fischer [24] to solve the carry look-ahead problem for binary addition. The prefix operation was used by many researchers to solve a variety of problems in the field of computer science. In [5], the prefix operation was used to solve recurrence equations, to find convex hulls of images, to route packets in interconnection networks, and to solve the problem of computing carries.

Let $\oplus$ be a binary associative operation on some domain X. For a given tuple $\{x_0, x_1, ..., x_{k-1}\} \in X$, the prefix problem is to compute each of the partial sums, assuming $\oplus$ is addition, $y_i = x_0 \oplus x_1 \oplus ... \oplus x_i$, $0 \le i \le k-1$. We assume that each processor P_i, $0 \le i \le 2^n-1$, contains the value x_i. The computation is considered to be complete when the partial sum $y_i = x_0 \oplus x_1 \oplus ... \oplus x_i$ has been completed at processor i, $0 \le i \le 2^n-1$. The local variables y_i and t_i accumulate the partial and total sums, respectively. The symbol $\overset{j}{\Leftarrow}$ denotes a data transfer from a processor to an adjacent processor by a link through dimension j. The function BIT (j) returns the j[th] bit of the node's label.

```
PREFIX (X)
begin
    for all Pi,  0 ≤ i ≤ 2^n-1, do
        yi ← xi
        ti ← xi
    end for
    for j ← n to 1 do
        for all Pi, 0 ≤ i  ≤ 2^n-1,  do
            tempk ⇐j ti, where Pk is a neighbor through
                      dimension j.
            ti ← ti ⊕ tempi
            if  BIT (j) = 1
            then yi ← yi ⊕ tempi
        end for
    end for
end PREFIX
```

It is obvious that the algorithm runs in n time steps, where n is the dimension of the twisted hypercube. During the j[th] step, each node sends its current total sum to its adjacent node through dimension j. The partial and total sums of each node are updated based on the value of the j[th] bit of its label. Figure 4 shows the prefix computation on a twisted hypercube of dimension 3. The initial value x_i, the current partial sum y_i, and the current total sum t_i of each node are given for each phase.

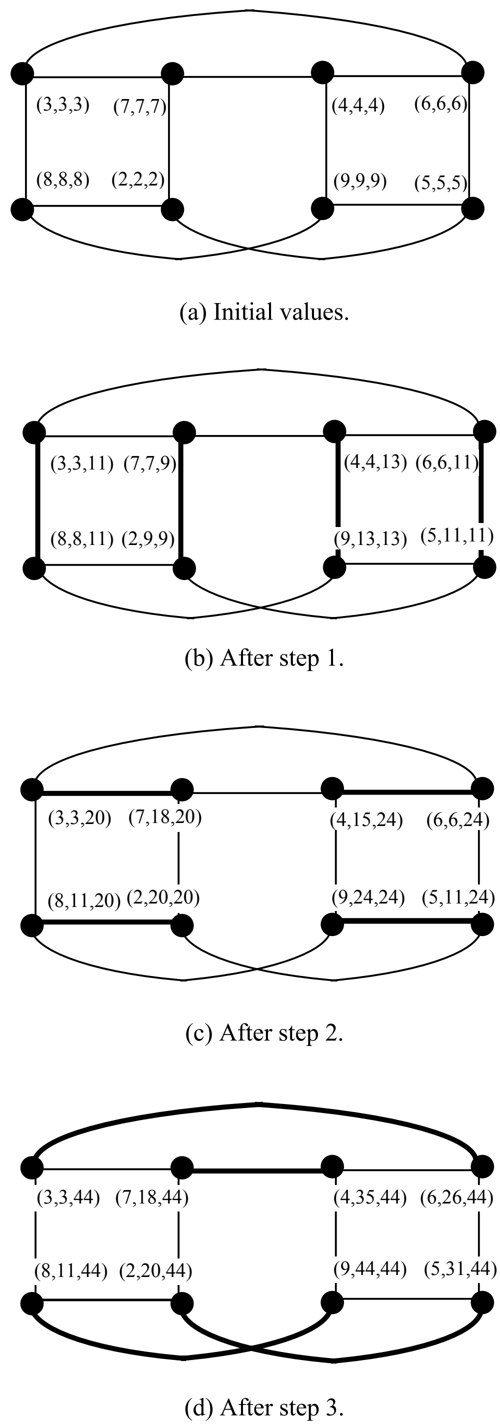

Figure 4. The prefix operation on a twisted hypercube of dimension 3.

V. CONCLUSIONS AND FUTURE WORK

This paper has presented some of the basic operations that are needed in designing parallel algorithms on a twisted hypercube. These operations usually appear as sub problems in solving other major problems in parallel computing. The preliminary investigations show that the twisted hypercube has attractive features; it preserves the good features of the hypercube and reduces the diameter by a factor of two. In this paper, we presented optimal routing and broadcasting algorithms. Also, we developed efficient algorithms for some of the basic parallel operations such as associative and prefix operations that appear usually as sup problems in other major problems. A good problem will be to uncover more of the appealing properties of the twisted hypercube. Another interesting problem is to show the ability of this structure to compute, simulate other interconnection networks, and reconfigure itself in the presence of faults.

REFERENCES

[1] J. Al-Sadi, K. Day, and M. Ould-Khaoua, "Unsafety Vectors: A New Fault-Tolerant Routing for Binary n-Cubes," *Journal of Systems Architecture*, vol. 47, no. 9, pp-783-793, 2002.

[2] G. Chiu and K. Chon, "Efficient Fault-Tolerant Multicast Scheme for Hypercube Multicomputers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 9, no. 10, pp. 952-962, 1998.

[3] K. Day and A. Al-Ayyoub, "The Cross Product of Interconnection Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 2, pp. 109-118, 1997.

[4] R. Klasing, "Improved Compressions of Cube-Connected Cycles Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 8, pp. 803-812, 1998.

[5] T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes*, Morgan Kaufmann, 1992.

[6] M. Quinn, *Parallel Computing: Theory and Practice*, McGraw Hill, 1994.

[7] Y. Saad and M. Schultz, "Topological Properties of the Hypercube," *IEEE Transactions on Computers*, vol. C-37, no. 7, pp. 867-872, 1988.

[8] V. Sharma and E. Varvarigos, "Circuit Switching with Input Queuing: An Analysis for the d-Dimensional Wraparound Mesh and the Hypercube," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, no. 4, pp. 349-366, 1997.

[9] L. Bhuyan and D. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network," *IEEE Transactions on Computers*, vol. C-33, no. 4, pp. 323-333, 1984.

[10] F. Preparata and J. Vuillemin, "The Cube-Connected Cycles: A Versatile Network for Parallel Computation," *Communications of the ACM*, vol. 24, no. 5, pp. 3000-309, 1981.

[11] A. El-Amaway and S. Latifi, "Properties and Performance of Folded Hypercubes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 1, pp. 31-42, 19991.

[12] A. Youssef and B. Narahari, "The Banyan-Hypercube Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 2, pp. 160-169, 1990.

[13] S. Zheng, B. Cong, and S. Bettayeb, "The Star-Hypercube Hybrid Interconnection Networks," *Proceedings of the ISCA International Conference on Computer Application in Design, Simulation, and Analysis*, pp. 98-101, 1993.

[14] E. Abuelrub, "Parallel Computation on Twisted Hypercubes," *Al-Manarah Journal*, vol. 5, no. 1, pp. 1-10, 2002.

[15] E. Abuelrub, "Embedding Quad Trees into Twisted Hypercubes," *Proceedings of the 2nd IASTED International Conference on Parallel and Distributed Systems*, pp. 155-160, 1998.

[16] E. Abuelrub and S. Bettayeb, "Embedding Rings into Faulty Twisted Hypercubes," *Computers and Artificial Intelligence*, vol. 16, no. 4, pp. 425-441, 1997.

[17] K. Efe, "The Crossed Cube Architecture for Parallel Computation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 5, pp. 513-524, 1992.

[18] W. Huang, J. Tan, C. Hung, and L. Hsu, "Fault-Tolerant Hamiltonicity of Twisted Cubes," *Journal of Parallel and Distributed Computing*, vol. 62, pp. 591-604, 2002.

[19] P. Kulasinghe and S. Bettayeb, "Embedding Binary Trees into Crossed Cubes," *IEEE Transactions on Computers*, vol. 44, no. 7, pp. 923-929, 1995.

[20] P. Kulasinghe and S. Bettayeb, "The Multiply-Twisted Hypercube with 5 or more Dimensions is not Edge-Transitive," *Information Processing Letters*, vol. 53, pp. 33-36, 1995.

[21] A. Awwad and J. Al-Sadi, "On the Routing of the OTIS-Cube Network in the presence of Faults," *The International Arab Journal of Information Technology*, vol. 2, no. 1, pp. 17-23, January 2005.

[22] C. Decayeux and D. Seme, "3D Hexagonal Network: Modeling, Topological Properties, Addressing Schemes, and Optimal Routing Algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 9, pp. 875-884, September 2005.

[23] J. Fu and G Chen, "Hamiltonicity of the Hierarchical Cubic Network," *Theory of Computer Systems*, vol. 35, pp. 59-79, 2002.

[24] R. Lander abd M. Fischer, "Parallel Prefix Computation," *Journal of the ACM*, vol. 27, pp. 831-838, 1980.