IMPROVE FAULT TOLERANCE, SHORTEST PATH ROUTING IN HYPER DE BRUIJN ASTER NETWORK

Nguyen Chi Ngoc, Nguyen Hong Thai, Pham Minh Tri
University of Technology, VNU-HCM

ABSTRACT: *In this article, we investigate the properties of Hyper de Bruijn Aster Networks in order to evolve the Fault Tolerance, Shortest Path Routing applying into the network of multiprocessor's application, parallel processing systems. By giving new method of routing basing on features of duplex mapping in Hyper de Bruijn Aster Network, we can achieve more possibilities in routing than other recent researches that refers to increase the performance of Fault Tolerance and Shortest Path Routing.*

Keywords: *de Bruijn graph, Hypercube graph, Hyper de Bruijn, Hyper de Bruijn Aster, shortest path, fault tolerant, interconnection network.*

1.INTRODUCTION

The properties of de Bruijn network has shown it be the next generation after the hypercube for parallel processing applications such as multiprocessor network, VLSI [1,2,4,6,7,8,12]. Hypercube has its own advantages including the degree and diameter are independent [4]. Therefore the combination between de Bruijn graph and Hypercube graph will make an ideal topology for fault tolerance and shortest path routing. In this article, we first present a topology named Hyper de Bruijn Aster and then communication on this topology is investigated.

Routing in de Bruijn graph has been investigated by Samantham, Liu and Mao [1,2,6], but their algorithms cannot achieve shortest path if there is a fault node along the path [7]. Broadcasting in de Bruijn graph has also been investigated by Esfahanian, Ganesan, Ohring [8,10]. However, only working on binary de Bruijn is their drawback. To address the problems of fault tolerance in shortest path routing and broadcasting in high degree de Bruijn network, please refer to the paper [7].

The combination of hypercube and de Bruijn as Hyper de Bruijn has been studied by Elango Ganesan, Dhiraj K Pradhan [3,12]. However, their topology [12] is based on binary de Bruijn graph, and their DeadLock-Free routing algorithm [3] can work in binary de Bruijn network only. Wei Shi and Pradip K Srimani [5] have proposed a very good routing algorithm based on Hyper Butterfy (the combination between Hypercube and Butterfly network). In their article, they criticized that Hyper de Bruijn network (proposed by Ganesan [12]) is not regular, not optimally fault tolerant, and complex routing. All of these criticisms are solved in our article. The Hyper de Bruijn Aster proposed in this article can be used for load balancing and parallel processing. Our fault tolerant properties and shortest path routing algorithm are proved to archive the best performance among routing algorithms in Hyper de Bruijn network. Consequence, Hyper de Bruijn Aster has shown to be the most suitable topology for multiprocessor, VLSI and parallel processing networks.

Section 2 presents some background and properties of Hyper de Bruijn Aster. Section 3 presents shortest path routing algorithm and fault tolerant characteristics of Hyper de Bruijn Aster. And in section 4 comes with the conclusion.

2.HYPER DE BRUIJN ASTER GRAPH

2.1.Hypercube Graph Hn and De Bruijn Graph Dn

Hypercube Graph order m, H(m) includes the set of node Z_2^m . For the two adjacent nodes, address is different in one bit.

Binary de Bruijn Graph(undirected) order n, D(n) includes the set of vertices Z_2^n. Given α,β ∈ Z2 and x ∈ Z_2^{n-2} , each node is presented as αxβ, and linked by:

- $x\beta\alpha$ by shuffle arc
- $x\beta\overline{\alpha}$ by shuffle-exchange arc
- $\beta\alpha x$ by inverse-shuffle arc
- $\overline{\beta}\alpha x$ by inverse-shuffle-exchange arc

Similarly, if we extend α,β ∈ Zd and x ∈ Z_d^{n-2} (d ≥ 2, degree higher than 2), then the number of link to each node and the number of adjacent nodes are increased. And hence, it improves fault tolerant in de Bruijn graph.

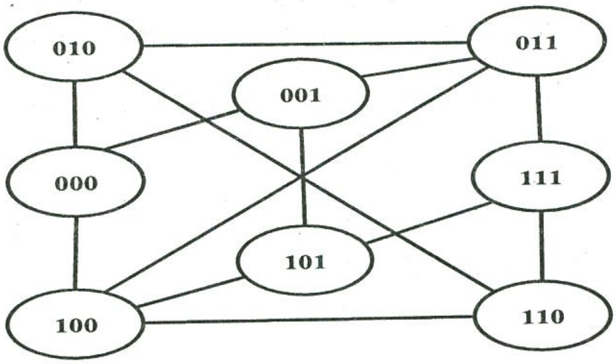

Figure 1. Hypercube graph H(3).

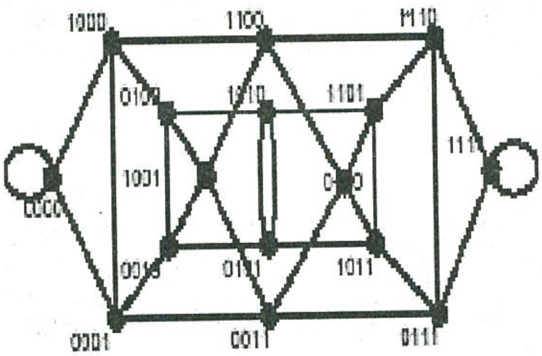

Figure 2. de Bruijn graph D(2,4).

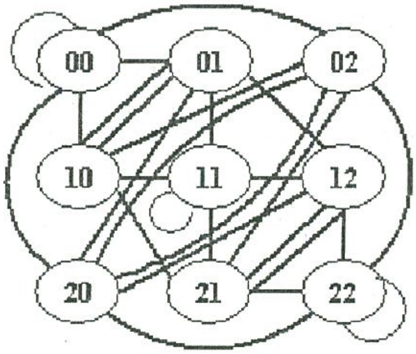

Figure 3. de Bruijn graph, D(3,2).

Corollary 1 D(d,n) is a symmetrical undirected graph, it has degree 2d, diameter n and total number of nodes dn [7].

Corollary 2 H(m) is a symmetrical undirected graph, it has degree m, diameter m and total number of nodes 2m [5].

2.2.Hyper de Bruijn Aster Graph HD*(m,d,n)

Definition: By extending de Bruijn graph to high degree (order) and combining with hypercube, we invent a new topology Hyper de Bruijn Aster order (m,n) symbolized by HD*(m,d,n), HD*(m,d,n) is a product of H(m)xD(d,n).

It is obviously to obtain links from a node $\langle x_{m-1}x_{m-2}\ldots x_0, y_{n-1}y_{n-2}\ldots y_0\rangle$ to the following nodes (in deBruijn part):

$\langle x_{m-1}x_{m-2}\ldots x_0, \alpha y_{n-1}y_{n-2}\ldots y_1\rangle$

$\langle x_{m-1}x_{m-2}\ldots x_0, y_{n-1}y_{n-2}\ldots y_1\alpha\rangle$, $\alpha \in Z_d$

And link to following node (in Hypercube part):

$\langle x_{m-1}x_{m-2}\ldots \overline{x_i}x_ix_{i-1}\ldots x_0, y_{n-1}y_{n-2}\ldots y_0\rangle$, $0 \le i \le m-1$

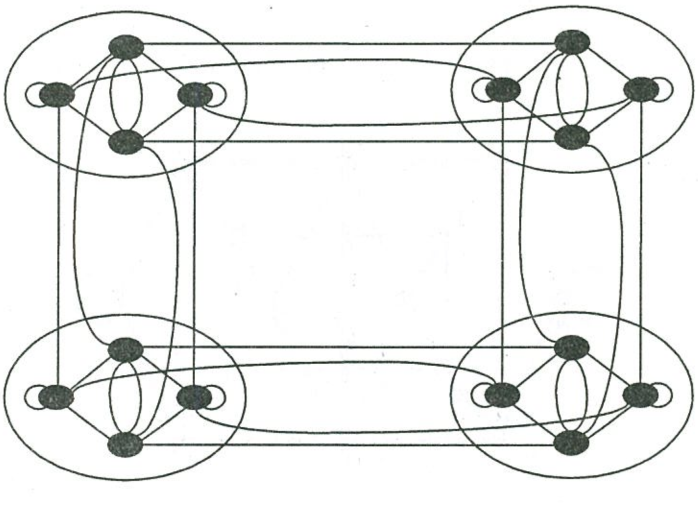

Figure 4. Another presentation of HD*(2,2,2).

Theorem 1 HD(m,d,n) is a symmetrical undirected graph with degree m+2d, diameter m+n and total number of node N(HD*) = 2mdn.*

Proof:

- Hypercube and de Bruijn graph are symmetrical (corollary 1,2). A Hyper de Bruijn Aster HD*(m,d,n)=H(m) x D(d,n), each node of hypercube contains a de Bruijn graph D(d,n). Therefore, HD*(m,d,n) is symmetrical.
- The total number of node in HD*(m,d,n) = total number of node in Hypercube H(m) x total number of node in de Bruijn D(d,n). The total number of node in Hypercube is 2m [4], the total number of node in de Bruijn is dn [1,2,7]. So N(HD*) = 2mdn
- Degree is the total number of link connecting to a node [4], because degree of hypercube H(m) is m (corollary 1), degree of de Bruijn D(d,n) is 2d (corollary 2) and HD*(m,d,n) is the combination of H(m) and D(d,n) (H(m) is the base cluster), each node in HD*(m,d,n) connects to H(m) and D(d,n) network, so the degree of HD*(m,d,n) is 2d+m.
- Diameter is the shortest distance between 2 farthest vertices in a graph [4]. Routing in HD*(m,d,n) from a vertex v(h,d) to a vertex v(h',d') can be done by 2 way: v(h,d) → v(h',d) → v(h',d') or v(h,d) → v(h,d') → v(h',d'), and diameter of H(m) is m, diameter of D(d,n) is n. Therefore, diameter of HD*(m,d,n) is m+n.

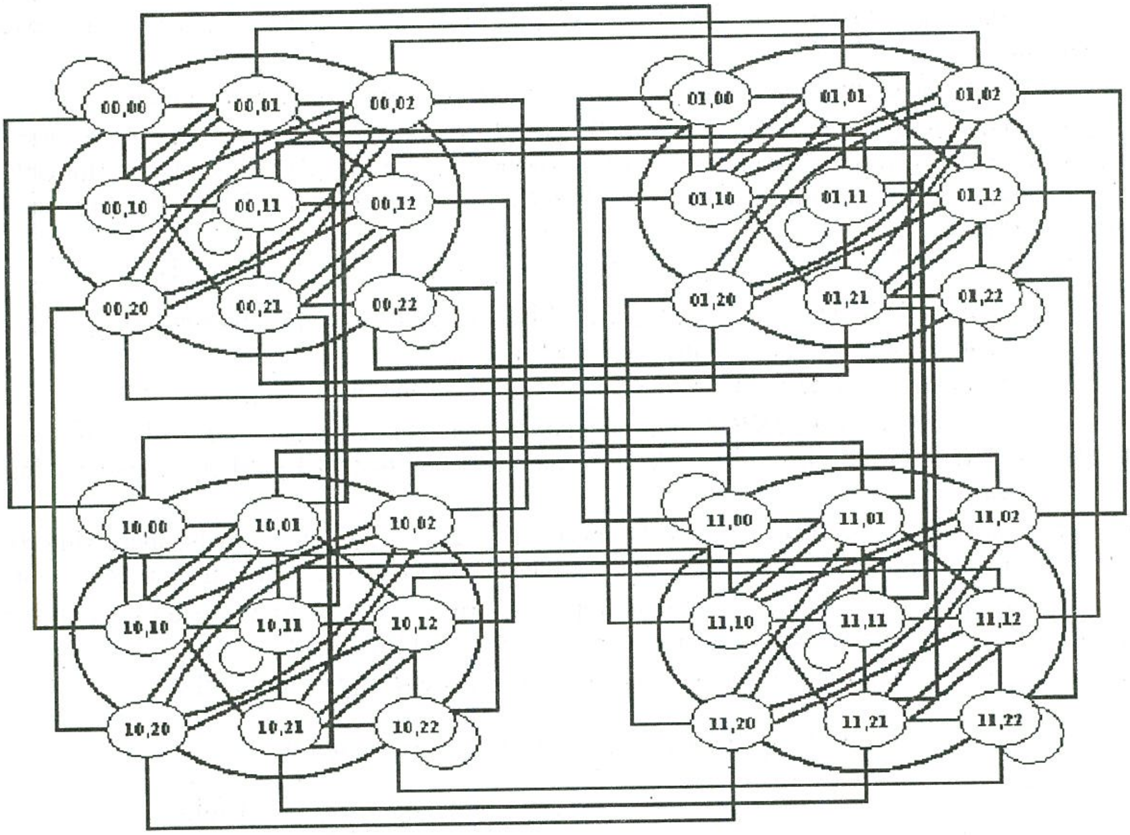

Figure 5. Hyper de Bruijn Aster HD*(2,3,2).

Corollary 3 Address of a node in HD* includes 2 parts: the first part is the address of Hypercube graph and the second part is the address of de Bruijn graph.

Proof: this is a corollary directly inferred by definition of Hyper de Bruijn Aster.

3.SHORTEST PATH ROUTING AND FAULT TOLERANCE IN HD*(M,D,N)

3.1.Shortest path routing

Shortest path routing from a vertex v(h,d) to a vertex v(h',d') can be established as follows,

- Shortest path from v(h,d) → v(h',d) can be achieved by shortest path routing scheme in hypercube graph, algorithms and proof are shown in references [4,5,12].
- Shortest path from v(h',d) → v(h',d') can be achieved by shortest path routing scheme in de Bruijn graph, algorithms and proof are shown in references [2,7].
- Or we can do the above steps inversely (routing in de Bruijn first and then routing in Hypercube)

Therefore, we have new theorem,

Theorem 2 ShortestRouting(HD) = ShortestRouting(H) + ShortestRouting(D) (*)*

Or ShortestRouting(HD) = ShortestRouting(D) + ShortestRouting(H)*

By theorem 2, we see that Shortest Path routing in HD* is its advantage in comparison to Hypercube, de Bruijn, Butterfly... In the next section, we will present a shortest path routing algorithm in Hyper de Bruijn Aster. By applying our algorithm, nodes in the network can perform smoother, faster and especially efficient in Fault Tolerant, Routing and Load Balancing.

The following compare some of properties of Hyper de Bruijn Aster to others,

- Suppose Hyper de Bruijn Aster *HD*(m1,d1,n1)*, m1 is hypercube's order(diameter), n1 is de Bruijn's order(diameter), and d1 is deBruijn's degree (d1 ≥ 2). Hyper Butterfly HB(m2,n2), m2 is hypercube's order, n2 is Buterfly's order. HD* and HB have the same total number of node.
- We have, $2^{m_1} d_1^{n_1} = n_2 2^{m_2+n_2} \Leftrightarrow m_1 + n_1 \log_2 d_1 = m_2 + n_2 + \log_2 n_2 (1)$

Degree of HD*: dHD* = m1 + 2d1 ≥ m1 +4 because of d1 ≥ 2. This makes HD* network performs better in fault tolerant routing and broadcasting.

Besides, diameter of HD*: DHD*= m1+n1= m2+n2 + log2n2 - n1log2d1 + n1

Increase d1 → decrease *DHD** (total number of node is not change) and if d1 ≥ 2n2 then DHD* ≤ DHB. It proves that HD* network is more efficient than HB in Shortest Path routing and broadcasting.

For Fault Tolerance *HD** is proportional to m1 + 2d1 — 2(2). Obviously, increasing d1 will improve Fault-Tolerance of *HD**. While, Fault-Tolerance of HB is proportional to m2 + 4.

By the above discussion, we see that Hyper de Bruijn Aster performs better than Hyper Butterfly in Fault Tolerant routing, Shortest path routing and Broadcasting.

The following table shows comparison among Hyper de Bruijn Aster to others,

Table 1. Hyper de Bruijn Aster in comparison to others.

Graph	Node	Degree	Diameter	Fault-Tolerance
Hypercube H(m+n)	2m+n	m+n	m+n	m+n
Butterfly B(m+n)	(m+n)2m+n	4	3n/2	4
de Bruijn D(d,m+n)	dm+n	2d	m+n	2d-2

HyperButterfly HB(m,n)	n2m+n	m+4	m+3n/2	m+4
Hyper de Bruijn Aster HD*(m,d,n)	2mdn	m+2d	m+n	m+2d-2

Explanations for algorithm in figure 6:

- Step 1: use shortest path routing algorithm in de Bruijn graph[2,7] to find all shortest paths from (hS,dS) to (hS,dD).
- Step 2: check fault tolerant characteristics of these paths from step 1.
- Step 3: if there exist a fault free shortest path from dS to dD, then route from (hS,dS) to (hS,dD). Otherwise, go to step 7.
- Step 4: use shortest path routing algorithm in hypercube [4,12] to find all shortest paths from (hS,dD) to (hD,dD).
- Step 5: check fault tolerant characteristics of these paths from step 4.
- Step 6: if there exists a fault free shortest path from hS to hD, then route from (hS,dD) to (hD,dD), go to END. Otherwise, ignore node, go to step 7.
- Step 7: use shortest path routing algorithm in hypercube [4,12] to find all shortest paths from (hS,dS) to (hD,dS).
- Step 8: check fault tolerant characteristics of these paths from step 7.
- Step 9: if there exists a fault free shortest path from hS to hD, then route from (hS,dS) to (hD,dS). Otherwise, go to END.
- Step 10: use shortest path routing algorithm in de Bruijn graph[2,7] to find all shortest paths from (hD,dS) to (hD,dD).

```
1     START
2     IF Node(hS,dS) = Node(hS,dD) THEN GOTO END;
3     ELSE
4         Node = Node(hS,dS)
5         Shortest Path Routing in de Bruijn from dS to dD
6     ENDIF
7     i,j belong to the set of nodes of shortest path routing in
Line 5
8     CALL CheckConnect(Node, Node[i,j])
9     IF CheckConnect(Node, Node[i,j]) = True  THEN
10        CALL Routing(Node, Node[i,j])
11        i = i+1
12        Node(hS,dS) = Node[i,j]
13        GOTO START
14    ELSE
15        IF (j<maxPath) THEN
16            j=j+1
17            GOTO Line 3
18        ELSE
19            Shortest Path routing in Hypercube
20            k belongs to the set of nodes of shortest path
```

```
routing in Line 19
21          CALL CheckConnect(Node, NodeCube[k])
22          IF CheckConnect(Node, NodeCube[k]=True  THEN
23              CALL Routing(Node, NodeCube[k])
24              Node(hS,dS)=NodeCube[k]
25              GOTO START
26          ELSE
27              IF (k<maxBit) THEN
28             k=k+1
29             GOTO Line 19
30              ELSE
31             CALL IgnoreNode[i]
32             RESET i,j
33             GOTO START
34              ENDIF
35          ENDIF
36       ENDIF
37    ENDIF
38    END
```

Figure 6. Shortest Path routing algorithm.

- Step 11: check fault tolerant characteristics of these paths from step 10.
- Step 12: if there exist a fault free shortest path from dS to dD, then route from (hD,dS) to (hD,dD). Otherwise, go to END.
- END.

Corollary 3 *from the formula (*), the total number of shortest path cross N nodes (including source and destination node, with the difference between source and destination address is M (in digit)) is N.M path.*

Corollary 4 *routing in Hypercube and de Bruijn in fault free mode can base on the difference between the source and destination address [4][2].*

Corollary 5 *Hyper de Bruijn Aster can be mapped following hypercube's edge (as presenting in figure 7). Therefore, if there is a fault when routing from a node to its adjacent node in the same cube then we can move this routing to the neighbor cube. However, we still keep address and routing direction in de Bruijn network*

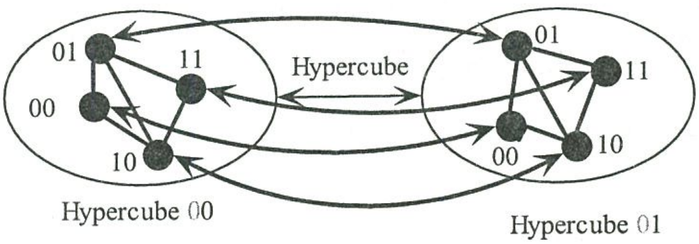

Figure 7. Mapping of each vertex of Hypercube 00 to Hypercube 01.

Example: to route from node (00,01) to node (01,11) in HD*(2,2,2) (figure 4), we do as follows,

- Use shortest path routing for de Bruijn graph in the same cube, i.e. (00,01) → (00,11) → (01,11).
- If node (00,11) is fault then it routes (00,01) → (01,01) → (01,11)

Corollary 6 *by routing follows our shortest path routing algorithm, there is a proportion of path length and total number of shortest path, and it belongs to the difference in the number of digit between source and destination address.*(Increasing the number of different digit makes longer path and hence increasing the total number of shortest path and fault possibility along the path).

3.2.Some fault tolerant characteristics of HD*

Fault Tolerance of network is the possibility to continuously work when there are failure nodes in the network.

In HD*, for a pair of source and destination node, we can find several shortest paths (≥1). Therefore, if there is a fault in a shortest path, then we can choose another shortest path to route. This improves fault tolerant. By applying our shortest path routing algorithm in section 3, we can avoid failure node along the path without reinitialize the whole process from the source node. Moreover, combining with "discrete set" concept in [7], we can provide fault free shortest path (optmum shortest path in the case of failure occurred) in HD*.

In any network, the requirement for a routing algorithm to be successful is that the adjacent nodes of source node and cubes in the middle have to be in good condition (ready for transmission the message). Otherwise, it cannot route the message. Some Shortest path routing algorithms cannot work well in HD* because of the above circumstances. It leads our shortest path routing algorithm to be optimum for fault tolerance.

Suppose, a node X <x0x1x2..xi…xn-1>, 0≤i≤n-1

In *HD*(m,d,n)* the total adjacent nodes of X, NHD* is:

$$N_{HD^*} = m + 2d \leftrightarrow \exists x_i \mid x_i \neq x_{i+2}, 0 \leq i \leq n-3 (3)$$

$$N_{HD^*} = m + 2d - 1 \leftrightarrow \begin{Bmatrix} \forall x_i \mid x_i \neq x_{i+2}, 0 \leq i \leq n-3 \\ x_0 \neq x_1 \end{Bmatrix} (4)$$

$$N_{HD^*} = m + 2d - 2 \leftrightarrow \forall x_i \mid x_i = x_{i+1}, 0 \leq i \leq n-2 (5)$$

Therefore, fault tolerance of HD* network when applying our shortest path routing algorithm is proportional to NHD* -1. By another mean, fault tolerance and time complexity of algorithm routing between 2 nodes belong to position of the failure nodes in the network.

4.CONCLUSION

Through our study in parallel processing systems, VLSI, multiprocessor networks, we invent a new topology called Hyper de Bruijn Aster HD*(m,d,n). Based on HD*, shortest path routing and fault tolerance are investigated. Our HD* has shown its superior in size (low diameter but large number of node), fault tolerance (because of the extending to high degree) and shortest path routing (our topology can provide more shortest paths than others). Consequently, our HD* is the best candidate for designing network of parallel processing systems, VLSI.

CẢI TIẾN KHẢ NĂNG CHỊU LỖI, CÁC GIẢI THUẬT TÌM ĐƯỜNG ĐI NGẮN NHẤT TRONG MẠNG HYPER DE BRUIJN ASTER

Nguyễn Chí Ngọc, Nguyễn Hồng Thái, Phạm Minh Trí
Trường Đại Học Bách Khoa – ĐHQG-HCM

TÓM TẮT: *trong bài báo này, chúng tôi nghiên cứu về cấu trúc, tính chất của một mô hình mạng mới được gọi là Hyper de Bruijn Aster (ký hiệu là HD*). Thông qua mô hình này, các giải thuật về Fault Tolerance (khả năng chịu lỗi), giải thuật tìm đường đi ngắn nhất được đề xuất và ứng dụng vào mạng multiprocessor và các hệ thống xử lý song song. Bằng cách đưa ra phương pháp lưu thông dựa trên các đặc tính liên kết kép trong mạng Hyper de Bruijn Aster, chúng tôi có thể tăng cường hiệu năng chịu lỗi và gia tăng số lượng đường đi ngắn nhất hơn các giải thuật được đưa ra bởi các nhà nghiên cứu khác trên cùng lĩnh vực.*

REFERENCES

[1]. Samantham, R. Maheswara , and D.K. Pradhan, *The De Bruijn Multi-processor Network: A Versatile Parallel Processing and Sorting Network for VLSI*, IEEE Trans. on Comp., Vol.38, NO.4, (1989).

[2]. Zhen Liu, Ting-Yi Sung, *Routing and Transmitting Problem in de Bruijn Networks*, IEEE Trans. On Comp., Vol. 45, Issue 9, pp 1056 – 1062, Sept. (1996).

[3]. Elango Ganesan, Dhiraj K Pradhan, *Wormhole Routing In De Bruijn Networks And Hyper de Bruijn Networks*, Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on Volume: 3 , 25-28 May (2003).

[4]. J. A. Bondy and U. S. R. Murty, *Graph Theory and Aplication.*

[5]. Wei Shi and Pradip K Srimani, *Hyper Butterfly Network: A Scalable Optimally Fault Tolerant Architecture.*

[6]. Jyh-Wen Mao and Chang-Biau Yang, *Shortest path routing and fault tolerant routing on de Bruijn networks,* Journal of Networks,Vol. 35,Issue 3,Pages 207-215, (2000).

[7]. Ngoc Chi Nguyen, Nhat Minh Dinh Vo and Sungyoung Lee, *Fault Tolerant Routing and Broadcasting in de Bruijn networks*, the 19th IEEE International Conference on Advanced Information and Networking Applications (AINA'05), Mar (2005).

[8]. A.H. Esfahanian, G. Zimmerman, *A distributed broadcast algorithm for binary De Bruijn networks,* Conference Proceedings, Seventh Annual International Phoenix Conference on Comp. and Comm., 16-18 March (1988).

[9]. Dally, W. J., *Performance analysis of k-ary n-cube interconnection networks*, IEEE Trans. on Computers, vol. 39, pp. 775-785, Jun (1990).

[10]. S.R.Ohring, D.H.Hondel, *Optimal Fault-Tolerant Communication Algorithms on Product Networks using Spanning Trees*, IEEE Symposium on Parallel Distributed Processing, October (1994).

[11]. Pradhan, D. K. and Reddy, S. M., *A fault-tolerant communication architecture for distributed systems,* IEEE Trans. On Computers, vol. C-31, pp. 863-870, Sep (1982).

[12]. Ganesan, E. and Pradhan, D.K., *The Hyper de Bruijn network: Scalable versatile architecture,* IEEE Trans. On Parallel and Distributed Systems, vol. 4, Sep. (1993).