Fault-tolerant Routing for Irregular Faulty Patterns in 2D-Mesh without Virtual Channel

Savio S.H. Tse*, Jipeng Zhou†, and Francis C.M. Lau‡
*Computer Engineering Department
Istanbul University, Avcilar Campus, 34320, Istanbul, Turkey
Email: sshtse@istanbul.edu.tr, sshtse@gmail.com
†Department of Computer Science
Jinan University, Guang Zhou 510632, China
Email: tjpzhou@jnu.edu.cn
‡Department of Computer Science
The University of Hong Kong, Hong Kong, China
Email: fcmlau@cs.hku.hk

***Abstract*—In wormhole meshes, many fault-tolerant routing algorithms have been proposed. None of them, however, can tolerate general (irregular) fault model without virtual channels. In this paper, a 0-virtual channel fault-tolerant deadlock-free routing algorithm for general fault model is proposed for wormhole routing in two-dimensional meshes.**

Considering XY-routing as a precious property of two-dimensional meshes, we try to keep as many XY-routings as possible without incurring any deadlock. During the pre-processing, the mesh is divided into a number of fault-free (healthy) rectangular regions, and a minimum spanning tree connecting all these regions is built. Routing within a region uses XY-routing mainly; and routing across regions will use the spanning tree. An online solution is given for handling dynamic faults and recovery.

***Keywords*-Wormhole Routing, virtual Channel, 2D-Mesh, fault-tolerance.**

I. Introduction

Multiprocessor systems rely on an interconnection network between processors to exchange data and synchronize with each other. Wormhole routing divides a message into packets and the packets into flits, it then routes the flits in each packet through the network in a pipeline fashion. Header flits contain all of the routing information for a packet and lead each packet through the network. When the header flits reach a node that has no suitable output channel available, all of the flits in the packet wait where they are for a suitable channel to become available. One disadvantage of wormhole routing relative to store-and-forward routing is that it tends to support routing that is less fault tolerant.

In multiprocessor systems, routing algorithms provide mechanisms for communication between processors, and the efficiencies of routing algorithms is important for achieving high performance in a multiprocessor system. Examples of commercial multi-computers and research prototypes based on mesh networks are Intel Paragon(2D mesh), IBM Blue Gene [1](3D mesh), Alpha 21364[11](2D torus),and Cray T3D/T3E [15](3D torus).

Recently, main microprocessor manufactures have shifted to chip multi-processor for their latest products. Intel has announced a research chip with 48 cores under the Tera-Scale Computing Research Program [8]. The most straightforward topology for on-chip networks is the 2D mesh structure as it offers regularity and simplicity for routing. Defective components may break topology regularity, thus efficient routing becomes a challenge. Currently, many solutions [7][12][13] [20] have been proposed to handle this problem. These schemes are focused on routing implementation on a chip. For example, an universal logic-based distributed routing (uLBDR) is proposed to any irregular topology derived from 2D meshes in [12][13]. The uLBDR relies on a compact representation of routing algorithm, and the routing algorithm implementation with uLBDR is based on a set of routing restrictions to ensure deadlock freedom. Y. Fukushima *et al.* [7] proposed a routing control algorithm for non-virtual channel router of irregular 2D mesh network-on-chips, their fault model is rectangle faulty regions, which needs to deactivate healthy nodes around faulty nodes.

Numerous fault-tolerant routing algorithms in general mesh networks have been proposed, and most of these algorithms augment the dimension-order routing algorithm to tolerate some faults. R.V. Boppana and S. Chalasani proposed a fault-tolerant routing algorithm in mesh networks [3][5], or in mesh and torus networks [4]. The deadlock can be prevented by using four virtual channels per physical channel for deterministic(dimension- order) fault tolerant routing, and their fault models are rectangle[3][4] and special convex [5]. P.H. Sui and S.D. Wang [16] improved Boppana and Chalasani's algorithm [3], and proposed a fault-tolerant routing algorithm with three virtual channels per physical channel to tolerate the rectangle fault model.

Tsai [18][17] proposed fault-tolerant routing algorithm

1087-4089/12 $26.00 © 2012 IEEE
DOI 10.1109/I-SPAN.2012.21

with two virtual channels, which prohibit routing to some locations and the fault blocks in the model could include non-faulty nodes. J. Wu [21] proposed a fault-tolerant extended XY-routing protocol, which is based on dimension-order routing and odd-even turn model and does not use any virtual channels in 2D meshes. The paper uses extended faulty block (disjointed rectangles), which consists of connected unsafe and faulty nodes. The proposed protocol in [24] can be applied in 2D meshes with orthogonal faulty blocks (convex polygon). The extended XY-routing protocol prohibits certain location of faults and destinations, their fault models includes non-faulty nodes. D. Wang [19] proposed a rectilinear-monotone polygonal model MCC (Minimal-Connected-Component) for fault-tolerant minimal routing in meshes. The contribution of the proposed fault model in [19] is that it includes fewer non-faulty nodes in fault block, many non-faulty nodes that would have been included in rectangular fault blocks can become candidate routing. However, MCC model includes non-faulty nodes, too.

D. Xiang *et al.* [22][23][24] proposed a fault-tolerant routing algorithm, which is based on a new virtual network partitioning scheme and block fault model. Two virtual channels are used to support deadlock free adaptive routing in meshes with arbitrarily shaped faults. A message is routed along a series planes. The fault-tolerant routing method in [9] provides planar-adaptive routing based on the 3D MCC fault model. The method implements planar-adaptive routing based on arbitrary orders of dimensions. Three virtual channels are used to support deadlock free routing in 3D meshes. F. Safaei and A. Mortazavi [14] proposed a fault-tolerant routing algorithm to prevent static faults in 2D mesh networks. It needs to know all faults in the network when the system is started. In their routing scheme, messages are avoided to enter congested faulty regions of the network. According to this scheme, the system cannot remain continuously operational when network components fail, and five virtual channels are needed for deadlock free routing.

In most situations, algorithms are designed to work only when fault regions have certain shapes, and it can be made to work for arbitrarily located faults by first deactivating certain nodes in such a way that the regions of faulty/deactivated nodes have the required shapes. Without assuming any specific shapes for faulty regions, and deactivating any nodes, our aim is to design a deadlock-free routing scheme in a faulty mesh, which uses only one 1-flit buffer at the ends of each link and no virtual channel routing.

For a fault-free mesh, the common XY-routing is enough. With the presence of faulty nodes, XY-routing may be blocked. Inevitably, we must apply YX-routing in some communication paths, and deadlock may occur. Therefore, YX-routing should be handled carefully. For the general fault model, previous algorithms use either more than one 1-flit buffer or virtual channel for routing. The underlining idea is to avoid any cycle formed by communication paths. However, to avoid any cycle using one 1-flit buffer without virtual channel is theoretically not difficult. In the extreme, spanning tree can be used for routing because of cycleless, at the expense of all useful properties of mesh. We will hardly satisfy with this solution, and in this paper, we give a solution which uses much of the remaining properties of a faulty mesh.

II. Definitions

Each node in a mesh is located in a unique (x, y)-coordinate. The coordinate of a node v is denoted as $(v.x, v.y)$. The $M \times N$ mesh in question is an undirected unit-cost graph $G = (V, E)$ such that $V = \{(x, y)|x \in [1, M], y \in [1, N]\}$ and $E = \{(v_1, v_2)|v_1, v_2 \in V, |v_1.x - v_2.x| + |v_1.y - v_2.y| = 1\}$, where $M, N \in I^+$, and $M \geq N$. We follow the convention that the node in the top left corner has coordinate $(0, 0)$ and that in the bottom right has coordinate $(M - 1, N - 1)$.

There are two kinds of nodes, namely *healthy* and *faulty* nodes. A healthy node can process and relay messages, and has four ingoing and outgoing edges. Each edge has a one-flit buffer at each end, and therefore, there are totally eight buffers in each node. A faulty node is assumed to be inactive in all aspects. In other words, we ignore the remaining properties of a faulty node. A *faulty block* is a submesh which nodes are all faulty. A healthy node can detect any faulty neighbours. Without loss of generality, we assume all healthy nodes are connected. Otherwise, we will handle each connected region of healthy nodes separately. A *rectangle* is a submesh of healthy nodes.

We now define the problem. The input consists of two integers M and N, and the coordinates of top left and bottom right nodes of each faulty block. The output is a routing table for each healthy node such that the routing is deadlook-free and uses no virtual channel.

III. The Algorithm

There are three phases in our algorithm. Phase I is to partition the healthy region into a number of disjoint rectangles such that the number of possible XY-routings within rectangles are reasonably large. It is because the XY-routings are optimal. First, find all rectangles and associate each rectangle with its area. Second, with the rectangles as input, we find disjoint rectangles to cover all the healthy nodes so as to maximize the sum of squares of areas. As the sum of squares of areas equals to the sum of the XY-routings plus the total number of healthy nodes which is determined by the input, maximum sum of squares of areas directly is equivalent to maximum sum of XY-routings. We design a greedy algorithm which is a modification from Chvàtal's [6]. The modification is that after a rectangle S giving the maximum area (which is equivalent to giving maximum square of area) is picked, its disjointness with the chosen

rectangles needed to be checked. If disjoint, S is accepted; otherwise, delete S from the rectangle pool and try another rectangle. This step will be executed until we can choose a disjoint rectangle returning maximum area. If there are more than one choice, choose the rectangle with more nodes. If there are still many choices, choose arbitrarily. Figure 1 is an example.

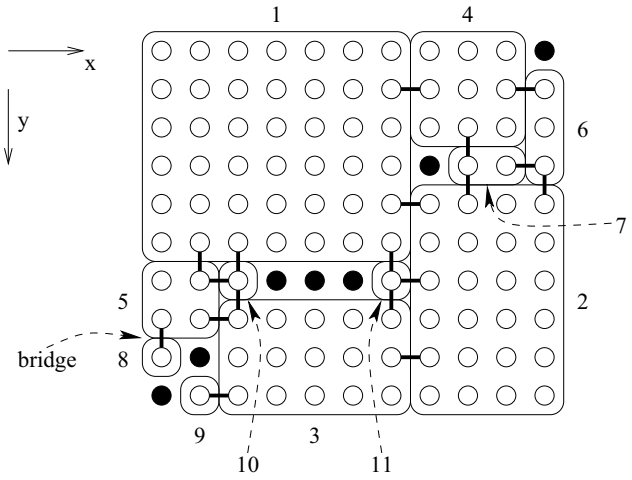

Figure 1. Partitioning a mesh into rectangles

Phase II is to output a spanning tree of rectangles such that the lengths of the routing paths along the tree edges are reasonably bounded. First, form an undirected weighted graph R of rectangles. In the graph, rectangles are nodes, and for any two adjacent rectangles, form an edge in R. In the mesh, choose a link such that the distance between their middle points through the link is minimized. If there are more than one possible choice, choose one arbitrarily for routing while the others are not used. Exact one link is used because we want to avoid cycles. This link is called *bridge*. Figure 1 gives examples that the rectangles are connected by bridges—the thick lines. The weight of the edge of R is defined by the distance between the middle points of the two rectangles. The reason will be given after the discussion of Phase III. Next, find the minimum cost spanning tree by Prim's algorithm [10]. The spanning tree will be used for routing between rectangles, and it is the reason why minimum cost is a requirement. Performance evaluation will be done on different techniques on choosing bridge, and different calculations on the costs of edges.

Phase III is to build the routing table for each healthy node. The routings inside a rectangle are all XY-routings. Consider inter-rectangle routings. First, route to the correct bridge by XY-routing. After crossing the bridge, use again XY-routing to go to the destination or another bridge, depending on whether the destination node resides in this rectangle. The worst case ratio on the path length is clearly $O(MN)$, because we need to avoid any cycle. The worst case is that a routing between neighbours needs to traverse the whole spanning tree.

Now we are back to the weights of edges of R. We consider the weight of an edge between rectangles P and Q as a measure of the cost not only between P and Q, but also between any other rectangles which use them in their communication paths. This cost depends on the position of the chosen link between P and Q, as well as the links connecting them to the other neighbour rectangles. However, it is bounded by the sum of X-lengths and Y-lengths of P and Q, as XY-routing is used inside each rectangle according to Phase III. The whole algorithm is illustrated in Figure III.

We now prove the correctness of the algorithm.

Theorem 1: Each healthy node is reachable if the buffers needed are eventually available.

Proof: We can easily check the completeness and disjointness of the sets of nodes directed by the routing table in each node. We now prove that the routing path between any two healthy nodes is simple.

Suppose the path is not simple and there exists a loop in a routing path. If the loop crosses over two rectangles, the routing table of the bridge in between violates the disjoint property. Therefore, the loop must appear within a single rectangle. The destination of the path must be outside the rectangle; otherwise XY-routing will be applied, and there does not exist any loop. However, according to the construction of the routing tables, XY-routing should be applied to direct the path to the correct and unique bridge, and then go out of the rectangle. There must not be any loop within the rectangle. This results in a contradiction, and completes the argument that the routing path is simple. The theorem follows. ■

Theorem 2: The routing scheme is deadlock-free.

Proof: It suffices to show that there is no buffer-waiting cycle. Assume the contrary that there exists such a cycle $\mathcal{C}$. Recalling that each routing path is simple, $\mathcal{C}$ must be formed by a number of routing paths.

As XY-routing is used for routings within rectangles, $\mathcal{C}$ does not occur in a single rectangle, and it must cross over more than one rectangle. Since the spanning tree of rectangles are used in inter-rectangle routings, there is one rectangle A such that $\mathcal{C}$ enters into A and leaves A through the same bridge b, without passing any other bridges. The paths of $\mathcal{C}$ inside A starts from b, turns at least a round and ends at b. However, there is no YX-routing inside A. Therefore, at each YX-turning, the Y-path must stop at the corner and the subsequent X-path must start at the same point. (Figure 5 gives two examples of YX-turnings.) It then breaks the buffer waiting cycle. A contradiction. ■

IV. Dynamic Faults and Recovery

Whenever there are few more faulty nodes after execution of the above algorithm, we can apply some simple tricks for temporary partitioning. Suppose a new faulty node u appears

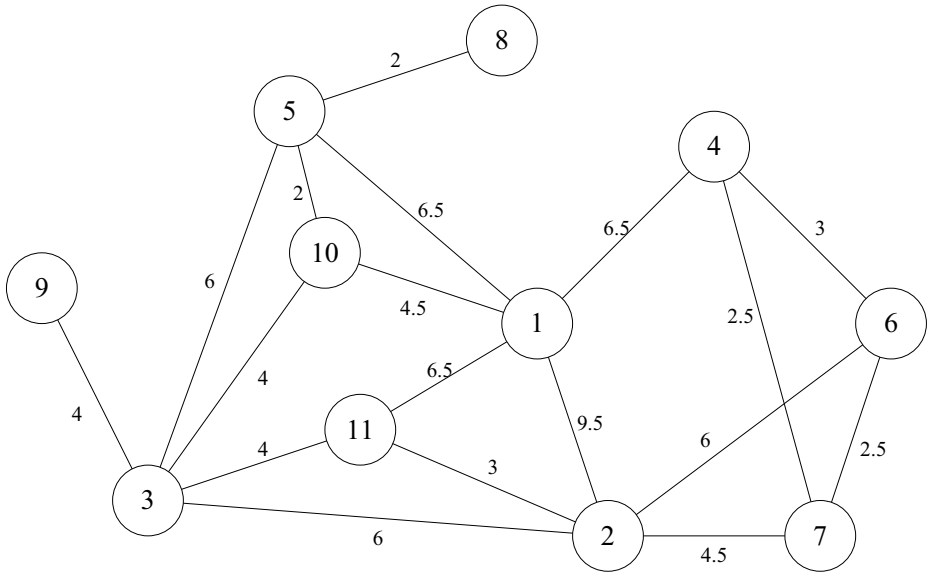

Figure 2. Graph of rectangles for the example in Figure 1

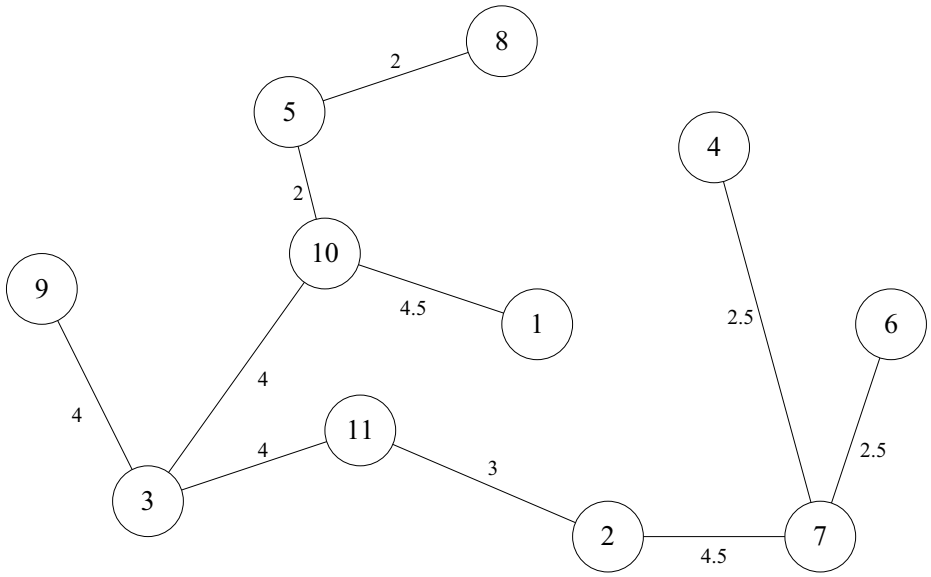

Figure 3. Minimum spanning tree of the above graph

in a rectangle B. Partition B into two to four rectangles, depending on the position of u, and the shape of B. If B is a linear array and the new faulty node appears at its end, then the remaining healthy nodes are a rectangle. In any case, the number of XY-routings inside B needs to be maximized. We use the example in Figure 1 to illustrate this idea and give the partitioning in Figure 6. In this figure, only the new rectangles and their neighbours are shown. B is the original rectangle 3 and u is the black node. The rectangle is partitioned into a, b, c and d. After partitioning, the graph of rectangles and its spanning tree need corresponding changes. First, remove B and its edges from the graph and spanning tree. So the spanning tree may become a forest of trees. Add the new rectangles as nodes and form a larger forest. Put the edges of the new rectangle into a pool of edges, together with the original unused edges, *i.e.*, the edges in the original graph but not in the spanning tree. Start Prim's algorithm, until the forest becomes a new spanning tree or all edges have been tested. In this section, the input is a forest, not nodes. As the new faulty node can disconnect the healthy nodes, there may be more than one spanning trees outputted. These spanning trees may not be minimum, because those unaffected edges

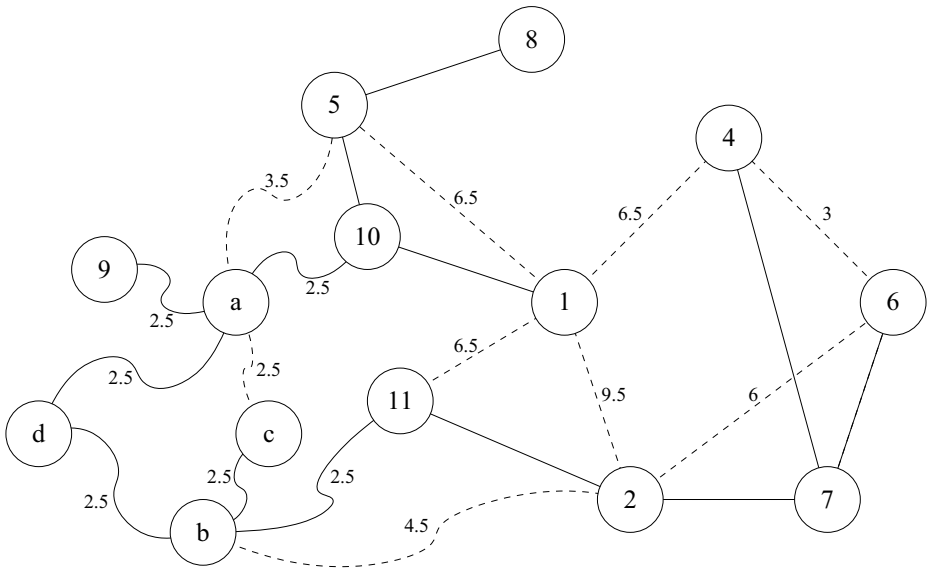

Figure 7. The new graph and its spanning tree.

in the previous tree are kept. Figure 7 shows the graph and spanning tree for the rectangles in the example in Figure 6. The straight lines represent the old edges, and the curves are for new ones. The solid and dotted styles are for chosen and un-chosen edges, respectively. The edges with cost are in the pool for re-building the new tree. The edges with their costs hidden do not participate in this process. Assignment of routing tables is similar as before.

In some situations, a node can be temporarily faulty, and will be recovered by itself shortly. Therefore, we need a dynamic approach to accept the return of a node. The new node can be treated as a rectangle and attached to one of its neighbours. However, it is not efficient in terms of benefit. We design a rectangle enlargement scheme for this purpose. Since it is a dynamic scheme, only the neighbours of the returned node are considered. The new healthy node and all its neighbours are grouped together and form a larger healthy region. Similar to Phase I, find all possible rectangles in this healthy region, and apply the modified Chvàtal's algorithm. Figure 8 shows the recovery of a faulty node in the example given in Figure 1. The rectangles with thicker surroundings are new. After the new rectangles are formed, perform the same procedures as for new faulty nodes. That is, remove the old nodes and their edges from the graph and its spanning tree, and then by Prim's algorithm, choose the minimum cost edges from the new and old unused edges to connect the new nodes to the forest and form a new spanning tree. Figure 9 shows the spanning tree for the above example. The meanings of solid and dotted, lines and curves are the same as above. The resulting spanning tree may not be minimum as the chosen edges in the existing forest or partial spanning tree may have higher weights. As for the purpose of dynamic recovery, "local" minimum is chosen.

V. Concluding Remarks

We propose an algorithm for the wormhole deadlock-free routing problem in faulty 2D-meshes. We consider asynchronous mesh and all-to-all routing between connected non-faulty nodes. The contribution of our algorithm is that it employs no virtual channel and can be applied to any fault pattern.

Although the problem has been studied for years, research is still much needed for the following areas, which include a metric for the performance of routing and one for the computational complexity for preprocessing. The performance of routing can be measured by the number of virtual channels used, the number of non-XY-routings used where XY-routing is unblocked, and their tradeoffs. It is a measure of the price for deadlock-free routing. The tradition stretch factor may not tell the quality of routing because $O(MN)$ is unavoidable for avoiding cycles. The computational complexity for constructing routing tables or rules can be a performance measurement. It further falls into categories of online, offline, centralized and distributed manners.

Another direction of research can be focused on the heterogeneities of faults. For example, we can allow some faulty nodes to forward messages in X-direction only, and some undirected edges to become directed due to faults. There are still much rooms for research.

Acknowledgment

This work is supported by Project 28046 under Istanbul University Scientific Research Project Unit (BAP), Istanbul University.

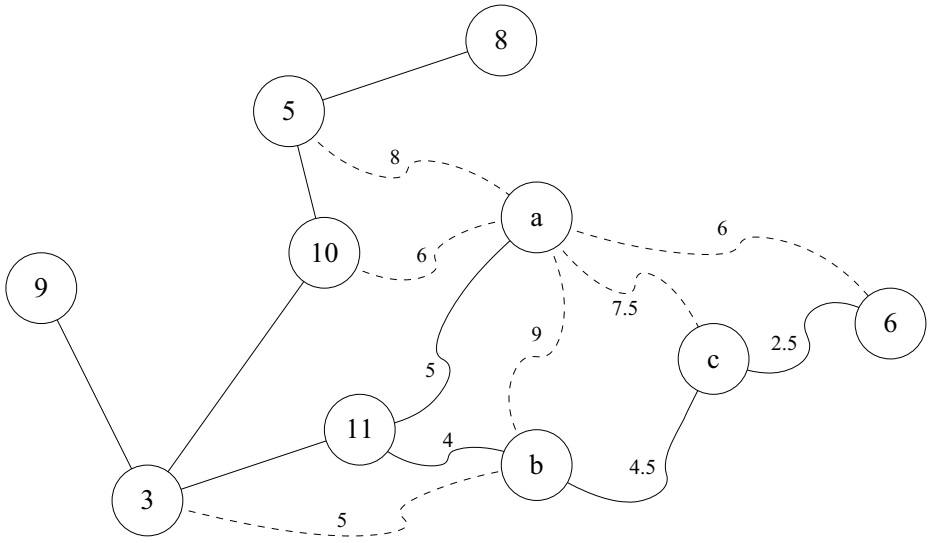

Figure 9. The new spanning tree, which is not necessarily minimum.

REFERENCES

[1] N.R. Adiga et al. "Blue Gene/L Torus Interconnection Network", IBM J. Research and Development, Vol.49, pp. 265-276. Mar.-May 2005.

[2] [1] F. Allen et al., "A Version for Protein Science Using Peta op Supercomputer", IBM Systems J., Vol. 40, pp.310-327, 2001.

[3] R.V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," IEEE Trans. Computers, Vol.44,No.7, pp.848-864, July, 1995.

[4] R.V. Boppana and S. Chalasani, Fault-tolerant communication with partitioned dimension-order routers," IEEE Trans. on Parallel and Distributes systems, Vol.10, No.10, pp. 1026-1039, Oct. 1999.

[5] S. Chalasani and R.V. Boppana, "Communication in multicomputers with nonconvex faults," IEEE Trans. Computers, Vol.46, No.5, pp.616-622, May 1997.

[6] V. Chvàtal, "A greedy heuristic for the set covering problem", *Mathematics of Operation Research* 4:233–235, 1979.

[7] Y. Fukushima, M.Fukushi, I.E. Yairi, and T. Hattori, "A hardware-oriented fault-tolerant routing algorithm for irregular 2D-mesh network-on-chip without virtual channels", 2010 25th International Symposium on Defect and Fault Tolerance in VLSI Systems, Kyoto, Japan, Oct. 6-8, 2010, pp.52-59.

[8] Intel.(2010). The single-chip cloud computer[online]. Avaiable: http://techresearch.intel.com/article/Tera-Scale/142.htm.

[9] Z. Jiang, J.Wu, and D. Wang, "A new fault information model for fault-tolerant adaptive and minimal routing in 3D networks", IEEE Trans. reliability, Vol.57, No.1, Mar. 2008, pp.149-162.

[10] J.B. Jr. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem", *Proc. AMS*, 7:1, 48–50.

[11] S.S. Mukherjee, R. Bannon, S. Lang, and A. Spink, "The Alpha 21364 Network Archi- tecture,", IEEE Micro, pp.26-35, 2002.

[12] S. Rodrigo, J.Flich, A. Roca, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato, "Addressing manufacturing challenges with cost-effi cient fault tolerant routing", 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip, Grenoble, France, May 3-6,2010, pp. 25-32.

[13] S. Rodrigo, J.Flich, S. Medardoni, D. Bertozzi, J. Camacho, F. Silla, and J. Duato, "Cost-effi cient on-chip routing implementations for CMP and MPSoC systems", IEEE Trans. on Computer-aided design of integrated circiuts and systems, Vol.30, No.4,April 2011, pp. 534-547.

[14] F. Safaei and A. Mortazavi, "A novel routing algorithm for achieving static fault-tolerance in 2-D meshes", 2010 10th IEEE International Conference on Computer and Information Technology(CIT 2010), June 29-July 1, Bradford, UK, pp.2621-2627.

[15] S.L. Scott, "Synchronization and communication in the T3E multiprocessor", Proc. of ASPLOS 7, Otc. 1996, pp. 26-36.

[16] P.H. Sui and S.D. Wang, "An improved algorithm for fault-tolerant wormhole routing in meshes," IEEE Trans. Computers, Vol.46, No.9, pp. 1040-1042, Sept. 1997.

[17] M.J. Tsai, "Fault-tolerant routing in wormhole meshes", Journal of Interconnection Networks, Vol. 4, No. 4, pp. 463-495, 2003.

[18] M.J. Tsai and S.D. Wang, "Adaptive and deadlock-free routing for irregular faulty patterns in mesh multicomputer", IEEE Trans. Parallel Distributed Systems. 11(2000), 50-63.

Initially, P, V, E, T, U and W are $\emptyset$.

Phase I:
1. For each healthy node a
1.1 Find all possible rectangles with top-left coordinates at a, and store them in rectangle pool P;
1.2 Associate each rectangle with its area;
2 While $P \neq \emptyset$ do
2.1 Find the rectangle A with maximum area from P;
2.2 $P := P \setminus \{A\}$;
2.3 If A is disjoint with all rectangles in V
2.3.1 $V := V \cup \{A\}$;

Phase II:
3. For any pair of rectangles in V
3.1 If they are adjacent to each other
3.1.1 Choose a bridge arbitrarily;
3.1.2 Create an edge with weight equal to the sum of their X-lengths and Y-lengths;
3.1.3 Store the edge into E;

// Prim's algorithm
4. Form a graph $G := (V, E)$;
5. Choose an arbitrary node in V and store it in U and all its edges in W;
6. While $W \neq \emptyset$ and $U \neq V$ do
6.1 Find the minimum-weight edge e from W;
6.2 $W := W \setminus \{e\}$;
6.3 If e is connected to a rectangle $A \notin U$
6.3.1 $U := U \cup \{A\}$; $T := T \cup \{e\}$;
6.3.2 Store all edges of A except e into W;
// T will be the set of edges in the minimum spanning tree.

Phase III:
7. For each rectangle $A \in V$
7.1 Associate each rectangle in $V \setminus \{A\}$ to the corresponding bridge according to the minimum spanning tree;
7.2 For each node a in A
7.2.1 For each node $b \neq a$
7.2.1.1 If b is in A
7.2.1.1.1 Add to the routing table that XY-routing is used;
7.2.1.2 If b is not in A
7.2.1.2.1 Add to the routing table that XY-routing is used to direct the messages for b to the corresponding bridge; and simply cross the bridge if a is already its end node.

Figure 4. The algorithm

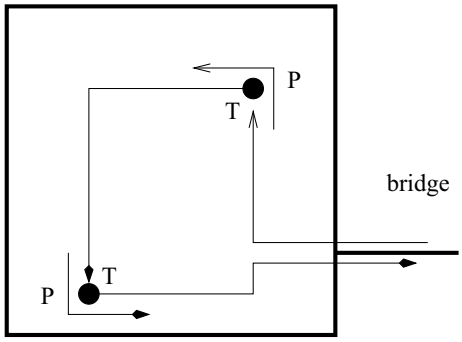

Figure 5. P: Forbidden paths. T: Turning points.

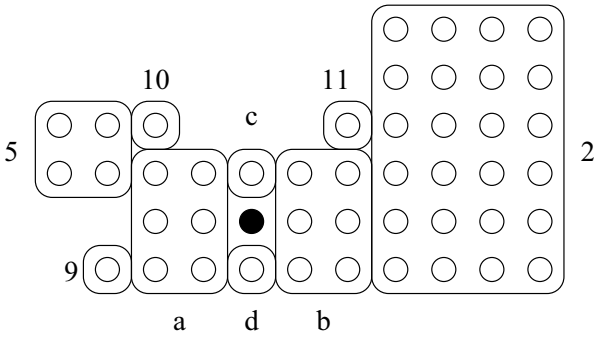

Figure 6. Partitioning of rectangle 3.

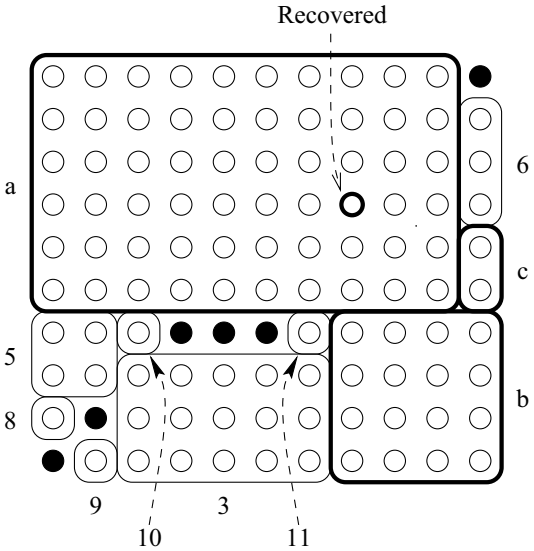

Figure 8. Re-confi guration of rectangles.

[19] D.Wang, ”A rectilinear-monotone polygonal fault block model for fault-tolerant min- imal routing in mesh”, IEEE Trans. Computers, Vol. 52, No.3, pp.310-320, March 2003.

[20] J.X. Wang, F.F. Fu, T.S. Zhang, and Y.P. Chen, ”A small Granularity solution on fault-tolerant in 2D-mesh network-on-chip”, 2010 10th IEEE International Conference on Solid-State and Integrated Cirtuit Technology (ICSICT), Nov. 1-4, 2010, Shanghai, China, pp.382-384.

[21] J. Wu, ”A fault-tolerant and deadlock-free routing in 2D meshes based on odd-even turn model”, IEEE Trans. Computers, Vol. 52, No.9, pp.1154-1169, Sept. 2003.

[22] D. Xiang, J.C. Sun, J. Wu, and K. Thulasiraman, ”Fault-tolerant routing in meshes/tori using planarly contructed fault blocks”, Proc. of 34th International Conference on Parallel Processing(ICPP 2005), Oslo, Norway, June 14-17, 2005, pp.577-584.

[23] D. Xiang,Y. Zhang, Y. Pan, and J. Wu, ”Deadlock-free adaptive routing in meshes based on cost-effective deadlock avoidance schemes”, Proc. of 36th International Conference on Parallel Processing(ICPP 2007), Xian, China, Sept, 10-14, 2007.

[24] D. Xiang, ”Deadlock-free adaptive routing in meshed with fault-tolerant ability based on channel overlapping”, IEEE Trans. on Dependable and Secure Computing, Vol. 8, No.1, Jan.-Feb. 2011, pp.74-88.