GPAA 103317—24/1/2005——134051

The International Journal of Parallel, Emergent and Distributed Systems,
Vol. 00, No. 0, 2005, 1–14

Fault-tolerant and deadlock-free routing in 2-D meshes using rectilinear-monotone polygonal fault blocks

JIE WU*† and DAJIN WANG‡

†Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA
‡Department of Computer Science, Montclair State University, Upper Montclair, NJ 07043, USA

We propose a deterministic fault-tolerant and deadlock-free routing protocol in 2-dimensional (2-D) meshes based on Wu's fault-tolerant odd–even turn model and Wang's rectilinear-monotone polygonal fault block model. The fault-tolerant odd–even turn protocol, also called *extended X–Y routing*, was originally proposed to achieve fault-tolerant and deadlock-free routing among traditional, rectangular fault blocks. It uses no virtual channels. The number of faults to be tolerated is unbounded as long as nodes outside fault blocks are connected in the mesh network. The recently proposed rectilinear-monotone polygonal fault blocks (also called *minimal-connected-components* or *MCCs*) are of the polygonal shapes, and are a refinement of rectangular fault blocks. The formation of MCCs depends on the relative locations of source and destination, and MCCs include far fewer healthy nodes in resultant fault blocks. In this paper, we show that with a simple modification, the extended X–Y routing can also be applied to 2-D meshes using extended MCCs.

Keywords: Deadlock-free routing; Deterministic routing; Fault models; Fault tolerance; Turn models; Virtual channels

1. Introduction

Mesh topology is one of the most important interconnection networks. This topology possesses many attractive properties. Multicomputer systems using mesh topology as their underlying architecture have been around for years. Because of its importance in achieving high performance, fault-tolerant computing for mesh topology has been the focus of extensive literature. A very important area of mesh fault tolerance is its ability to route packets from a source to a destination, while avoiding all faulty nodes in the system. Routing algorithms are either *deterministic* or *adaptive*. Deterministic routing uses only one path to route packets from a source to a destination, while adaptive routing makes use of many different routes. Most commercial systems use deterministic routing because of its deadlock freedom and ease of implementation. X–Y routing is an example of dimension-order routing used in 2-dimensional (2-D) meshes and tori. In X–Y routing, the packet is routed first in the x dimension and then in the y dimension. Unfortunately, X–Y routing is not fault-tolerant, and it cannot tolerate even a single fault.

*Corresponding author. Email: jie@cse.fau.edu

The International Journal of Parallel, Emergent and Distributed Systems
ISSN 1744-5760 print/ISSN 1744-5779 online © 2005 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/17445760500033341

GPAA 103317—24/1/2005——134051

Designing a deterministic routing protocol that is both fault-tolerant and deadlock-free poses a major challenge. The *wormhole switching* [8] technique used in the latest generation of multicomputers is subject to more deadlock than packet switching. In addition, wormhole switching tends to support routing with less fault tolerance. Wormhole routing divides a message into packets and packets into flits. It then routes flits through the network in a pipelined fashion. When the header flits reach a node that has no output channel available, all of the flits are blocked where they are (in place). A deadlock occurs when some packets from different messages cannot advance toward their destinations because the channels requested by them are not available. All the packets involved in a deadlocked configuration are blocked forever. Deadlock avoidance is a commonly used approach in which channels are granted to a packet in such a way that a request never leads to a deadlock.

To achieve fault tolerance, faults are normally contained in a set of disjoint rectangular regions called *fault blocks*. The convexity of fault blocks facilitates a simple design for deadlock-free routing. To design a deadlock-free routing, *virtual channels* [4] are generally used to provide a certain degree of routing freedom when routing around a fault block. However, using virtual channels has some disadvantages, for example, the routers based on virtual channels require more gates and time compared with those not based on virtual channels.

Many studies have been done on deadlock-free routing in 2-D meshes based on the fault block model [1,2,6,7,9,10,11,14]. Most approaches try to reduce either the number of non-faulty nodes in a faulty block by considering different types of fault regions or the number of virtual channels. Thus far the best results can reduce the number of virtual channels to two or three depending on the type of faulty blocks used. To our knowledge, there is no deadlock-free dimension-order routing that can tolerate an unlimited number of faults without using virtual channels.

In this paper, we apply Wu's extended X−Y routing to 2-D meshes that use a new fault block model called minimal-connected-component (MCC). The *extended X−Y routing* is a deterministic fault-tolerant and deadlock-free routing protocol in 2-D meshes. It is based on the well-known X−Y routing and the recently proposed *odd−even turn model* [3], which is an extension to Glass and Ni's *turn model* [5] where certain types of turns are prohibited to avoid deadlock. The extended X−Y routing does not use any virtual channels. The number of faults to be tolerated is unbounded as long as nodes outside fault blocks are connected in the resultant mesh network. However, a conservative approach is used in constructing a fault block; that is, many non-faulty nodes are unnecessarily contained in the block in order to form a rectangular shape block.

The MCC model was proposed by Wang [12] as a refinement to the rectangular fault block model. Basically, fault blocks in MCC are of the so-called "rectilinear-monotone" polygonal shapes. The size of fault blocks are greatly reduced by considering the relative locations of source and destination nodes. The original purpose of MCC is to facilitate minimal routing in the presence of faults—a node will be included in a MCC block only if using that node in a routing will definitely make the route non-minimal. It turns out that MCC block is the absolute minimal fault block; that is, no healthy node contained in MCC will be useful in any routing, minimal or non-minimal.

In this paper, we show that the extended X−Y routing can also be applied to 2-D meshes using an extended MCC model through a simple modification. The following are assumptions used in this paper: (1) The fault model is static, that is, no new faults occur

GPAA 103317—24/1/2005——134051

during a routing process. (2) Both source and destination nodes are outside any fault block. In addition, the destination is not a boundary node of any fault block. (3) Faults are at least 2 hops away from the four edges of a mesh. (4) A connected 2-D mesh is used with four directions: North (positive x), south (negative x), east (positive y), and west (negative y). (Unlike the convention, the north–south axis is used here as the x axis to match the notation used in the odd–even turn model.)

The remainder of this paper is organized as follows: Sections 2 and 3 give introduction to odd–even turn, extended X–Y routing, and MCC fault model. Section 4 first extends the definition of MCC model to accommodate deadlock-free routing and, then, applies the extended X–Y routing to 2-D meshes using extended MCC model. Section 5 concludes the paper and discusses possible future work.

2. 2-D meshes and odd–even turn model

In this section, we first briefly review the mesh topology. We then give an introduction to Chiu's odd–even turn model [3], which is an extension to Glass and Ni's turn model.

2.1 2-D meshes

A 2-D mesh with $n \times n$ nodes has an interior node degree of 4. Each node u has an address u:(u_x,u_y), where u_x,$u_y \in \{0,1,2,\ldots,n-1\}$. Two nodes u:(u_x, u_y) and v:(v_x, v_y) are connected if their addresses differ in one and only one dimension, say x, and $|u_x - v_x)| = 1$. Each non-boundary node in a 2-D mesh has four neighbors, one in each of four directions: East, south, west, and north.

Routing is a process to send *packets* from a source node to a destination node, passing some intermediate nodes. There are basically two types of routing: *Deterministic* routing and *adaptive* routing. In deterministic routing, a fixed path is used to send/receive packets for a particular pair of source/destination. The obvious advantage of this routing is its simplicity and ease of implementation. However, it suffers the shortcoming of weak fault tolerance—when one or more nodes on the dedicated path fail, either routing cannot be carried out, or only very limited detours can be taken. In adaptive routing, there is no dedicated path for a pair of source and destination. The path is adaptively constructed in the process of routing. However, the algorithm/implementation for adaptive routing is more complicated.

2.2 Odd–even turn model

Both deterministic and adaptive routings suffer from possible deadlocks when multiple packets are flowing in the mesh. Many strategies, with or without resorting to extra resources, have been proposed. Recently, Chiu [3] proposed an odd–even turn model, an extension to Glass and Ni's turn model [5], to prevent deadlocks without using the popular means of virtual channels. In general, deadlock avoidance tries to avoid the formation of a cycle of dependencies, which is a necessary condition for deadlock. Such deadlock is also called *communication deadlock* where routing packets are (active) processes and channels are resources. A cycle in a mesh consists of several (at least four) turns. As illustrated in figure 1,

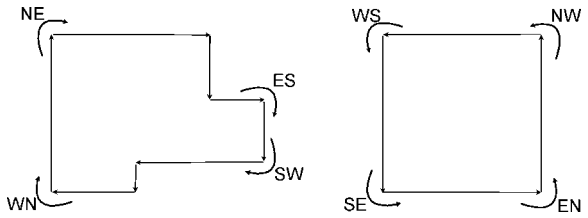

Figure 1. SW, WN, NE, and ES turns are essential in forming a clockwise cycle. Dually, to form a counterclockwise cycle, SE, EN, NW, and WS turns are necessary.

south−west (SW), WN, NE, and ES turns are essential in a clockwise cycle. Likewise, to form a counterclockwise cycle, SE, EN, NW, and WS turns are necessary.

In Chiu's odd−even turn model, certain types of turns are prohibited at certain columns. Refer to figure 1 again: In clockwise cycle, if we always command an ES turn and SW turn to take place at different columns, then the essential *rightmost column segment* of a cycle will never be formed. Similarly, in counterclockwise cycle, if we always command EN and NW turns to take place at different columns, then the essential rightmost column segment of a cycle will never be formed. That is the basic idea of the odd−even turn model.

In odd−even turn model, the above four types of turns are disallowed at certain columns: Two in a clockwise cycle and two in a counterclockwise cycle. The basic concept behind the turn model is to keep disallowed turns as few as possible to increase the routing adaptivity. Chiu gives two rules for making turns [3]:

Rule 1 (to prevent counterclockwise cycle): A packet is not allowed to take an EN turn at any node located in an even column, and it is not allowed to take an NW turn at any node located in an odd column.

Rule 2 (to prevent clockwise cycle): A packet is not allowed to take an ES turn at any node located in an even column, and it is not allowed to take an SW turn at any node located in an odd column.

Figure 2 shows the EN, NW, ES, and SW turns and the columns at which these turns are allowed under the above two rules. These four turns are called *sensitive turns*. Turns without

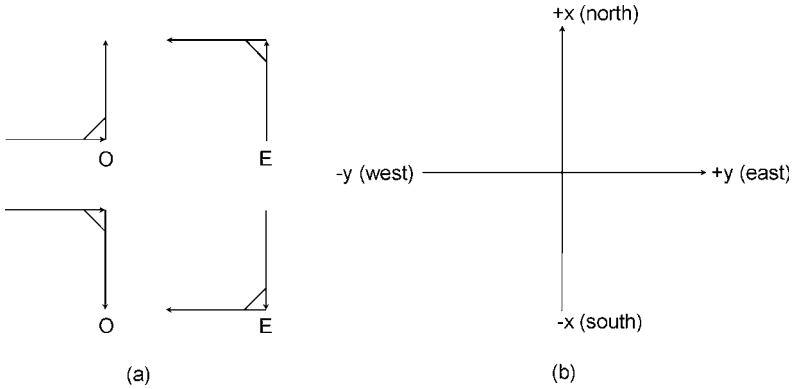

Figure 2. (a) EN, ES turns are permissible only at odd columns; NW, SW turns are permissible only at even columns. (b) The coordinate system used in this paper.

GPAA 103317—24/1/2005——134051

restriction are called *insensitive turns*. A small triangle is placed at each sensitive turn that is permissible (as shown in figure 2). A turn in an even (odd) column is represented by E(O).

3. Extended X−Y routing and MCC fault block model

3.1 Extended X−Y routing

In the fault-tolerant, deadlock-free routing in [13], called extended X−Y routing, a set of disjointed rectangular fault blocks is first constructed. The distance between two fault blocks along the *x* dimension (column) is 2 and that along the *y* dimension (row) is 3. The 3-hop distance requirement along the *y* dimension is to ensure that two boundary lines (one even column and one odd column) can be constructed at the east and west sides of each fault block. All faulty and non-faulty nodes are initially labelled *faulty* and *fault-free*, respectively. A fault-free node is changed to *disabled* if it has two faulty or disabled neighbors that are not all in the *x* dimension; or it has a faulty or disabled neighbor in the *x* dimension and a faulty or disabled 2-hop neighbor (neighbor's neighbor) in the *y* dimension. A fault block consists of connected faulty and disabled nodes.

The extended X−Y routing consists of two phases, similar to a regular X−Y routing. In phase 1, the offset along the *x* dimension is reduced to zero, and in phase 2, the offset along the *y* dimension is reduced to zero. Assume source and destination nodes are both fault-free.

Extended X-Y routing:

1. /* the packet is sent to an even column before phase 1 starts */
   (a) If the source is in an odd column and Δ_x is non-zero, then the packet is sent to its west neighbor in an even column.
2. /* phase 1: reduce Δ_x, the offset in the x dimension */
   (a) (Normal mode) reduce Δ_x to zero by sending the packet north (or south). (with no 180° turn).
   (b) (Abnormal mode) when a north-bound (south-bound) packet reaches a boundary node of a fault block, it is routed around the block clockwise (counterclockwise) by following the boundary ring of the fault block as shown in Figure 4 (a) (Figure 4 (b)). The packet takes the first even column turn whenever possible and step (a) is followed.
3. /* phase 2: reduce Δ_y, the offset in the y dimension */
   (a) Once Δ_x is reduced to zero, a NW or NE turn is performed for the north-bound packet (see Figure 4 (a)) and a SW or SE turn is performed for a south-bound packet (see Figure 4 (b)). The selection of a turn depends on the relative location of the destination to the current node.
   (b) (Normal mode) reduce Δ_y to zero by sending the packet east (west) (with no 180° turn).
   (c) (Abnormal mode) when an east-bound (west-bound) packet reaches a boundary node of a fault block, it is routed around the block, clockwise or counterclockwise, along odd columns of the boundary ring as shown in Figure 4 (c) (even columns of the boundary ring as shown in Figure 4 (d)). Routing around the block is completed when Δ_x is again reduced to zero and step (b) is followed.

Figure 3. Extended X−Y routing.

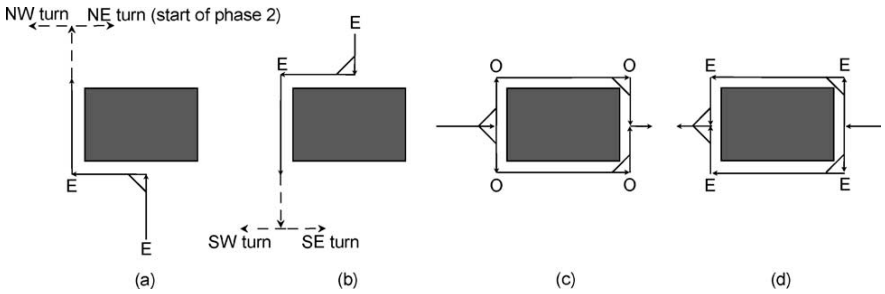

Figure 4. Two cases of routing along the x dimension (column) (a) north-bound and (b) south-bound; and two cases along the y dimension (row) (c) east-bound and (d) west-bound.

Let s:(s_x, s_y) and d:(d_x, d_y) be the source and destination nodes, respectively. $\Delta_x = |d_x - s_x|$ and $\Delta_y = |d_y - s_y|$ are offsets along dimension x and dimension y, respectively.

The extended X–Y routing protocol, shown in figure 3, follows the regular X–Y routing (and the packet is in a "normal" mode) until the packet reaches a boundary node of a fault block. At that point, the packet is routed around the block (and the packet is in an "abnormal" mode) clockwise or counterclockwise based on the following rules: When a routing packet routes around a fault block following the boundary ring, the corresponding block is called the *routing block*. During phase 1 [refer to figure 4(a) and (b)] the packet is routed around the routing block through the west side of the block. Even columns are used to route the packet along the x dimension (column). In phase 2 [refer to figure 4(c) and (d)], odd columns (even columns) will be used to route around a routing block when the packet is east-bound (west-bound). The packet can be routed around the routing block either clockwise or counterclockwise, since either way is permissible. During the normal mode of routing the

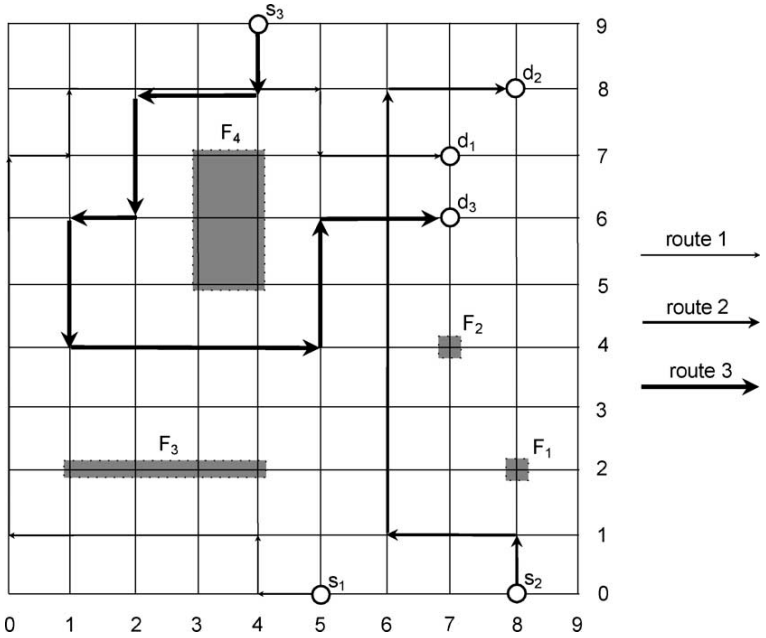

Figure 5. Three routing examples in a 10 × 10 mesh with four fault blocks.

GPAA 103317—24/1/2005——134051

packet along the $x(y)$ dimension, 180° turn is not allowed. For example, the positive x direction cannot be changed to the negative x direction. Note that in extended X−Y routing, south-bound corresponds to quadrant III or quadrant IV while north-bound corresponds to quadrant I or quadrant II.

Referring to figure 2 and figure 4, it can be seen that the rules of turn-making are followed during the block-avoiding manoeuvring. In [13], Wu provided a proof that the extended X−Y routing is both deadlock-free and livelock-free in a 2-D mesh where faults are contained in a set of disjointed fault blocks. Figure 5 shows three routing examples (s_i,d_i), $i \in \{1,2,3\}$, in a 10 × 10 mesh with four fault blocks F_1, F_2, F_3, and F_4.

3.2 MCCs

In each step of routing, a node can take one and only one direction to its immediate neighbor. Therefore, a routing can be represented as a sequence of directions. Minimal routing in a 2-D mesh can take at most two directions (the corresponding routing is also called *Manhattan routing*). For example, if the destination is in the first quadrant with respect to the source (origin of the coordinate system), only north and east directions are used. The formation of MCC fault block is based on the notions of *useless* and *can't-reach* nodes: A node labelled useless is such a node that once it is entered in a routing, the next move must take either west or south direction, making a Manhattan routing (in the first quadrant) impossible. A node labelled can't-reach is such a node that to enter it in a routing, a west or south move must be taken, making a Manhattan routing impossible. The node status (faulty, fault-free, useless, and can't-reach) can be determined through a labelling procedure. Connected faulty, useless, and can't-reach nodes form a MCC. The labelling procedure given in figure 6 is for quadrants I routing, i.e. the destination is in quadrant I (northeast). The labelling procedure for quadrant III (the destination is in southwest) can be obtained just by exchanging the roles of useless and can't-reach. In fact, MCCs generated from quadrant I labelling procedure and from quadrants III labelling procedure are the same, and they are called *type-one MCCs*.

The labelling procedure for quadrant II (the destination is in northwest) can be derived from that for quadrant I by exchanging the roles played by east and west neighbors. Again, the labelling procedure for quadrant IV is from the one for quadrant II by exchanging the role of useless and can't-reach. MCCs generated from quadrants II and IV are the same, and they

Labelling procedure for MCC:

1. Initially, label all faulty nodes as *faulty* and all non-faulty nodes as *fault-free*;
2. If node u is fault-free, but its north neighbor and east neighbor are faulty or useless, u is labelled *useless*;
3. If node u is fault-free, but its south neighbor and west neighbor are faulty or can't-reach, u is labelled *can't-reach*;
4. The nodes are recursively labelled until there are no new useless or can't-reach nodes.

Figure 6. Labelling procedure for MCC block for quadrant I destination.

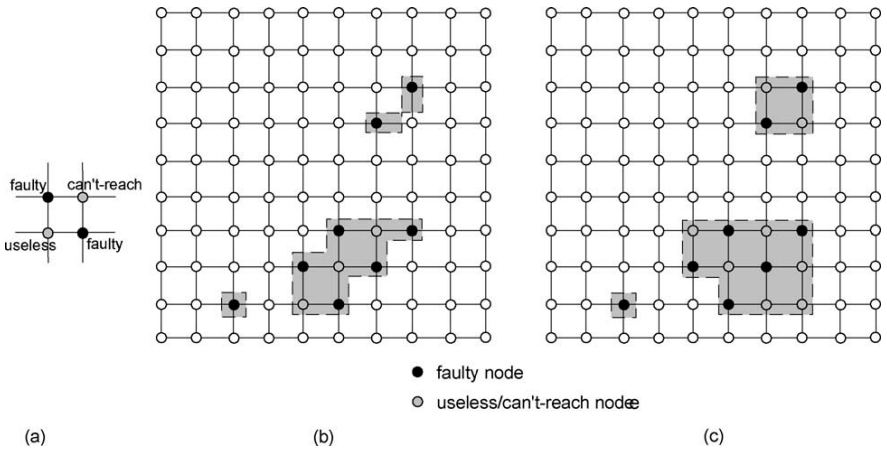

Figure 7. (a) Definition of useless and can't-reach nodes for quadrant I destination. (b) Sample type-one MCCs. (c) Sample type-two MCCs.

are called *type-two MCCs*. Note that useless and can't-reach nodes correspond to disabled nodes in the regular fault block model. Figure 7(a) shows the definition of useless and can't-reach nodes (for quadrant I destination), figure 7(b) shows sample type-one MCCs (for quadrants I and III destinations). For the same faulty nodes, the corresponding type-two MCCs (for quadrants II and IV destinations) are shown in figure 7(c). Figure 8 gives the general shapes of MCCs for routing in the four quadrants.

MCC was originally proposed to facilitate minimal routing (Manhattan routing). MCC block is a *minimum* connected block, so that a node v of the MCC is either faulty, or if fault-free, then entering v would force a step that violates the requirement for a Manhattan route (i.e. a third direction would be used in the route). In other words, the MCC is a fault block model that provides the *maximum* possibility to find a Manhattan route among faults. Therefore, if there exists no Manhattan route under MCC fault model, there will be absolutely no Manhattan route. In [12], a sufficient and necessary condition was provided for the existence of Manhattan routes in presence of MCCs.

Experimental results showed that MCC model includes far fewer non-faulty nodes in fault blocks than the conventional rectangular model. Many non-faulty nodes that would have been included in rectangular fault blocks now can become candidate routing nodes. As a matter of fact, MCC block is the "ultimate" minimal fault block; that is, no fault-free node contained in MCC will be useful in any routing, minimal or non-minimal.

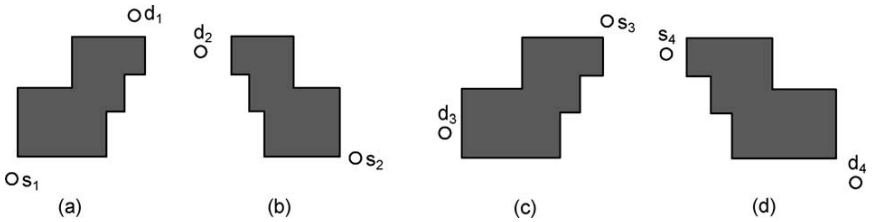

Figure 8. (a) MCC for quadrant I routing (type-one); (b) MCC for quadrant II routing (type-two); (c) MCC for quadrant III routing (type-one); (d) MCC for quadrant IV routing (type-two).

GPAA 103317—24/1/2005——134051

4. Extended X−Y routing in 2-D meshes with extended MCCs

4.1 Extended MCCs

By the definition of MCC, the distance between two MCCs is at least 2 hops along each dimension. To ensure that the distance is at least 3 along the *y* (row) dimension, we give a definition for *extended MCCs* to ensure two boundary columns (at east and west) of each MCC. In the labelling procedure for extended MCC, a new label *connector* is used to connect two MCCs that are 2 hops away along the *y* (row) dimension. Again, initially all faulty nodes are labelled *faulty* and non-faulty nodes are labelled *fault-free*. The nodes are recursively labelled until there are no new useless, can't-reach, connector nodes. All useless, can't-reach, and connector nodes are disabled (i.e. included in extended MCCs) and the remaining fault-free nodes are not included in any extended MCCs.

The labelling procedure for extended MCC is given in figure 9. In figure 10, the role of connector nodes is illustrated. It can be seen that MCCs, which are 2-hop apart will be merged into one (extended) MCC. Note that the extended MCCs are not minimal fault blocks anymore. However, this modification is necessary to accommodate the application of odd−even turn model. Also, its rectangular counterparts will undergo the same merge process with even more—as a matter of fact many more—fault-free nodes included. Therefore, in the context of odd−even turn model, the extended MCCs are still the smallest fault blocks.

All nodes included in extended MCCs are *disabled* for routing. Note that in the formation of extended MCCs, certain disabled nodes can be labelled either useless (can't-reach) or connector. For example, node (5,3) in figure 11(a) can be either can't-reach or connector. Since useless, can't-reach, and connector are all disabled, the way each node is labelled will make no difference on the shape of each extended MCC.

Since a node may have a different status (i.e. whether it is a MCC or non-MCC node) for quadrant I/III or quadrant II/IV routing, each node carries two status: (*status*1; *status*2) where *status*1 is for quadrant I and III routing and *status*3 is for quadrant II and IV routing. Referring to figure 11 again, the status of node (2,3) is (fault-free, fault-free). That for (3,3) is (fault-free, disabled), for (5,3) is (disabled, disabled), and for (3,1) is (disabled, fault-free). It is assumed that source and destination nodes both have (fault-free, fault-free) status.

Labelling procedure for extended MCC:

1. First, generate all original MCCs;
2. If node u is fault-free, but its east neighbor and west neighbor are both MCC nodes, then u is labelled *connector*;
3. If node u is fault-free, but its north neighbor and east neighbor are faulty/useless/connector, u is labelled *useless*;
4. If node u is fault-free, but its south neighbor and west neighbor are faulty/can't-reach/connector, u is labelled *can't-reach*;
5. The nodes are recursively labelled until there are no new useless, can't-reach, connector nodes.

Figure 9. Labelling procedure for extended MCC block for quadrant I destination.

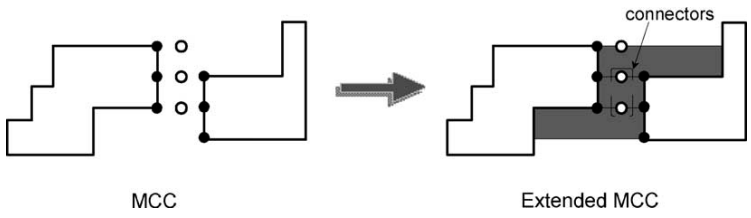

Figure 10. MCCs that are 2-hop apart are merged into one (extended) MCC.

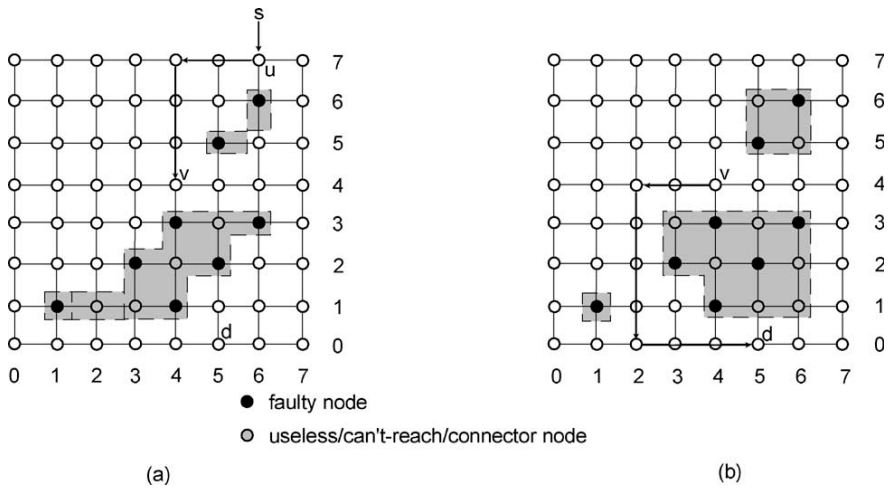

Figure 11. (a) A quadrant III routing from u to d. (b) A quadrant IV routing from v to d.

4.2 Extended X−Y routing in 2-D meshes with extended MCCs

Once the extended MCCs are defined, the extended X−Y routing for rectangular blocks can be applied with some simple modification. In phase 1, the routing proceeds along the x dimension (vertical) at an even column, until a MCC boundary is hit. At that point, the routing takes a "zigzag" detour along the boundary. Notice that SW and NW turns are sensitive turns and can only be made in even columns. Figure 12(a) and (b) illustrate an example of north-bound detouring for quadrant I and quadrant II routing, respectively, and figure 12(c) and (d) illustrate examples of south-bound detouring for quadrant III and quadrant IV routing, respectively.

In phase 2, when a packet reaches a boundary node of an extended MCC, it is routed around the block *clockwise for quadrants I and III routing*, and *counterclockwise for quadrants II and IV routing*. The above rule is to avoid the formation of the rightmost column of a cycle as shown in figure 13(b). Figure 13(a) shows a quadrant I routing and a quadrant III routing, while figure 13(b) shows a quadrant IV routing.

Note that during the routing process, *the selection of block type does not depend on the relative locations of source and destination nodes, but rather relative locations of current and destination nodes*! However, block type of the routing block will not be changed once decided. Figure 11 shows such an example. When a south-bound packet reaches a boundary node u [see figure 11(a)], since destination d is in the quadrant III

GPAA 103317—24/1/2005——134051

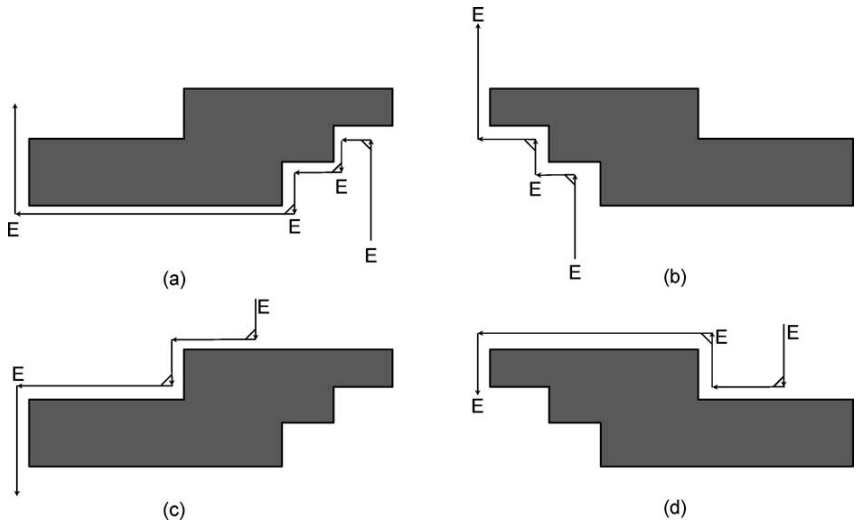

Figure 12. (a) North-bound quadrant I detouring in phase 1; (b) North-bound quadrant II detouring in phase 1; (c) South-bound quadrant III detouring in phase 1; (d) South-bound quadrant IV detouring in phase 1.

(w.r.t. u), type-one MCC is selected. Once the packet reaches node v which is a boundary node of another MCC, since destination d is in the quadrant IV [w.r.t. v, see figure 11(b)], type-two MCC is selected.

THEOREM 1. Using the extended MCC fault block model, the modified extended X–Y routing is deadlock-free and livelock-free.

Proof. In phase 1, assume that the packet is north-bound (the south-bound case can be treated in a similar way), routing around a routing block involves a sequence of the following turns (refer figure 12(a) and (b)): NW*, (WS, SW)*, WN, where NW* represents a zero or more repetition of NW turns, and (WS, SW)* represents a zero or more repetition of WS, SW turns. All sensitive turns NW and SW occur at even columns and they are permissible.

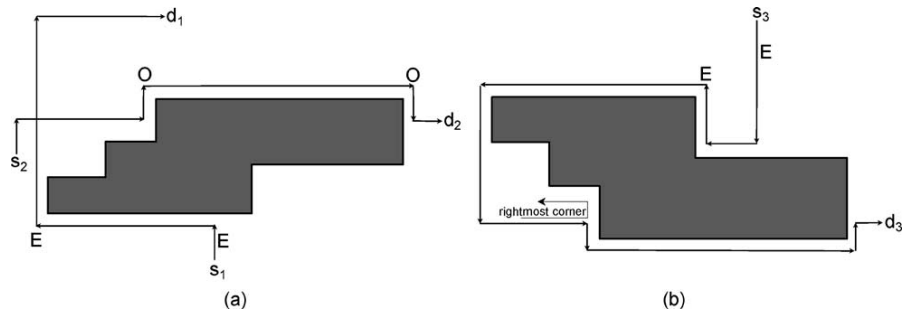

Figure 13. (a) Quadrants I and III routing with extended MCCs. (b) Quadrants II and IV routing with extended MCCs.

GPAA 103317—24/1/2005——134051

In the transition between phase 1 and phase 2, either a NW (permissible) or NE (insensitive) turn is performed in an even column.

In phase 2, assume that the packet is east-bound (the west-bound case can be treated in a similar way), routing around a routing block (if any) involves a sequence of the following turns if the packet is routed in the counterclockwise direction: ES, (SE, ES)*, SE, EN. If the packet is routed in the clockwise direction, the following sequence of turns is used: EN, (NE, EN)*, NE, ES. All sensitive turns are performed in odd columns and they are permissible. Once Δ_x is reduced to zero, either an SE or NE turn is performed in an odd column to reduce Δ_y.

Since all involved turns follow the rules of odd–even model, by the result of [3], the modified extended X–Y routing is deadlock-free and livelock-free. □

The advantage of MCCs over traditional rectangular fault blocks is obvious—MCCs are much smaller than rectangular fault blocks. Basically, a type-one (type-two) MCC is obtained by removing the unnecessarily included fault-free nodes in the SE (SW) and NW (NE) corner sections of a rectangular block.

The MCC model is not the only effort for reducing the size of fault blocks. In [13], Wu also proposed a similar polygonal-shaped fault model, called *orthogonal fault blocks*, which are fault blocks by removing all four corner sections (see figure 14). Although an orthogonal fault block is in general smaller than either type-one or type-two MCC, it has the following drawbacks, which can be overcome by using MCC model.

- To avoid generating the rightmost column of a cycle, a rather complex process is used in the orthogonal fault block model. In that process, *directional information* is associated with boundary nodes of fault blocks to avoid formation of the rightmost column. Directional information tells direction of routing around the block (clockwise or counterclockwise) and such direction is dependent on the location of current boundary node. No such process is needed in the extended MCC model. In other words, the directional information is already *implied* by the type of the MCC.
- Additional detours may occur in orthogonal fault blocks. For example, additional detours occur whenever the quadrant I routing packet enters the SW corner section of the block. Such case will never occur in the extended MCC model.

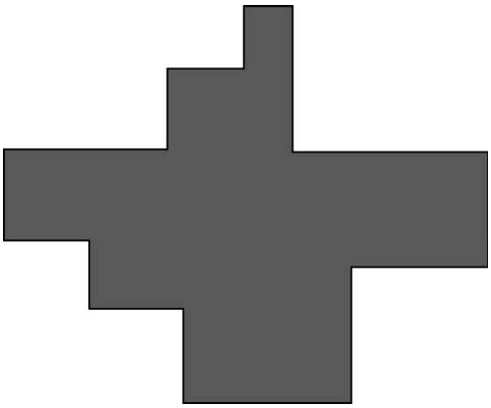

Figure 14. A fault block of orthogonal shape.

GPAA 103317—24/1/2005——134051

The only overhead in the MCC is two different status maintained at MCC node.

5. Conclusion

We have proposed a deterministic, fault-tolerant and deadlock-free routing protocol in 2-D meshes. The protocol is based on Wu's fault-tolerant routing scheme, called extended X−Y routing [13], using odd−even turn model [3] which is a fault-tolerant and deadlock-free routing protocol without using any virtual channels. The recently proposed rectilinear-monotone polygonal model (MCC) [12] has been employed as the fault block model. It has been shown that with a simple modification, the extended X−Y routing can be applied to 2-D meshes using extended MCCs, and the advantage of using MCC model in the context of odd−even turn model has been discussed. Our future work will include performance study to verify the effectiveness of the proposed scheme. An open issue is whether we can devise a more efficient scheme that can remove more non-faulty nodes from the convex-type of fault block, while at the same time can support a fault-tolerant and deadlock-free routing scheme without using virtual channel.

Q3 ## Acknowledgements

The work was supported in part by NSF grants ANI 0083836, CCR 9900646, CCR 0329741, CNS 0422762, CNS 0434533, and EIA 0130806.

References

[1] Boppana, R.V. and Chalasani, S., 1995, Fault-tolerant wormhole routing algorithms for mesh networks, *IEEE Transactions on Computers*, **44**(7), 848−864, July.

Q4 [2] Boura, Y.M. and Das, C.R., 1995, Fault-tolerant routing in mesh networks. *Proceedings of 1995 International Conference on Parallel Processing*, pp. I 106−I 109, August.

[3] Chiu, G.M., 2000, The odd−even turn model for adaptive routing, *IEEE Transactions on Parallel and Distributed Systems*, **11**(7), 729−737, July.

[4] Dally, W.J. and Seitz, C.L., 1987, Deadlock-free message routing in multiprocessor interconnection networks, *IEEE Transactions on Computers*, **36**(5), 547−553, May.

[5] Glass, G.J. and Ni, L.M., 1994, The turn model for adaptive routing, *Journal of ACM*, **40**(5), 874−902, Sept..

[6] Ho, C.T. and Stockmeyer, L., 2004, A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers, *IEEE Transactions on Computers*, **53**(4), 427−438, April.

[7] Kim, S.P. and Han, T., 1997, Fault-tolerant wormhole routing in meshes with overlapped solid fault regions, *Parallel Computing*, **23**, 1937−1962.

[8] Ni, L.M. and McKinley, P.K., 1993, A survey of routing techniques in wormhole networks, *Computer*, **26**(2), 62−76, Feb.

Q4 [9] Shih, J.D., 1997, Adaptive fault-tolerant wormhole routing algorithms for hypercube and mesh interconnection networks. *Proceedings of the 11th International Parallel Processing Symposium*, pp. 333−340, April.

[10] Sui, P.H. and Wang, S.D., 2000, Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes, *IEEE Transactions on Parallel and Distributed Systems*, **11**, 50−63.

[11] Tsai, M.J., 2003, Fault-tolerant routing in wormhole meshes, *Journal of Interconnection Networks*, **4**(4), 463−495.

Q4 [12] Wang, D., 1999, Minimal-connected-component (MCC)—a refined fault block model for fault-tolerant minimal routing in mesh. *Proceedings of IASTED International Conference on Parallel and Distributed Computing and Systems*, pp. 95−100, Nov.

[13] Wu, J., 2000, A deterministic fault-tolerant and deadlock-free routing protocol in 2-d meshes without virtual channels. *Technical Report* (Florida Atlantic University), TR-CSE-00-26, Nov .

[14] Zhou, J. and Lau, F., 2001, Adaptive fault-tolerant wormhole routing in 2d meshes. *Proceedings of the 15th International Parallel & Distributed Processing Symposium (IPDPS 2001)*, p. 56.