

## Corrections to Chen and Chiu's Fault Tolerant Routing Algorithm for Mesh Networks

RICKARD HOLSMARK AND SHASHI KUMAR  
*Department of Computer and Electrical Engineering  
School of Engineering  
Jönköping University  
SE-551 11, Jönköping, Sweden*

Chen and Chiu published a fault tolerant routing algorithm for mesh topology networks [1] which they claimed was deadlock free in the presence of multiple faults. In this paper we give a counter-example to show that their Message-Route algorithm [1] fails to provide deadlock free routing in a 2 dimensional mesh network. We also point out certain cases where the algorithm fails to route messages to their destinations. We identify an error in the proof of the main theorem in their paper [1] which was used for proving the property of deadlock freeness. Changes to their algorithm are proposed to make it deadlock free and complete. We also discuss a new application of fault tolerant routing algorithms for non-homogeneous 2-dimensional mesh topology networks for on-chip communication.

**Keywords:** deadlock, routing algorithms, wormhole routing, fault tolerance, mesh networks, networks on chip

### 1. INTRODUCTION

Two dimensional mesh topology networks have been popular as communication architecture for multi-processor systems in the past; currently they are the most favorite choice for designing on-chip communication among cores in systems on chip (SoC) built using Network on Chip (NoC) paradigm. A regular two dimensional topology like mesh is particularly suitable for layout inside a chip. Efficient routing of messages within the network is essential in order to fully exploit the power of the computing resources and achieve good performance for applications running on them. A good routing algorithm should not only provide low latency for messages but should also be deadlock free when the network is concurrently routing multiple messages.

Designing a routing algorithm which is efficient, deadlock free and also can tolerate faults or failures in the network is a hard problem. Wormhole switching technique used in communication networks is more efficient than store and forward technique but is more prone to deadlocks. Many deadlock free routing algorithms have been proposed for mesh topology networks in literature [2, 3]. These algorithms can be classified into two categories, namely those using virtual channels and those not using virtual channels. Virtual channels have also been used to facilitate design of deadlock free *and* fault tolerant routing schemes [4]. However, the use of virtual channels increase implementation cost as it adds both resources and design complexity. Therefore, some researchers have

---

Received May 13, 2005; revised February 24 & June 12, 2006 & January 22, 2007; accepted May 15, 2007.  
Communicated by David H. C. Du.

proposed algorithms which are both deadlock free and fault tolerant without the use of virtual channels [5, 6]. Algorithm by Chen and Chiu [1] is an algorithm in this category.

There is a new significance to the fault tolerant and deadlock free routing algorithms in the context of NoC paradigm based system design. A homogeneous mesh topology NoC architecture consists of an array of routers. Cores are positioned in the slots left in the layout [7]. Homogeneous mesh topology NoC has many good properties but it has one significant assumption. The assumption is that each core required for the NoC design will fit into the slot within routers. A solution to remove this assumption was proposed through the concept of *Region* [7]. The concept of *Region* makes the mesh network non-homogeneous since routers covered by the region are no more available for routing. Routing algorithms for homogeneous mesh topology NoC may not guarantee communication between all nodes and at the same time, freedom from deadlocks in a non-homogeneous mesh topology. For the purpose of routing, it is possible to consider regions as faulty blocks, where routers may become unavailable due to the placement of regions over them. This allows us to treat the problem of routing in the presence of faults as equivalent to routing in the presence of regions. This equivalence of region and fault help us to reuse available fault tolerant and deadlock free algorithms for message routing in NoC architectures with regions.

Chen and Chiu's algorithm claims to provide deadlock free routing in mesh networks in the presence of multiple faults. Unfortunately, the published algorithm contains some errors and is also incomplete. In this paper, we not only identify the errors in their paper but also present corrections to make it complete. The rest of the paper is organized as follows. In section 2 we describe the basic idea in [1] and describe a situation where the algorithm results in a deadlock. We also show a case where the algorithm is not able to route. In section 3, we present an analysis of the proof of their main theorem which is used for proving the property of deadlock freeness and identify an error. In section 4, we present a corrected algorithm and give proof of its deadlock freeness. Section 5 concludes the paper with highlighting the importance of this algorithm.

## 2. ERRORS IN CHEN AND CHIU'S ROUTING ALGORITHM [1]

Chen and Chiu's algorithm uses the idea of rings and chains borrowed from [4] to isolate the faulty nodes from the rest of the network. Further, in order to avoid deadlocks they allowed none-adaptive routes, thus using only a few turns for the flits from source to destination for normal routing. In the presence of faults it becomes necessary to allow some turns which are forbidden during normal routing. Their idea was to allow only a few combinations of forbidden turns in a clever manner such that these turns can never combine with each other (or allowed normal turns) to form a cycle.

Their algorithm has the following two problems:

- (i) In certain situations, the turns combine to form a cycle which can lead to a deadlock. Therefore, their algorithm is **not deadlock free**.
- (ii) There are certain nodes which can not be reached from a set of source nodes using this algorithm. We call routing algorithms which cannot find a path between two nodes when a path exists as **incomplete**. Their algorithm is also incomplete in this sense.

In the next two subsections, we give examples to show that their algorithm can result in a deadlock and do not route messages from a node to another node when a path actually exists.

## 2.1 Possibility of Deadlock

We give an example to show that a deadlock can occur using the algorithm presented in [1]. Consider Fig. 1 where four different messages are traveling in the network. The channels between nodes have been omitted for increased readability. The faulty nodes are colored dark and messages are routed around them using either the rules of a fault ring or a fault chain (*s-chain*). The four faulty nodes at the south boundary are surrounded by a special chain called *s-chain*. The three faulty nodes not touching any border are surrounded by a fault-ring. A message is denoted as a source and destination pair ( $S_i, D_i$ ). In [1] messages are classified as column first (CF), row first (RF) or row only (RO), depending on the relative locations of the source and the destination of the message. A CF message travels to north or south from its current position before taking a turn to the east. An RF message travels to the west before taking a turn to north or to south. An RO message always travels to east and do not take a turn. Further, Normal-Route, Ring-Route and Chain-Route are sub-behaviors in the Message-Route algorithm for handling messages which are not blocked, blocked by a ring or blocked by a chain.

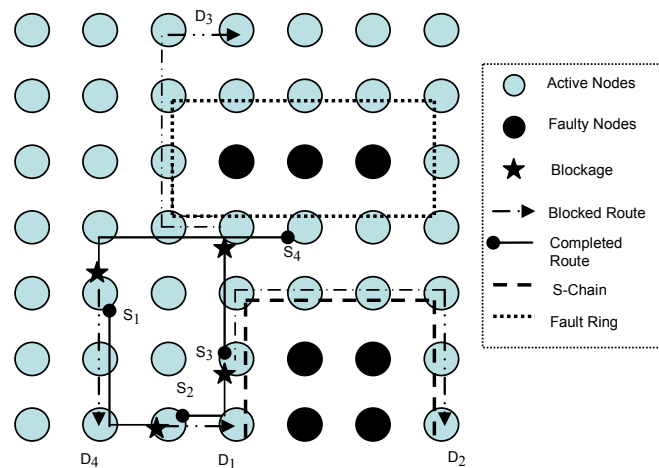


Fig. 1. Four messages involved in a deadlock situation.

Message ( $S_1, D_1$ ) is a CF message as its destination is south-east with respect to the source. Its path is not blocked by any fault and the message can therefore proceed using the Normal-Route, taking a south-east turn when it reaches the destination row. However, before reaching  $D_1$  it is blocked by the message ( $S_2, D_2$ ).

Message ( $S_2, D_2$ ) is an RO message since it has the destination in the same row to its east. The path is routed using the Chain-Route behavior as the *s-chain* that surrounds the faulty nodes blocks its normal route. According to Chain-Route an RO message should

be routed clockwise around the chain. It therefore turns north at the chain but is blocked from further advancement by message  $(S_3, D_3)$ .

Message  $(S_3, D_3)$  is a CF message as the destination is in the same column at its north. It is first guided by the Chain-Route behavior because of the  $s$ -chain. Chain-Route tells that a CF message heading SN (south to north) at the west boundary of an  $s$ -chain should use SN channel if available and if the destination ( $D$ ) is not to the west of the current node ( $C$ ).  $D$  is not to the west of  $C$  and thus SN is selected. It is then blocked traveling north by the fault-ring, where Ring-Route behavior tells that it should be routed clockwise around the ring. This direction is to be used as  $D$  is not lower than the reference node (top right corner) of the ring. Thus, it should turn west but is blocked by message  $(S_4, D_4)$ .

Message  $(S_4, D_4)$  is an RF message since it has the destination at south-west. It is first guided by the rules of Ring-Route behavior at the fault-ring. According to this an RF message should use EW (east to west) channel if it is available. One hop outside the ring it makes a west to south turn according to Normal-Route behavior of the algorithm. It can however not proceed because it is blocked by message  $(S_1, D_1)$ .

As these four messages are involved in a circular wait, which can not be resolved by Message-Route algorithm, a situation of deadlock has occurred.

## 2.2 Incompleteness of the Routing Algorithm [1]

Chen and Chiu's routing algorithm have missed certain cases with a result that messages can not be routed to a set of destination nodes from a certain set of source nodes. We illustrate two such cases where messages will be stopped from advancing in the network and they will, as a result of this, never reach their destinations.

**Case 1:** The first case is either an RF or a CF message heading from north to south (NS) and having destination on the west border of a chain, below the north-west corner. At the north-west corner of the  $s$ -chain any message with destination in the same column is CF. Chain-Route determines these should proceed clockwise and the only alternative is to turn east. In the next node these messages are stopped because the desired route is to turn back again. However 180 degree turns are not allowed using Message-Route behavior and as a consequence these messages are blocked. The overall result of the algorithm can be illustrated as in Fig. 2, where two faulty nodes are surrounded by an  $s$ -chain. White nodes are destinations which can not be reached from any of the nodes marked with grey color.

**Case 2:** The second case is an RO or CF message proceeding clockwise on any fault chain with the destination to the east of the east border of the chain. In this case, Chain-Route procedure decides a clockwise direction for the message.

Thus the message will propagate around towards the east border, pass the row of the destination node even when it becomes an RO message. This is because there is no alternative in the Chain-Route procedure but to always proceed clockwise for an RO message. Therefore, the message is blocked and can not make any progress towards the destination in a similar way as in case 1. An illustration of this case with affected source and non-reachable destination nodes around an  $s$ -chain is given in Fig. 2.

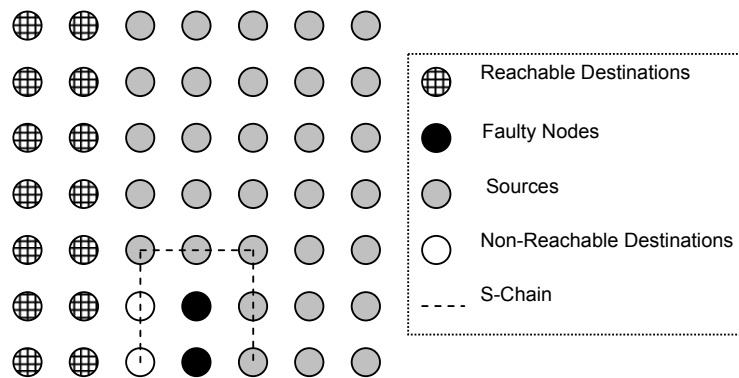


Fig. 2. Case 1 example of sources with non-reachable destinations.

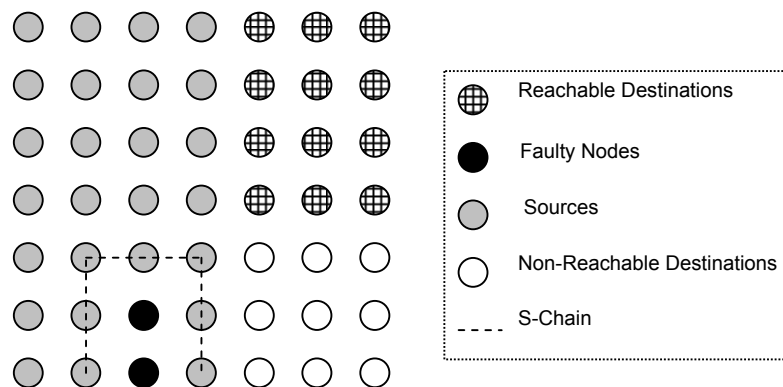


Fig. 3. Case 2 example of sources with non-reachable destinations.

### 3. ANALYSIS OF IDENTIFIED ERRORS

#### 3.1 Error in Proof of Lemma 1

As we have demonstrated in the previous section, a situation of deadlock is possible using Message-Route algorithm. In order to understand how this error occurred we analyzed the theoretical proof in the paper regarding deadlock freeness. The basic idea used in the proof is that certain combinations of turns which can lead to a deadlock will never occur or get aligned to form a circular path. The authors organized the proof using two lemmas before stating the theorem regarding deadlock freeness. Lemma 1 says that an EN (a message heading east changes direction to north) turn cannot be aligned with an NW turn as long as the EN turn does not occur at the south-east corner of a fault chain. This lemma is proved by enumerating different conditions where an EN turn can occur. They identify three different cases where this turn can occur and the algorithm ensures that the dangerous alignment does not take place. But the authors missed another case where an EN turn can occur. In this case an RO message makes an EN turn at the west border of an *s*-chain. This follows from the Message-Route algorithm where an RO mes-

sage should always be routed clockwise when traveling on a fault chain. There is a possibility of this EN turn to align with an NW turn which can create situation for a deadlock. Therefore Lemma 1 is not correct. Since Lemma 1 is used in the proof of Theorem 1 it makes this theorem also incorrect. Hence, Message-Route algorithm is not deadlock free, at least not in the presence of  $s$ -chains in the network.

### 3.2 Blocked Messages

As we have also shown in the previous section, some messages may be blocked in a node indefinitely and cannot reach their destinations under certain circumstances. The authors seem to have missed consideration of these cases and it makes the algorithm **incomplete**. The first such missed case occurs for a CF message heading from north to south on an  $s$ -chain. There is only one alternative and that is to proceed in clockwise direction. This alternative is good for a non  $s$ -chain proceeding from north to south. It could be suspected that this error occurred because of a mix-up between the chain types. The second case involves both chain types and the problem is related to an RO message not being able to leave a chain. As this case is handled properly when a ring is involved we suspect this was simply left out by mistake in the chain behavior.

## 4. CORRECTIONS TO CHEN AND CHIU'S ALGORITHM

Chen and Chiu's algorithm [1] is based on good ideas but it overlooks certain network traffic situations. At some traffic situations the routing algorithm leads to deadlock and at some other situations the messages never reach their destinations. These situations are related to traffic routing around chains. In this section, we propose modifications to their Chain-Route procedure to take care of the problems related to routing around chains. Our solution to avoid a deadlock situation while routing around  $s$ -chain is based on their method to route CF messages which are going from north to south at the west border of a fault-ring. We prohibit all messages, except those blocked by the  $s$ -chain, from traveling from south to north on the west border of an  $s$ -chain. Other messages are made to first take a west hop before proceeding northwards. This will break the link between the EN turn at the  $s$ -chain and the NW turn at a fault-ring or chain positioned over the  $s$ -chain. We have also made the other necessary corrections to avoid blockage of messages.

### 4.1 Assumptions for the Creation of Faulty Blocks and Fault Rings

The following terminology and important assumptions about nodes, faulty blocks, fault rings and fault chains have been borrowed and extended from [1] and used in our algorithm.

1. Faulty and deactivated nodes are contained in rectangular regions called faulty blocks. Active nodes are nodes that are not faulty and do not belong to a faulty block. To make these faulty blocks rectangular, non-faulty nodes which have at least two neighbors that are faulty or deactivated also become deactivated. A deactivated node that has at least one active node as neighbor is called unsafe. Unsafe nodes can com-

municate via their active neighbors but are not used in the routing of messages which originate outside the faulty block.

2. A faulty block is surrounded by either a fault ring or a fault chain. A faulty block not touching any border of the mesh is surrounded by a fault ring. A faulty block touching a border is surrounded by fault chain. A fault chain touching the north or east border of the mesh uses the rules of a fault ring. A fault chain touching only the south border of the mesh uses the rules of the *s*-chain. Any other fault chain uses the rules of non *s*-chain.
3. To support deadlock free routing, two fault rings/chains should have at most one overlapping node between them. This means that there can be at most one node that is part of two different fault rings/chains. If this happens each of the fault ring/chain has a corner node in this overlap. If any fault ring/chain has an entire side facing another fault ring/chain they cannot share any node, as the distance between any two faulty blocks have to be at least two nodes.

#### 4.2 Corrected Message-Route Algorithm

Although, we have made corrections only to procedure Chain-Route, we give the full algorithm for the sake of completeness. As the original algorithm is somewhat unclear in how overlapped fault rings/chains shall be treated we have also added a procedure, *Overlapped-Ring-Chain-Route*, to resolve this situation. This procedure lists rules of handling this situation as described in [1]. The corrections are typed in bold Arial font. We use the terminology of the original algorithm [1].

```

Procedure Message-Route-Modified
/* Message mg is sent from source S to destination D; C is the current node of the header flit */
if (C is the destination D)
  Consume mg;
else
  if (current node C is S, and is unsafe)
    Send mg to an active neighbor;
  Else /* active node */
    begin
    Determine the message type (RF, CF, or RO) of mg;
    if (C is not on a fault ring or fault chain)
      Normal-Route(mg);
    else
      if (C is on a single fault ring)
        Ring-Route(mg)
      else
        If (C is on a single fault chain)
          Chain-Route-Modified(mg)
        else
          Overlapped-Ring-Chain-Route(mg)
    End

```

```

Procedure Overlapped-Ring-Chain-Route(mg)
if (mg is RO message)
  Select Ring-Route(mg) or Chain-Route(mg) as used in previous node;
else
  switch (mg's direction)
    case (EW message)
      Compare the reference point of the overlapped chains/rings
      Select Ring-Route(mg) (Chain-Route(mg)) if reference point
      of ring (chain) was more westwards;
      exit;
    case (WE message)
      Compare the reference point of the overlapped chains/rings
      Select Ring-Route(mg) (Chain-Route(mg)) if reference point
      of ring (chain) was more eastwards;
      exit;
    case (SN message)
      Compare the reference point of the overlapped chains/rings
      Select Ring-Route(mg) (Chain-Route(mg)) if reference point
      of ring (chain) was more northwards;
      exit;
    case (NS message)
      Compare the reference point of the overlapped chains/rings
      Select Ring-Route(mg) (Chain-Route(mg)) if reference point
      of ring (chain) was more southwards;
      exit;
  exit;

```

```

Procedure Normal-Route(mg)
switch (mg's type)
  case (RF message)
    Use EW channel to forward mg;
    exit;
  case (CF message)
    if (mg is NS message)
      Use NS channel to forward mg;
    else Use SN channel to forward mg;
    exit;
  case (RO message)
    Use WE channel to forward mg;
    exit;

```

```

Procedure Ring-Route(mg)
switch (mg's type)
  case (RF message)
    if (EW channel is available)
      Use EW channel to forward mg;
    else
      Route mg clockwise;
    exit;
  case (CF message)
    if (mg is SN message)
      if (C is on the north boundary of the fault ring)
        Normal-Route(mg);
      else
        if (C is on the west boundary of the fault ring and D is in the same column
          as C)
          Normal-Route(mg);
        else
          if (D is lower than the reference node of the ring)
            Route mg counter-clockwise;
          else
            Route mg clockwise;
    else /* NS message */
      if (C is on the east or south boundary of the fault ring)
        Normal-Route(mg);
      else
        if (C (including S) is on the west boundary of the ring and EW channel is
          available)
          Route mg along EW channel;
        else
          Route mg counter-clockwise;
    exit;
  case (RO message)
    if (C is in the same row as D and WE channel is available)
      Use WE channel to route mg;
    else
      Route mg counter-clockwise;
exit;

```

```

Procedure Chain-Route-Modified(mg)
switch (mg's type)
  case (RF message)
    if (the fault chain is an s-chain)
      if (EW channel is available)
        Route mg along EW channel;
      else
        Route mg counter-clockwise;
    else /* not s-chain */
      if (C is in the same row as D)
        Route mg along EW channel;
      else
        if (D is higher than C)
          Route mg counter-clockwise;
        else
          Route mg along clockwise direction;
    exit;
  case (CF message)
    if (mg is an NS message)
      if (the chain is an s-chain)
        /* Correction 1: Reach destinations on west border of s-chain */
        if (C (including S) and D is on the west border of the chain)
          Route mg along NS channel;
        else
          Route mg clockwise;
      else /* not s-chain */
        if (NS channel is available and D is not to the west of C)
          Route mg along NS channel;
        else
          Route mg clockwise;
    else /* SN message */
      /* Correction 2: Avoid deadlock when RO makes EN turn at s-chain */
      if (the chain is an s-chain)
        if (C is on the north or east boundary of the chain)
          Normal-Route(mg);
        else
          If (C (including S) is on the west boundary of the ring and EW
          channel is available)
            Route mg along EW channel;
      else /* not s-chain */
        if (SN channel is available and D is not to the west of C)
          Route mg along SN channel;
        else
          Route mg counter-clockwise;
    exit;
  case (RO message)
    /* Correction 3: Reach destinations located east of chain */
    if (C is in the same row as D and WE channel is available)
      Use WE channel to route mg;
    else
      Route mg clockwise;
    exit;

```

### 4.3 Discussion on the Changes to the Algorithm

There are three corrections in the procedure Chain-Route.

The first correction gives CF messages traveling from north to south a possibility to reach destinations south of the north-west corner of the  $s$ -chain. If the current node or the source node is on the west border and the destination also is on this border then the message should be routed via the north to south channel. This will not violate any proof since no message makes an SW turn on or below the east border in this situation.

The second correction is made to avoid the occurrence of a deadlock when making an EN turn at the west border of an  $s$ -chain. It makes a CF message going from south to north on the west border of an  $s$ -chain divert one hop west before proceeding north. This will stop these messages to use the link immediately below the north-west corner of the  $s$ -chain in order to prevent this link to be involved in a deadlock situation.

The third correction is made to make it possible for messages to reach destinations located to the east of a chain. As we have described these were unreachable from sources at certain locations. The correction makes it possible for these messages to leave a chain using a WE channel when the destination row is reached.

We have also added a new procedure, Overlapped-Ring-Chain-Route, to clarify the routing decision when a message is on a node which is part of two rings or a ring and a chain or two chains.

### 4.4 Proof of Deadlock Freeness

As described in the section 3 an important case was missed in the proof of Lemma 1 [1] which resulted in a non-valid proof of the main theorem regarding their algorithm. We restate both the lemmas and the theorem for the modified algorithm. It should be noted that Lemma 1 is more general with respect to an EN turn as compared to the corresponding lemma in [1]. Because of the modifications to the algorithm the errors pointed out in the previous section are removed.

**Lemma 1** Using the *Message-Route-Modified* algorithm, an EN turn cannot be aligned with an NW turn as long as the column segment between the turning nodes of the EN and NW turn does not include the east boundary of an  $f$ -chain.

**Proof:** We write the proof of the lemma in the same style as the proof of the original algorithm [1]. According to Message-Route-Modified algorithm, an EN turn can only be made in four different cases:

1. An RO message encounters a fault ring, and makes an EN turn at the south-east corner node of the ring.
2. An SN message encounters a fault ring, and the destination of the message is lower than the reference node of the ring.
3. A message makes an EN turn at the south-east corner node of a fault chain.
4. An EN turn is made by an RO message on encountering an  $s$ -chain.

The proof for the first three cases for the original Message-Route algorithm given in [1] still holds for our modifications. We will only prove the fourth case here.

There are only two situations in which an NW turn can be made using our algorithm. In the first case a message is routed around an  $f$ -chain in counter-clock wise direction and makes an NW turn at the north-east corner of an  $f$ -chain. The second case is when a CF message is blocked by a ring. According to the lemma we only need to prove that an EN turn made by a message blocked by an  $s$ -chain does not align with a CF message blocked by a ring. We will prove by contradiction that this situation is not possible.

Assume now that an EN turn is aligned with an NW turn. Let  $T_1$  represent the EN turn and  $T_2$  represent the NW turn. For the turns to be aligned there must now exist a set of messages,  $m_1, m_2, \dots, m_l$  such that  $m_1$ , which is an RO message, takes  $T_1$ ,  $m_l$  takes  $T_2$ ,  $m_i$  waits for  $m_{i+1}$  for all  $1 \leq i \leq l-1$ , and the column segment that starts from  $T_1$  and continues to  $T_2$  is a sub-path of the waiting path of the messages. According to Chain-Route-Modified procedure the turning node of  $T_1$  cannot be the north-west corner of the  $s$ -chain in this case. The following shows that  $T_1$  and  $T_2$  cannot be aligned.

An NW turn can be made by  $m_l$  if  $m_l$  is an SN message and encounters a fault ring. There are two different sub-cases where  $T_2$  can be positioned.

**Case 1:** The turning node of  $T_2$  is no higher than the north-west corner of the  $s$ -chain.

In this case the column channel of  $T_2$  is located on the west boundary of the  $s$ -chain. Our algorithm requires that an SN message, which starts from a source node, located below the north-west corner node, on the west boundary of the  $s$ -chain, be routed one step to the west as long as such channel is available. Recall that two rings/chains can overlap at most one node. Thus, in this overlapping situation such a channel is available at and below the source node of the column channel of  $T_2$ . Therefore,  $m_l$  cannot possibly travel through the SN channel of  $T_2$ . Hence,  $T_2$  cannot be made by  $m_l$ .

**Case 2:** The turning node of  $T_2$  is located higher than the north-west corner of the  $s$ -chain.

Consequently, there must exist some message,  $m_i$ , traveling north through the SN channel immediately below and above the north-west corner of the  $s$ -chain. Note that  $m_1$  must make NE turn at this node, thus  $m_1$  cannot be  $m_i$ . Actually any RO must proceed east here, hence  $m_i$  cannot be RO. Neither can  $m_i$  be an RF message as these must be routed east to west in the case considered. Next, assume that  $m_i$  is a CF message. In this case  $m_i$  must be an SN message but our algorithm requires that an SN message, which starts from a source node, below the north-west corner node, on the west boundary of the  $s$ -chain, is routed one step to the west as long as such channel is available. In the case considered such a channel is available below the north-west corner node of the  $s$ -chain. Therefore  $m_i$  cannot possibly travel through the SN channel immediately below the south-east corner of the  $s$ -chain. Hence, the existence of such an  $m_i$  is not possible.  $\square$

As the algorithm for the cases covered by the original Lemma 2 in [1] have not been changed in Message-Route-Modified and its proof does not have any errors we can use this lemma unchanged. But we restate this lemma for the modified algorithm.

**Lemma 2** Using the *Message-Route-Modified* algorithm, an ES turn cannot be aligned with an SW turn as long as the column segment between the turning nodes of the ES and the SW turn does not include the east boundary of any  $f$ -chain.

As Theorem 1 and the original proof in [1] are based on the validity of these two lemmas we also use them to prove deadlock freeness for Message-Route-Modified algorithm. The proof for the modified algorithm will be identical to the proof of corresponding theorem in [1].

**Theorem 1** *Message-Route-Modified* algorithm is deadlock-free in a 2D mesh.

The theorem can easily be proved using Lemmas 1 and 2 and following the same steps as were followed for the proof of the original Message-Route algorithm [1].

## 5. CONCLUSIONS

In this paper we have shown that the Message-Route algorithm described in [1] is not deadlock free and in some cases even fails to deliver messages to active destinations. We have identified errors in the algorithm leading to the above problems and proposed corrections. A new algorithm Message-Route-Modified has been proposed which incorporates the necessary changes to the original Message-Route algorithm [1]. The proposed algorithm is proved to be deadlock free. Though published with some errors, we believe [1] is an important contribution to the area of fault-tolerant routing algorithms without virtual channels. We also believe these types of algorithms will be even more important in the context of fault-tolerant and deadlock free network design for on-chip communication. In NoC design it is quite critical to keep the size of network routers as simple and small as possible. Therefore, the overhead of using virtual channel implementation will directly compete with other communication and computational enhancements. We are currently using the modified Chen and Chiu's algorithm described in this paper to evaluate various aspects and issues related to NoC design incorporating region concept [8].

Algorithm proposed in [1] as well as its corrected version has some limitations. There is a possibility that Form-Ring procedure might partition the network into unconnected parts. This might make routing between some pairs of nodes impossible using the algorithm in [1] even when there exist non-blocking paths. More research is required to remove this limitation of the algorithm and improve its routing capabilities.

## REFERENCES

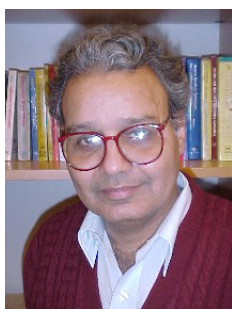
1. K. H. Chen and G. M. Chiu, "Fault-tolerant routing algorithm for meshes without using virtual channels," *Journal of Information Science and Engineering*, Vol. 14, 1998, pp. 765-783.
2. W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 4, 1993, pp. 466-475.
3. C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Proceedings of the 19th International Symposium on Computer Architecture*, 1992, pp. 278-287.
4. R. V. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," *IEEE Transactions on Computer*, Vol. 44, 1995, pp. 848-864.
5. C. J. Glass and L. M. Ni, "Fault-tolerant wormhole routing in meshes without virtual

channels,” *IEEE Transactions on Parallel and Distributed Systems*, Vol. 7, 1996, pp. 620-635.

6. J. Wu, “A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model,” *IEEE Transactions on Computers*, Vol. 52, 2003, pp. 1154-1169.
7. S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrjä, and A. Hemani, “A network on chip architecture and design methodology,” in *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, 2002, pp. 117-124.
8. R. Holsmark and S. Kumar, “Design issues and performance evaluation of mesh NoC with regions,” in *Proceedings of the 23rd Norchip Conference*, 2005, pp. 40-43.



**Rickard Holsmark** is a Ph.D. student with the Embedded System Group at School of Engineering, Jönköping University, Sweden. His research is focused towards specialized architectures and routing algorithms for Networks on Chip. Other areas of interest are embedded systems in general, system level design and processor architectures. He received a Bachelor of Science degree (2001) in Electronics, with specialization in microcontroller systems. After this he completed a Master of Science degree (2003) in Electronics, with specialization in embedded systems. Both of these degrees were received at Jönköping University.



**Shashi Kumar** is a professor of Embedded Systems at School of Engineering, Jönköping University. His research interests include system-level modeling and synthesis, parallel architectures and algorithms, reconfigurable computing and heuristic search algorithms. He was member of the team which was the first to propose the idea of packet switched communication for on-chip communication and coined the term Network on Chip (NoC) in 2000. Prof. Kumar has interest in various aspects of NoC design including NoC topologies, QoS issues in NoC communication, NoC architectural modeling and evaluation, application specific NoC architecture design, mapping applications to NoC platforms and testing of NoC. He received B.Tech., M.Tech., and Ph.D. degrees from the Indian Institute of Technology Delhi in 1974, 1976 and 1985 respectively.