A Simple and Efficient Deadlock Recovery Scheme for Wormhole Routed 2-Dimensioal Meshes

Shih-Chang Wang, Hung-Yau Lin, Sy-Yen Kuo*

Department of Electrical Engineering, Rm. 415
National Taiwan University
Taipei, Taiwan, R.O.C.
Email: sykuo@cc.ee.ntu.edu.tw

Yennun Huang

AT&T Labs – Research
Florham Park, NJ 07932
USA

Abstract

In order to avoid deadlocks, prevention-based routing algorithms impose certain routing restrictions which lead to high hardware complexity or low adaptability. If deadlock occurrences are extremely rare, recovery-based routing algorithms become more attractive on hardware complexity and routing adaptability. A simple architecture that each router is provided with an additional special flit buffer was developed to achieving deadlock recovery in the literature. Disha_SEQ and Disha_CON are two deadlock recovery schemes based on such architecture to accomplish sequential recovery and concurrent recovery, respectively. In this paper, we propose a simple recovery scheme for a 2-dimensioal mesh with the same router architecture and reduce drawbacks in Disha_SEQ or Disha_CON such as hardwired token, finding the Hamiltonian cycle, Hamiltonian-path labeling for each node, and non-minimal path routing. Moreover, the simulation result shows that the proposed scheme has similar performance as Disha_CON and is better than Disha_SEQ.

1. Introduction

In interconnection network computers, tasks are executed by a set of intercommunicating nodes or processors. The communication is usually carried out by means of passing messages from one node to another over the interconnection network. Since direct networks utilize the locality of the message references more efficiently, most of the existing systems use direct networks such as *k*-ary *n*-cube or *n*-dimensional mesh[4].

The performance of the inter-processor communication scheme depends largely on the network dimension, the switching technique, and the routing algorithm. Most contemporary systems have used two or three dimensions. Store and forward, virtual cut-though, and wormhole routing are the main switching techniques used for inter-processor communication. Due to the lower latency and smaller buffer requirements, wormhole routing is preferred and is widely used in recent multi-computers [1,3,4].

Most wormhole adaptive schemes focus on deadlock prevention rather than deadlock recovery. Preventive schemes suffer from high hardware complexity and losses in adaptability [5,7-8,11-12]. In general, the routing algorithm should be fully adaptive. If deadlock occurrences are extremely rare [13], it does not make sense to limit the adaptability of the routing scheme to solve an infrequent event. In addition, the router design should be extremely simple. Many deadlock-prevention approaches devote virtual channels [6] specifically to prevent deadlocks. However, virtual channels need more hardware complexity.

A simple and efficient deadlock recovery scheme was proposed in [8]. This scheme requires nominal hardware to be devoted for the deadlock recovery. Each router is provided with an additional special deadlock buffer to be used in the case of deadlocks. These deadlock buffers are central to the routers and essentially form a dedicated deadlock-free lane, which can be viewed as a "floating" virtual channel. Routing on this deadlock-free lane can be fully adaptive. On deadlock, one of the packets in the deadlock cycle is switched into the deadlock-free lane and routed using this resource until it reaches its

* Acknowledgment: This research was supported by the National Science Council, Taiwan, R.O.C., under Grant NSC 88-2213-E-002-040. Sy-Yen Kuo is currently a visiting researcher at AT&T Labs-Research, New Jersey.

destination where it will be consumed (sunk) to break the dependency cycle.

The above structure offers the following advantages: 1) deadlock-free fully adaptive wormhole routing, 2) suitable for arbitrary interconnection network, 3) no virtual channels required, 4) simple router design, 5) no padding, storing or retransmission of packets required. Simulation results have shown that it is very efficient compared with existing deadlock-prevention techniques: Turn Model[2], Dally[10], Duato[11] and Dimension_Order[4].

The Disha_SEQ proposed in [8] requires packets to gain exclusive access to the deadlock buffers by capturing a circulating hardwired token. As recovery from cycles is sequential, the bandwidth available for recovery is small which could degrade performance if deadlocks are temporally clustered or otherwise appear more frequently. The Disha_CON proposed in [9] is based on structured buffer routing that offers higher bandwidth for deadlock recovery on demand and eliminates the need for mutual exclusion on recovery resources. Disha_CON makes simultaneous recovery from multiple deadlock cycles possible with the additional deadlock buffers. However, network bandwidth is allocated to deadlock recovery in some instances when potential deadlocks are detected. In some cases, Disha_CON has the scenery of the mis-routing and needs additional information on the header flit to overcome such situation.

In this paper, a simple deadlock recovery method based on deadlock buffers for a 2-dimensional mesh is proposed. The proposed method has the following advantages compared with Disha_SEQ or Disha_CON: 1)no hardwired token, 2)no need to find Hamitonian cycle, 3)no additional global information about Hamiltoian-path labeling, 4) minimal path routing, 5)partially or fully concurrent recovery, 6)no network bandwidth is allocated to deadlock recovery.

The paper is organized as follows. In Section 2, we introduce the preliminaries used in this paper. The Motivation is described in Section 3, and a simple deadlock recovery algorithm for a 2-dimensional mesh is given in Section 4. Simulation results are shown in Section 5. Section 6 concludes this paper.

2. Preliminaries

This section introduces the network topology and the switching technique used in this paper.

- The wormhole routing network we consider is the 2-dimensional (2D) mesh which has $k_0 \times k_1$ nodes, where $k_0 \geq 2$ and $k_1 \geq 2$. Each node is identified by two coordinates, (x,y), where $0 \leq x,y \leq k_i - 1$ for $0 \leq i \leq 1$. Two nodes (x_0,y_0) and (x_1,y_1) are neighbors if and only if $x_0=x_1$ and $y_0=y_1 \pm 1$, or $y_0=y_1$ and $x_0=x_1 \pm 1$. Clearly, nodes have from 2 to 4 neighbors, depending on their locations in the mesh. Two neighbor nodes are connected by two opposite unidirectional channels (links). Each node has its own router to perform switching, routing, flow control, multiplexing physical channels, inter-chip signaling and clock recovery. An example mesh is shown in Fig. 1. In general, lower-dimensional meshes, in which each node has small and fixed degrees, are preferred over higher-dimensional meshes for scalability consideration.
- The switching technique used is the wormhole routing. A packet consists of a sequence of flits(flow control digits). The size of a flit depends on system parameters, in particular the channel width. The header flit(s) of the packet governs the route, and the remaining flits of the message follow in a pipelined fashion. Each unidirectional channel has its own flit buffers (queue), and once a queue accepts the header flit of a packet, it must accept the remainder of the message before accepting any flits from another packet. In addition, each node can generate packets destined for any other node and a packet arriving at its destination node is eventually consumed.

3. Problem Description

Under wormhole routing, each packet may reserve more than one channel at a time for its exclusive use. The order in which the channels of the network are reserved must be restricted to prevent deadlocks among packets. A deadlock refers to the situation that in a set of packets each is blocked indefinitely because each packet in the set holds some resources also needed by another packet. An example deadlock in a 5×5 mesh is shown in Fig. 1. In general, a circular wait condition should be

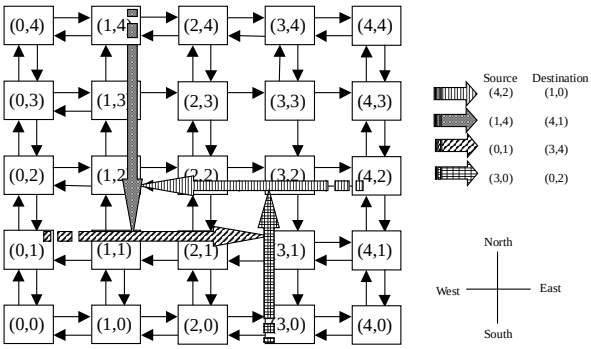
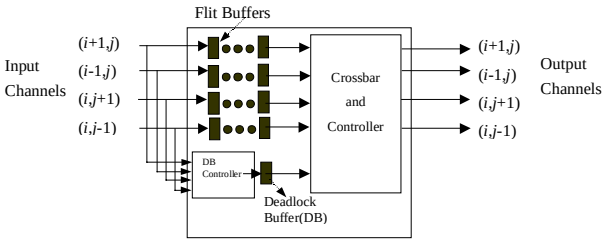


Figure 1. Deadlock in a 5×5 2D mesh.

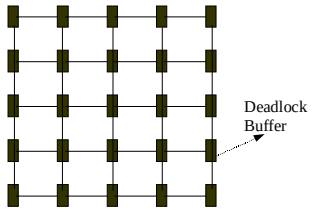
inhibited to prevent the deadlock.

The Disha which is based on sequential recovery(Disha_SEQ) was proposed by Anjan K. V. and

T. M. Pinkston[8] to optimize routing performance in the absence of deadlocks. A simple architecture for the Disha_SEQ router is shown in Fig. 2(a). In Disha_SEQ, the router for each node has an additional deadlock buffer (DB) whose size is the same as a flit buffer. These deadlock buffers are central to the routers and essentially form a "floating" lane (see Fig. 2(b)). Routing for a packet can be fully adaptive through the flit buffers in the network until the deadlock does occur. On deadlock, one of the packets in the deadlock cycle is switched into the floating lane until it reaches its destination. Clearly, the deadlock dependency cycle is broken.



(a) The simple router for node (i,j).



(b) The floating lane formed by the DBs for a 5×5 mesh.

Fig. 2. Illustrations for Disha_SEQ architecture in a 2D mesh.

The mutual exclusive use of the floating lane is necessary to avoid deadlock. Disha_SEQ has a token to circulate in the network along a fixed predetermined cyclic path to include all routers. A simplistic approach to implement an asynchronous token-passing protocol is based on the Hamiltonian cycle, that is, the token is passed circularly along a predetermined Hamiltonian cycle. Only the packet which has captured the token can be switched into the floating lane and reach it's destination under fully adaptive routing on the floating lane. The destination which receives the header flit of the packet from the floating lane will regenerate the token and the floating lane becomes free for other deadlock packets to use.

Deadlocks are detected using a time-dependent selection function. When the header flit H of a packet arrives, the router sets the counter T_H for the flit H to 0. T_H keeps track of the number of clock cycles that the flit H can not be sent out. If $T_H > T_{out}$(a predetermined time-out value), the state of this flit H is changed to "deadlock" and the floating lane is used to solve the "deadlock" condition. Note that it may not denote a real deadlock situation when $T_H > T_{out}$. However, it denotes the flit H can not be sent out due to the channel contention for the heavy system traffic. So, it is critical to determine the value of T_{out}.

The main disadvantage for Disha_SEQ is that it does not allow simultaneous recovery from a number of deadlock situations and needs a hardwired token. Under the heavy traffic or multiple-deadlock conditions, it is difficult for sequential recovery to obtain better system performance. It is clear that a deadlock area will extend its blocked buffer area until the deadlock is broken. Thus, the partially or fully concurrent recovery is needed to deal with the conditions of the heavy traffic or multiple-deadlocks. Another problem for Disha_SEQ is that the Hamiltonian cycle to implement the token-passing path can not be found for some 2D meshes. For example, there is no Hamiltonian cycle in Fig. 1. This implies that it is difficult to implement a fair token-passing path and each node has different waiting time for capturing the token.

The Disha that is based on concurrent recovery (Disha_CON) was proposed in [9]. Disha_CON offers higher bandwidth for deadlock recovery on demand and eliminates the need for mutual exclusion on the resources of the deadlock buffers. However, network bandwidth is allocated to deadlock recovery in the rare instance when probable deadlocks are detected. For Disha_CON, a Hamiltonian path is constructed first on the deadlock buffers, and each node is labeled according to its sequence along the Hamiltonian path. The deadlock buffer at a node can be used only by a deadlocked packet at a neighboring node for which the packet's destination has a higher label than the node to which the buffer belongs. Once a deadlocked packet is placed on the floating lane to break the deadlock cycle, it is restricted to using only deadlock buffers at subsequent nodes until it is delivered. Moreover, successive deadlock buffers can be used only in increasing label order. For the deadlocked packets at nodes higher than the destination label, normal flit buffers are used to reach an intermediate node with lower label than the destination, following which they use the acyclic deadlock buffer lane to arrive at the destination.

An example for Dish_CON to deal with the deadlock situation of Fig. 1 is given in Fig.3. Fig. 3(a) shows one possible Hamiltonian labeling way for the nodes. The moving direction for the packet on the floating lane is described in Fig. 3(b). Clearly, it is deadlock-free for the packets moving on the floating lane. Fig. 3(c) shows the possible normal channel buffers used by the deadlocked packets that can not be switched into the floating lane immediately. Four deadlocked packets in Fig. 1 will take the following paths to reach their destinations.

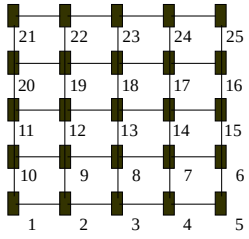
The packet blocked at (1,2)⇒(1,1)⇒(1,0)

The packet blocked at (1,1)⇒(1,0)→(2,0)→(3,0)→(4,0)→(4,1)
The packet blocked at (3,1)→(3,2)→(3,3)→(3,4)
The packet blocked at (3,2)⇒(3,1)→(2,1)→(1,1)→(0,1)→(0,2)

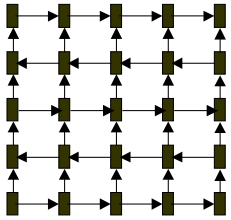
Note that the symbol, ⇒, denotes that the normal channel buffer is used and the symbol, →, denotes that the deadlock buffer is used. We can find that two packets with source nodes (1,1) and (3,2) are mis-routing; therefore, the minimal routing is not guaranteed by the Disha_CON.

Compared with Disha_SEQ, Disha_CON provides a way to achieve concurrent recovery without any hardwired token. But, the disadvantages of Disha_CON are:

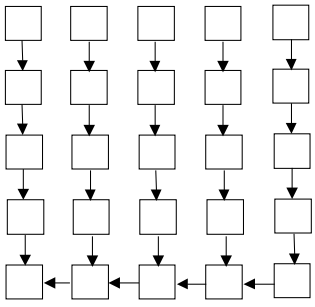
1) Each node needs to know the global information of the Hamiltonian-path labeling.
2) Different labeling ways may cause the unbalanced traffic on the normal channel buffers or the floating lane.
3) Routing path for a deadlocked packet may be non-minimal.
4) Additional information needs to be carried in the header flit for the deadlocked packet to deal with the mis-routing.



(a). The Hamiltonian-path labeling.



(b). The moving direction on the floating lane.



(c). The moving directions for the deadlocked packet traversed on the normal channel buffers.

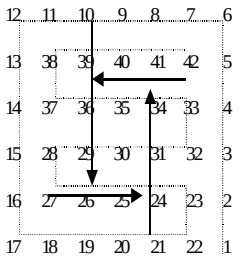
Fig. 3. Deadlock recovery from Fig. 1 by using Disha_CON.

Fig. 4 shows a deadlock situation in a 7×6 mesh. In this 7×6 mesh, a special Hamiltonian-path labeling is used. According to the recovery algorithm Disha_CON, four deadlocked packets will be transferred along the following paths:

The packet blocked at label 26⇒19→20→21→24
The packet blocked at label 24⇒21⇒20⇒19⇒18⇒17⇒16⇒
15⇒14⇒13⇒12⇒11⇒10⇒9⇒8
The packet blocked at label 41⇒8→9→10→11→12→13
The packet blocked at label 39⇒10→11→12→13→14→15→16→
17→18→19→20→21
→22→23→24→25→26→29

Recall that the symbol, ⇒, denotes that the normal channel buffer is used and the symbol, →, denotes that the deadlock buffer is used. Obviously, each deadlocked packet has non-minimal routing, and each packet needs normal channel buffer first when Disha_CON is used to accomplish deadlock recovery. That is, the deadlocked packets can not be moved into the floating lane until the required channel buffers are captured. If other packets hold the required channel buffers, Disha_CON can not break the deadlock cycle immediately.

In this paper, a simple approach for deadlock recovery in a 2D mesh is developed. Our approach has the following properties compared with Disha_SEQ and Disha_CON: 1)no hardwired token is used, 2)no additional global information such as Hamiltonian-path labeling is needed for each node, 3)guarantee the minimal-path routing for the deadlocked packet, 4)no additional control information is carried in the header flit of a packet if deadlock recovery is needed, 5)partially or fully concurrent recovery.



Source label	Destination label
10	24
27	8
21	13
42	29

Fig. 4: A special Hamiltonian-path labeling to illustrate the mist-routing when Disha_CON is used to achieve deadlock recovery in a 7×6 mesh.

4. The proposed deadlock recovery approach for the 2D mesh

Deadlocks occur when cyclic resource dependencies are formed by consumers holding some resources while waiting to acquire others. As we know, a deadlock cycle can be broken if one of the deadlocked packets in this cycle is switched into the floating lane. However, there may occur deadlocks when more than one packets are switched into the floating lane. Disha_SEQ uses a

hardwired token to guarantee only one deadlocked packet is moved on the floating lane until the destination is reached. If multiple deadlock cycles exist, Disha_SEQ is sequential recovery and thus degrades the system performance.

The main idea of the proposed approach is very simple and is based on the following concepts: 1) select only one particular deadlocked packet for each deadlock cycle to be switched into the floating lane, 2) the number of packets in the floating lane should be as many as possible while the floating lane is deadlock-free. The difficulty of our idea is how to select the "particular" packet for each deadlock cycle and how to prove the floating lane is guaranteed deadlock-free. Fig. 5 shows the proposed algorithm to solve the deadlock recovery for a 2D mesh.

Algorithm Deadlock_Recovery_for_a_2D_Mesh
/* Note that this algorithm is executed by each intermediate node when some received packets are blocked for a long time */
Step 1: If the blocked time for a packet P is not greater than T_{out}, **Go to Step 3**. /* P is not a potentially deadlocked packet. */
Step 2: If the destination node of P is in the north of the current node, switch P into the floating lane to reach its destination.
Step 3: **END**.

Fig. 5: The proposed deadlock recovery approach for a 2D mesh.

The algorithm in Fig. 5 is performed by each node in the 2D mesh. If one incoming packet is blocked on a intermediate node for a long time, this node executes this algorithm to check whether the deadlock recovery is needed. In Step 1, the blocked packet is checked whether the potential deadlock occurs by comparing the blocked time with the deadlock time T_{out}. If the blocked time is greater than T_{out}, the blocked packet is supposed to be inside some deadlock cycle. Step 2 in Fig. 5 checks the blocked packet whether it is the packet switched into the floating lane. In our algorithm, the potentially deadlocked packet with the destination node in the "north" of the current intermediate node is selected as the particular packet for some deadlock cycle. Let the label of the current intermediate be (x_i,y_i) and the label of the particular packet's destination be (x_j,y_j). The following expression can check easily whether the packet's destination is in the north of the current node.

$$(x_i=x_j) \text{ and } (y_j>y_i) \text{ ?}$$

Lemma 1: *The floating lane is deadlock-free when the algorithm Fig. 5 is used.*
Proof: From the Step 2 in Fig. 5, only the deadlocked packet moving toward the north is switched into the floating lane. Thus, the possible moving direction for packets in the floating lane is shown in Fig. 6. It is easy to see that the floating lane is deadlock-free. □
Lemma 2: *For the 2D mesh adopting fully adaptive routing, if there exists any deadlock cycle, the algorithm in Fig. 5 breaks the cycle and the potentially deadlocked packets can be sent out progressively.*
Proof: The moving directions for a packet in the 2D mesh are west, east, south, and north. From these four directions, eight 90-degree turns can be formed: left and right turns when travelling west, east, south, and north[2]. The eight turns form two abstract deadlock cycles as shown in Fig. 7. Clearly, deadlock recovery can

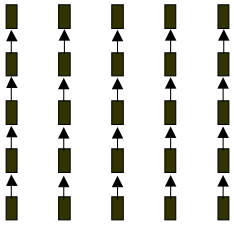


Fig. 6: The possible moving directions for the packets in the floating lane.

be achieved if one of the four turns for a deadlock cycle is switched into the floating lane.

Without loss of generality, consider the counterclockwise deadlock cycle in a 2D mesh. If we can prove that the algorithm of Fig. 5 can break any deadlock cycle, the potentially deadlocked packets in this cycle can be sent out progressively. There are three essential situations for a deadlock cycle occurring.
Case 1: illustrated by Fig. 8(a). In this situation, packet M_1 is one of the potentially deadlocked packets in cycle 1 and is blocked to turn the north(Note that the destination D_1 of packet M_1 is in the north direction). From the algorithm of Fig. 5, packet M_1 will be switched into the floating lane when its blocked time is greater than T_{out}. So, the deadlock cycle 1 can be broken. By the symmetry property, it is true if the deadlock cycle 1 is a clockwise cycle.
Case 2: illustrated by Fig. 8(b). In this situation, the destination D_1 of the deadlocked packet M_1 is in the east-north direction while packet M_1 is blocked to turn the north. Because packet M_1 has alternative path in the east direction by using the minimal fully adaptive routing to reach its destination D_1, we can suppose that there exists one packet M_2 holding the channel buffers in the east direction requested by packet M_1. If M_2 is not involved in any deadlock cycle, the resource held by packet M_2 will be released later. Thus, packet M_1 can occupy the

channel buffers in the east direction, and the deadlock cycle 1 can be broken. By the symmetry property, it is also true if the deadlock cycle 1 is a clockwise cycle.

Case 3: illustrated by Fig. 8(c). In this situation, packet M_2 is blocked by the counterclockwise deadlock cycle 2 or a clockwise deadlock cycle 3. Let M_2 be blocked by deadlock cycle 2. It is straightforward to see that the situation of deadlock cycle 2 is the same as the Case 1 or the Case 2, that is, the deadlock cycle 2 can be broken following by that packet M_2 can occupy the required channel buffer. Finally, the situation of the deadlock cycle 1 is also the same as the Case1 or the Case 2, and the deadlock cycle 1 can be broken. If packet M_2 is blocked by the clockwise deadlock cycle 3, the discussion is similar as the above. In conclusion, three deadlock cycles can be broken.

When the relationship of the deadlock cycles is more complex than Case 3, it is the "finite" extended case of the Case 3 because the size of a 2D mesh is finite. So the similar discussion as in Case 3 can be used to show that the algorithm Fig. 5 can break all deadlock cycles occurring in a 2D mesh. We conclude that the proposed algorithm shown in Fig. 5 can break any deadlock cycle and all relative deadlocked packets can be sent out progressively. □

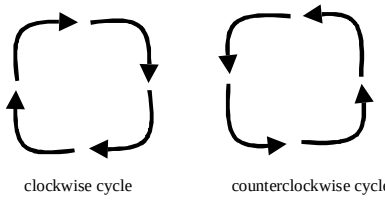


Fig. 7: The possible abstract turns and cycles in a 2D mesh.

Theorem 1: *Under minimal fully adaptively routing, a 2D mesh with deadlock buffers can safely recover from potentially deadlocks when the algorithm shown in Fig.5 is used.*

Proof: From **Lemma 2**, we know that each deadlock cycle can be broken by switching one particular deadlocked packet into the floating lane if it is necessary. Thus, all remaining deadlock packets on the channel buffers can continue to route. From **Lemma 1**, we know that the floating lane is deadlock-free, and each particular packet switched into the floating lane will reach its destination. Therefore, a 2D mesh with deadlock buffers can safely recover from potential deadlocks when the algorithm shown in Fig. 5 is used. □

For the step 2 in Fig. 5, a deadlocked packet turning north is switched into the floating lane. From Fig. 7, it is easy to see that two turns "from west to north" and "from east to north" are switched out the channel buffers for the clockwise and counterclockwise cycles, respectively. From the turn model [2], it is also shown that the proposed simple algorithm can recover from deadlock for a 2D mesh with deadlock buffers. By the symmetry property, the "north direction" in the step 2 of Fig. 5 can be changed to the "south direction", or "west direction", or "east direction".

An example is shown in Fig. 9. In this situation, there are three cycles in the network. First, because packets M_8 and M_9 satisfy the condition of the Step 2 in Fig. 5, packets M_8 and M_9 are switched into the floating lane, and the cycles C_2 and C_3 are broken. All blocked packets in these two cycles continue to transmit. Next, packet M_5 releases its resources and the blocked packet M_1 becomes active(Note that the destination of packet M_1 is in the east-north direction). Thus, cycle C_1 is broken. Finally, all blocked packets in cycle C_1 also continue to transmit. The proposed algorithm in Fig. 5 successfully recovers the deadlock situation in Fig. 9. From this example, we can find that the proposed algorithm is a kind of partially concurrent recovery if there is the dependent relationship among deadlock cycles, however, it is a kind of concurrent recovery from deadlock cycles if there does not exist any dependent relationship among those cycles.

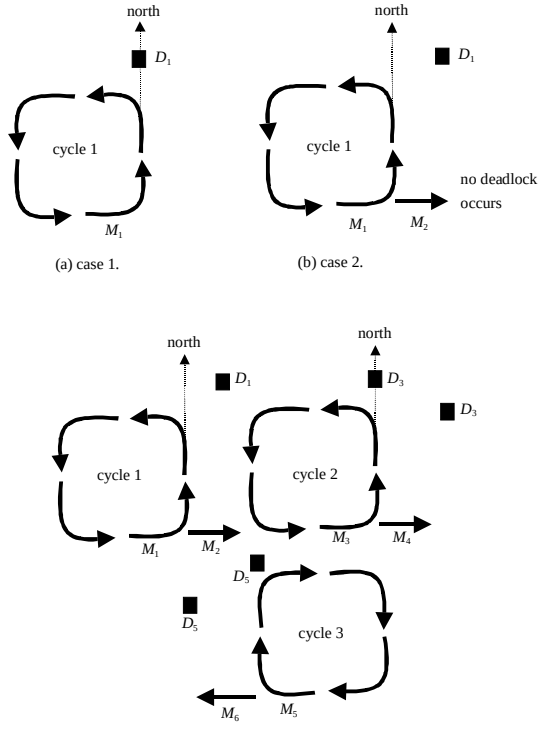


Fig. 8: Three essential deadlock situations for any counterclockwise deadlock cycle occurring in a 2D mesh.

The summaries of three schemes Disha_SEQ, Disha_CON and the proposed scheme are shown in

Table 1. Disha_SEQ needs additional hardware to implement its token mechanism. Disha_SEQ and Disha_CON construct the Hamiltonian cycle and Hamiltonian path, respectively. For Disha_SEQ, there arises the problem of token passing while the Hamiltonian cycle cannot be found in a network. Both Disha_SEQ and the proposed scheme in this paper are minimal routing, however, Disha_CON may have mis-routing. The proposed scheme becomes partially concurrent recovery when the deadlock cycles in network have the dependent relationship. In most cases, the proposed scheme accomplishes concurrent recovery due to the dependent relationship among deadlock cycles is rare.

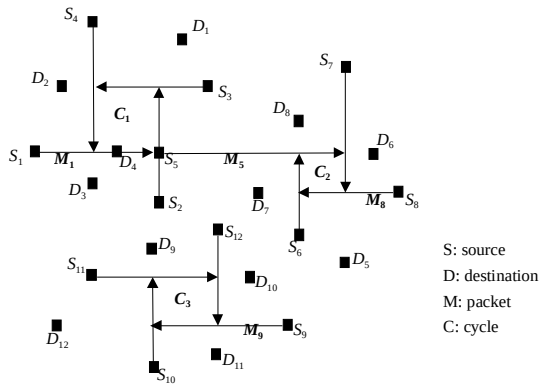


Fig. 9: An example to illustrate the deadlock recovery of Fig. 5 for a 2D mesh.

5. Simulation Result

In this section, we present the simulation result for Disha_SEQ, Disha_CON and the proposed scheme. All simulations are performed in a 16×16 mesh with bi-directional channels, a virtual channel buffer depth of two flits, one deadlock buffer for each node, and a buffer arbitration policy which favors the FCFS(First Come First Serve) allocation. Random traffic patterns are simulated. Messages are 32 flits long and equal clock cycle is assumed for all schemes being compared. A minimal true fully adaptive *X-Y* routing algorithm is used to allow unrestricted routing in any profitable dimension using any available virtual channel. Three schemes are simulated with a time-out of 16 clock cycles. For Disha_CON, each node (x,y) in the 16×16 mesh uses the following Hamiltonian labeling:

$16\times y+x$, if y is even;

$16\times(y+1)-x-1$, if y is odd

	Disha_SEQ	*Disha_Con*	*The proposed scheme*
Adapt deadlock buffer?	*yes*	*yes*	*yes*
Hardwired token?	*yes*	*no*	*no*
Node labeling	*yes*	*yes*	*no*
Minimal path routing?	*yes*	*no*	*yes*
Sequential recovery?	*yes*	*no*	*rare*
Concurrent recovery?	*no*	*yes*	*in most cases*
Suitable for which network?	*arbitrary network*	n-dimensional mesh	*2D mesh*

Table 1: The summary for three deadlock recovery schemes.

Fig. 10 shows the performance results for three deadlock recovery schemes under uniform traffic. Three schemes have the similar performance, however, Disha_SEQ saturates at about 0.65 load rate. The proposed scheme and Disha_CON satuates at about 0.7 load rate.

Simulations indicate that the proposed scheme has similar performance as Dish_CON and cannot even be distinguished in some cases. Compared with Disha_SEQ, the proposed scheme has better performance for a 2D mesh without the need for a token resource. Also, the proposed scheme achieves the similar performance as Disha_CON without the need for the global information of the Hamiltonian labeling.

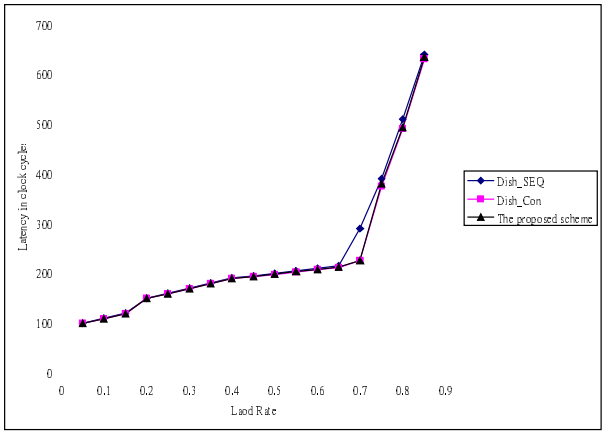


Fig.10: Comparison for uniform traffic pattern.

6. Conclusions

In contrast with the deadlock prevention algorithm, the deadlock recovery scheme has shown the superior performance in the literature. Disha_SEQ is a sequential deadlock recovery scheme based on the single deadlock buffer architecture while needing an additional token resource. Disha_CON is a concurrent deadlock recovery scheme to achieve better performance than Dish_SEQ without the need of the token resource. However, Disha_CON needs a global information for each node when the Hamiltonian-path labeling is proceeded, and achieves non-minimal path routing in some labeling ways. Such non-minimal routing increases the

implementation complexity for the header flit of a mis-routed packet.

For the 2D mesh, a simple deadlock recovery scheme is proposed in this paper. The proposed scheme does not need the token resource or the Hamiltonian-path labeling while achieving the minimal path routing. Simulations show that the proposed scheme has the similar performance as the scheme Disha_CON. Our scheme is based on exploiting the relationship of the deadlock dependencies. One "particular" deadlocked packet in a deadlock cycle is rerouted through the deadlock buffers to break the dependency relationship. In addition, the proposed scheme is suitable for the fully adaptive routing algorithm in a 2D mesh when the deadlock is absent, and no any overhead is added.

References

[1] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *in IEEE Trans. Comput.*, Vol. 4, No. 12, pp.1320-1331, Dec. 1993.

[2] C. J. Glass and L. Ni, "The Turn Model for Adaptive Routing in Multicomputer Networks," *in Proc. of the Int'l Symp. on Computer Architecture*, pp. 278-287, May 1992.

[3] J. Upadhyay, V. Varavithya and P. Mohapatra, "A Traffic-Balanced Adaptive Wormhole Routing Scheme for Two-Dimensional Meshes," *in IEEE Tran. Comput.*, Vol. 46, No.2, pp. 190-197, Feb. 1997.

[4] L. M. Ni. And P. K. McKinley, "A Survey of Wormhole Routing Techniques *in Direct Networks," in IEEE Comput. Mag.*, Vol. 26, No. 2, pp. 62-76, Feb. 1993.

[5] K. M. Al-Tawil, M. Abd-El-Barr and F. Ashraf, "A Survey and Comparison of Wormhole Routing Techniques in Mesh Networks," *in IEEE Network Mag.*, pp38-45, March/April 1997.

[6] W. J. Dally, "Virtual Channel Flow Control," *in IEEE Tran. Parallel and Distributed Systems*, Vol. 3, No.3, pp.194-205, Mar. 1992.

[7] W. J. Dally, "Performance Analysis of k-ary n-cube Interconnection Networks," *in IEEE Tran, Comput.*, Vol. 39, No. 6, pp.775-785, June 1990.

[8] Anjan K. V. and T. M. Pinkston, "Disha: A Deadlock Recovery Scheme for Fully Adaptive Routing," *in Proc. of the 9th International Parallel Processing Symposium*, pp.537-543, April 1995.

[9] Anjan K. V., T. M. Pinkston and J. Duato, "Generalized Theory for Deadlock-Free Adaptive Wormhole Routing and its Application to Disha Concurrent," *in Proc. of the 10th International Parallel Processing Symposium*, pp. 815-821, April 1996.

[10] W. Dally and H. Aoki, "Deadlock-free Adaptive Routing in Multicomputer Networks using Virtual Channels," *in IEEE Tran. Parallel and Distributed Systems*, Vol. 4, No. 4, pp.466-475, April 1993.

[11] J. Duato, "On the Design of Deadlock-free Adaptive Routing Algorithms for Multicomputers: Design Methodologies," *In Proc. of Int'l Conf. On Parallel Processing*, pp.I142-I149, Aug. 1994.

[12] Andrew A. Chien, "A Cost and Performance Model for k-ary n-cube Wormhole Routers," *in Proc. of Hot Interconnects Workshop*, Aug. 1993.

[13] T. M. Pinkston and S. Warnakulasuriya. "On Deadlock in Interconnection Networks", *in Proc. of Int'l Sym. on Computer Architecture*, pp.38-49, 1997.