Fault-Tolerant Routing Schemes in RDT(2,2,1)/α-Based Interconnection Network for Networks-on-Chip Designs

Mei Yang†, Tao Li‡, Yingtao Jiang†, and Yulu Yang‡

† Dept. of Electrical & Computer Engineering
University of Nevada, Las Vegas
Las Vegas, NV 89154, USA
{meiyang, yingtao}@egr.unlv.edu

‡ Dept. of Computer Science and Technology
Nankai University
Tianjing, 300071, China
litao@mail.nankai.edu.cn, yangyl@nankai.edu.cn

Abstract

It has been well recognized that the fault-tolerance capability is vital for a NoC system, since one faulty link/processor may isolate a large fraction of processors. Continuing from a previous paper [13] where a RDT(2,2,1)/α-based interconnection network for NoC designs was proposed, in this paper, we investigate fault-tolerant routing schemes on NoC systems featuring a RDT-based interconnection network. In specific, we propose two fault-tolerant routing schemes in the presence of either single link/node failure or multiple link/node failures. The proposed routing schemes are based on deterministic routing. Alternative routes are discovered by properly selecting the intermediate nodes between the source and the destination nodes on the rank tori. As of the single link/node failure case, we show that the number of routers on the detoured route generated by the proposed routing scheme is at most 2 more than the number of routers on the original route.

1. Introduction

Advances in VLSI technology will soon allow a single chip to contain more than one billion transistors [8], indicating that a large number of processing units (such as CPU, DSP, multimedia processor) shall be integrated into one packaged chip. In these new systems, communication resources are competed by the vast volume computational resources. With a communication-centric design style, Networks-on-Chip (NoC) [1] was proposed to mitigate the complex communication problems. A NoC system is composed of a large number of processing units communicating to other units through routers across the interconnection network. Packets travel through the network by passing one or more hops between the source and the destination tiles.

As semiconductor technology scales down to nanometer domain, processing units (PUs), routers (nodes), and interconnect links (links) are all subject to new types of malfunctions and failures that are harder to predict and avoid [5]. Particularly, failures of nodes/links may isolate a large number of fault-free PUs. A major challenge in NoC designs thus is to provide fault tolerance under link/node failure(s).

In [13], we proposed an interconnection network architecture for NoC based on a special Recursive Diagonal Torus (RDT) structure [12], named as RDT(2,2,1)/α. In RDT(2,2,1)/α, each node has links to form base torus (rank-0) and 2 upper tori with the cardinal number 2. RDT(2,2,1)/α provides a promising solution for interconnection networks of NoC designs with its distinct advantages: 1) high scalability due to its recursive structure, 2) low power consumption due to its smaller diameter and average distance, 3) architectural customizability with its embedded mesh/torus topologies, 4) fault-tolerance capability with a constant node degree of 8, and 5) feasibility of layout compatible with current and future VLSI technologies [7].

In this paper, we attempt to investigate various fault-tolerant routing schemes applicable to the RDT(2,2,1)/α-based interconnection network. As pointed out in [4][7], adaptive routing which employs virtual channels [2][3] is infeasible for NoCs due to the need of large buffers and lookup tables as well as complex shortest-path algorithms. Instead, we consider the deterministic routing based fault-tolerant routing schemes which reroute the packets by properly selecting the intermediate routers between the source and the destination nodes. The proposed fault-tolerance routing schemes have no adverse impacts on routing in the absence of faults.

The rest of the paper is organized as follows. In Section 2, we give an overview of the RDT structure. In Section 3, we describe the NoC architecture, the delay model, and the routing algorithms when no faults are present in the interconnection network. In Section 4, we present the fault-tolerant routing algorithm under single link/node failure. In Section 5, the fault-tolerant routing algorithm under multiple link/node failures is presented. Section 6 concludes the paper.

2. RDT(2,2,1)/α Structure

The RDT structure [12] is constructed by recursively overlaying 2-D diagonal meshes (tori). The *base torus* is a two-dimensional square array of N by N nodes, each of which is numbered with a two-dimensional number as follows:

$$\begin{array}{ccccc} (0,0) & (1,0) & (2,0) & \cdots & (N\text{-}1,0) \\ (0,1) & (1,1) & (2,1) & \cdots & (N\text{-}1,1) \\ (0,2) & (1,2) & (2,2) & \cdots & (N\text{-}1,2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (0,N\text{-}1) & (1,N\text{-}1) & (2,N\text{-}1) & \cdots & (N\text{-}1,N\text{-}1) \end{array}$$

where $N = n^k$ and both n and k are natural numbers. The torus network is formed with four links between node(x, y) and its neighboring four nodes: (mod(x±1, N), y) and (x, mod(y±1, N)). This base torus is also called *rank-0 torus*.

On the rank-0 torus, a new torus-like network (*rank-1 torus*) is formed by adding four links between node (x, y) and nodes (x±n, y±n). The direction of the new torus-like network is at an angle of 45 degrees to the original torus. On the rank-1 torus, another

torus-like network (*rank-2 torus*) can be formed by adding four links in the same manner. Similarly, a *rank-(r+1) torus* can be formed upon *rank-r torus*. An *independent torus* on rank-*i* is one that does not have links to other tori on rank-*i*.

RDT(n, R, m) is a class of networks in which each node has links to form base torus (rank-0) and *m* upper tori (the maximum rank is *R*) with the cardinal number *n*. Thus, the degree of the RDT(*n*, R, *m*) is 4(*m*+1) [12]. One of the upper rank torus is assigned to each node in the RDT(*n*, *R*, *m*) after *n*, *R*, and *m* are set. Thus, the structure of the RDT(*n*, *R*, *m*) also varies with different assignments for upper rank tori to each node. This assignment is called *torus assignment*.

A network in which every node has links to form all possible upper rank tori (i.e. RDT(*n*, *R*, *R*)) is called a *perfect RDT* (PRDT(*n*, *R*)), where *n* is the cardinal number, and *R* is the maximum rank. We refer a *perfect torus* as one that contains all links and no two links overlap.

A PRDT is unrealistic due to its large degree (4(*R*+1)). Various structures of the practical RDT can be formed by changing *n* and *m*, which provide various characteristics and can be applied to different applications. We consider a practical RDT structure, RDT(2, 2, 1)/α, with its torus assignment shown in Fig. 1. In this assignment, each node has eight links, four for the base (rank-0) torus and four for rank-1 or rank-2 tori. Fig. 1 only shows part of these links. In RDT(2,2,1)/α, we only use four independent rank-1 tori with their source nodes located at (0,0), (1,1), (2,0) and (3,1), respectively. Thirty-two rank-2 tori are formed on four rank-1 tori (1,0), (0,1), (2,1) and (3,0), respectively. Compared with other RDT(*n*, *R*, *m*) structures (such as RDT(2, 4, 1)/α, RDT(2, 4, 1)/ß [12]), RDT(2, 2, 1)/α is much simpler and suitable for networks with a thousand or less nodes.

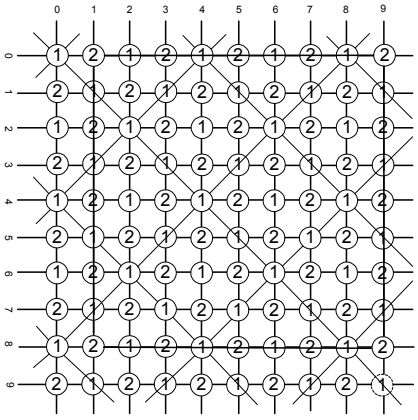

Figure 1. Torus assignment for the RDT(2,2,1)/α.

3. Routing Algorithms without Link/Node Faults

3.1 The NoC Architecture

Our study is based on the NoC architecture shown in Fig. 2, where each PU is attached on a switch/router which is connected to the interconnection links. Fig. 3 shows a possible router structure which consists of input and output ports and interface to the attached PU. At each input and/or output port, a limited number of buffers are provided to store packets, and a line controller is used to select the packet to be received from the input link or sent on the output link. A central controller performs the functions of routing and arbitration.

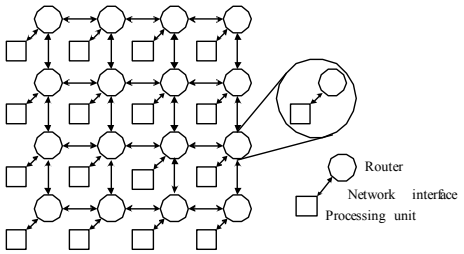

Figure 2. NOC architecture.

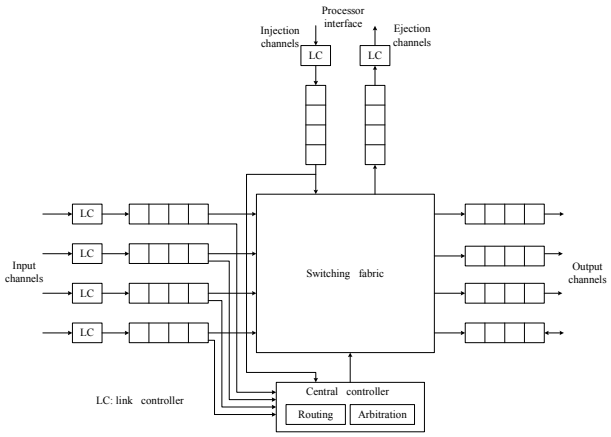

Figure 3. Router structure.

3.2 Delay Model

For an NoC application, an important performance metric is the delay, which depends on the traffic pattern of the application and many other factors, such as packet/flit size, router's arbitration scheme, etc. [7]. In this study, we employ a simple delay model as follows.

For a NoC network shown in Fig. 2, the delay experienced by a packet on the route from the source PU to its destination PU includes the transmission delay on the wires (or wire delay), the processing delay of the routers and the queuing delay at the routers. We assume the processing delay and queuing delay are the same for all routers, denoted as T_p, T_q, respectively. There may exist different types of wires with various lengths. We denote the unit length wire delay as T_0, the length of wire type i as l_i, and the number of wires of type i as n_i. Let N be the number of hops a packet travels between a source to a destination router and m be the number of different types of wires. Then we can determine the total delay experienced by a packet as

$$\mathrm{T}_{\mathrm{total}} = (T_p + T_q)(N+1) + \sum_{i=1}^{m} cn_i T_0 l_i^2 \qquad (1)$$

where c represents the constant proportional to the product of the resistance per unit length and the capacitance per unit length.

As discussed in [10], the effect of length to wire delay can be alleviated by breaking long lines into shorter sections with a repeater driving each section. Hence, it is possible to match the wire delay of a long line to the total wire delay of a set of shorter lines assuming the length of the long line is less or equal to the sum of the lengths of shorter lines in the set. And we assume the processing and the queuing delays experienced at a router are far longer than the wire delay of the line connecting two adjacent routers. Hence the total delay experienced by a packet is

dominated by the sum of the processing and queuing delays on the routers (including the intermediate routers, and the source and destination routers). Hence, Eqn. (1) can be simplified to

$$T_{total} \approx (T_p + T_q)(N+1) \quad (2)$$

As one can see from Fig. 4, on the RDT(2, 2, 1)/α structure, if a route on rank-1 torus is selected to connect S_1 and D_1, the number of routers on this route is 2. This is much less than an alternative route on rank-0 torus that has 5 routers. In another example, the number of routers on the route from S_2 to D_2 on rank-2 torus is 2, which is much less than that of routers on the corresponding route on rank-1 and rank-0 tori, 5 and 9, respectively. These two examples indicate that, according to Eqn. (2), the total delay of a route can be reduced if more links on higher rank tori are included.

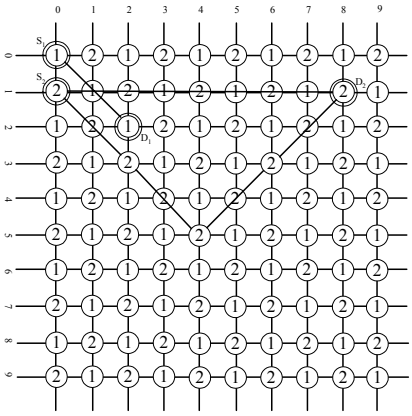

Figure 4. Wires on RDT(2, 2, 1)/α. For clearness, we only show the nodes representing routers.

All the routing schemes discussed in this paper are based on the delay model shown in Eqn. (2). For routing in the RDT(2, 2, 1)/α-based interconnection network without link/router failure, we can directly apply the floating vector routing algorithm [9][12], which is based on the vector routing algorithm [12] proposed for PRDT structures. In the following, we will briefly introduce the two routing algorithms.

3.3 Vector Routing Algorithm

The basic idea of the vector routing algorithm is to represent the route from a source node to the destination node with a combination of unit vectors, each of which corresponds to each rank of tori. Fig. 5 shows the directions of the unit vectors for each rank torus for RDT(2, 2, 1)/α, which rotate in clockwise direction at an angle of 45 degrees as the rank increases.

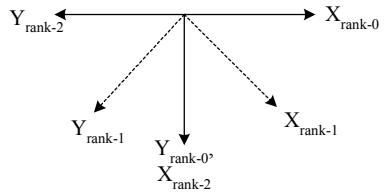

Figure 5. Directions of unit vectors for RDT(2, 2, 1)/α.

We are given a vector from a source node to the destination node represented as $\vec{A} = a\vec{X}_0 + b\vec{Y}_0$ where $\vec{X}_0$ and $\vec{Y}_0$ are the unit vectors of the base (rank-0) torus. $\vec{A}$ can be represented with a combination of the unit vectors $\vec{X}_R$, $\vec{Y}_R$, …, $\vec{X}_0$, $\vec{Y}_0$ on rank-R to rank-0 tori as

$$\vec{A} = a\vec{X}_0 + b\vec{Y}_0$$
$$= v_{Rh}\vec{X}_R + v_{Rv}\vec{Y}_R + \ldots + v_{rh}\vec{X}_r + v_{rv}\vec{Y}_r + \ldots + v_{0h}\vec{X}_0 + v_{0v}\vec{Y}_0$$

where (v_{rh}, v_{rv}) represents the vector on rank-r torus, where $R \geq r \geq 0$. And v_{rh} and v_{rv} are maximized in order to use the upper rank torus as much as possible. Given the vector $\vec{A} = a\vec{X}_0 + b\vec{Y}_0$, we list the vector routing algorithm as follows.

Algorithm Vector Routing (a, b):
begin
for $r = 0$ **to** R **do**
  $g \leftarrow (a + b)/2$
  $f \leftarrow -(a - b)/2$
  $v_{rh} \leftarrow a - (n * g - n * f)$
  $v_{rv} \leftarrow b - (n * g + n * f)$
  $a \leftarrow g$
  $b \leftarrow f$
end

The vectors are computed at the source node and encapsulated into the packet. Then we route the packet to the nodes following the order of rank-R, rank-(R-1), …, rank-0 vectors. On the same rank torus, the routing is performed according to a predetermined order, for example, XY routing [11].

Theorem 1 The maximum number of routers that a packet travels on rank-r torus is n+1 when using the vector routing algorithm, where n is the cardinal number.

The proof of Thm. 1 follows the proof of Thm. 5 in [12].

3.4 Floating Vector Routing Algorithm

The floating vector routing algorithm [12] was proposed for practical RDTs, such as RDT(2, 2, 1)/α. In this algorithm, the vector computation is done at the source node according to the vector routing algorithm for the PRDT, and stored in the packet header. Each router maintains a simple table indicating the direction to reach the nearest node for each rank of torus. A router checks the vector in the packet header, and sends the packet to the nearest router on the highest rank torus according to the simple table. This process is called *floating*, through which each router can use any rank of torus by only a single step packet transfer to a neighbor router. Based on Thm. 1 and the description of the floating vector routing algorithm, we have:

Corollary 1 The maximum number of routers that a packet travels on rank-r torus is n+1 when using the floating vector routing algorithm, where n is the cardinal number.

In [13], we have showed the feasibility of using the RDT(2, 2, 1)/α structure as the interconnection network for a NoC design. In this paper, we further explore the fault-tolerance capability of the RDT(2, 2, 1)/α structure. In specific, we propose two fault-tolerant routing schemes that introduce the least extra processing and queuing delays (equivalent to the least increase of the number of routers as shown in Eqn. (2)) on the RDT(2, 2, 1)/α-based interconnection networks.

4. Fault-Tolerant Routing Algorithm under Single Link/Node Failure

4.1 Fault Model

The following assumptions are used in this paper. 1) Any link or node in the network can fail, and the faulty components are unusable; that is, data will not be transmitted over a faulty link or routed through a faulty node. 2) The fault model is static, that is, no new faults occur during a routing process. 3) Both source and

destination nodes (on any rank torus) are fault-free. 4) The faults occur independently. 5) If a node fails, the four links associated with the node on rank-*r* torus also fail. 6) Faulty link(s)/node(s) are known to all other nodes in the same rank.

4.2 Single Link/Node Failure

Given a pair of source and destination nodes *S* and *D* on the RDT(2, 2, 1)/α structure, using the floating vector routing algorithm, we know that the route from *S* to *D* is composed of route segments on different rank tori and floated route segments on rank-0 torus. The route thus can be represented as S~S_2-D_2~S_1-D_1/S_0-D_0/D, where ~ represents a floating on rank-0, and S_r and D_r represent the source node and the destination node on the route segment on a rank-*r* torus, respectively. Note that a route may consist of all the route segments or subsets of these route segments. For single link/node failure on one segment floating on rank-0 torus, we can always find one alternative route without increasing the number of routers since each router has four possible routes (each with the same number of hops/routers) to float to a router on an upper rank torus. In the following, we enumerate the cases of single link/node failure happening on the route segment on one rank torus.

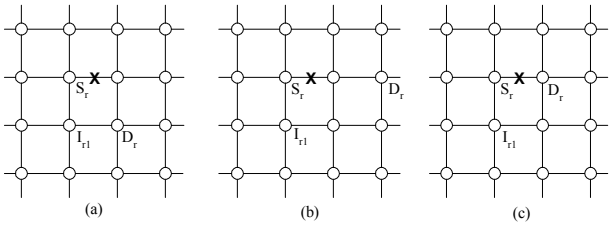

Figure 6. Three distinct cases of single link failure on rank-*r* torus of the RDT(2, 2, 1)/α structure.

By Corollary 1, we know that on the RDT(2, 2, 1)/α structure, using the floating vector routing algorithm, the number of routers a packet travels from S_r to D_r is no more than 3. Hence, there are three possible single link failures on the route segment on rank-*r*, as shown in Fig. 6, where faulty links are marked by X.

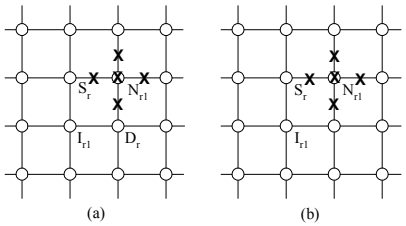

Figure 7. Two distinct cases of single node failure on rank-*r* torus of the RDT(2, 2, 1)/α structure.

For the same reason of single link failure, there are two possible cases of single node failure, as shown in Fig. 7, where faulty nodes are marked by X.

4.3 Fault-Tolerant Routing Algorithm under Single Link/Node Failure

As one can see from Figs. 6 and 7, if the failure is on the *X* (or *Y*) direction, then we should detour the packet by sending it through the *Y* (or *X*) direction first. The vector needs to be changed to reflect the detour if necessary.

In each packet, we add a field *flag*, which is set as "floating" when the packet is floated to a node or "routing" when the packet is routed to a node on a rank-*r* link, and a variable *r* to indicate the current rank value. Fig. 8 shows the packet format.

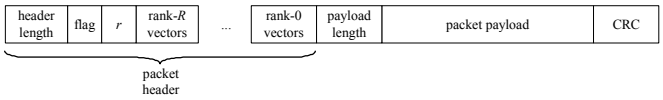

Figure 8. Packet format.

Let X_r^+ denote the +*X* direction on rank-*r*, X_r^- denote the -*X* direction on rank-*r*, Y_r^+ denote the +*Y* direction on rank-*r*, and Y_r^- denote the -*Y* direction on rank-*r*. A distributed fault-tolerant floating routing algorithm is listed as follows, where (i_x, i_y) represents the vector of a node *i* on rank-0 torus.

Algorithm Floating Vector Routing under Single Failure (FVRSF):
begin
//Step 1: the source node computes the vectors, other nodes
//check the flag and decrement the vector value accordingly
if $i = S$ **and** $S \neq D$ **then**
call Vector Routing to compute vectors from rank *R* to 0
set *r* as $\max\{i \mid (v_{ih}, v_{iv}) \neq (0, 0)\}$
else if *flag* = "routing" **then**
if the packet is received from X_r^- **then** $v_{rh} \leftarrow v_{rh}$ -1
else $v_{rh} \leftarrow v_{rh}$ +1
if the packet is received from Y_r^- **then** $v_{rv} \leftarrow v_{rv}$ -1
else $v_{rv} \leftarrow v_{rv}$ +1
else if *flag* = "floating" **then**
if the packet is received from X_0^- **then** $v_{oh} \leftarrow v_{oh}$ -1
else $v_{oh} \leftarrow v_{oh}$ +1
if the packet is received from Y_0^- **then** $v_{ov} \leftarrow v_{ov}$ -1
else $v_{ov} \leftarrow v_{ov}$ +1
//Step 2: check the availability of the link and send the packet //accordingly
if *i* is on rank-*r* **then**
set *flag* as "routing"
if $(v_{rh}, v_{rv}) \neq (0, 0)$ **then**
if $v_{rh} \neq 0$ **and** the link on the v_{rh} direction (i.e., X_r^+ for $v_{rh}>0$ or X_r^- for $v_{rh}<0$) is not faulty **then**
send the packet to v_{rh} direction
else if $v_{rh} = 0$ **and** the link on the v_{rv} direction (i.e., Y_r^+ for $v_{rv}>0$ or Y_r^- for $v_{rv}<0$) is not faulty **then**
send the packet to v_{rv} direction
else
if $v_{rh} \neq 0$ **and** the link on the v_{rh} direction is faulty **then**
if $v_{rv} = 0$ **then** send the packet to Y_r^+
else send the packet to v_{rv} direction
else if the link on the v_{rv} direction is faulty **then**
send the packet to X_r^+
else if find $p = r$-1 to 0 that $(v_{ph}, v_{pv}) \neq (0, 0)$ **then**
$r \leftarrow p$, **goto** Step 2
else goto end
//Step 3: floating to a node nearest to node *i* on rank-*r*
find the non-faulty node $j(j_x, j_y)$ on rank-*r* torus with $\min\{\text{abs}(i_x - j_x - v_{0h}) + \text{abs}(i_y - j_y - v_{0v})\}$ by checking the table on node *i*
set *flag* as "floating"
if $i_x \neq j_x$ **then** send the packet to $(j_x - i_x)$ direction
else send the packet to $(j_y - i_y)$ direction
end

Theorem 2 Consider a route from a source node *S* to a destination node *D* on the RDT(2, 2, 1)/α structure. If there exists a single link/node failure on the original route generated by the floating vector routing algorithm, the number of routers on the detoured route (also the shortest route) generated by the fault-tolerant floating vector routing algorithm is at most 2 more than the number of routers on the original route.

Proof: As discussed in the fault model, single link/node failure only affects routing on one rank torus. Based on Corollary 1, we can derive three possible cases of single link failure as shown in Fig. 6, and the two possible cases of single node failure as shown in Fig. 7. One can verify that, the fault-tolerant floating routing algorithm, with the mbedded vector routing algorithm, computes the shortest route (with the least number of routers) from S_r to D_r on each rank-*r*, and further the shortest route from *S* to *D*. The number of routers on the detoured route generated by the fault-tolerant floating vector routing algorithm, compared with the original route, is either equal to the number of routers on the original route (as the case shown in Fig. 7 (a)) or 2 more than the number of routers on the original route (as the case shown in Fig. 7(b) and the cases shown in Fig. 6). The increase of the number of routers on the detoured route is minimized for all three cases. Hence the theorem follows. ■

The above theorem indicates that the extra queuing and processing delays is minimized using the FVRSF algorithm. Another advantage of the FVRSF algorithm is that each router decides the best route based on the fault information of its four links on one rank torus. No global fault information thus needs to be maintained at each router. In this way, the overhead introduced is very low.

Fig. 9 illustrates two examples of the FVRSF algorithm. The original route from S_1 (2, 1) to D_1 (7, 6) generated by the floating vector routing algorithm is (2, 1)-(3, 1)-(5, 3)-(7, 5)-(7, 6), which travels through 5 routers. Assume that the link between (3, 1)-(5, 3) on rank-1 fails. The detoured route generated by the FVRSF algorithm is (2, 1)-(3, 1)-(1, 3)-(3, 5)-(5, 7)-(7, 5)-(7, 6) with 7 routers, which is 2 more than the number of routers on the original route.

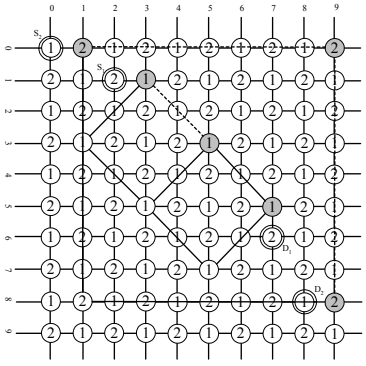

Figure 9. Two examples of the FVRSF algorithm.

As another example, the original route from S_2 (0, 0) to D_2 (8, 8) generated by the floating routing algorithm is (0, 0)-(1, 0)-(9, 0)-(9, 8)-(8, 8), which includes 5 routers. In the case of failure of node (9, 0), the detoured route generated by the FVRSF algorithm is (0, 0)-(1, 0)-(1, 8)-(9, 8)-(8, 8) with 5 routers. Obviously, this is the best detoured route with the least number of routers that a packet has to travel.

5. Fault-Tolerant Routing Algorithm under Multiple Failures

5.1 Multiple Link/Node Failures

Multiple link/node failures may exist on different rank tori of the RDT(2,2,1)/α structure. As discussed in Section 4.1, a node failure can be modeled as multiple link failures. As such, we only consider link failures here. Based on the construction of the RDT(2,2,1)/α structure, we can decompose multiple link failures on different rank tori into sets of link failures on the same independent torus on one rank torus. In the following, we discuss the fault-tolerance routing scheme under multiple link failures on the same independent torus on one rank-*r* torus.

When multiple link/node failures exist on a rank-*r* torus, the FVRSF algorithm cannot generate the detoured route with the least number of routers since the decision made by each router is not based on global view of other failures. In this case, we adopt the method discussed in [6] to identify the best intermediate router through which the detoured route uses the least number of routers. Note that there may exist more than one such intermediate routers. We use I_{ri} to denote such intermediate routers between S_r and D_r.

5.2 Fault-Tolerant Routing Algorithm under Multiple Failures

When applying the method of identifying the best intermediate router in combination with the vector routing algorithm, the following requirements have to be satisfied for an intermediate router I_{ri} to ensure that the faulty links are avoided when routing from S_r via I_{ri} to D_r on rank-*r* torus.

1) All the faulty links shall not appear on all possible S_r-I_{ri} routes generated by the vector routing algorithm.
2) All the faulty links shall not appear on all possible I_{ri} -D_r routes generated by the vector routing algorithm.
3) There is no I_{ri}'giving a shorter route than $I_{ri.}$

The first requirement guarantees that packets can be routed from S_r to I_{ri}, and the second requirement guarantees that packets can be routed from I_{ri} to D_r. Since the vector routing algorithm is a deterministic routing scheme, these two requirements are sufficient to ensure that packets can avoid the faulty links. The third requirement guarantees that the final route is the shortest possible route.

Let R_S^d be the set of routers reachable through vector routing from *S*, let R_D^d be the set of routers that can reach *D* through vector routing. Let $n(i, j)$ be the number of routers on the minimal delay route from router *i* to router *j*. We define R_k as follows: a router *I* is in R_k if and only if $n(S, I) + n(I, D) = n(S, D) + k$. Let k' be the smallest *k* for which $R_k \cap R_S^d \cap R_D^d$ is non-empty. Then we have:

Theorem 3 A router *I* satisfies all three requirements if and only if $I \in R_k \cap R_S^d \cap R_D^d$.

The fault-tolerant routing algorithm under multiple failures is described as follows, where (Iv_{ih}, Iv_{iv}) represents the vector for the intermediate router on the rank-*r* torus with link failures. Correspondingly, in the packet header, a new field of the intermediate node vector on the current rank will be inserted.

Algorithm Floating Vector Routing under Multiple Failures (FVRMF):
begin
  //Step 1: the source node computes the vectors, other nodes
  //check the flag and decrement the vector value accordingly
  if $i = S$ **and** $S \neq D$ **then**
    call Vector Routing to compute vectors from rank *R* to 0
  else if *flag* = "routing" **then**
    set *r* as the highest rank with $(v_{rh}, v_{rv}) \neq (0, 0)$

if $(Iv_{ih},Iv_{iv}) \neq (0, 0)$ **then**

if the packet is received from X_r^- **then** $Iv_{ih} \leftarrow Iv_{ih}$ -1

else $Iv_{ih} \leftarrow Iv_{ih}$ +1

if the packet is received from Y_r^- **then** $Iv_{iv} \leftarrow Iv_{iv}$ -1

else $Iv_{iv} \leftarrow Iv_{iv}$ +1

if $(v_{rh},v_{rv}) \neq (0, 0)$ **then**

if the packet is received from X_r^- **then** $v_{rh} \leftarrow v_{rh}$ -1

else $v_{rh} \leftarrow v_{rh}$ +1

if the packet is received from Y_r^- **then** $v_{rv} \leftarrow v_{rv}$ -1

else $v_{rv} \leftarrow v_{rv}$ +1

else if *flag* = "floating" **then**

if the packet is received from X_0^- **then** $v_{oh} \leftarrow v_{oh}$ -1

else $v_{oh} \leftarrow v_{oh}$ +1

if the packet is received from Y_0^- **then** $v_{ov} \leftarrow v_{ov}$ -1

else $v_{ov} \leftarrow v_{ov}$ +1

//Step 2: if the router is the source node on rank-*r*, identify
//the intermediate router and update the vector list
if *i* is on rank-*r* **then**
if *flag* = "floating" **then**
set *flag* as "routing"
$S_r \leftarrow i$
$D_r \leftarrow (v_{rh}, v_{rv})$
compute $R_{Sr}^{\ d}$ as defined
compute $R_{Dr}^{\ d}$ as defined
$k \leftarrow 0$
compute R_k as defined
while $R_k \cap R_S^{\ d} \cap R_D^{\ d} = \varnothing$ **do**
$k \leftarrow k + 1$
compute R_k as defined
set *I* as one router in $R_k \cap R_S^{\ d} \cap R_D^{\ d}$
insert $I=(Iv_{ih}, Iv_{iv})$ to the intermediate router list on rank-*r*
if $(Iv_{ih,}\ Iv_{iv}) \neq (0, 0)$ **then**
if $Iv_{ih} \neq 0$ **then** send the packet to Iv_{ih} direction
else send the packet to Iv_{iv} direction
if $(v_{rh}, v_{rv}) \neq (0, 0)$ **then**
if $v_{rh} \neq 0$ **then** send the packet to v_{rh} direction
else send the packet to v_{rv} direction
else if find $p = r$-1 to 0 that $(v_{ph,}\ v_{pv}) \neq (0, 0)$ **then**
$r \leftarrow p$, **goto** step 2
else goto end
//Step 3: floating to a node nearest to node *i* on rank-*r*
find the non-faulty node $j(j_x, j_y)$ on rank-*r* torus with min{abs($i_x - j_x - v_{0h}$)+ abs($i_y - j_y$-v_{0v})} by checking the table on node *i*
set *flag* as "floating"
if $i_x \neq j_x$ **then** send the packet to $(j_x - i_x)$ direction
else send the packet to $(j_y - i_y)$ direction
end

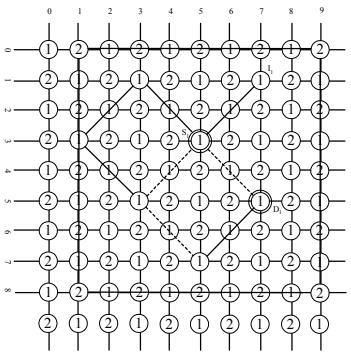

Figure 10. An example of the FVRMF algorithm.

Fig. 10 shows an example of the FVRMF algorithm under multiple failures on the same rank torus. There are three link failures on rank-1 torus shown in dotted lines. According to the FVRMF algorithm, I_1 will be selected as the best intermediate router between S_1 D_1 with the minimized number of routers.

If there exists a possible route between a source node to a destination node, we can find a best detoured route with the least number of routers using the FVRMF algorithm. However, due to the complexness of multiple failures, there is no bound on the number of routers that will be used on the detoured route.

6. Conclusion

In this paper, we investigate fault-tolerant routing schemes on NoC systems featuring a RDT-based interconnection network. We proposed the FVRSF and FVRMF algorithms for fault-tolerant routing under single link/node failure or multiple link/node failures respectively. Both algorithms are based on deterministic routing, and hence, no virtual channel is needed. We showed that using FVRSF algorithm, the number of routers on the detoured route is at most 2 more than the number of routers on the original route. The FVRMF algorithm guarantees that an alternative route is always found by identifying the best intermediate router. Future work includes the study of the delay performance under real traffic patterns and the study of fault-tolerant routing schemes under different delay and fault models.

References

[1] L. Benini and G.D. Micheli, "Networks on chips: a new SoC paradigm," *IEEE Computer*, pp. 70-78, Jan. 2002.

[2] R. Casado *et al.*, "A protocol for deadlock-free dynamic reconfiguration in high speed local area networks," *IEEE Trans. Parallel & Distributed Systems*, vol. 12, no. 2, pp. 115-132, 2001.

[3] W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputers networks using virtual channels," *IEEE Trans. Parallel & Distributed Systems,* vol. 4, no. 4, pp. 466-475, 1993.

[4] Dally W. J. and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," in *Proc. Design Automation Conf.*, 2001, pp. 684-689.

[5] T. Dumitras, S. Kerner, and S. Marculescu, "Towards on-chip fault-tolerant communication," in *Proc. ASP-DAC*, 2003, pp. 225-232.

[6] M.E. G´omez, J. Duato, J. Flich, P. L´opez, A. Robles, N.A. Nordbotten, O. Lysne, and T. Skeie, "An efficient fault-tolerant routing methodology for meshes and tori," *Computer Architecture Letters*, vol. 3, pp. 10-13, Jul. 2004.

[7] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Trans. Computer-Aided Design*, vol. 24, no. 4, Apr. 2005.

[8] ITRS, International Technology Roadmap for Semiconductors, Update 2002.

[9] T. Li, Y. Yu, P. Li, J. Xu, X. Ma and Y. Yang, "Practical routing and torus assignment for RDT," in *Proc. ISPAN*, 2004, pp. 30-35.

[10] J. Liu, M. Shen, L. Zheng, and H. Tenhunen, "System level interconnect design for network-on-chip using interconnect IPs," in *Proc. Int'l Workshop System-Level Interconnect*, 2003, pp. 117-124.

[11] L.M. Ni and P.K. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, no. 2, pp. 62-76, 1993.

[12] Y. Yang, A. Funahashi, A. Jouraku, H. Nishi, H. Amano, and T. Sueyoshi, "Recursive diagonal torus: an interconnection network for massively parallel computers," *IEEE Trans. Parallel & Distributed Systems,* vol. 12, no. 7, pp. 701-715, Jul. 2001.

[13] Y. Yu, M. Yang, Y. Yang, and Y. Jiang, "A RDT-based interconnection network for scalable NoC designs," in *Proc. IEEE ITCC*, 2005, pp. 729-734.