We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

186,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS
BOOK CITATION INDEX
INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

A Novel Approach for the Design of Fault-Tolerant Routing Algorithms in NoCs: Passage of Faulty Nodes, Not Always Detour

Masaru Fukushi and Yota Kurokawa

Abstract

Due to the faults in system fabrication and run time, designing an efficient fault-tolerant routing algorithm with the property of deadlock-freeness is crucial for realizing dependable Network-on-Chip (NoC) systems with high communication performance. In this chapter, we introduce a novel approach for the design of fault-tolerant routing algorithms in NoCs. The common idea of the fault-tolerant routing has been undoubtedly to detour faulty nodes, while our approach allows passing through faulty nodes with the slight modification of NoC architecture. As a design example, we present an XY-based routing algorithm with the passage function. To investigate the effect of the approach, we compare the communication performance (i.e. average latency) of the XY-based algorithm with well-known region-based algorithms under the condition of with and without virtual channels. Finally, we provide possible directions of future research on the fault-tolerant routing with the passage function.

Keywords: network-on-chip (NoC), fault-tolerant routing, two-dimensional mesh, passage, dependability

1. Introduction

Demand for computation power will never stop, and it is ever increasing year by year in a variety of scientific research fields. As can be seen in the modern multi-processor system-on-chips and many core systems [1–3], this makes computing hardware devices equip with hundreds or thousands of processor cores for providing high computation power by parallel processing on a chip. For the implementation of such highly-integrated parallel systems, Network-on-Chip (NoC) has emerged as a promising paradigm. In NoCs, each node (i.e. a processor core with a router) is connected by an on-chip network and communication among them are done by transferring packets on the network. Using global interconnection structure reduces the difficulty of wiring design and latency of signal transmission and offers high scalability, in comparison with point-to-point signal wires or shared busses [4].

One of the most important and fundamental issues that must be addressed for NoCs is fault-tolerant routing. Definitely, routing of packets plays a key role in

parallel systems because it has significant impact on the overall system performance. Meanwhile, the occurrence of faults during system fabrication and run time is inevitable, and it is almost impossible to completely remove their adverse effects from the systems even if some redundancy is incorporated. A single faulty node disrupts packet routing between many pairs of nodes, resulting in the failure of the entire system. Besides, a deadlock (i.e. circular waiting of packets) will occur if an adopted routing algorithm is imperfect. Once the deadlock occurs, packets can never proceed to the destinations, and thus resulting in the malfunction of the entire system. Therefore, designing an efficient fault-tolerant routing algorithm with the property of deadlock-freeness is crucial for realizing dependable NoC systems with high communication performance.

So far, extensive research has been devoted to fault-tolerant routing not only for NoCs but for traditional parallel computers. Although there exist several basic approaches, as we reviewed in Section 2, the common idea of the fault-tolerant routing remains unchanged from the earliest, and it has been undoubtedly to detour faulty nodes. This is quite natural because the purpose of the fault-tolerant routing is to route packets from source to destination nodes without entering faulty parts. Meanwhile, it is also obvious that detouring faulty nodes increases the communication latency as the packet is misrouted apart from the minimal path to the destination. One may consider that the increase in the communication latency is very little. This is true if packets are routed without interfered by other packets. However, it can be substantial increase under the situation where a number of packets are routed simultaneously and thus frequently blocked by others.

In this chapter, we introduce a novel approach for the design of fault-tolerant routing algorithms. In contrast to the common idea of detouring faulty nodes, our approach allows passing through them with the slight modification of NoC architecture. We provide a general methodology for designing a fault-tolerant routing algorithm with the passage of faulty nodes. As a design example, we describe an XY-based routing algorithm with the passage function. By computer simulations, we reveal the communication performance of the algorithm under the condition of with and without Virtual Channels (VCs), in comparison with well-known region-based routing algorithms.

The rest of this chapter is organized as follows: Section 2 presents the architecture of NoC, the basis of packet routing, and the related works of fault-tolerant routing algorithms. Section 3 presents the basic idea of the proposed approach and XY-based fault-tolerant routing algorithm, inclusive of the proof of the deadlock-freeness. Section 4 presents the results of the performance evaluation. Finally, Section 5 concludes the chapter with some possible direction of future research.

2. NoC architecture and fault-tolerant routing

2.1 2D mesh NoC

Target NoC topology in this chapter is a popular 2D mesh which has nodes of m rows and n columns. **Figure 1** shows the general architecture of the 2D mesh NoC. Each node is composed of a processor core and a router. The processor core runs assigned computation tasks, which can be either independent one or a part of parallel programs, while the router forwards packets to one of the neighbor routers or its local processor core to support the communication among cores. Each node has a unique address (i, j), where $i \in X = \{1, 2, \cdots, m\}$ and $j \in Y = \{1, 2, \cdots, n\}$. In the 2D mesh NoC, a node (i, j) is connected to at most four neighbor nodes, $(i \pm 1, j)$ and $(i, j \pm 1)$, via two unidirectional links, if $i \pm 1 \in X$ and $j \pm 1 \in Y$. For the ease of
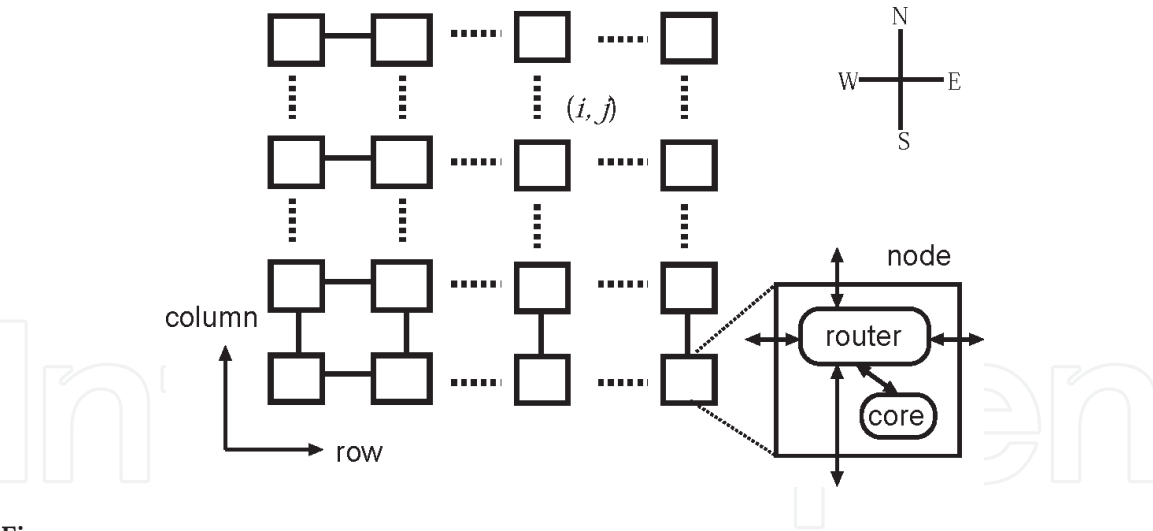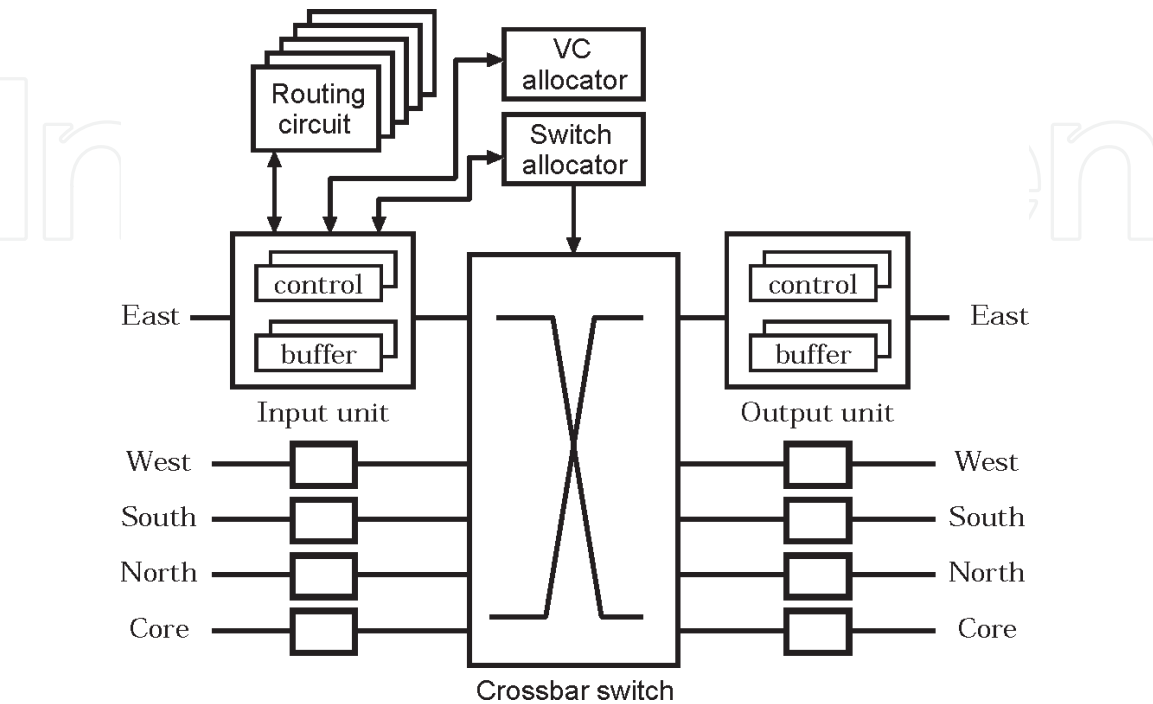
Figure 1.
Architecture of 2D mesh NoC.

explanation, positive/negative directions of x (row) and y (column) axes are called east/west and north/south, respectively.

Figure 2 shows the block diagram of the router. In the typical wormhole routing adopted in NoCs, a packet is divided into a sequence of fixed-size units of data, called flits, and transferred by routers one after another. Each router consists of five input/output units, five routing circuits, a VCs allocator, a crossbar switch, and a switch allocator. When a head flit (i.e. a flit having routing information) is transferred to a router and stored into a buffer in the input unit, the following processes are applied.

1. An output port to which the flit is forwarded is determined by the routing circuit.

2. A VC (i.e. buffer) to be used is determined by the VC allocator.

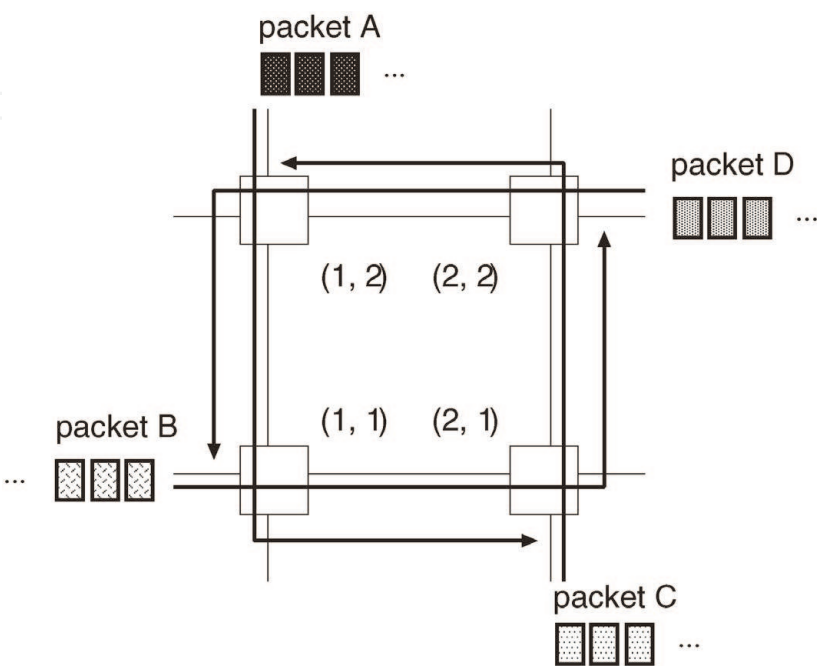**Figure 2.**
Architecture of router.

3. The crossbar switch is set up by the switch allocator to connect the input unit and the output unit associated with the determined output port.

4. The flit is moved from the input to the output units.

5. Finally, the flit is forwarded to the corresponding input unit of the next router.

The incoming head flit moves to the next router at the fifth cycle if there are no contentions, and the subsequent flits follow it in a pipeline fashion. This is a standard five-cycle router [5]. If no VC is used in an adopted routing algorithm, the router is reduced to a four-cycle router, as the second process (i.e. VC allocation) is omitted.

2.2 Deadlock

In routing packets in accordance with a routing algorithm, the algorithm must care about the occurrence of deadlocks. Deadlock is a situation where packets wait on one another to release the buffers. **Figure 3** shows an example of a deadlock. In this example, a packet A is routed to the node $(2, 1)$ via $(1, 1)$, which is blocked by a packet B at $(1, 1)$. The packet B is also routed to $(2, 2)$ via $(2, 1)$, which is blocked by a packet C. The packets C and D are also routed similarly, but blocked by the packets D and A, respectively, resulting in circular waiting of packets. Once a deadlock occurs, packets involved in the circular waiting cannot proceed toward the destinations forever. Therefore, deadlock-freeness must be guaranteed in the routing algorithm.

There have been two approaches to preventing deadlocks; approaches with and without VCs. In the approach with VCs, the original network is multiplexed into several virtual networks by VCs. For example, in **Figure 3**, if packets A and C are supposed to be routed on a virtual network with a VC and packets B and D are on a different virtual network with other VC, then the circular waiting is decomposed and packets get to proceed to the destinations. In the approach without VCs, a routing algorithm is carefully designed so that deadlocks never occur in the original

Figure 3.
Example of a deadlock.

physical network. For example, in **Figure 3**, if packets A and C are routed via $(2,2)$ and $(1,1)$, respectively, i.e., forced to move in x-direction first, then no deadlocks occur. This approach has an area advantage over the former approach because the implementation of VCs involves the replication of buffers and control circuits in all input/output units in all routers.

2.3 Related works

Fault-tolerant routing has been the subject of extensive research not only for NoCs but for traditional parallel computers over the past few decades. Most of the existing fault-tolerant routing algorithms for 2D mesh networks fall into the following three categories: (1) those employ a routing table, (2) those relax the constraints of guaranteed delivery or deadlock-freeness, and (3) those define some form of fault information on routers and detour paths.

In the first category, a routing table is employed in each router to route packets to the destinations. Routing tables contain routing information such as next hops for destinations, status of the network, and/or fault information. Hsin et al. [6] proposed an algorithm which employs ant colony optimization for traffic balancing. Liu et al. [7] proposed an algorithm which introduces coarse and fine-grained look-ahead schemes to obtain the information of other routers within the range of four hops. This algorithm requires two VCs for each input/output port to route packets. Zhao et al. [8] proposed an algorithm to provide minimal paths using the information of whole network. In general, those algorithms offer flexible route selection; however, they require a large amount of circuits to implement a routing table and complex calculation mechanism to create/update the table in all routers.

In the second category, constraints of guaranteed delivery or deadlock-freeness is relaxed to ease the design of routing algorithms. Janfaza et al. [9] proposed an adaptive routing algorithm which employs timeout and packet reinjection. Information of intermediate nodes is recorded in each packet and two VCs are used to route packets. Sinha et al. [10] proposed an algorithm based on the common XY and YX routing. This algorithm allows U-turn using several VCs. Wang et al. [11] proposed an algorithm which relaxes transmission accuracy for the applications that allow lossy communication. This algorithm discards conflicting approximate flits without retransmission and recovers them after packet transmission. Those algorithms are imperfect in that 100% packet reachability or deadlock-freeness are not guaranteed by the routing algorithms. Retransmission of packets generally results in a high communication latency.

In the third category, some form of fault information is defined for routers to detour faulty parts. Usually, clusters of faulty nodes, called fault blocks, are defined in the networks with the detour paths. Chen et al. [12], Holsmark et al. [13], Fu et al. [14], and Fukushima et al. [15] proposed routing algorithms which generate rectangular fault blocks and detour them without using VCs. Wu [16] and Chalasani et al. [17] proposed routing algorithms which can deal with convex and nonconvex fault blocks, respectively. In [17], four VCs are used to choose shorter detour paths. Those algorithms called region-based algorithms provide simple but strict routing rules to guarantee the deadlock-freeness and 100% packet reachability, and thus, they can be implemented as a small circuit in the routing circuit of each router. They are practical and suitable for NoCs. However, one drawback is that fault blocks may include several non-faulty nodes, which are to be deactivated (i.e. unused nodes); therefore, the number of unused nodes and the length of detour paths are prone to increase if there exist a number of faulty nodes in the network.

Although extensive research has been devoted to designing fault-tolerant routing algorithms inclusive of the above ones, the common idea remains

unchanged from the earliest, and it has been undoubtedly to detour faulty nodes. If a packet must detour a faulty node (i, j), the hop count between $(i - 1, j)$ and $(i + 1, j)$ is increased by two, which increases the communication latency by at least ten cycles in an NoC with five-cycle routers. This can be substantial increase in the situation where the network gets congested (i.e. by packet blocking) or includes a number of faulty nodes (i.e. by detouring). This is a serious problem for a large-scale parallel system on a single VLSI chip.

3. Proposed method

### 3.1 Basic approach and NoC architecture

Motivated by the problem presented in the previous section, we introduce a novel approach based on the opposite idea of the common approach; our approach allows packets to pass through the faulty nodes with slight modification of NoC architecture (originally proposed in [18]). Basic idea behind this approach is to reduce communication latency by saving detouring as much as possible.

Figure 4 shows the modified NoC architecture for supporting the proposed approach. Four electrical switches, bypass links, buffers to store one flit are added around each router. Each switch has two states, either normal or passage, as shown in this figure. In the state of passage, packets from the neighbor node are input to the bypass link not to the node. The switch states can be determined easily, once the node is tested and judged as faulty or not. In other words, they are determined so that the node becomes passage state if it is faulty or remains normal state otherwise. It is worth to note that buffers can be removed if packets are transmitted between routers in an asynchronous way.

3.2 Design methodology for fault-tolerant routing algorithms

Here, we provide a design methodology for fault-tolerant routing algorithms based on the passage of faulty nodes.

First, we clarify the fault model. A common assumption is made for faults [6, 12–16, 18, 19]; that is, permanent faults are considered to be associated only with nodes. In practice, the probabilities of links, switches, and buffers being faulty are not zero, though they will be substantially small because of the simplicity of their
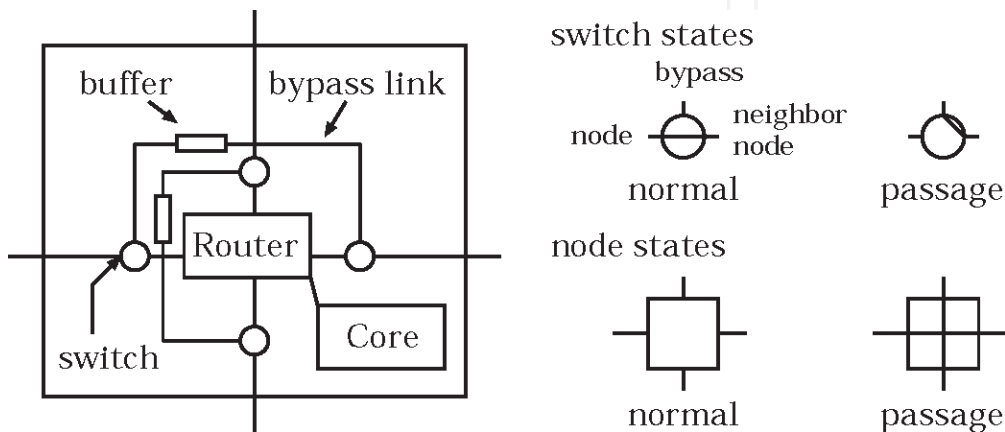
Figure 4.
Modified NoC architecture.

circuits [19]. For the faults on those circuits, one can employ some popular redundancy technique such as duplication and triplication if necessary.

Below is the general methodology for designing fault-tolerant routing algorithms with the passage function.

Step 1 Choose a base routing algorithm from the existing algorithms or design a new one. This algorithm is not necessary to be fault-tolerant, but should be deadlock-free.

Step 2 Decide which faulty nodes can be passed through and define routing rules for the remaining faulty nodes to be detoured. The resultant routing algorithm, denoted by R, is a candidate for the final algorithm.

Step 3 Verify if the candidate routing algorithm R is deadlock-free or not. If not, return to Step 2 to modify R.

Step 4 Repeat Steps 2 and 3 until a fault-tolerant and deadlock-free routing algorithm R is obtained.

3.3 Routing algorithm based on XY routing

As a design example, we introduce a new fault-tolerant routing algorithm based on the popular dimension order routing (i.e. XY routing for 2D meshes) [18]. In the following, we explain the details of each step in the design methodology.

In Step 1, we choose XY routing as a base routing algorithm. In XY routing, packets first proceed along x-direction until they reach the nodes having the same x-coordinates as the destinations, then proceed to the destinations along y-direction without changing the x-coordinates.

In Step 2, we must consider the case where passage must be restricted. For example, suppose that a packet moves from node $(i-1,j)$ to $(i+1,j)$ passing through a faulty node (i,j). If the destination node is (i,j') where $j' \neq j$, the packet keeps moving between $(i-1,j)$ and $(i+1,j)$ because the x-coordinate of the current node will never be the same as that of the destination node. The same kind of thing never happens in the y-direction. Therefore, we allow packets passing through faulty nodes only in the y-direction and let them detour faulty nodes through the south side in the x-directional movement. (This restriction is relaxed a bit in the final routing algorithm).

Then, we need to consider the case where a faulty node is on the south boundary of the network. In this case, packets cannot detour it through the south side, as they face the south boundary. To cope with this, we give the following definitions.

Definition 1 A faulty node (i,j) which is on the south boundary of a mesh network is defined as a South Faulty (SF) node, where $j = 0$.

Definition 2 A faulty node (i,j) which exists in the eight neighbor of any SF node (i',j') is also defined as an SF node, where $(i'-1 \leq i \leq i'+1)$ and $i \in X$, and $(j'-1 \leq j \leq j'+1)$ and $j \in Y$.

The process in Definition 2 is repeated until no SF nodes are generated. For SF nodes, we give a new routing rule such that packets must detour them through the north side.

In Step 3, we check the deadlock-freeness of the resultant routing algorithm R where packets detour faulty nodes/SF nodes through south/north side in the x-directional movement of XY routing and always pass through faulty nodes in the y-directional movement. Unfortunately, it is not hard to find the case where a deadlock occurs. **Figure 5** illustrates the example of a possible deadlock. Packets generated at nodes S1/S2 detour faulty and SF nodes in accordance with the routing algorithm R, but finally they are blocked by each other, resulting in circular waiting. Note that, generally, the deadlock in **Figure 5** can be occurred by more than two packets.
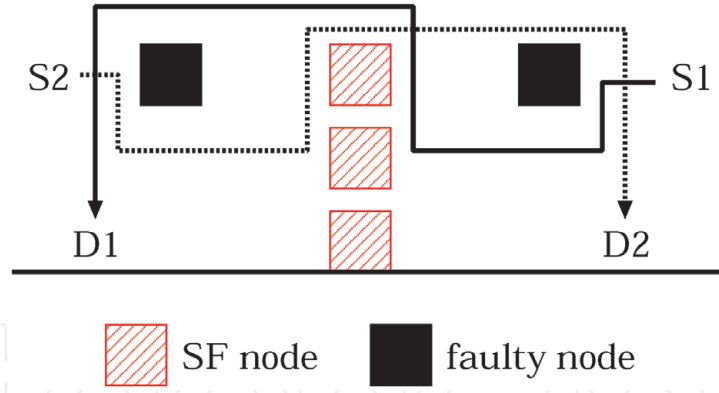
Figure 5.
Example of a possible deadlock.
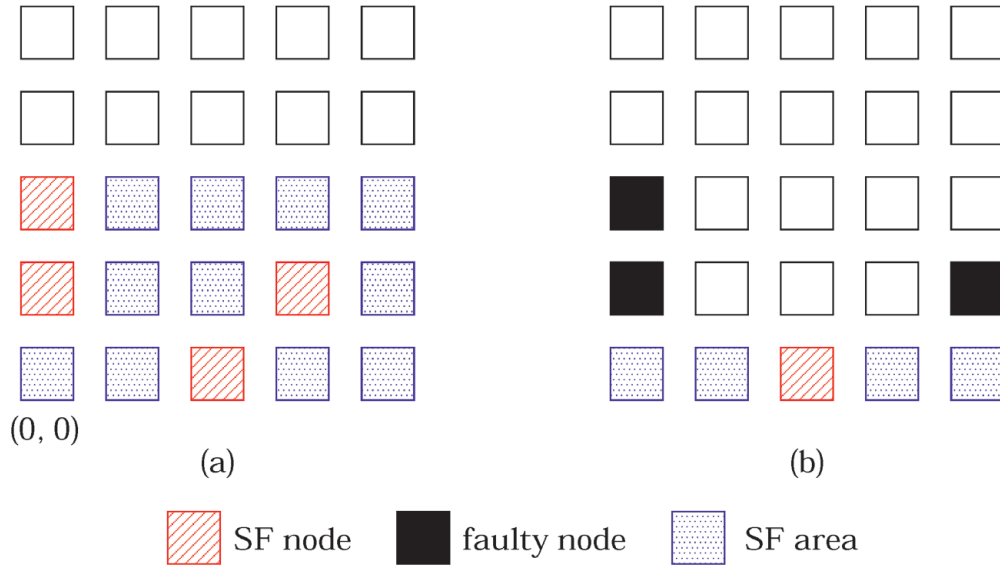
Figure 6.
SF area for xy-based routing algorithm.

To cope with the deadlock, we give the following definitions.

Definition 3 Let (i', j') be the coordinates of the north most SF node generated by repeating Definition 2. SF area is defined as the area consisting of all nodes (i, j) such that $j \leq j'$ for any $i \in X$. All faulty nodes in the SF area are changed to SF nodes.

For the newly generated SF nodes in Definition 3, the processes in Definitions 2 and 3 are repeated until no SF nodes are generated.

Figure 6 illustrates examples of the SF area. In the case of **Figure 6** (a), faulty node $(2, 0)$ is changed to an SF node by Definition 1 and subsequently $(3, 1)$ is changed to an SF node by Definition 2. Then, faulty node $(0, 1)$ on the west boundary is included in the SF area and thus changed to an SF node by Definition 3. Finally, faulty node $(0, 2)$ is changed to an SF node by Definition 2. According to the above processes, the SF area is configured as shown in the figure. In the case of **Figure 6** (b), faulty node $(4, 1)$ is not included in the SF area; hence, faulty nodes $(0, 1)$ and $(0, 2)$ are not changed to SF nodes.

By the above definitions, the deadlock in **Figure 5** can be solved. By Definition 3, two faulty nodes in **Figure 5** are changed to SF nodes and the SF area is defined as shown in **Figure 7**. Then, two packets detour the SF nodes, not faulty nodes, through the north side and get to proceed to the destinations as shown in the figure.

Figure 8 describes the finally obtained proposed routing algorithm. In this figure, C and D represent a current and a destination node, respectively. The
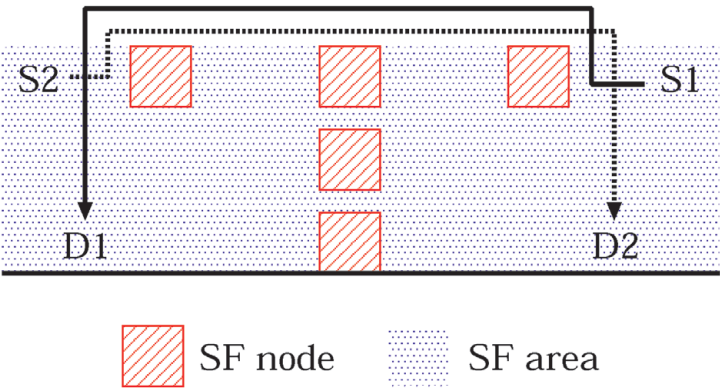
Figure 7.
Routing example without deadlocks.

proposed routing algorithm allows packets to pass through faulty and SF nodes in the movement of x-directions only if C and D are on the same row (i.e. lines 8 and 18 in **Figure 8**), while it always allows passage in the movement of y-directions.

Next, we prove the deadlock-freeness of the proposed algorithm described in **Figure 8**. First, we define turns of packets.

Definition 4 ES turn is a turn in which an incoming packet from the East neighbor is sent to the South neighbor at a router. Other seven turns are also defined similarly as shown in **Figure 9**.

Theorem 1 The routing algorithm in **Figure 8** is deadlock-free.

Proof. We prove that circular waiting of packets never occurs in both clockwise and counter-clockwise directions.

For the clockwise direction, we show that an SW turn is never aligned with an NE turn. The SW turn occurs in a non-SF area; however, the NE turn never occurs

```
1    XY-based fault-tolerant routing(C, D)
2    if (C is D)
3      consume the packet
4    elsif (D is to the west of C)
5      if (west neighbor is faulty)
6        if (west neighbor is an SF node)
7          Next_Route = North
8        elsif (C and D are on the same row)
9          Next_Route = West
10       else
11         Next_Route = South
12     else
13       Next_Route = West
14   elsif (D is to the east of C)
15     if (east neighbor is faulty)
16       if (east neighbor is an SF node)
17         Next_Route = North
18       elsif (C and D are on the same row)
19         Next_Route = East
20       else
21         Next_Route = South
22     else
23       Next_Route = East
24   elsif (C and D are on the same column)
25     if (D is to the north of C)
26       Next_Route = North
27     else
28       Next_Route = South
```

Figure 8.
A pseudo-code of the proposed routing algorithm.

9
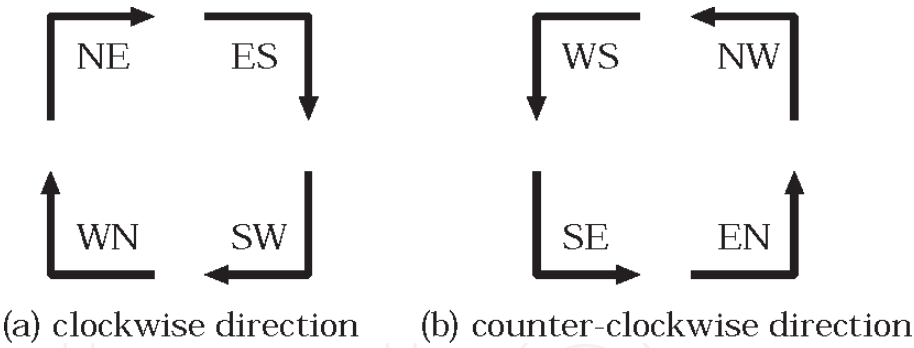
Figure 9.
Possible eight turns of packets.

in the area because it only occurs in an SF area. Conversely, the NE turn occurs in an SF area; however, the SW turn only occurs in a non-SF area. From the above, circular waiting never occurs in the clockwise direction.

For the counter-clockwise direction, we omit the proof because it is symmetrical to the proof for the clockwise direction.

Thus, the proposed routing algorithm is proved to be deadlock-free.   □

4. Performance evaluation

4.1 Evaluation condition

To investigate the effect of the proposed approach, we have conducted computer simulations with a cycle-accurate custom simulator developed in C. This simulator accurately simulates the behavior of flits in all routers in a 2D mesh NoC. As explained in Section 2, if there are no contentions, each flit takes five (or four) cycles to move to the next node when VCs are used (or not used) in the adopted routing algorithm. Note that, as flits are transmitted in a pipeline fashion, a subsequent flit moves to the next node one cycle after the movement of the precedent flit if buffer space is available in the input unit of the router. It also takes one cycle to pass through a faulty node, as a buffer is placed on the bypass link.

Following three methods are evaluated in the simulations with the parameters listed in **Table 1**.

- Fukushima's method [15]: packets detour *rectangular* fault blocks with no additional VCs (denoted by M_r).

Parameter	Value	Unit
Network size	10×10	Nodes
Fault rate (f)	2, 4, 6, 8, 10	%
Packet length	16	Flits
Packet generation rate (p)	0.05 ∼ 1.0	Packets/cycle/network
Input (Output) buffer depth	8 (1)	Flits
Simulation (Stabilization) cycle	50,000 (5000)	Cycles

Table 1.
Simulation parameters.

- Chalasani's method [17]: packets detour *nonconvex* fault blocks, such as L, T, and + shapes, as well as rectangular one using four VCs per link (denoted by M_{nc}).

- Our method [18]: packets can *pass through* or detour faulty nodes with no additional VCs (denoted by M_p).

The number of VCs required for each algorithm is different, and VCs can also be employed in the algorithms which require no VCs for the purpose of congestion avoidance. We use the notation of *M-n* to indicate the number of VCs (i.e. buffers), where M is either M_r, M_{nc} or M_p and n represents the number of VCs. For example, M_{nc}-4 denotes Chalasani's method with four VCs; M_p-1 denotes our proposed method with one buffer (i.e. no additional VCs).

In the simulations, faulty nodes are generated randomly according to the fault rate f, and packets are also generated randomly at each cycle according to the packet generation rate p during the simulation time of 50,000 cycles. Latency is not measured up to 5000 cycles to stabilize the network. The same fault patterns are used for all methods for fair comparison. The above trial is repeated 1000 times and the following metrics are measured.

Average latency is defined by the average cycles required for packets from the generation to the arrival.

Average node utilization rate is defined as the percentage of available nodes among all non-faulty nodes, i.e., given by $\{mn(1-f) - u\}/mn(1-f)$, where m and n is the number of rows and columns, respectively, f is the fault rate, and u is the number of unused nodes.

To make a quantitative evaluation of average latency, we define maximum latency reduction rate of an algorithm M_a for an algorithm M_b by

$$R(M_a, M_b) = max_p \, r_p(M_a, M_b), \qquad (1)$$

where $r_p(M_a, M_b)$ represents latency reduction rate of M_a for M_b at the packet generation rate p and is defined by the following expression.

$$r_p(M_a, M_b) = \frac{L_b - L_a}{max \, (L_a, L_b)} \times 100, \qquad (2)$$

where L_a and L_b is the average latency of M_a and M_b at p, respectively, and $max \, (L_a, L_b)$ is a function to return the larger of L_a and L_b.

4.2 Evaluation results

4.2.1 Overall trend

Figures 10–14 show the average latency as a function of packet generation rate p for each fault rate f. In the figures, x axis represents p, and a larger value indicates a higher request load; meanwhile, y axis represents average latency, and a larger value indicates higher delay in delivery of packets. When p is relatively low, the average latency of three algorithms is almost the same. On the other hand, when it is high, the difference becomes significant. The average latency of M_p and M_{nc} is smaller than that of M_{nc} and M_r, respectively, regardless of f and the number of VCs. M_p outperforms M_{nc} without using VCs, indicating that passage of faulty nodes has a significant impact on reducing average latency. As f increases, the
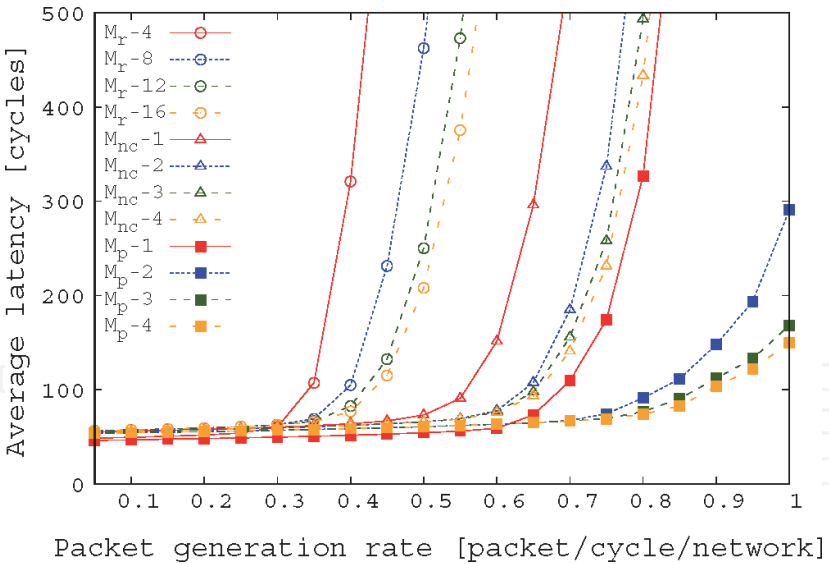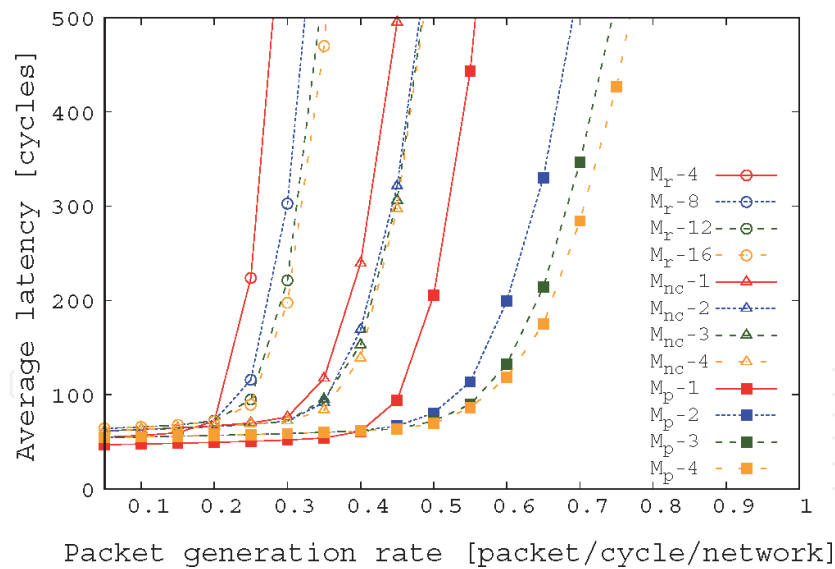
Figure 10.
Average latency for f = 2%.
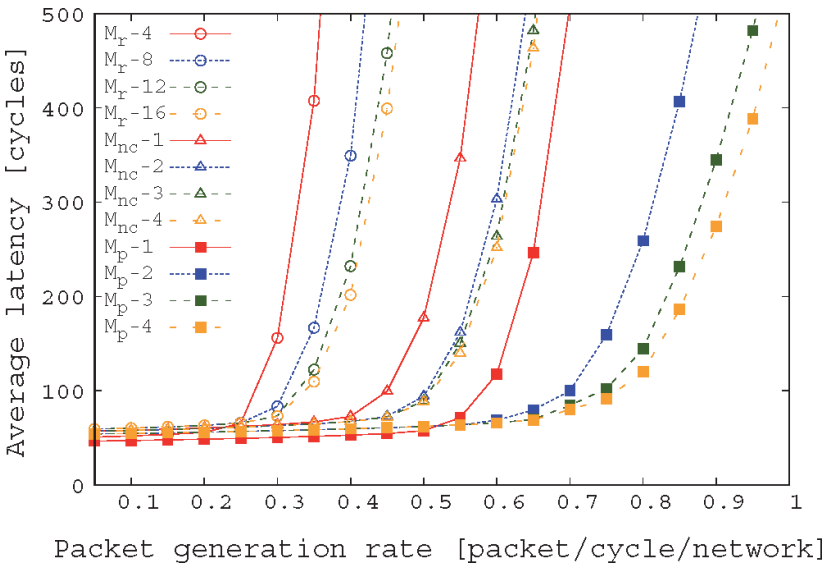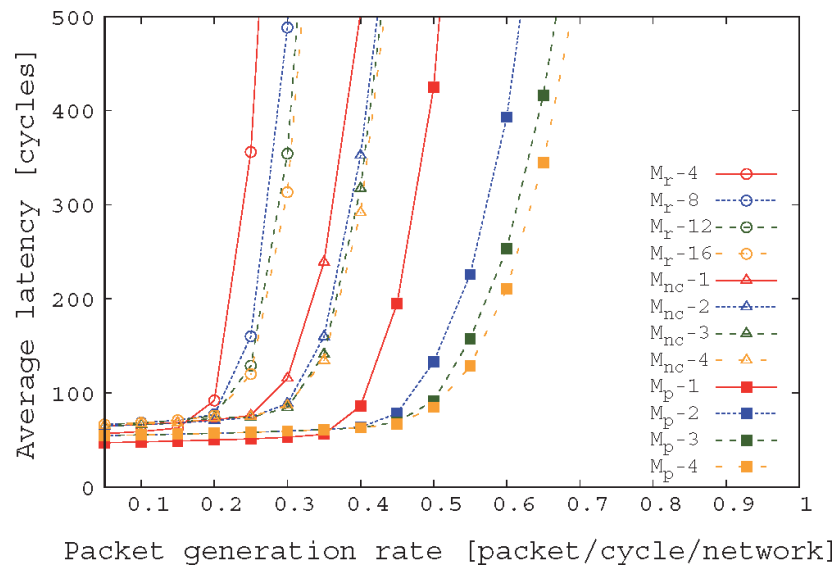

Figure 11.
Average latency for f = 4%.
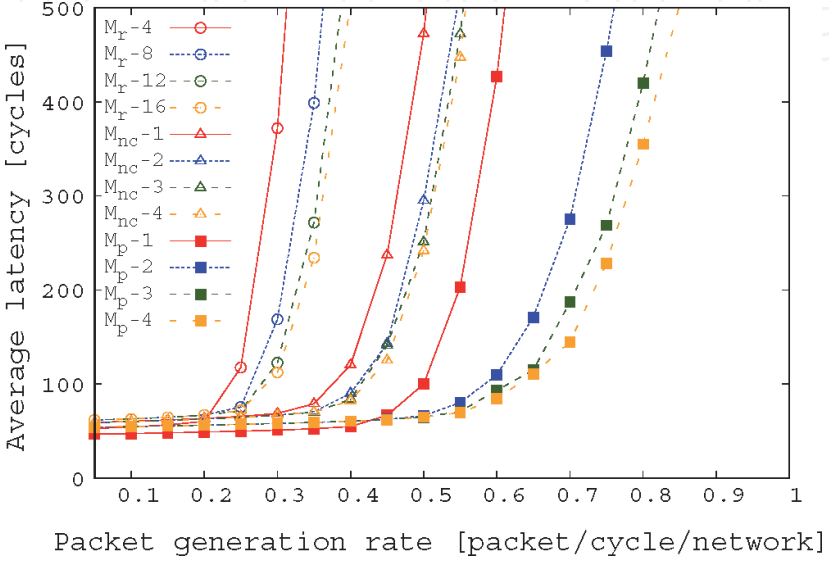
Figure 12.
Average latency for f = 6%.

Figure 13.
Average latency for f = 8%.

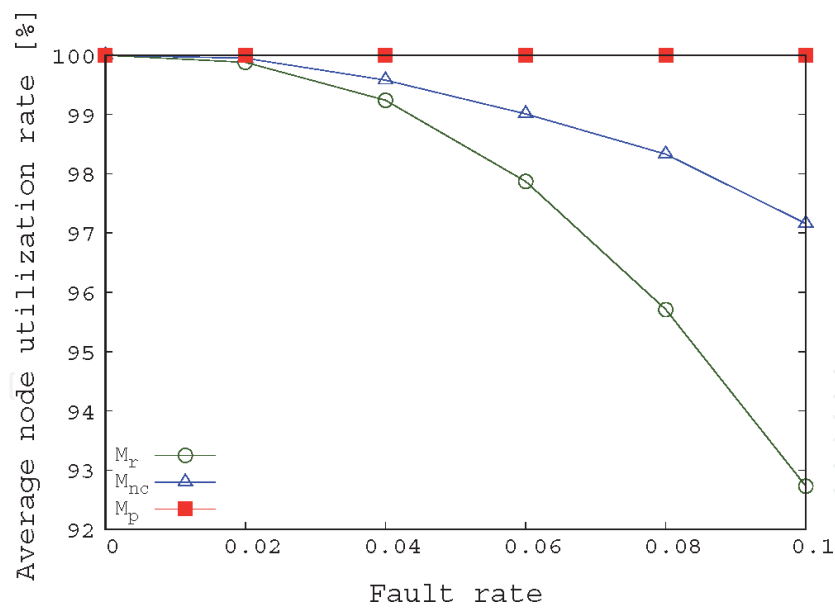**Figure 14.**
Average latency for f = 10%.

average latency of those algorithms is increased due to the increased number of faulty nodes.

Figure 15 shows the average node utilization rate. M_r and M_{nc} generate rectangular and nonconvex fault blocks, and accordingly, about 7% and 3% of non-faulty nodes become unused nodes, respectively. This is a cause of longer detour paths. Meanwhile, M_p does not generate any fault blocks and always keeps 100% utilization rate.

For the results shown in **Figures 10–14**, we make performance comparison of the routing algorithms in the following three conditions.

4.2.2 Performance comparison of the original routing algorithms

The average latency of the original routing algorithms is compared numerically (i.e. comparison of M_r-1, M_{nc}-4, and M_p-1). **Table 2** shows the maximum reduction rate of M_r-1 and M_p-1 for M_{nc}-4. The value of p at which the maximum reduction rate is attained is noted in parenthesis. As we saw in **Figures 10–14**, average latency of M_r is higher than that of M_{nc}; hence, all rates of $R(M_r$-1, M_{nc}-4) are negative

Figure 15.
Average node utilization rate vs. fault rate.

				f		
M_a	M_b	2%	4%	6%	8%	10%
M_r-1	M_{nc}-4	−94 (0.55)	−93 (0.45)	−92 (0.40)	−91 (0.35)	−89 (0.30)
M_p-1	M_{nc}-4	82 (0.75)	82 (0.60)	79 (0.50)	81 (0.45)	83 (0.40)

Table 2.
Maximum latency reduction rate $R(M_a, M_b)$ for the original algorithms.

values for any f. From this table, we found that M_p-1 reduces the average latency of M_{nc}-4 by about at least 79% without using additional VCs.

4.2.3 Performance comparison of routing algorithms with increased VCs

Next, the average latency of the three algorithms is compared by increasing the number of VCs twofold, threefold, and fourfold from the original (i.e. comparison of M_*-n, M_*-$2n$, M_*-$3n$, and M_*-$4n$). **Table 3** shows the results. The following can be found in the evaluation results:

1. With twofold VCs, the average latency can be reduced by about at least 66% and 33% for $f = 2$ and 10, respectively, compared with the original.

2. Fourfold increase in the number of VCs have only a marginal effect in reducing average latency.

3. Effect of latency reduction is higher in the algorithms with no VCs (i.e. M_r and M_p), and M_p shows the highest reduction rate for any f.

4.2.4 Performance comparison of routing algorithms with fixed number of VCs

Finally, the average latency of the three algorithms is compared under the same number of VCs (i.e. comparison of M_r-4, M_{nc}-4, and M_p-4). **Table 4** shows the maximum reduction rates of M_r-4 and M_p-4 for M_{nc}-4. By using four VCs, the

M_a	M_b			f		
		2%	4%	6%	8%	10%
M_r-1	M_r-2	67 (0.40)	63 (0.40)	55 (0.30)	55 (0.30)	55 (0.25)
	M_r-3	81 (0.45)	75 (0.40)	69 (0.35)	67 (0.30)	65 (0.30)
	M_r-4	84 (0.45)	78 (0.40)	73 (0.35)	71 (0.30)	69 (0.30)
M_{nc}-4	M_{nc}-8	66 (0.70)	54 (0.60)	40 (0.45)	35 (0.45)	33 (0.35)
	M_{nc}-12	73 (0.75)	60 (0.60)	47 (0.50)	38 (0.45)	41 (0.35)
	M_{nc}-16	76 (0.75)	62 (0.60)	49 (0.50)	40 (0.45)	44 (0.35)
M_p-1	M_p-2	88 (0.95)	82 (0.75)	78 (0.70)	76 (0.60)	75 (0.55)
	M_p-3	92 (1.00)	90 (0.80)	85 (0.70)	85 (0.65)	82 (0.55)
	M_p-4	92 (1.00)	91 (0.80)	89 (0.70)	87 (0.65)	86 (0.55)

Table 3.
Maximum latency reduction rate $R(M_a, M_b)$ for the algorithms with increased VCs.

M_a	M_b			f		
		2%	4%	6%	8%	10%
M_r-4	M_{nc}-4	−76 (0.60)	−75 (0.45)	−76 (0.40)	−75 (0.35)	−69 (0.35)
M_p-4	M_{nc}-4	96 (0.90)	96 (0.75)	94 (0.70)	94 (0.60)	94 (0.50)

Table 4.
Maximum latency reduction rate $R(M_a, M_b)$ for the algorithms with four VCs.

maximum reduction rates of M_r-4 and M_p-4 can be improved from the rates shown in **Table 2**, and M_p-4 always achieves more than 94% reduction rates for any f.

From the above results, we can conclude that, for reducing average latency of packet transmission, the reduction of hop count by the passage of faulty nodes, not always detour, is more effective than the avoidance of congestion using additional VCs.

4.3 Circuit amount

To evaluate the overhead of additional circuits such as switches, buffers, and links in the proposed approach, we designed two routers for M_r-1 and M_p-1 with Verilog HDL. In those routers, M_r and M_p are implemented into the routing circuits and the depth of input/output buffers is eight/one flits, respectively, as in the simulation setting. We used Xilinx Vivado EDA tool for synthesizing the routers for the target FPGA device of Vertex 7 xc7vx485tffg1761–2.

From the EDA tool, the router for M_r needs 1865 Look Up Tables (LUTs) in the FPGA device, while that for M_p needs 664 LUTs, which indicates about 64% LUT reduction. This is mainly because of the difference in the routing circuit; one routing circuit costs 193 LUTs for M_r and 18 LUTs for M_p. The small routing circuit is also benefit from the passage function. The additional circuits require only 27 LUTs, which is substantially small compared with the overall router circuit.

5. Conclusion and future work

We have introduced a novel approach for the design of fault-tolerant routing algorithms in 2D mesh NoCs. In contrast to the common idea of detouring faulty

nodes, our approach allows passing through them with the slight modification of NoC architecture. We have provided a general methodology for designing fault-tolerant routing algorithms with the passage of faulty nodes, and as a design example, we have described the XY-based routing algorithm, showing how to prevent deadlocks in the routing rules. The XY-based routing algorithm allows passage of faulty nodes in the x-directional movement if the current and destination nodes are on the same row, while always allows in the y-directional movement.

To demonstrate the effect of the XY-based routing algorithm, we measured the average latency of packet transmission by computer simulations and compared with those of the well-known region-based algorithms proposed by Fukushima et al. and Chalasani et al. The results revealed that the XY-based algorithm reduced average latency of Chalasani's algorithm by about 79% without additional VCs and 94% with the same number of VCs. From the evaluation, we have found that passage is highly effective approach to reducing the average latency rather than employing VCs for congestion avoidance. We have also designed router circuit for the XY-based algorithm and showed that the overhead of additional circuit required for the proposed approach is substantially small compared with the overall router circuit.

As the passage of faulty nodes is a simple but effective approach, we have even more room to fully investigate the effect. For example, in this chapter, we selected popular XY routing as a base algorithm, which is a deterministic routing algorithm. Designing a new routing algorithm with the passage function based on some adaptive routing algorithm is a possible future research. As the passage is not limited to 2D mesh NoCs, designing passage-based fault-tolerant routing algorithms for other popular topology such as 2D torus, 3D mesh/torus is also one of the interesting future researches.

Acknowledgements

Author details

Masaru Fukushi*† and Yota Kurokawa†
Graduate School of Sciences and Technology for Innovation, Yamaguchi University, Ube, Japan

*Address all correspondence to: mfukushi@yamaguchi-u.ac.jp

† These authors contributed equally.

IntechOpen

References

[1] Mattson TG, Wijngaart RFV, Riepen M, Lehnig T, Brett P, Haas W, et al. The 48-core SCC Processor: the Programmer's View. In: Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis. New Orleans: IEEE; 13–19 Nov. 2010. p. 1–11.

[2] Sodani A, Gramunt R, Corbal J, Kim HS, Vinod K, Chinthamani S, et al. Knights landing: Second-generation intel xeon phi product. IEEE Micro. 2016;**36**(2):34-46. DOI: 10.1109/MM.2016.25

[3] Bohnenstiehl B, Stillmaker A, Pimentel JJ, Andreas T, Liu B, Tran AT, et al. KiloCore: A 32-nm 1000-processor computational array. IEEE Journal of Solid-State Circuits. 2017;**52**(4): 891-902. DOI: 10.1109/JSSC.2016.2638459

[4] Dally WJ, Towles B. Route packets, not wires: on-chip interconnection networks. In: Proceedings of the 38th Design Automation Conference. Las Vegas: IEEE; 22 June 2001; p. 684–689. DOI: 10.1109/DAC.2001.156225

[5] Dally WJ, Towles B. Principles and practices of interconnection networks. Morgan Kaufman Publishers; 2004.

[6] Hsin HK, Chang EJ, Lin CA, Wu AY. Ant colony optimization-based fault-aware routing in mesh-based network-on-chip systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2014;**33**(11): 1693-1705. DOI: 10.1109/TCAD.2014.2347922

[7] Liu J, Harkin J, Li Y, Maguire LP. Fault-tolerant networks-on-chip routing with coarse and fine-grained look-ahead. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2016;**35**(2):260-273. DOI: 10.1109/TCAD.2015.2459050

[8] Zhao H, Bagherzadeh N, Wu J. A general fault-tolerant minimal routing for mesh architectures. IEEE Transactions on Computers. 2017;**66**(7): 1240-1246. DOI: 10.1109/TC.2017.2651828

[9] Janfaza V, Baharlouei E. A new fault-tolerant deadlock-free fully adaptive routing in noc. In: Proceedings of IEEE East-West Design & Test Symposium (EWDTS); 29 Sept.-2 Oct. 2017. Novi Sad: Serbia. IEEE; 2017. p. 1-6. DOI: 10.1109/EWDTS.2017.8110139

[10] Sinha D, Roy A, Kumar KV, Kulkarni P, Soumya J. Dn-FTR: Fault-tolerant routing algorithm for mesh based network-on-chip. In: Proceedings of the 4th International Conference on Recent Advances in Information Technology (RAIT); 15–17 March 2018; Dhanbad, India. IEEE; 2018. p. 1–5 DOI: 10.1109/RAIT.2018.8389083

[11] Wang L, Wang X, Wang Y. An approximate bufferless network-on-chip. IEEE Access. 2019;7:141516–141532. DOI: 10.1109/ACCESS.2019.2943922

[12] Chen KH, Chiu GM. Fault-tolerant routing algorithm for meshes without using virtual channels. Journal of Information Science and Engineering. 1998;**14**(4):765-783

[13] Holsmark R, Kumar S. Corrections to chen and chiu's fault tolerant routing algorithm for mesh networks. Journal of Information Science and Engineering. 2007;**23**(6):1649-1662

[14] Fu B, Han Y, Li H, Li X. ZoneDefense: A fault-tolerant routing for 2-D meshes without virtual channels. IEEE Transactions on Very Large Scale Integration Systems. 2014; **22**(1):113-126. DOI: 10.1109/TVLSI.2012.2235188

[15] Fukushima Y, Fukushi M, Yairi IE. A region-based fault-tolerant routing algorithm for 2D irregular mesh network-on-chip. Journal of Electronic Testing. 2013;**29**:415-429. DOI: https://doi.org/10.1007/s10836-013-5377-9

[16] Wu J. A fault-tolerant and deadlock-free routing protocol in 2D meshes based on odd-even turn model. IEEE Transactions on Computers. 2003;**52**(9): 1154-1169. DOI: 10.1109/TC.2003.1228511

[17] Chalasani S, Boppana RV. Communication in multicomputers with nonconvex faults. IEEE Transactions on Computers. 1997;**46**(5):616-622. DOI: 10.1109/12.589238

[18] Kurokawa Y, Fukushi M. Passage of faulty nodes: A novel approach for fault-tolerant routing on nocs. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. 2019;**E102-A**(12): 1702-1710. DOI: 10.1587/transfun.E102.A.1702

[19] Takanami I, Horita T, Akiba M, Terauchi M, Kanno T. A built-in self-repair circuit for restructuring mesh-connected processor arrays by direct spare replacement. Transactions on Computational Science XXVII. 2016; **9570**:97-119. DOI: https://doi.org/10.1007/978-3-662-50412-3_7