A Low-overhead Fault-aware Deflection Routing Algorithm for 3D Network-on-Chip

Chaochao Feng, Minxuan Zhang,
Jinwen Li, Jiang Jiang
School of Computer
National University of Defense Technology
Changsha, China
Email: {fengchaochao, mxzhang}@nudt.edu.cn

Zhonghai Lu, Axel Jantsch
Department of Electronic Systems
Royal Institute of Technology
Stockholm, Sweden
Email: {zhonghai, axel}@kth.se

***Abstract*—This paper proposes a low-overhead fault-tolerant deflection routing algorithm, which uses a layer routing table and two TSV state vectors to make efficient routing decision to avoid both TSV and horizontal link faults, for 3D NoC. The proposed switch is implemented in hardware with TSMC 65nm technology, which can achieve 250MHz. Compared with a reinforcement-learning-based fault-tolerant deflection switch with a global routing table, the proposed switch occupies 40% less area and consumes 49% less power consumption. Simulation results demonstrate that the proposed switch has 5% less average packet latency than the switch with the global routing table under real application workloads and with only 5% performance degradation under synthetic workloads in the presence of 10% link faults.**

***Keywords*-3D NoC; deflection routing; fault-tolerance**

I. Introduction

Recently, advances in three-dimensional (3D) manufacturing technologies have enabled to integrate the processors and memories on multiple vertical stacking layers to overcome the interconnect problems and design complexity of the multiprocessor SoC [1]. 3D Network-on-Chip (NoC) connects elements on different vertically stacked dies by the through silicon vias (TSV) [2]. Unfortunately, the low yield for current TSV fabrication process seriously affects the reliability of 3D NoC [3]. For now, there are many researches on the fault-tolerant routing algorithm for 2D NoC [4][5][6]. However, they cannot be used for 3D NoC directly.

Deflection routing is a non-minimal adaptive routing algorithm which can be implemented with small hardware overhead for NoC. Its non-minimal routing characteristic provides the potential to achieve fault-tolerance. In [7], a reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning has been proposed for 2D mesh NoC. In the presence of link/switch faults, the routing table can be reconfigured through the reinforcement learning method. Although it can be used for 3D NoC directly by extending the size of the routing table from $n \times 4$ to $n \times 6$ entries (where *n* is the number of nodes in the network), the area overhead of the routing table will be large as the size of the network increases. In this paper, we propose a low-overhead fault-aware deflection routing algorithm for 3D NoC, which uses a routing table of 2D mesh (layer) and two TSV state vectors instead of the global routing table for the whole network to make routing decision efficiently to avoid both horizontal link faults and TSV faults. Simulation results illustrate that the proposed switch has 5% less average packet latency than the switch with the global routing table under real application workloads and with only 5% performance degradation under synthetic workloads in the presence of 10% link faults. We also implement the switch with TSMC 65nm standard-cell library. Compared with the switch with the global routing table, the proposed switch can save area and power consumption by up to 40% and 49% respectively.

The rest of paper is organized as follows. Related work is reviewed in Section II. Section III describes the NoC architecture and fault model. The detailed routing algorithm and hardware implementation are proposed in Section IV. In Section V, simulation experimental results are presented and analyzed, followed by the conclusion in Section VI.

II. Related Work

Researches in 3D NoC are now emerging at different aspects. Pavlidis and Friedman [1] have compared 2D mesh NoC with its 3D counterpart by analyzing the zero-load latency and power consumption of each network. Addo-Quaye [8] has presented an algorithm for the thermal-aware mapping and placement of 3D NoC. A systematic performance evaluation of 3D NoC architectures has been performed in [9] to demonstrate their advantages compared to the 2D implementations. In the work of [10], a performance analysis method based on network calculus has been proposed for 3D NoC. In [11], the performance of several alternative vertical interconnection topologies has been studied. The routing algorithm is based on the dimensional XYZ routing which is only suitable for specific topologies.

Many fault-tolerant routing algorithms have been proposed for 2D mesh NoC. Routing packets through a cycle free contour surrounding a faulty router by a reconfigurable routing algorithm for a fault-tolerant 2D mesh NoC has been investigated in [4]. In [5], a fault adaptive deflection routing algorithm, which makes routing decision based on

a cost function considering the route length and local fault state, has been proposed. A resilient routing algorithm for fault-tolerant NoC based on turn model is described in [6]. However, research on the reliability issues of 3D NoC is still in its infancy. For general interconnection network, a fault-tolerant routing scheme for 3D mesh based on the limited-global safety information is presented in [12]. It is both adaptive and minimal, however, it can only handle faulty cube model which is built around faulty nodes and contains faulty and disabled nodes. A robust and defect-tolerant vertical link architecture for 3D NoC has been proposed in [3] to overcome challenges of low yield for the current TSV fabrication process. This work integrates the defect-tolerant 3D link into a complete three-dimensional NoC design flow.

III. NoC Architecture and Fault Model

A. NoC Architecture

We extend the Nostrum NoC [13], which is a 2D mesh topology, to a 3D mesh topology. Besides four ports connecting four neighboring switches in the same layer, each switch has two additional vertical ports to connect switches above and below the current switch, as shown in Fig. 1. The network is bufferless network (the switch has only one input register for each input port). Deflection routing is used to route packets based on the packet priority which is the number of hops the packet has been routed. For the routing algorithm without fault-tolerance, the switch tries to route packets to the direction with the minimal number of hops to the destination along the x, y or z axis. If there are two or more directions satisfying the requirement, one direction with the smallest traffic load, which is the number of packets handled by neighboring switches in the last 4 cycles, will be selected. The switch can handle at most 6 packets simultaneously. If two or more packets contend for one output port, a packet with a higher priority will be selected to route through this port and other packets will be deflected to a free output port with the smallest traffic load, which can balance the network traffic load.

The packet format of the 3D NoC architecture is shown in Fig. 2. The width of a packet is 128 bits, including a 48-bit head and an 80-bit payload. The packet head contains 5 fields. A valid bit (*V*) is used to mark a packet valid or not. An 18-bit destination address field (*D_x, D_y* and *D_z*), is divided into 3 parts, each of which has 6 bits using complement code, denoting the relative address to the destination along *x, y* and *z* axis. The relative addresses are updated when the packet has passed a switch. A temporary address field (*T_x, T_y* and *T_z*, 6 bits for each) is used to set a relative address to an intermediate destination along *x, y* and *z* axis. When the packet will be routed to the up/down layer and the vertical link of the current switch is broken, the switch will set this field and forward the packet to an intermediate switch with a healthy vertical link to the up/down layer. The *TV* bit denotes the temporary address valid or not. When the *TV* bit is set to be '1', the packet will be routed to the intermediate switch first. The hop count field (*HC*, 10 bits) is used as the packet priority to denote the number of hops the packet has been routed. The switch makes routing decision based on this field to avoid livelock.

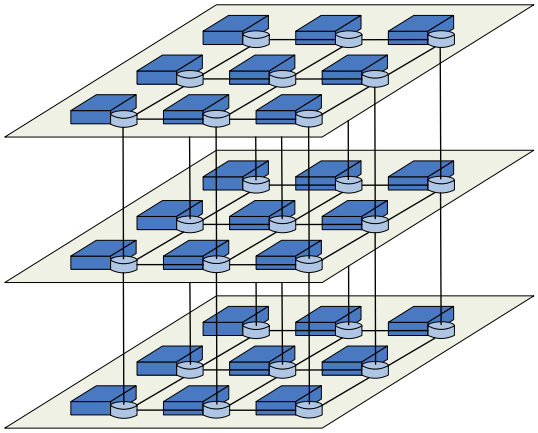

Figure 1. 3D mesh NoC.

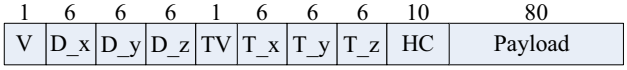

Figure 2. Packet format.

B. Fault Model

In this paper, we consider the faults as completely broken links (permanent faults). For 3D NoC, the faulty link can be divided into two classes: vertical (TSV) and horizontal link. Due to the number of input ports being equal to the number of output ports in deflection switch, the faulty link is assumed to be bidirectional. For the focus on the routing algorithm, we also assumed that there exists a fault diagnosis mechanism to detect faults. In order to simulate the faulty link of the switch, a 6-bit fault vector is used to represent the fault state of six links for each switch (a '1' in the fault vector represents the corresponding bidirectional link is broken). In addition, we assume that the faulty vertical links do not disconnect different layers and the faulty horizon links do not disconnect the 2D mesh for each layer to guarantee a path existing for each node pairs.

In order to get the fault state of TSV, each switch contains two n-bit TSV state vectors which record the fault state of the up and down links of the current layer. The two vectors are transmitted to four neighboring switches which are in the same layer as the current switch. For each clock cycle, the switch will update its own TSV state vectors based on the TSV state vectors transmitted from its neighbors. After a short period, the switch will get all TSV state of the layer.

During run-time, if one or more TSVs have been detected to be faulty, the TSV state vectors can also be updated though transmission.

IV. Routing Algorithm Design and Hardware Implementation

A. Routing Algorithm Design

The algorithm can be divided into two parts: packets routed on the same layer and packets routed across layers. A reinforcement-learning-based deflection routing algorithm is used to route packets on the same layer [7]. Each switch contains an $n \times 4$ routing table which is constructed by the minimum number of hops to all destinations on the 2D mesh layer from four output ports (*North, East, South, West*). Table I shows a layer routing table of a $3 \times 3 \times 3$ 3D mesh. The routing table is reconfigured by equation (1). $Q^x(d, y)$ denotes the minimum number of hops from x to d through neighbor y. When x sends a packet to d through y, y will return 1 plus the minimum number of hops from itself to d ($\min_z Q^y_{t-1}(d, z)$) back to x to reconfigure the corresponding routing table entry of x. Through this reinforcement learning method, after a learning period the routing table will be reconfigured to achieve fault-tolerance.

$$Q^x_t(d, y) = 1 + \min_z Q^y_{t-1}(d, z) \quad (1)$$

Table I
Routing table of switch 5 in a layer of $3 \times 3 \times 3$ 3D mesh

	North	East	South	West
Number of hops to S1	2	4	4	2
Number of hops to S2	1	3	3	3
Number of hops to S3	2	2	4	4
Number of hops to S4	3	3	3	1
Number of hops to S5	0	0	0	0
Number of hops to S6	3	1	3	3
Number of hops to S7	4	4	2	2
Number of hops to S8	3	3	1	3
Number of hops to S9	4	2	2	4

For packets routed across layers, the switch makes routing decision based on the TSV state vectors. When a packet reaches a switch with the same row and column addresses but different layer as the destination switch, if the up/down link of the switch is faulty, it will try to find an intermediate switch with a healthy vertical link at the same layer, which has a minimal manhattan distance to the current switch, based on the TSV state vector. Then the packet will be routed to the intermediate switch according to the routing table of the layer. Fig. 3 shows a routing example in a $3 \times 3 \times 3$ mesh with three broken vertical links. S1 sends a packet to S7. First, it will send the packet to S2 through up link. At S2, the up link is broken, so it will find a nearest switch with a healthy up link (Here is S4) as an intermediate switch. Then

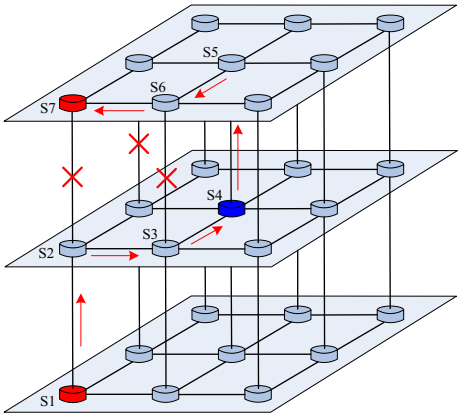

Figure 3. A routing example.

the packet will be routed to S4. After the packet is routed to S4, it can be routed to S7 through S5 and S6.

The pseudo code of the routing algorithm is shown in Fig. 4. When a packet reaches a switch, it will check whether the packet has reached the destination first (Step 1). If the packet reaches the destination, then it will be routed to the local port (Step 2). If the temporary address is valid (*TV* bit is '1') and the packet has reached the intermediate switch, the productive direction is set to be a vertical direction (*up/down*) and the *TV* bit is set to be '0' (Step 4-7). If the packet has not reached the intermediate switch, the current switch will look up the layer routing table based on the temporary address to get the productive direction(s) to the intermediate switch (Step 8 and 9). When the temporary address is not valid (Step 11), the switch gets the productive direction(s) based on the layer routing table and the vertical relative address (Step 12 and 13). At this point, if the vertical direction is the only one productive direction and is faulty, the switch will find an intermediate switch with a good TSV, set the relative address to the intermediate node into the temporary address field of the packet and look up the layer routing table to get the productive direction(s) to the intermediate switch (Step 14-16). The *TV* bit is also set to be '1' (Step 17). After getting the productive direction(s), the switch will choose a free one with the smallest traffic load to route the packet (Step 21 and 22). If all productive ports are not available, the switch will choose a remaining free port with the smallest traffic load to route the packet (Step 23 and 24).

B. Deadlock and Livelock Avoidance

Deflection routing is inherently free from deadlock due to the fact that packets never have to wait in a switch. Because of the non-minimal routing characteristic, deflection routing must avoid livelock by limiting the number of misroutings. The reinforcement-learning-based deflection routing algorithm for 2D mesh is livelock free as long as the

Algorithm

For each input packet (from the highest priority to the lowest priority)
1: ***if*** *dst_addr = 0* ***then*** //reach the destination node
2: Route the packet to local port
3: ***else*** //not reach the destination node
4: ***if*** *TV = 1* ***then*** //temporary address is valid
5: ***if*** *tmp_addr = 0* ***then*** //reach the intermediate node
6: $\{d_{productive\_vertical}\}$ ← get_prefer_vertical_dir($dst\_addr_{vertical}$)
7: *TV* ← 0
8: ***else*** //not reach the intermediate node
9: $\{d_{productive\_row/col}\}$ ← table_lookup(tmp_addr_{row}, tmp_addr_{col})
10: ***end if***
11: ***else*** //temporary address is not valid, use destination address
12: $\{d_{productive\_row/col}\}$ ← table_lookup(dst_addr_{row}, dst_addr_{col})
13: $\{d_{productive\_vertical}\}$ ← get_prefer_vertical_dir($dst\_addr_{vertical}$)
14: ***if*** $\{d_{productive\_row/col}\} = \emptyset$ ***and*** $d_{productive\_vertical}$ is faulty ***then*** //vertical link faulty
15: $\{tmp\_addr_{row}, tmp\_addr_{col}\}$ ← get_intermediate_addr(*row*, *col*, *VerticalLinkStatus*);
16: $\{d_{productive\_row/col}\}$ ← table_lookup(tmp_addr_{row}, tmp_addr_{col})
17: *TV* ← 1
18: ***end if***
19: ***end if***
20: ***end if***
21: ***if*** there are free ports in $\{d_{productive}\}$ ***then***
22: Choose a free productive port with the smallest traffic load to route the packet
23: ***else*** //all productive ports are not free
24: Choose a free port with the smallest traffic load to route the packet
25: ***end if***

Figure 4. Pseudo code of routing algorithm.

faulty links do not disconnect the network [7]. So for this algorithm, packets routed on the same layer are free from livelock. In addition, each switch can get the TSV state of the whole layer and the TSV faults do not disconnect the whole network. For the packets routed across different layers, the switch will indeed find an intermediate node with a healthy TSV to route packets to the up/down destination layer. So the packet will finally advance towards its destination.

C. Hardware Implementation

The proposed switch is developed in VHDL. For an $n \times n \times n$ 3D mesh, each switch contains an $n^2 \times 4$ layer routing table and two n-bit TSV state vectors. Here, we synthesize a switch in the $4 \times 4 \times 4$ 3D mesh with TSMC 65nm standard-cell library. To make a comparison, the reinforcement-learning-based deflection switch with a global routing table (GR) for 3D mesh is also implemented. For this switch, it contains a 64×6 global routing table. The area and power consumption of the two switches with 250MHz are shown in Table II. Compared with the switch with the global routing table, the area and power consumption of the optimized switch (OPT) with the layer routing table and TSV state vectors can reduce up to 40% and 49% respectively.

Table II
HARDWARE IMPLEMENTATION COMPARISON OF TWO SWITCHES

	Area (μm^2)	Power (mW)
GR	193672	17.1
OPT	117392	8.8

V. SIMULATION RESULTS

In this section, we evaluate the performance of the routing algorithm using a cycle-accurate NoC simulator developed in VHDL under both synthetic and application workloads.

A. Experimental Setup

We perform the experiments on a $4 \times 4 \times 4$ 3D mesh. For synthetic workloads, each switch is connected to a packet generator which can generate three synthetic traffic patterns (uniform random traffic, transpose traffic and local traffic). For uniform random traffic, each resource node sends packets randomly to other nodes with an equal probability. In transpose traffic, resource node positioned at (x, y, z) sends packets to destination node $(X-1-x, Y-1-y, Z-1-z)$ for all $x \in \{0, \ldots, X-1\}, y \in \{0, \ldots, Y-1\}, z \in \{0, \ldots, Z-1\}$, where *X*, *Y* and *Z* are the number of nodes for each dimension. For local traffic, the resource node sends packets to the near neighbors with a higher probability than the remote nodes. The probability depends on the source-destination manhattan distance d as follows [14]:

$$P(d) = \frac{1}{A(D)2^d} \tag{2}$$

where *D* is the diameter of the network and $A(D) = \sum_{d=1}^{D}(1/2^d)$ is a normalizing factor guaranteeing that the sum of all probabilities is 1.

To simulate real applications, we use traces from parallel application benchmark suites. The Splash-2 suite [15] traces are obtained from the full-system simulator Simics [16] with GEMS2.1 [17] and Garnet network model [18]. The detailed full-system configurations are listed in Table III. The multicore system contains 64 processors connected by a $4\times4\times4$ 3D mesh network. A 32KB L1 I-Cache and D-Cache and a 512KB L2 Cache are attached on each processor. The Cache coherence protocol is MOESI_CMP_directory protocol. 8 on-chip memory controllers are attached on 4 processors of the top and bottom layers respectively.

Table III
FULL-SYSTEM CONFIGURATION FOR TRACE GENERATION

Number of Processors	64
ISA	SPARC
L1 Cache	32K-I/D, 4-way associative, 64B/line
L2 Cache	fully shared S-NUCA, 512KB/bank 64 banks, 64B/line, 8-way associative
Cache coherence protocol	MOESI_CMP_directory
Memory	8 on-chip memory controllers
Splash applications	barnes, cholesky, fft, fmm lu, radix, raytrace, water

We measure the average packet latency under both synthetic and real application workloads. The packet latency *T* is calculated by equation (3), where T_{net} is the *network delivery time* which is the hop count the packet being routed and T_{src} is the time a packet waiting in the source queue.

$$T = T_{net} + T_{src} \quad (3)$$

B. Results under Synthetic Workloads

In this subsection, we evaluate the performance of the switch with the global routing table (GR) and the proposed switch (OPT) under three synthetic workloads. Fig. 5 (a)-(c) show the average packet latency of the two switches in the presence of no faults and 10% link faults. Two switches perform similar in the case of no faults in the network. Compared with the switch with the global routing table, in the presence of 10% link faults, the proposed switch has only 5%, 6% and 4% performance degradation under three synthetic workloads respectively, while occupies 40% less area and consumes 49% less power consumption.

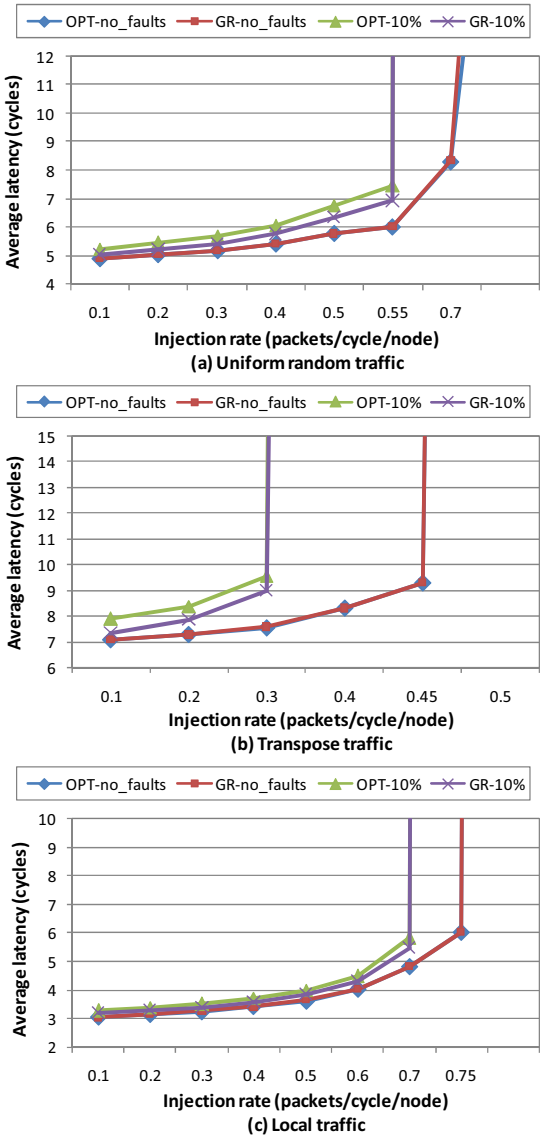

Figure 5. Average latency under synthetic application workloads with 10% link faults.

C. Results under Application Workloads

We measure the performance of the two routing algorithms under eight application traffic traces from Splash-2. Fig. 6 (a)-(c) show the average packet latency of the two routing algorithms under application workloads in the network with 10% horizontal, vertical and mixed faulty links respectively. In the presence of 10% horizontal faulty links, the switch with the global routing table performs slightly better than the proposed switch, while the average latency of the proposed routing algorithm is 6% and 5% less than the routing algorithm with the global routing table under 10% vertical and mixed faulty links respectively.

VI. Conclusion

In this paper, we propose a low-cost fault-tolerant deflection switch for 3D Network-on-Chip. Instead of the global routing table, the switch uses only a layer routing table and two TSV state vectors to make routing decision efficiently to avoid both TSV faults and horizontal faulty links. The proposed switch, which can achieve 250MHz, is synthesized with TSMC 65nm technology. Compared with a fault-tolerant deflection switch with a global routing table, the switch occupies 40% less area and consumes 49% less power consumption. Simulation results illustrate that the proposed switch outperforms the switch with the global routing table under splash-2 application workloads and with only small performance degradation under synthetic workloads.

Acknowledgment

The research is partially supported by the National Natural Science Foundation of China under Grant No. 60970036, No. 60873212 and No. 61003301.

References

[1] V. F. Pavlidis and E. G. Friedman, "3-d topologies for networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1081–1090, 2007.

[2] I. Loi, F. Angiolini, and L. Benini, "Supporting vertical links for 3d networks-on-chip: toward an automated design and analysis flow," in *Proc. 2nd Int. Conf. on Nano-Networks*, 2007, pp. 15:1–15:5.

[3] I. Loi, S. Mitra, T. H. Lee, S. Fujita, and L. Benini, "A low-overhead fault tolerance scheme for tsv-based 3d network on chip links," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 2008, pp. 598–602.

[4] Z. Zhang, A. Greiner, and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip," in *Proc. 45th ACM/IEEE Design Automation Conf.*, 2008, pp. 441–446.

[5] A. Kohler and M. Radetzki, "Fault-tolerant architecture and deflection routing for degradable noc switches," in *Proc. 3rd ACM/IEEE Int. Symp. Networks-on-Chip*, 2009, pp. 22–31.

[6] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant nocs," in *Proc. Design, Automation & Test in Europe Conf. & Exhibition*, 2009, pp. 21–26.

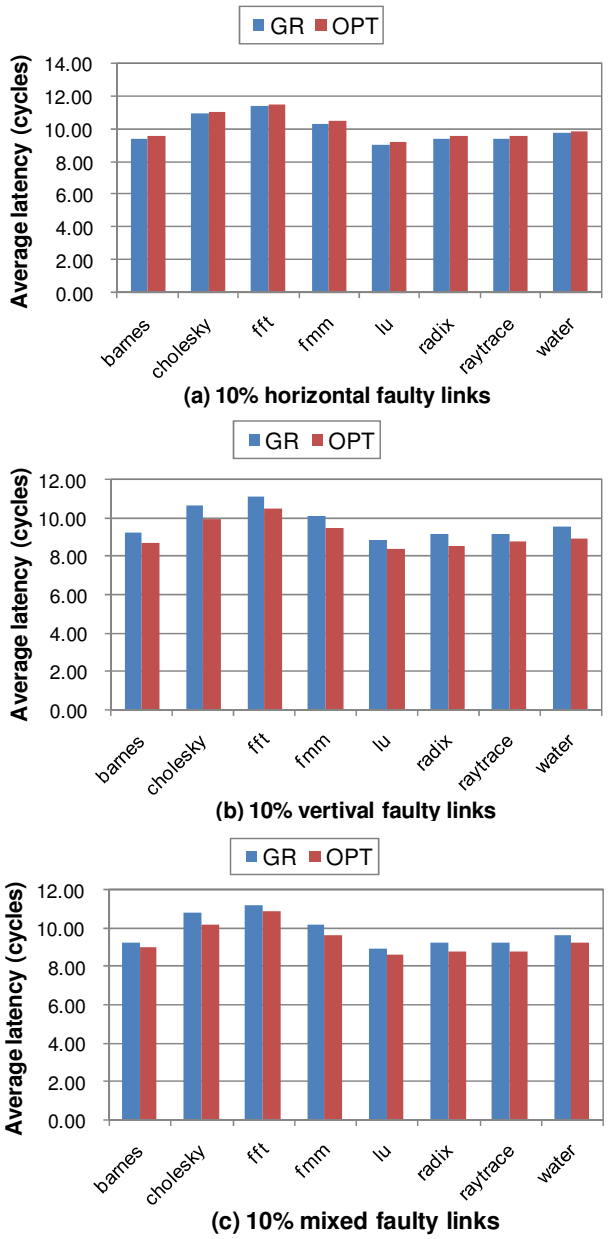

Figure 6. Average latency under 8 real application workloads with 10% link faults.

[7] C. Feng, Z. Lu, A. Jantsch, J. Li, and M. Zhang, "A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip," in *Proc. 3rd Int. Workshop on Network on Chip Architectures*, 2010, pp. 11–16.

[8] C. Addo-Quaye, "Thermal-aware mapping and placement for 3-d noc designs," in *Proc. IEEE Int. SOC Conf.*, 2005, pp. 25–28.

[9] B. S. Feero and P. P. Pande, "Networks-on-chip in a three-dimensional environment: A performance evaluation," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32–45, 2009.

[10] Y. Qian, Z. Lu, and W. Dou, "From 2d to 3d nocs: A case study on worst-case communication performance," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 2009, pp. 555–562.

[11] A. Bartzas, N. Skalis, K. Siozios, and D. Soudris, "Exploration of alternative topologies for application-specific 3d networks-on-chip," in *Proc. Workshop on Application Specific Processors*, October 2007.

[12] J. Wu, "A simple fault-tolerant adaptive and minimal routing approach in 3-d meshes," *Journal of Computer Science and Technology*, vol. 18, no. 1, pp. 1–13, 2003.

[13] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, "The nostrum backbone-a communication protocol stack for networks on chip," in *Proc. 17th Int. VLSI Design Conf.*, 2004, pp. 693–696.

[14] Z. Lu, A. Jantsch, E. Salminen, and C. Grecu, "Network-on-chip benchmarking specification part 2: micro-benchmark specification," in *OCP-IP*, May 2008, pp. 7–8.

[15] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: Characterization and methodological considerations," in *Proc. 22nd Annual Int. Symp. on Computer Architecture*, 1995, pp. 24–36.

[16] P. S. Magnusson, M. Christensson, J. Eskilson, D. Forsgren, G. Hallberg, J. Hogberg, F. Larsson, A. Moestedt, and B. Werner, "Simics: A full system simulation platform," *IEEE Computer*, vol. 35, no. 2, pp. 50–58, 2002.

[17] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood, "Multifacet's general execution-driven multiprocessor simulator (gems) toolset," *SIGARCH Comput. Archit. News*, vol. 33, pp. 92–99, November 2005.

[18] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "Garnet: A detailed on-chip network model inside a full-system simulator," in *Proc. IEEE Int. Symp. Performance Analysis of Systems and Software*, 2009, pp. 33–42.