FON: FAULT-ON-NEIGHBOR AWARE ROUTING ALGORITHM FOR NETWORKS-ON-CHIP

Chaochao Feng[1,2], Zhonghai Lu[1], Axel Jantsch[1], Jinwen Li[2], Minxuan Zhang[2]

[1]Royal Institute of Technology, Stockholm, Sweden
[2]School of Computer, National University of Defense Technology, Changsha, P.R. China
[1]{cfeng, zhonghai, axel}@kth.se [2]{fengchaochao, lijinwen, mxzhang}@nudt.edu.cn

ABSTRACT

Reliability has become a key issue of Networks-on-Chip (NoC) as the CMOS technology scales down to the nanoscale domain. This paper proposes a Fault-on-Neighbor (FoN) aware deflection routing algorithm for NoC which makes routing decision based on the link status of neighbor switches within 2 hops to avoid fault links and switches. Simulation results demonstrate that in the presence of faults, the saturated throughput of the FoN switch is 13% higher on average than a cost-based deflection switch for 8x8 mesh. The average hop counts can be up to 1.7x less than the cost-based switch. The FoN switch is also synthesized using 65nm TSMC technology and it can work at 500MHz with small area overhead.

I. INTRODUCTION

Networks-on-Chip (NoC) design paradigm which utilizes packet-switch communication has been proposed as a promising solution for Multiprocessors System-on-Chip (MPSoC) [1]. However, as the CMOS technology scales down to the nanoscale domain, some physical effects such as crosstalks, electro-magnetic interferences, alpha and neutron particle strikes and power supply disturbances may seriously affect the reliability of NoC. A possible solution to design a reliable NoC is to make the routing algorithm fault-tolerant.

Deflection routing is a non-minimal adaptive routing algorithm which can be implemented easily in hardware, since it does not need buffers for packets in transit [2]. A main motivation for the use of non-minimal routing is fault-tolerance, so it can be easily modified to achieve fault-tolerance at the cost of small hardware overhead. This paper proposes a Fault-on-Neighbor (FoN) aware deflection routing algorithm which makes efficient routing decision based on the fault information transmission within 2 hops to avoid fault links and switches. Simulation results demonstrate that the saturated throughput of the FoN switch is 13% higher on average than a cost-based deflection switch [3] for 8x8 mesh with faults. The average hop counts can be at most 1.7x less than the cost-based switch in the presence of link failures. We also synthesize the FoN switch using TSMC 65nm standard-cell library, which can achieve 500MHz with area less than half of the cost-based switch and only 18% increase compared to the non fault-tolerant deflection switch.

The rest of paper is organized as follows: Related work is reviewed in Section II. Section III describes the NoC architecture, fault model and introduces the 2-hop fault information transmission mechanism. The detailed FoN aware routing algorithm and implementation are proposed in section IV. In section V, simulation experiment results are analyzed, followed by the conclusion and future work in section VI.

II. RELATED WORK

Fault-tolerant routing can be divided into two categories: stochastic and deterministic. Stochastic communication transfers redundant packets through different paths to avoid faults. A probabilistic flooding algorithm derived from the randomized gossip protocol has been proposed in [4]. In this switch, a packet is sent to a randomly chosen subset of its neighbors such that the packet is broadcasted from one switch to the entire NoC. In order to limit the number of packet copies, redundant random walk [5] only replicates packets at the source node. The flood-based scheme can be highly fault-tolerant such that if a path exists between source and destination, a packet can be always routed to the destination. But it wastes much bandwidth to transfer redundant packets and limits the performance of the network.

Deterministic algorithm utilizes the structure redundancy of NoC to route packets to the destination through different paths to achieve fault-tolerance. Force-Directed Wormhole Routing (FDWR) [6] makes routing decision based on the routing table and the buffer status of neighbor switches. It uses the first flit of a packet as a look-ahead flit to investigate the buffer and fault status of neighbor switches. However, it needs an additional flit to check neighbor state and with the increase of the network size, more buffers are needed to store the routing table. In [7], a reconfigurable routing

algorithm for fault-tolerant 2D mesh NoC has been presented to route packets through a cycle free contour surrounding a faulty switch. But it can only be used in one faulty switch topology. A resilient routing algorithm for fault-tolerant NoC based on turn model is described in [8]. The switch can be reconfigured around faulty components while maintaining correct operation without using virtual channels. A fault adaptive deflection routing algorithm which makes routing decision based on a cost function has been proposed in [3]. The switch implements an on-line fault diagnosis mechanism and makes routing decision based on a cost function which considers the route length and local fault status. It can not only handle link and switch faults, but also crossbar faults. Because it makes routing decision only based on the fault information of the current switch, the hop count field of the packet can easily overflow in some fault patterns.

III. NOC ARCHITECTURE AND FAULT MODEL

A. NoC Architecture Overview

The NoC architecture is based on Nostrum NoC [9,10], which is a 2D mesh topology. The difference from the ordinary 2D mesh is that the boundary output is connected to the input of the same switch so that the packet sent in that direction returns to the same switch. This can be used as a packet buffer. Deflection routing is used to make routing decision based on the packet priority and stress value which is the traffic load of neighbor switches in last 4 cycles. All incoming packets are prioritized based on its hop counts which record the number of hops the packet has been routed. The packet with the largest hop counts has the highest priority. The switch makes routing decision for each packet from the highest priority to the lowest. If a desired output port has already been occupied by a higher priority packet, a free port with the smallest stress value will be chosen, which means the packet has to be deflected.

B. Fault Model

In our work, faults are considered as completely broken links and switches. Link failures are assumed to be bidirectional. In each switch, a 4-bits fault vector is used to represent the fault status of its four links. A switch fault is modeled by making all four links attached on it faulty. Packets are only destined to fault-free switches. It is difficult to handle arbitrary fault patterns without the shape constraint. Without loss of generality, the fault regions handled by the FoN aware routing algorithm can be constrained to be convex and concave shapes [11], as shown in Fig. 1(a) (I-shape, □-shape, U-chain, +-shape, L-shape and T-shape), which do not contain two sequential concave points such as U-block and H-shape, shown in Fig. 1(b). A concave point is a corner on the boundary of the faulty region which has an interior angle of 270 degrees, as the gray nodes shown in Fig. 1(b). In addition, it is assumed that fault regions do not disconnect the network.

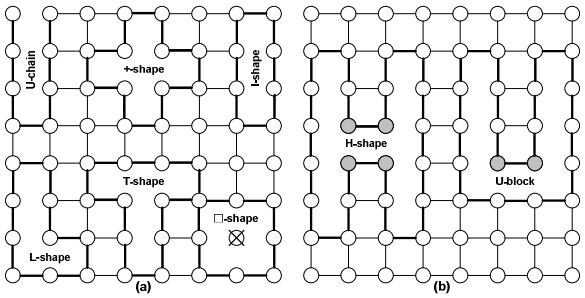

Figure 1: Fault patterns

C. Fault Information Transmission Mechanism

The key issue of designing a fault-tolerant routing algorithm is that each switch gets fault information such that it can make routing decision efficiently. However, it will cost a large amount of resources to store and transmit fault information and also take much time to search for an available path. Due to the limited on-chip resources, it is important to decide to which extent of the fault information should be available at a switch. In this paper, to make a compromise between purely local and purely global fault information we propose a 2-hop fault information transmission mechanism which means each switch can get the link status within 2 hops.

In the 2-hop fault information transmission mechanism, each switch not only transmits its link status to its four neighbors but also collects the link status from its three neighbors and transmits to the fourth neighbor, as shown in Fig. 2, for example, switch A can get the status of 16 links. The signal *FoN_to[d]* (Fault-on-Neighbor to direction *d*) collected by the current switch is a 3-bits vector to denote link status along the other three directions except *d* and is transmitted to the neighbor along *d*. For example, *FoN_to[W]* denotes the link status of the North, East and South links and is transmitted to the West neighbor. Due to the limited fault information, packets can be routed in a circle on H-shape and U-block fault regions. So this mechanism can also be easily extended to n-hop fault information transmission to handle these more complicated fault patterns.

Only four additional signals are required to transmit fault information between a switch and its four neighbors. They are totally 8 bits for each

direction. For each direction $d \in$ {North, East, South, West}, these signals are represented as:

- *fault_from[d]* (1 bit): the input link status along the direction *d*;
- *fault_to[d]* (1 bit): the output link status along the direction *d*;
- *FoN_from[d]* (3 bits): the link status from the neighbor along the direction *d*;
- *FoN_to[d]* (3 bits): the link status collected by the current switch based on link status of three directions except the direction *d* and transmitted to the neighbor along *d*.

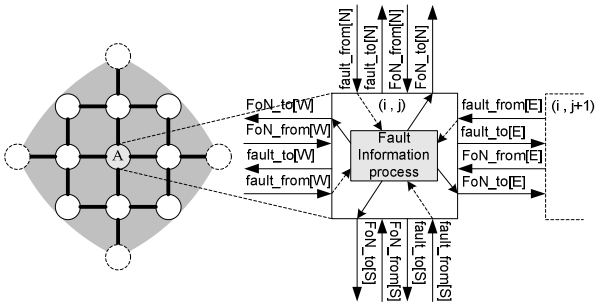

Figure 2: Fault information transmission mechanism

IV. FON AWARE ROUTING ALGORITHM AND IMPLEMENTATION

A. FoN Aware Routing Algorithm

The basic idea of the FoN aware routing algorithm is that each switch makes routing decision based on the link status within 2 hops. Fig. 3 shows an example of routing decision on a 4x4 mesh. Suppose switch (2,2) sends a packet to switch (4,4). First it will check two productive links (East and South) faulty or not. If both are not faulty, then it will check status of the productive links to switch (4,4) from its neighbors (switch (2,3) and (3,2)). In this case, both of productive links to switch (4,4) from switch (2,3) are faulty, so switch (2,2) will choose the south link finally.

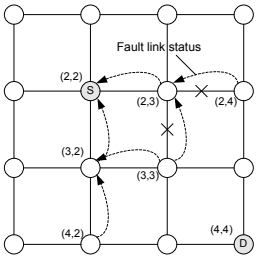

Figure 3: Example of fault information transmission

The pseudo code of the FoN aware deflection routing algorithm executed at node n_c is shown in Fig. 4. There are at most four packets arriving at a switch at the same time. The switch makes routing decision from the highest priority packet to the lowest. The switch first checks if the packet has reached the destination n_d. If not, then it calculates the productive direction(s) based on the relative address to n_d, which is provided by the function *get_prefer_Dir(n_c, n_d)*. A packet may have at most two productive directions to the destination. The remaining algorithm can be divided into two parts: 1) n_d is in the same row or column as n_c; 2) n_d is not in the same row or column as n_c.

1) n_c and n_d are in the same row or column

In this case, there is only one productive direction to n_d. When the productive link is faulty and the input direction is not one of orthogonal links, the switch will check the *FoN_from* vectors along the two orthogonal directions by the function *Check_FoN,* which checks the productive link of the neighbor along the orthogonal direction (step 10). If both of orthogonal links are available, it will choose one of them with a smaller stress value. If the input direction is one of orthogonal links, it will check the *FoN_from* vector along the other orthogonal link (step 13). If only one orthogonal link is not fault, then it will check the *FoN_from* vector along this direction (step 20). If both orthogonal links are not available, it will choose the remaining link (step 15, 22, 26). If all other links are not available, the packet will be routed back (step 18, 25, 29).

2) n_c and n_d are not in the same row or column

In this case, the packet can be routed to n_d through two productive directions. If both of them are not faulty and the input direction is not one of them, for each productive direction the switch first gets the productive direction(s) from the neighbor to n_d (step 35, 36) and then checks *FoN_from* vectors along this path (step 39) and chooses one of them base on the *FoN_from* vectors and stress values. If the input direction is one of productive directions, it will check *FoN_from* vector along the other productive direction (step 42). If only one productive link is not faulty, it will check the *FoN_from* vector along this direction (step 47). If both of productive links are faulty, it will choose a remaining link with a smaller stress value (step 54).

In order to avoid the packet oscillation between two switches, the algorithm tries to avoid routing the packet back to the input direction. Only if all the other links are not available, the packet will be routed back.

B. Deadlock and Livelock Avoidance

Deflection routing is deadlock free since packets never have to wait in a switch. Livelock is a key problem that deflection routing must avoid. Packet priority mechanism can be used in a fault-free network to avoid livelock. However, because each switch only has the limited fault information of the

network, it is difficult to avoid livelock under arbitrary fault patterns. The simplest way to avoid livelock is by limiting misrouting. Constraining the shape of the fault regions to be convex and concave without sequential concave points can be helpful to avoid unlimited misrouting which is shown in the following livelock-free proof.

```
Algorithm FoN
For each input packet (from the highest priority to the lowest priority)
1:  if packet reaches destination then
2:    Route packet to local port
3:  else
4:    d_productive ← get_prefer_Dir(n_c, n_d)
5:   if n_c and n_d are in the same row or column then
6:    if d_productive is faulty then
7:     if both of orthogonal links are not faulty then
8:      if d_in ∉ d_orthogonal then
9:       for each d_r ∈ d_orthogonal then
10:        Check_FoN(d_r ∈ d_orthogonal) // Check FoN_from(d_i)(d_productive)
11:      Choose one orthogonal link based on FoN_from vectors and stress value
12:     else if d_in is one of orthogonal links then
13:      if Check_FoN(d_orthogonal_i ≠ d_in) OK then
14:       Choose d_orthogonal_i
15:     else if the remaining link d_r is not fault then
16:       Choose d_r
17:     else
18:       Route back from d_in
19:   else if only one orthogonal link d_i is not faulty then
20:    if d_in ≠ d_i and Check_FoN(d_i) OK then
21:      Choose d_i
22:    else if the remaining link d_r is not fault then
23:      Choose d_r
24:    else
25:      Route back from d_in
26:   else if the remaining link d_r is not fault then
27:     Choose d_r
28:   else
29:     Route back from d_in
30:  else
31:    Choose d_productive
32: else
33:  if both of d_productive are not faulty then
34:   for each d_i ∈ d_productive
35:     n_i ← get_neighbor(d_i)
36:     d_ni_productive ← get_prefer_Dir(n_i, n_d)
37:   if d_in ∉ d_productive then
38:    for each d_i ∈ d_ni_productive
39:      Check_FoN(d_i)  //based on FoN_from(d_i) (d_i)
40:    Choose one productive link based on FoN_from vectors and stress value
41:   else if d_in is one of productive links then
42:    if Check_FoN(d_productive_i ≠ d_in) OK then
43:      Choose d_productive_i
44:    else
45:      Choose one of remaining links based on fault status and stress value
46:  else if only one d_productive_i is not faulty then
47:   if d_in ≠ d_productive_i and Check_FoN(d_productive_i) OK then
48:     Choose d_productive_i
49:   else if the remaining link d_r is not fault then
50:     Choose d_r
51:   else
52:     Route back from d_in
53:  else //both of productive links are faulty
54:    Choose one link of the remaining links based on fault status and stress value
```

Figure 4: FoN aware routing algorithm

Theorem: FoN aware routing algorithm is livelock-free under convex and concave fault regions with at most one concave point in sequence which do not disconnect the network.

Proof: Packets routed on non-faulty region are livelock-free, because the algorithm gives the highest priority to the oldest packets, which guarantees a packet can eventually advance towards its destination. When a packet reaches the corner of the fault region which is not a dead end, it can be routed the same way as routed on the non-faulty region. Depending on how to route packets to the corner of the fault region, packets routed around the fault region can be categorized into three possible cases, which are described as follow.

1) Packet routed on the edge of the fault region without contention

When a packet reaches one switch on the edge of the fault region and the switch chooses a direction *d* to route the packet, in order to avoid the packet oscillation between two switches, the following switches on the edge will try to route the packet along *d* even when the packet is far away from the destination until the packet reaches the corner of the fault region or reaches a dead end of the fault region which can be met on U-chain. In the later case, the packet will be routed back along the original path until it reaches another corner of the fault region, as shown Path 1 in Fig. 5.

2) Packet routed to the concave point of the fault region without contention

When a packet reaches a concave point of the concave fault regions (+-shape, L-shape and T-shape), it will be routed along the orthogonal direction of the input direction and follow the same way on another edge as case 1) to the corner, as shown Path 2 in Fig. 5.

3) Packet deflected away from the fault region due to the contention

In the case of a packet first routed on the edge of the fault region and then deflected away from the fault region temporarily due to the contention, when the priority of the packet becomes higher, it will be rerouted on the edge of the fault region again and there is no contention to hinder the packet routed to the corner of the fault region as described in case 1) and 2), as shown Path 3 in Fig. 5. □

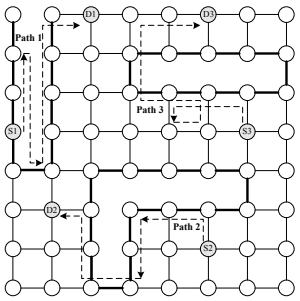

Figure 5: Packets routed around fault region

C. FoN Switch Implementation

The structure of the FoN switch is shown in Fig. 6. Because deflection routing does not need buffers to store packet, the switch can be easily implemented with hardware. Each switch sends its link status and stress value to its neighbor switches and receives link status and stress values from its

neighbors. Routing controller makes routing decision based on the fault information and stress value.

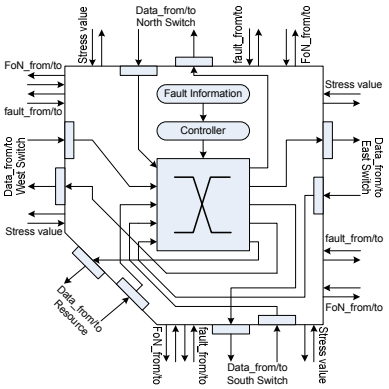

Figure 6: FoN switch implementation

The FoN switch is synthesized using 65nm TSMC standard-cell library by Synopsys Design Compiler. In order to make a comparison, the cost-based deflection switch [3] is also implemented. For the focus on the routing algorithm, we only implement the cost-based deflection routing algorithm and do not implement the on-line fault diagnosis and CRC check of the switch. Results for frequency, area and power consumption are shown in Table 1. For frequency, the FoN switch can achieve the same frequency (500MHz) as the non fault-tolerant switch, while the cost-based switch can only achieve 166MHz. The areas of the non fault-tolerant and FoN switch are less than half of the cost-based switch. Compared to the non fault-tolerant switch, the area of the FoN switch is increased by 18%. All power consumptions are measured at the same frequency and voltage (166MHz, 1.0V). The power consumption of the FoN switch is 4.7x less than the cost-based switch. The cost-based switch has high hardware overhead because it has to find the best permutation among all 4!=24 permutations of input and output ports based on a cost function.

Table 1: Synthesize results

	Frequency (MHz)	Area (μm^2)	Power (mW)
Non fault-tolerant	500	28784	1.2
Cost-based	166	76184	6.6
FoN	500	34109	1.4

V. EXPERIMENT RESULTS

The experiment evaluates the FoN switch and the cost-based switch in terms of throughput and hop count by simulation. The simulation is constructed on an 8x8 2D mesh.

A. Simulation Platform

The FoN switch is developed in VHDL and performed a cycle-accurate RTL-level simulation. Six common synthetic traffic patterns (uniform random, transpose, bit complement, bit reverse, shuffle and tornado) [12] are used in the simulation. A packet generator is attached to each switch and there is a FIFO in it to buffer the packets which can not be injected into the network because of no free output port. The packet generator delivers packets into the network with an injection rate which can be set at the beginning of the simulation. The specified number of faults is randomly selected before the simulation.

B. Results and Analysis

The experiments first evaluate the throughput of the network with different number of link failures. Throughput is measured in packets/cycle/node at the saturation point of the network which means the maximum accepted traffic. Fig. 7 shows the throughput degradation with an increasing number of link faults under six traffic patterns for the FoN and cost-based switch. The number of link faults varies from 10% to 30% of the total. In the situation of no faults, the throughput of the cost-based switch is slightly higher than the FoN switch for all traffic patterns. This is because the cost-based switch can always find the best permutation for each packet in a congested network with no faults instead of unnecessary deflections. As the number of failures increase, the number of available routing paths also decreases which results in a throughput degradation for both switches. In the presence of faults, the cost-based switch can not always make optimized routing decision, so the network can be easily saturated. The throughput of the FoN switch is 13% higher on average than the cost-based switch in the presence of link faults. For individual traffic patterns, the throughput of the FoN switch is higher than the cost-based switch on average 17% for uniform random traffic, 14% for transpose traffic, 14% for bit complement traffic, 8% for bit reverse traffic, 12% for shuffle traffic and 14% for tornado traffic.

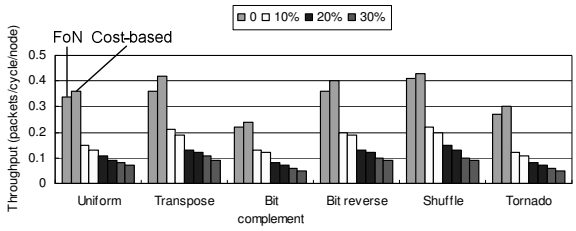

Figure 7: Throughput with increasing number of link failures

Fig. 8 (a) ~ (f) reveal a comparison of the two switches in terms of average hop counts under various link failure rates from 10% to 30% for the six traffic patterns respectively. The packet injection rate is 0.1 packets/cycle/node. The FoN switch can

achieve 1.5x, 1.4x, 1.2x, 1.5x, 1.7x and 1.3x less hop counts on average than the cost-based switch for the six traffic patterns respectively. The reason is that the cost-based switch makes routing decision only based on local fault status which leads some packets to be routed in a circle among several switches or oscillation between two switches. However, the FoN switch always tries to route packet advanced to the destination, so the average hop counts are significantly less.

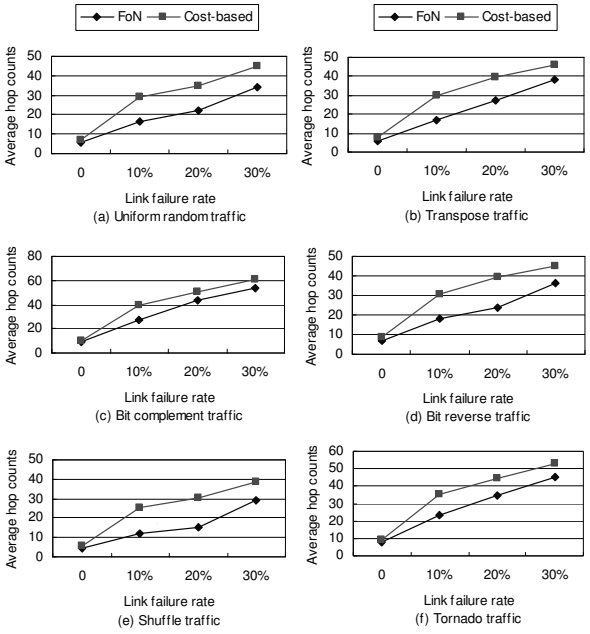

Figure 8: Average hop counts under various link failure rates

The worst case hop counts for the two switches are also compared in the experiments. Fig. 9 illustrates the packet number distribution based on hop counts under uniform random traffic with a fault pattern of 10% link faults. The max hop counts of the FoN switch are between 40 and 59, while the max hop counts of the cost-based switch are larger than 120. From the simulation it can be found that 10 bits of the hop count field are even not enough for the cost-based switch, in some fault patterns the hop counts can easily overflow.

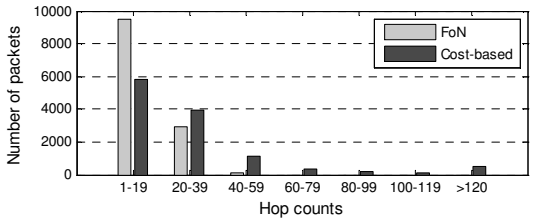

Figure 9: Packet distribution based on hop counts

VI. CONCLUSION AND FUTURE WORK

This paper proposes a fault-tolerant deflection routing algorithm for Networks-on-Chip. The routing algorithm utilizes a 2-hop fault information transmission mechanism to make routing decision efficiently and can tolerate common convex and concave fault regions without deadlock and livelock. Simulation and synthesize results show that our switch outperforms a cost-based deflection switch in terms of throughput, average hop count, area and power consumption in the presence of faults.

In future work, we will focus on on-line fault detection mechanism and explore intelligent routing strategy to handle more complicated fault patterns.

ACKNOWLEDGEMENT

The research is partially supported by the National Natural Science Foundation of China (No. 60873212).

REFERENCES

1. S. Kumar, A. Jantsch, J.P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja and A. Hemani, “A network on chip architecture and design methodology”, *IEEE Computer Society Annual Symposium on VLSI*, pp. 117-122, April 2002.
2. Z. Lu, M. Zhong, and A. Jantsch, “Evaluation of on-chip networks using deflection routing”, *ACM Great Lakes symposium on VLSI*, pp. 296-301, May 2006.
3. A. Kohler and M. Radetzki, “Fault-tolerant architecture and deflection routing for degradable NoC switches”, *IEEE International Symposium on Networks-on-Chip*, pp. 22-31, May 2009.
4. T. Dumitras, S. Kerner, and R. Marculescu, “Towards on-chip fault-tolerant communication”, Asia and South Pacific Design Automation Conference, pp. 225-232, January 2003.
5. M. Pirretti, G.M. Link, R.R. Brooks, N. Vijaykrishnan, M. Kandemir and M.J. Irwin, “Fault tolerant algorithms for Network-on-Chip interconnect”, *IEEE Computer Society Annual Symposium on VLSI*, pp. 46-51,February 2004.
6. T. Schonwald, J. Zimmermann, O. Bringmann and W. Rosenstiel, “Fully adaptive fault-tolerant routing algorithm for Network-on-Chip architectures”, *Euromicro Conference on Digital System Design Architectures, Methods and Tools*, pp. 527-534, August 2007.
7. Z. Zhang, A. Greiner and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2D-mesh Network-on-Chip", *ACM/IEEE Design Automation Conference*, pp. 441-446, June 2008.
8. D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant NoCs", *Design, Automation & Test in Europe Conference & Exhibition*, pp.21-26, April 2009.
9. E. Nilsson, M. Millberg, J. Oberg and A. Jantsch, “Load distribution with the proximity congestion awareness in a network on chip”, *Design, Automation & Test in Europe Conference & Exhibition*, pp.1126-1127, March 2003.
10. Z. Lu and A. Jantsch, “TDM Virtual-Circuit Configuration for Network-on-Chip”, *IEEE Transaction on VLSI Systems*, Vol. 16, No. 8, pp. 1021-1034, August 2008.
11. Y.J. Suh, B.V. Dao, J. Duato and S. Yalamanchili, “Software-based rerouting for fault-tolerant pipelined communication”, *IEEE Transaction on Parallel and Distributed Systems*, Vol. 11, No. 3, pp. 193-211, March 2000.
12. W.J. Dally and B. Towles, “Principles and practices of interconnection network”, *Morgan Kaufmann publishers*, pp. 50-51, 2004.