Implementation of a Novel Fault Tolerant Routing Technique for Mesh Network on Chip

Akshay B P[1], Ganesh K M[1], Thippeswamy D R[1], Vishnu S Bhat[1], Anitha Vijayakumar[1], Ananda Y R[2], and John Jose[2]

[1] Dept. of ECE, Dayananda Sagar College of Engineering, Bengaluru, India
[2] MARS Research Lab, Dept. of CSE, Indian Institute of Techonology Guwahati, India
{akshaybp456,ganesh.km.n7,dr.thippu,vishnu.s.bhat}@gmail.com
{anithavijaya,yrananda}@gmail.com, johnjose@iitg.ac.in

Abstract. The continuous advancements in the Network on Chip technology emphasizes the need for fault tolerant designs. In this work, we propose a routing technique that handles multiple link faults. We use flit parameters to handle the fault in the routing path. Experimental analysis show that the proposed routing technique is capable of routing packets even with two fault locations and the packets are received in the destination router without any error. In addition, hardware implementation done using ZedBoard Zynq FPGA hardware kit shows that our design is having minor area overhead compared to the standard XY routing and it's a significantly better choice than the other fault tolerant algorithms.

Keywords: Router micro-architecture · double faults.

1 Introduction

As technology increases, design and scalability issues associated with multi-core processors become evident. In addition, increasing number of transistors on a single chip has invaded the concept of system on chip [12] that scales up deep sub-micron effects like cross-talk and interference. Network on chip (NoC) design paradigm has been proposed to replace traditional bus based interconnect. Here data communication takes place through data packets with the help of routing algorithm and packet switching technology. Figure 1 shows the basic two-dimensional mesh NoC. It is a 16-core tiled chip-multiprocessor (CMP) architecture. Each router has 5x5 crossbars with 4 virtual channels per port.

Each processing element (PE) consists of an L1 cache block, a shared L2 cache block and also connected to a router. Usually, a source core generates packets and divides them into smaller units called as flits. Each packet is segmented into head flit, single or multiple body flits and a tail flit. Head flit carries routing information of the packet, body flit has the required data and tail flit performs bookkeeping to close the connection between the nodes. These flits are injected into local router when an L1 miss occurs. Wormhole virtual channel switching occurs between the source and the destination routers for all flits of a packet. Packets follow an embedded routing algorithm. The rapid growth in the NoC

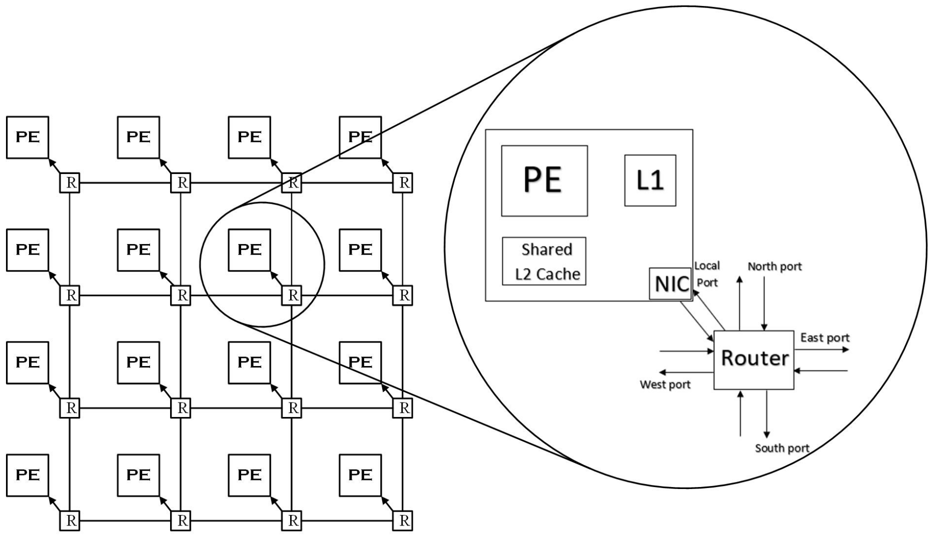

Fig. 1. Basic Two-Dimensional Mesh NoC

framework for multi-core systems demands fault-tolerant architecture. Generally, the quality of chip is directly influenced by its fault tolerance limit. Faults are categorized into two types in NoC, namely transient faults and permanent faults. The former occurs due to bit error in the transmitted flit and the latter occurs due to electron-migration, temperature instability and oxide break down [13]. The aforementioned faults are a threat to the reliability of Network on Chip. To address these issues researchers are focusing on building a fault tolerant NoC. As the Standard XY routing is known for its better traffic management and network latency, we build our fault tolerant algorithm on top of traditional XY routing. In this paper, we propose a fault-tolerant adaptive XY routing algorithm that can handle one or more faults in the routing path with minimal traffic management and area overhead. To evaluate the effectiveness of our algorithm, we model our proposed algorithm on BookSim2.0 simulator. In addition, we validated our algorithm on ZedBoard Zynq-7000 FPGA hardware kit [15] using Xilinx Vivado HLS 2016.2. The rest of the paper is organized as follows. Section II discusses about the related work. Section III presents about the motivation of our work. Section IV addresses our proposed mechanism, Section V describes the experimental setup, Section VI depicts our experimental results and findings, and finally Section VII concludes the paper with future work.

2 Related Work

In an approach given by Jin-Xiang et al. [1], the authors consider two types of faults, one is node fault and another one is link fault. The algorithm aims at bypassing this node fault and the link fault with the help of new paths that passes through the neighbouring nodes. This new path strategy across all the neighbouring nodes results in increased hops to destination and high traffic conditions across corners.

An approach given by Jianfei et al. [2], uses the XY algorithm as their default routing function to make a fault tolerant routing algorithm. When the fault is detected, if the faulty router is not the edge router, then it bypasses the fault by north last routing algorithm. If it is an edge router, then routing list information (table) is stored in the edge router. This approach requires a larger hardware and latency overhead.

Andrew DeOrio et al. [3], explains fault tolerant architecture and a routing algorithm called VICIS. This algorithm can address permanent faults with its features like BIST (Built In Self Test) units, fault diagnosis and reconfiguration of router architecture. Fault diagnosis is carried out with the help of error correcting codes (ECC). This approach adds many hardware components like port swapper, ECC units and FIFO buffers which increases the area overhead. This approach may also become susceptible to deadlock in some pathological cases.

It includes a detection mechanism with the help of Error Correcting Codes, followed by a diagnosis phase which determines the fault location. In short VICIS includes a reconfigurable router architecture and a routing algorithm. VICIS explores in finding solution through changing the architecture and also changing the routing algorithm that may incur high performance overhead. A Built-In Self-Test (BIST) at each router is also employed for diagnosis. Architecture features including error correction a crossbar bypass bus and port swapping result in overall increase in the latency.

Timo Schonwald et al. [4], explains a fully adaptive and fault tolerant routing algorithm for NoCs called force directed wormhole routing (FDWR). FDWR mainly concentrates on distributing the traffic across the entire network using wormhole routing principle. In order to adapt to different network topologies, FDWR uses a routing table concept which updates the number of hops to the destination. This in turn increases the memory overhead.

An approach given by Yusuke Fukushima et al. [5], consists of fault tolerant implementation for both regular and irregular network due to permanent link failures. The technique is deadlock free and guarantees a path between every pair of nodes. The segment routing is used for irregular topology. In this approach, three sets of bits are used for each of the switch. First set of bits are used to represent routing option and are called routing bits, second set of bits are used to capture the connection pattern and are called connectivity bits and the last set of bits are used to capture the faulty routing options and are called faulty bits. Routing logic has two phases and are termed as comparator phase and computation phase. In the comparator phase, comparing of the x and y coordinates takes place while the appropriate routing is computed in the computation stage. The algorithm is deadlock free, but it restricts switch design and also loads the communication traffic by extra messages. This approach guarantees the connectivity without additional hardware overhead. It provides high performance for a regular network, but degrades performance for an irregular network.

A unique approach by Navonil Chatterjee et al. [13], presents a fault tolerant reconfigurable NoC architecture using router redundancy. In this paper, a double router is implanted in all the nodes of a mesh, instead of a single router, one active router and a spare router. Both routers use the same links with the help of a multiplexer. In case of failure of a router the routing algorithm remains the same since the faulty router is replaced by spare router and the topology is not affected. This type of approach increases area overhead. In this architecture a controller is designed, it has a look up table for storing individual router information. Therefore, this approach requires additional memory with increased complexity.

An approach by Zhen Zhang et al. [14], have considered 4 main neighbours and 4 indirect-neighbours. As in an 8x8 2-D mesh a router can fit into 9 different contours i.e., four borders, four corners and one in the center. The NE router of center contour is not said to be deadlock-free as there will be two complete cycles formed due to the turn-based algorithm. In-order to overcome that aforementioned problem, the authors have come up with a solution that is to remove two turns at the NE router of the contour, thus reducing these cycle count from two to one. The authors have also proposed another approach by completely replacing the turn-based algorithm with some unique paths for every contour at the expense of increased area overhead.

3 Motivation

Due to increase in the number of cores on a chip, the hardware complications arising due to link failures and process variation is significant. As the links between each router in the network is very small in size, there might be a case in which these links get down permanently. Then this may lead to discarding the entire chip, which is not an economical solution. This signifies the need for fault tolerant algorithms.

In earlier discussed approaches [1][2][3][5][14], the fault tolerant routing algorithms involve a complex hardware design with various limitations and requires an additional memory space for routing table information[13]. This motivates us to design a fault tolerant algorithm which should be minimal in implementation as well as efficient enough to by-pass upto two faults in the entire network without any hardware complexities.

4 Proposed Design

In our proposed fault-tolerant algorithm (Algorithm 1), we are considering permanent faults. For the explanation purpose let's consider a single router. Whenever a flit arrives at the router, the *valid bit*, *fault bits* and *destination ID* is extracted. If the *valid bit* is set, *fault bits* are checked first before applying the routing logic. If the *fault bits* are 00 it is considered as normal flit which is following its natural path, and the flit is routed according to the standard XY routing. When the *fault bits* are 01, the flit is deflected before coming to the present router, due to an X-axis fault. Then that flit has to be re-routed by resetting the source as the current router; i.e, that is the path starts freshly from the current router.If the *fault bits* are 10, then the flit is deflected to the current router because of a Y-axis fault, then that type of flits are re-routed by pushing such flits through east port. This is done in all type of routers in a network except last column router. In last column router these Y-axis deflected flits are re-routed by pushing them through west-port. If continuous parallel faults are observed, the flits are moved to columns aside according to the algorithm and further decision is taken depending on the destination router location. We had analysed various cases for our algorithm as given below.

Algorithm 1 Proposed Fault-Tolerant Algorithm

```
if $validbit == 1$ then
  if $faultbits == 00$ then
    Calculate the output port using XY routing
  else if $faultbits == 01$ then
    Based on destination address apply XY routing
  else if $faultbits == 10$ then
    Based on destination row, calculate north or south port
  end if
  if Calculated port is not faulty then
    Calculated port is assigned
    Reset $faultbits$ to 00
  else
    if East or West port is faulty then
      Based on destination row, calculate north or south port
      Set $faultbits$ to 01
    end if
    if North or South port is faulty then
      East port is assigned
      Set $faultbits$ to 10
    end if
  end if
else
  Invalid input
end if
```

Case 1 (Single X-axis fault): Consider Figure 2, where for a packet source is 9 and destination is 45, and a fault in the east port of router 11 (marked as 'x'). The algorithm first checks the *destination ID*, if the destination row is above or same as that of the current row, then the algorithm deflects the flit to north. If the destination row is below the current row, then the algorithm deflects the flit to south. In both the cases, before deflecting the flit we set the *fault bits* to 01. When a flit arrives at a particular router with the *fault bits* set as 01, this means that the flit was deflected due to an X-axis fault and has reached the

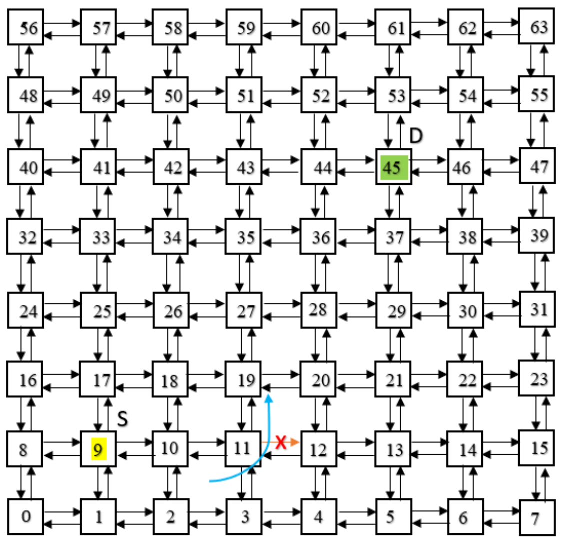

Fig. 2. 8x8 Mesh NoC Architecture With One X-axis Fault

current router from a north or south port. The current router resets the *fault bits* to 00 and routes the packets as normal XY routing.

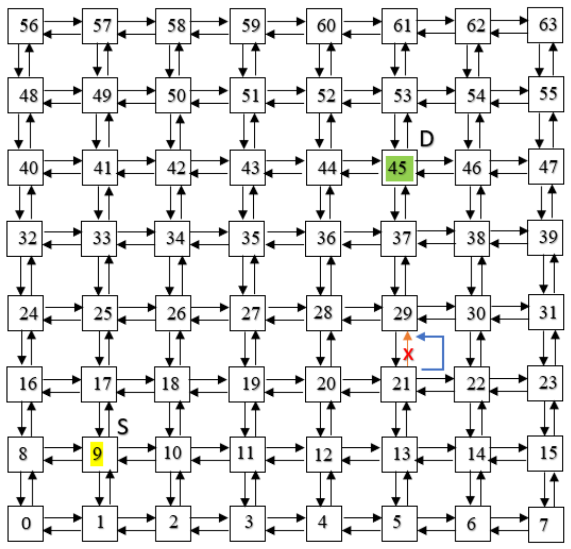

Fig. 3. 8x8 Mesh NoC Architecture With One Y-axis Fault

Case 2 (Single Y-axis fault): Consider Figure 3, where for a packet source is 9 and destination is 45, and a fault in the north port of router 21 (marked as 'x'). Whenever this type of fault occurs, it means that the flit has arrived its destination column. According to standard XY routing, a flit has to take Y turn only when the destination column is reached. We deflect such flit to east. Before deflecting the flit we set the *fault bits* to 10. When a flit arrives a particular router with the *fault bits* set as 10, it means that the flit was deflected due to a Y-axis fault and has arrived the current router from a west port. The current router resets the *fault bits* to 00 and routes the packets to north if the destination row is above the current row or to south if the destination row is below the current row. In the case of the right most column, such flits are pushed to the network through west port as there will be no more east port available. Then the immediate next router will guide that flit to reach it's destination. In our example, the flit is pushed to east port from router 21 and sets the *fault bits* 10, it will be taken back to the destination column through a new path i.e., 21-22-30-29. Once it reaches the router 29 it is treated separately as explained before.

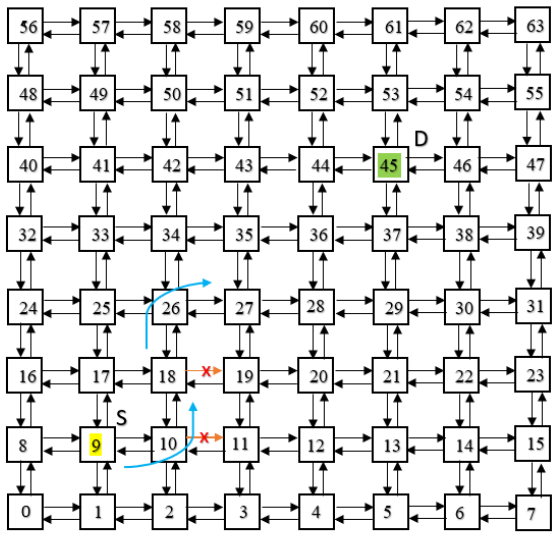

Fig. 4. 8x8 Mesh NoC Architecture With Two X-axis Faults

Case 3 (Double X-axis faults): Consider Figure 4, where for a packet source is 9 and destination is 45, and a fault in the east port of router 10 (marked as 'x') and in the east port of router 10 (marked as 'x'). Whenever this type of fault occurs, if the destination row is above or in the same row as that of the current row, then we deflect the flit to north. If the destination row is below the current row, then we deflect the flit to south. In both the former and latter cases before deflecting the flit we set the *fault bits* to 01 in that particular flit. If the next router also has an X-axis fault, the *fault bits* are kept as it is that is 01 and again deflected based on the destination row. When a flit arrives a particular router either from north or south port with the *fault bits* set as 01, the current router resets the *fault bits* to 00 and routes the packets as normal XY routing. In our example, the routing follows the path 10-18-26 then from 26 it starts afresh and routes the packet according to XY-routing.

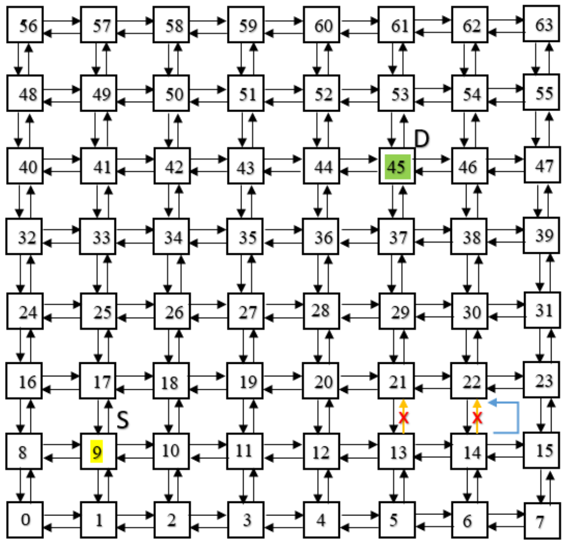

Fig. 5. 8x8 Mesh NoC Architecture With Two Y-axis faults

Case 4 (Double Y-axis faults): Consider Figure 5, where for a packet source is 9 and destination is 45, and a fault in the north port of router 13 (marked as 'x') and in the east port of router 14 (marked as 'x'). Whenever this type of fault occurs, it means that the flit has arrived its destination column, we deflect the flit to the east, before deflecting the flit we set the *fault bits* to 10 in that particular flit. If the next router also has a Y-axis fault, the *fault bits* are kept as it is (i.e. 10) and again deflected to east. When a flit arrives a particular router with the *fault bits* set as 10, this means that the flit was deflected due to a Y-axis fault and has arrived the current router from a west port. The current router resets the *fault bits* to 00 and routes the packets to north if the destination row is above the current row or to the south if the destination row is below the current row. In our example, the routing takes the path 13-14-15-23-22-21 then it takes its final turn.

5 Experimental Setup

5.1 Simulation Setup:

We use Booksim2.0, the cycle accurate NoC simulator for modelling 8x8 CMP with 2D topology [11]. Booksim supports various kinds of routing algorithms, traffic patterns and network topologies. It can generate NoC traffic from real traffic traces in addition to the synthetic traffic patterns. We incorporate an algorithm which handles the traffic in the entire NoC. The simulation is carried out in latency mode, in which the simulation is carried out by injecting batch of packets. The simulation is run till the average network latency is calculated for the current network configuration. A flit history is taken out to trace the flit traffic path, which will reflect the entire path taken by a particular flit. We verify the fault bypassing strategy by this flit-tracing file.

5.2 Hardware Implementation Setup:

We use Vivado tool from Xilinx to implement an NoC router architecture with and without fault tolerant routing technique on a FPGA kit. In our case we have used ZedBoard Zynq-7000 XC7Z020CLG484-1 [15]. The design and the synthesised code is flashed on to the board, with the help of an IP catalogue VIO. This IP catalogue helps in creating a top module VIO wrapper which can directly be flashed on the board, and verified successfully.

6 Result and Analysis

Figure 7 contains the injection rate vs average packet latency plot for normal XY algorithm without fault and proposed fault tolerant algorithm with single and double faults. As expected normal XY performs better. Our fault tolerant algorithm due to flit deflections around the fault location takes more number of hopes to reach the destination. This increases the latency of such flits. Due to the additional hops and hence the extra packet movement in the network with fault tolerant routing, it saturates easily. Our approach guarantees packet delivery at destination at the additional overhead of higher average packet latency.

We also analyse the count of such deflections flit due to our algorithm. Figure 8 contains the number of deflected flits. Higher the traffic higher the number of deflected flits. We also observe that the result for single X fault and single Y fault cases are almost same. Hence they are not separately plotted. Same is the case with double X fault and double Y fault. Our algorithm cannot ensure guaranteed delivery of packets if one fault is in X and other is in Y links. We could address this at the expense of additional fault bits. We reserve that part for a future extension.

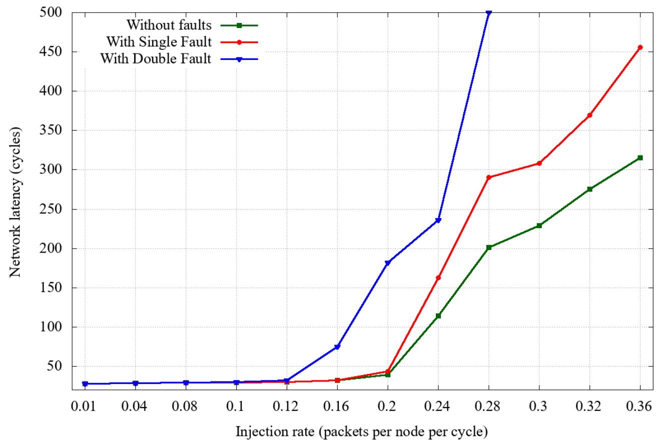

Fig. 6. Injection rate versus average packet latency for single and double fault

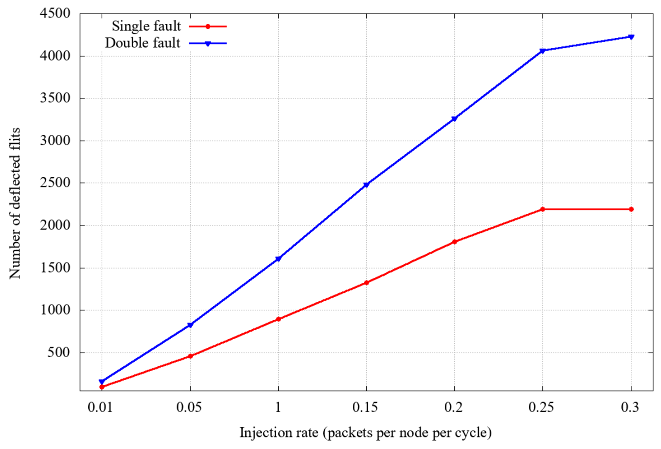

Fig. 7. Injection rate versus deflected flits for single and double fault

Routing-Algorithm	LUTs-Used (count)	BUFGs-Used (count)	Power consumption (mW)
Basic-XY	493	3	120
Fault-tolerant	718	3	125

Table 1. Area overhead comparision

Terms used:- LUTs - Look Up Tables and BUFG - Buffer Gates

For the hardware implementation, we have considered a single router with five ports (north, south, east and west along with a local port) representing links of a node in the network. The Verilog design of the proposed routing module is done and the synthesized RTL was dumped on ZedBoard Zynq-7000 FPGA hardware kit [15].

We use Virtual Input Output(VIO) to realise our implementation as there is a need of more than 40 input pins as well as nearly 80 output pins. Since these many pins are not available physically on the board. VIO in the Vivado tool for pin virtualization. The area overhead is calculated by looking at the increase of number of LUTs and BUFGs as listed in Table 1. Even though the routing logic takes 1.45 times the LUTs than basic XY routing. The overall router area overhead is approximately 4%. Power is one of the major factors to be considered in NoC architecture. For our proposed algorithm the power consumption increase is about 5mW more than the standard XY algorithm. These numbers are taken for ZedBoard Zynq-7000 FPGA hardware kit [15] and may vary from one FPGA chip to other.

7 Conclusion

In this paper we proposed a new algorithm for guaranteed delivery of packets in an NoC with faulty links. In the proposed algorithm we re-routed a packet approaching a faulty link, by deflecting it to the direction closer to the destination. There are different cases considered such as a fault in x-axis, a fault in y-axis, parallel faults and others. The proposed algorithm is efficient for all the types of faults present in the network. The output for different types of faults are explained. The various statistical analysis and graphs are shown and a number of parameters compared between single and double faults can be observed. The proposed algorithm is adaptive to any type of faults in the network, saves a large number of flits from getting discarded in the network due to faulty links by deflecting it to a new path and is hardware implementable. Few disadvantages such as increase in average latency, congestion and heat dissipation is also seen which can be minimised. We hope that the proposed fault tolerant algorithm can ensure NoC design with higher reliability.

References

1. J. Wang, F. Fu, T. Zhang and Y. Chen, “A small-granularity solution on fault-tolerant in 2D-Mesh Network-on-Chip”, 10th IEEE International Conference on Solid-State and Integrated Circuit Technology, 2010.
2. Jianfei Wu, Hua Cai, Fuheng Qu and Yong Yang, “The reconfigurable fault tolerance routing algorithm in mesh topology structure”, 10th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2014), 2014.
3. A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu and G. Chen, “A Reliable Routing Architecture and Algorithm for NoCs”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 31, no. 5, pp. 726-739, 2012.

4. T. Schonwald, J. Zimmermann, O. Bringmann and W. Rosenstiel, “Fully Adaptive Fault-Tolerant Routing Algorithm for Network-on-Chip Architectures”, 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007), 2007.
5. Y. Fukushima, M. Fukushi, I. Yairi and T. Hattori, “A Hardware-Oriented Fault-Tolerant Routing Algorithm for Irregular 2D-Mesh Network-on-Chip without Virtual Channels”, IEEE 25th International Symposium on Defect and Fault Tolerance in VLSI Systems, 2010.
6. M. Ebrahimi, M. Daneshtalab and J. Plosila, “High Performance Fault-Tolerant Routing Algorithm for NoC-Based Many-Core Systems”, 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, 2013.
7. M. Pirretti, G. Link, R. Brooks, N. Vijaykrishnan, M. Kandemir and M. Irwin, “Fault tolerant algorithms for network-on-chip interconnect”, IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2004.
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 and Test in Europe Conference and Exhibition (DATE), 2009.
9. Y. Jojima and M. Fukushi, “A fault-tolerant routing method for 2D-mesh Network-on-Chips based on components of a router”, IEEE 5th Global Conference on Consumer Electronics, 2016.
10. X. Gu, H. Cai, Y. Zhang, N. Huang and Y. Yang, “Research on network fault tolerance method on chip”, 2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2016.
11. W. Dally and B. Towles, “Route packets, not wires: on-chip interconnection networks”, Design Automation Conference, 2001.
12. P. Yan, S. Jiang and R. Sridhar, “A novel fault-tolerant router architecture for network-on-chip reconfiguration”, 28th IEEE International System-on-Chip Conference (SOCC), 2015.
13. N. Chatterjee, S. Chattopadhyay and K. Manna, “A spare router based reliable Network-on-Chip design”, IEEE International Symposium on Circuits and Systems (ISCAS), 2014.
14. Z. Zhang, A. Greiner and S. Taktak, “A reconfigurable routing algorithm for a fault-tolerant 2D-Mesh Network-on-Chip”, Proceedings of the 45th annual conference on Design automation (DAC), 2008.
15. Zedboard.org Zedboard. Available at: http://www.zedboard.org/product/zedboard