International Journal of Research and Review
www.gkpublication.in E-ISSN: 2349-9788; P-ISSN: 2454-2237

Original Research Article

Performance Evaluation of Fault-Tolerant Routing For Network-On-Chip in Hotspot and Local Traffic

Ladan Momeni[1], Arshin Rezazadeh[2]

[1,2]Faculty Member,
[1]Sama Technical and Vocational Training College, Islamic Azad University, Ahvaz Branch, Ahvaz, Iran
[2]Department of Computer Engineering, Shahid Chamran University of Ahvaz, Ahvaz, Iran

Corresponding Author: Ladan Momeni

Received: 06/06/2015 *Revised: 09/06/2015* *Accepted: 24/06/2015*

ABSTRACT

NoC is a scalable and flexible communication medium for the design of multi-core based SoC. Communication performance of NoC depends seriously on efficient routing algorithms. Fault-tolerant routing algorithm is the ability to survive failure of individual components. We evaluated a deterministic fault-tolerant, deadlock-free routing algorithm in two-dimensional (2D) mesh based topology on Fault-Tolerant-Routing (FTR) to increase performance of the messages over the on-chip interconnection networks. The FTR algorithm is a wormhole-switched routing for 2-D mesh networks and has been used for block faults. This algorithm uses virtual channels to pass faulty regions. We have evaluated the FTR algorithm by two different traffic patterns, hotspot and local, to show message delays and performance in the network which led to an Improved-Fault-Tolerant-Algorithm (i-FTR). Moreover, to simulate FTR and i-FTR algorithms, same network conditions namely network size, message length and number of generated messages has been considered. It can be deduced from results that i-FTR performance is better compared to FTR algorithm. Furthermore, results show that the interconnection network of NoC which has been used for i-FTR can deal with higher message rates and can tolerate higher traffic loads with higher utilization.

Keywords: routing algorithm; wormhole switching; 2D-mesh interconnection networks; virtual channel

I. INTRODUCTION

With the improvements in process technology, multi-core architecture has become the norm. A new communication paradigm, called Network-on-Chip (NoC), has been widely investigated for future Systems-on-Chips (SoCs) Fault-tolerance is an important feature of NoC since it usually has inherent redundancy in communication which can be used for hardware reconfiguration. [1]

In NoC paradigm, cores are connected to each other through a network of routers and they communicate among themselves through packet-switched communication. The protocols used in NoC are generally the simplified versions of general communication protocols used in data networks. This makes it possible to accept mature concepts of communication networks routing algorithms, switching

techniques, congestion control etc. in NoC. [2]

Two kinds of faults (permanent and transient) need to be addressed in NoC architectures. There are two methods to cope with transient and permanent faults in NoC. Flow-control-based methods combine the error control code with the retransmission mechanism to tolerate transient faults occurring in transmission; the other is fault-tolerant routing which utilizes the inherent structure redundancy of NoC to route packets around the permanent faulty routers or links to achieve fault-tolerance. A good fault-tolerant routing algorithm should ensure zero lost packet in whatever fault patterns as long as a path exists. [3]

The most common template that proposed for the communication of NoC is a 2-D mesh network topology where each resource is connected with a router. In these networks, source nodes (an IP-Core), generate packets that include headers as well as data, then routers transfer them through connected links to destination nodes. Every node has four neighbors; one on North, one South, one West and one on East of any node except border nodes.

The two main components of interest when designing NoCs are the routers, designed for performance, and secondly, the underlying topology and its associated routing algorithm, designed to facilitate data transfer. [4]

Wormhole routing is the preferred flow control strategy for application-specific topologies, providing the NoC components with low latency and buffering requirements. Although there are advantages to this type of routing, it is also prone to contention as the packets tend to spread throughout the network during transmission. Contention within a network occurs when different packets require the same resources at a particular moment in time. If a contention point propagates throughout the system, congestion is formed, causing performance degradation. As a result, the system can have long delays and may not meet throughput and utilization requirements as is necessary to adhere to its system demands. A bottleneck is thus a single contention point that limits the overall system performance, [5]

In the context of NoCs, a communication failure can happen due to a fault on either a router or a link between two routers. [1]

Deadlock is a situation in a network where a number of messages wait for each other and none of them can proceed. It is a result from a cyclic dependency between the packets. Deadlocks can be handled in two ways: to prevent them from happening (deadlock avoidance) or cope with the situation when a deadlock appears (deadlock recovery). Well known avoidance methods include turn models and virtual channels. [6]

We have been modified the way of usage of virtual channels to improve the performance of routing algorithm such as latency and utilization. As simulation results showed, delay of messages and also delay of messages in source node decreased by use of modified algorithm, i-FTR fault-tolerant algorithm. Moreover, utilization of i-FTR has higher in comparison with FTR algorithm.

The organization of the paper is as follows. Routing algorithms are discussed in section 2. In section 3 fault-tolerant algorithms explained. FTR algorithm and evaluated algorithm of this research are described in section 4. Section 5 covers the simulation results and finally section 6 presents the conclusion.

II. ROUTING-ALGORITHM

A routing algorithm which consists of a routing function and a selection strategy determines a path that a packet traverses from source to destination. According to the

coordinates of current and destination nodes, the routing function returns a set of available output ports. Then, the selection strategy chooses one from the set based on the parameters which are applied to weight the output port. [7]

In NoCs, routing algorithms are used to determine the path of a packet from the source to the destination. These algorithms are classified as deterministic and adaptive. The implementations of deterministic routing algorithms are simple but they are not able to balance the load across the links in non-uniform or bursty traffic. Adaptive routing algorithms are proposed to address these limitations. By better distributing load across links, adaptive algorithms improve network performance and also provide tolerance if link or router failure occurs. [8]

Deadlock is an anomalous network state in which a circular hold-and-wait dependency relation is formed among the network resources, causing packet routing to be indefinitely postponed. Meanwhile, in livelock situation, a packet travels continuously around the network without ever reaching its destination because the requested channels are constantly occupied by other packets. In any routing scheme, it is essential to avoid both deadlock and livelock. [9]

In XY routing algorithm, the packet is first routed across X axis and then across Y axis until it reaches the destination node. However, applying XY routing for the torus topology may cause deadlock due to the channel dependency in each dimension between different messages as a result of added wrap-around links (with respect to the mesh topology). By using more than one virtual channel, there will be the flexibility of designing different deadlock-free routing algorithms for the cost of extra hardware complexity, more area, and thus higher power consumption. [10]

III. FAULT-TOLERANT ROUTING

Fault-tolerant methods require the use of redundancies which in turn deteriorate other concerns of NoCs such as performance and power consumption. In other words, reliability, performance and power consumption in NoCs are conflicting objectives, as improvement of one objective may deteriorate the other objectives. [11] As mentioned above, we focus on performance metrics in this article such as utilization and latency of packets.

Fault-tolerance is defined as the ability of a system to continue operation despite presence of faults. In this sense, fault-tolerance is closely related to concepts such as reliability, availability, and dependability, as it serves by providing these features. Faults in a network take many forms, such as hardware faults, software bugs, or malicious sniffing or removal of packets. The first step in dealing with errors is to understand the nature of component failures and then to develop simple models that allow us to reason out the failures and the methods for handling them. Classification of faults by nature is either random or systematic faults. Random faults are usually hardware faults affecting the system components, which occur with a certain probability, while systematic faults such as software failures are faults which are not random, whether a component has it or not. We assume that such permanent failures are detected and contained on a node or link. [12]

The proposed fault tolerant architecture takes advantage of the fact that one can change the size of each buffer in accordance to the application needs. When a fault occurs in a buffer slot, instead of disabling the entire input channel, one isolates the faulty slot. To sustain performance the input channel can borrow buffer slots from the neighbors according the monitoring architecture. [13]

There are two main kinds of thermal problems in Network-on-Chip, including regional temperature differential and hotspot. Regional temperature differential is caused by the thermal unbalanced distribution in the network. It makes link latency and gate latency hard to predict thus increasing the possibility of system synchronization failure. Hotspot is the node whose temperature is much higher than the others' in the network. A hotspot is formed when processing too much data and generating large amount of dynamic power consumption. The hotspot node will easily get damaged for its high temperature. Both regional temperature differential and hotspot decrease system reliability and infect system performance. [14]

IV. IMPROVED-FAULT-TOLERANT-ROUTING ALGORITHM

This section describes how we evaluate an existing technique for fault-tolerant wormhole routing in NoC with a mesh-based topology. The routing algorithm considered in this paper is a deterministic e-cube routing as long as no faults occur. When facing a faulty link or node a given flit cannot be routed along its normal e-cube route and its direction would be changed according to a bracket of rules and it would be re-routed along a fault chain or ring around the faulty nodes or links. These rules have been put forward by Chalasani and Boppana. [15] The main idea is described in the rest of this section.

A. Fault-Tolerant-Routing (FTR): Primitive Algorithm

The algorithm presented by Chalasani and Boppana, FTR, uses four virtual channels (VCs). This algorithm is able to pass faulty blocks and overlapped faulty regions. Each message is injected into the network as a row message (message must travel horizontally at first) and its status is set to normal. Messages are routed along their deterministic e-cube hop if they are not blocked by faults. When faults are happened, its status would be set to misrouted and depending on the message type and relative position of destination nodes to source nodes, direction of messages are set to clockwise or counter-clockwise by use of table 1. [15] Messages are routed on border of faulty block according to specific directions. The status of a message which is passed the faulty region would be configured to normal again.

B. Improved- Fault-Tolerant-Routing (i-FTR): Modified Algorithm

The e-cube routes a message in a row until the message reaches a node that is in the same column as its destination, and then routes it in the column. For fault-free meshes, the e-cube provides deadlock-free shortest path routing without requiring multiple virtual channels to be simulated. At each point during the routing of a message, the e-cube specifies the next hop which should be taken by the message. The message is assumed to be blocked by a fault, if its e-cube hop is on a faulty link. [16] The evaluated modification uses number of VCs as same as primitive algorithm. An entire column/row fault which disconnects meshes has not been considered. [17]

To route messages around faulty rings (f-rings), they are classified into one of the following types: EW (East-to-West), WE (West-to-East), NS (North-to-South), or SN (South-to-North). A message is labeled as either an EW or WE message (row direction) when it is generated, depending on its direction of travel along the row. Once a message completes its row hops, it becomes a NS or a SN message (column direction) depending on its travel direction along the column. Thus, EW and WE messages will become NS or SN messages; however, NS and SN messages cannot change their types because of live-lock and dead-lock. [17] Proposed algorithm and

procedures needed are given in fig. 1 and fig. 2.

Procedure Set-Message-Type (M)
/* Comment: The current host of M is (s_1,s_0) and destination is (d_1, d_0).
When a message is generated, it is labeled as WE if $d_0 \geq s_0$ and as WE otherwise. */
If M is an EW or WE message and $s_0 = d_0$,
Change its type to NS if $s_1 < d_1$ or SN if $s_1 > d_1$.

Procedure Set-Message-Status (M)
/* Comment: Determine if the message M is normal or misrouted. The current host of M is (s_1, s_0) and destination is (d_1, d_0). */
1) **If** M is a column message and $s_0 = d_0$, and its next e-cube hop is not on a faulty link, **then** set the status of M to normal and return.
2) **If** M is a row message and its e-cube hop is not blocked, **then** set the status of M to normal and return.
3) **Set** the status of M to misrouted, determine using Table 1 the f-ring orientation to be used by M for its misrouting.

Figure 1: Set-Message-Type and Set-Message Status procedures.

The technique evaluated in this paper has one primary advantage over the one presented in the previous work. According to, [8] as long as no fault occurs, a flit always uses a fixed virtual channel (channel c0 for EW message, c1 for WE, c2 for NS and channel c3 used for SN messages). When faults are occurred and a flit is re-routed, it uses specific virtual channel depending on pre-defined set of rules. However, in the current paper, a flit is allowed to use all virtual channels instead of just one fixed virtual channel when its type is NS or SN and located on the boundary of a fault block. [17] But in FTR algorithm using from just two virtual channels is permitted in this situation. Using this modification, simulations are performed to evaluate the performance of the enhanced algorithms compared to the algorithms proposed in prior work. Simulation results indicate an improvement in the average message delays and average message wait times (for source nodes) for different fault rates. Furthermore, the enhanced approach can handle higher message injection rates, it means it has higher saturation rate in hotspot and local traffic. Moreover, utilization of evaluated algorithms showed that the new algorithm has higher performance in comparison with old one. Utilization illustrates the number of flits in each cycle, which passed from one node to another, in any link over bandwidth. Bandwidth is defined as the maximum number of flits could be transferred across the normal links in a cycle of the network.

Table 1: Direction to be used for misrouted messages on faulty rings

Message Type	Traversed on the f-ring	Position of Destination	F-Ring Orientation
NS or SN	No	Don't care	Either orientation
EW	No	In a row above its row of travel	Clockwise
EW	No	In a row below its row of travel	Counter Clockwise
EW	No	In the same row	Either orientation
WE	No	In a row above its row of travel	Clockwise
WE	No	In a row below its row of travel	Counter Clockwise
WE	No	In the same row	Either orientation
Any message	Yes	Don't care	Choose the orientation that is being used by the message

Procedure i-Fault-Tolerant-Route (Message M)
/* Specifies the next hop of M */
1) **Set-Message-Type (M)**.
2) **Set-Message-Status (M)**.
3) **If** M is **normal**, **select** the hops specified by thex-y algorithm and use all 4 virtual channels.
4) **If** M is misrouted, **select** the hop along its f-ring orientation.
5) **If** the selected hop is on an f-ring link, **route** the message using all 4 virtual channels when M not blocked yet.
6) **If** the selected hop located on an f-ring link, **route** the message using virtual channel c0 if M's type is EW, c1 if WE, c2 if NS, or c3 if SN.
7) **If** the selected hop is not on an f-ring link, **route** the message using the virtual channel specified by the base algorithm.

Figure 2: i-Fault-Tolerant-Route (i-FTR) procedure.

V. RESULTS AND DISSCUTIONS

In this section, we will describe how we perform the simulation and acquire results from simulator. Furthermore, we show the improvements of the modified algorithm.

A. Simulation Structure

In order to model the interconnection network, an object-oriented simulator was developed base on. [18-20] The simulator is structured so that classes, such as routing algorithm or message traffic can be changed without any change in other components. A flit-level simulator has been designed. We record average message latencies measured in the network with the time unit equal to the transmission time of a single flit (one clock cycle). Our study is performed for 10% fault rates for all links faulty. In our simulation studies, we assumed message length to be equal to 32 flits and we used an 8 × 8 2-D mesh network. Two traffic patterns are simulated:

a) Hotspot – Messages are destined to a specific node with a certain probability and are otherwise uniformly distributed.
b) Local traffic – The source node sends messages to any other node with equal probability but with fixed maximum distance. We use Manhattan distance calculated as follows:

$$D_m = | x_s - x_d | + | y_s - y_d | \quad (1)$$

In equation above, D means distance and m denotes to Manhattan. Furthermore, x denotes to dimension x and y denotes to dimension y. Likewise, s used for source node and d is destination node.

The number of messages generated for each simulation result depends on the network size and traffic distribution. It is in the range of 2,000,000 to 4,000,000 messages. The simulator has three phases: start-up, steady-state, and termination. The start-up phase has been used to ensure that network is in steady-state before measuring message latency. For this reason the statistics for the first 10% of generated messages have not been gathered. All measurements are obtained from the remaining of messages generated in steady-state phase. The termination phase would continue till all the messages generated are delivered. [18]

In the remaining part of this section, we described the effect of using our modification on the performance of deterministic routing in the mesh network in details.

B. Simulation Results

Figures 3to 5 show the simulation results for Average Message Delay (AMD), Average Message Waiting in Source Nodes (AMWS), and utilization over AMD with 32 flit messages on 8*8 2-dimentional mesh network with hotspot traffic. [18]

In order to generate hotspot traffic we used a model proposed in. [21] According to this model each node first generates a random number. If it is less than a predefined threshold, the message is sent to the hotspot node. Otherwise, it is sent to other nodes of the network with a uniform distribution. [22]

As the mesh interconnection network is not a symmetric network, we have considered two types of simulation for hotspot traffic in this network. In one group of simulations, a corner node is selected as the hotspot node and in the other group; a node in the middle of the network is chosen as the hotspot node, and finally averaged. Hotspot rate is also considered in our study, namely 10%. [22]

We also considered local traffic for this comparison and to show the effect of mapping of elements. As learned by simulation results for uniform traffic at our study, it is founded that average number of hopes for this 8 x 8 mesh with this algorithm and other similar algorithms such as if-cube3 [23] is about 5. As the result of this practice, we used 5 Manhattan distance for messages to learn if any node needs no more than 5 hopes for destination what happened. Fig. 6 to 8 shows the effects of such work for evaluated algorithm.

Fig. 3 shows the average message delay (AMD) over the message injection rate (MIR). This latency illustrates the number of cycles between the time in which the first flit of a message injected into the network and the time that last flit of that message reached to the destination node. One can deduce that network which uses FTR algorithm is saturated with lower MIR while i-FTR algorithm has higher saturation point, even with the same virtual channels. For instance, by FTR algorithm, the AMD for 0.001 MIR is over 103 cycles, whereas the other algorithm, i-FTR algorithm, has less than 62 AMD in the network. In fact the evaluated fault-tolerant routing algorithm has lower AMD.

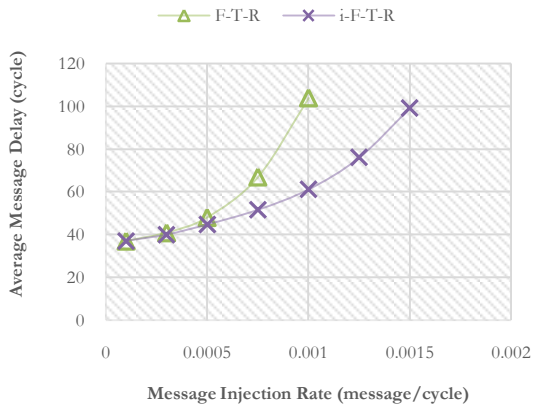

Figure 3: Average Message Delay (AMD) of FTR and i-FTR routing algorithms. 10% of faulty links with Hotspot traffic.

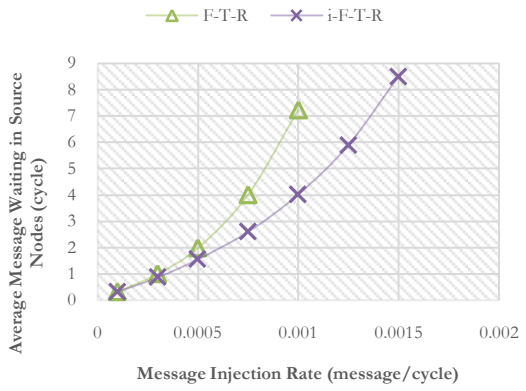

Figure 4: Average Message Waiting in Source Nodes (AMWS)of FTR and i-FTR routing algorithms. 10% of faulty links with Hotspot traffic.

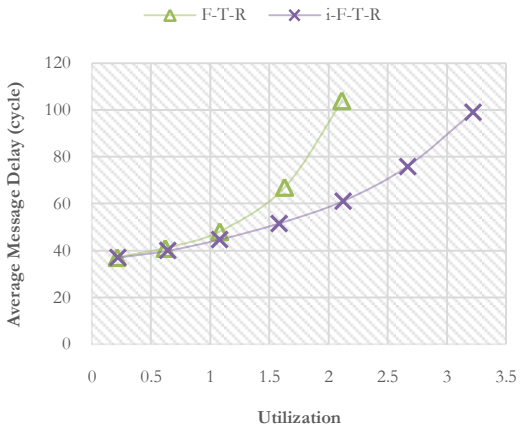

Figure 5: Performanceof FTR and i-FTR routing algorithms. 10% of faulty links with Hotspot traffic.

The next parameter which has been studied is average message waiting in source node (AMWS) which illustrates average number of cycles that a message waits to inject into the network due to lack of buffer. As it is shown in fig.4somepart of delays which messages are encountered with, is the delay of waiting for an empty buffer in source nodes. For instance, comparing fig. 3 and fig. 4significantly shows that about7 cycles of over 103 cycles of AMD in 0.001 MIR are caused by waiting in source nodes which is about 7% of AMD.

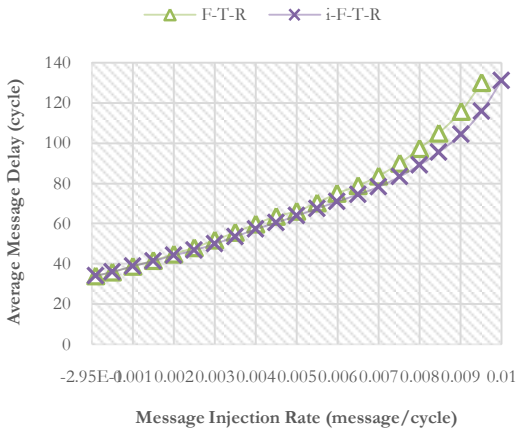

Figure 6: Average Message Delay (AMD) of FTR and i-FTR routing algorithms. 10% of faulty links with Local traffic.

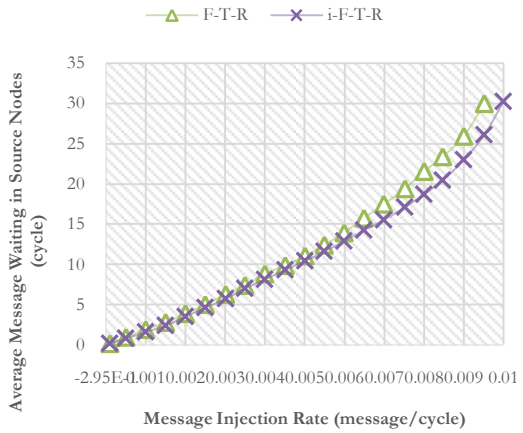

Figure 7: Average Message Waiting in Source Nodes (AMWS) of FTR and i-FTR routing algorithms. 10% of faulty links with Local traffic.

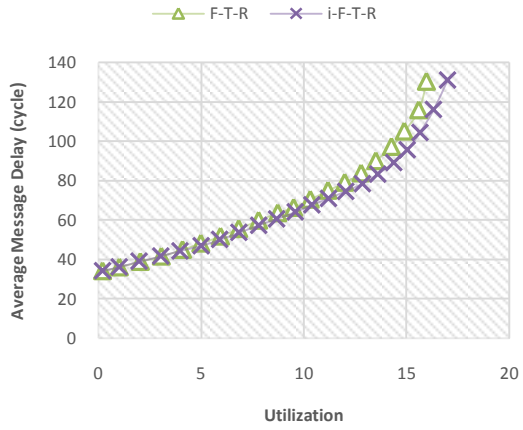

Figure 8: Performance of FTR and i-FTR routing algorithms. 10% of faulty links with Local traffic.

We have examined average message delay (AMD) over utilization. Utilization is the last parameter which has been evaluated to compare performance of i-FTR algorithm with FTR algorithm to work in faulty conditions. The most valuable comparison we have done between these two algorithms is the rate of average message delay over utilization. As fig.5 illustrates, the utilization of the network channels which uses the FTR algorithm is lower while the i-FTR algorithm has been higher utilization. As an example in fig. 5, we can look at the amount of average message delay for both algorithms with 2.1% utilization. In this point of utilization, the network which is using FTR has more than 103 AMD at 100% traffic load while the other network, using i-FTR, has about62 AMD, and it has not been saturated. Comparing the utilization of these algorithms for 100% traffic load, it is obvious the network using i-FTR has 3.22% utilization, whereas the other one has just 2.11% utilization in hotspot mode. Utilization improvement of network is more than 65% at 100% traffic load for this case.

We also consider abovementioned parameters for local traffic. As showed in fig. 6 i-FTR routing algorithm has lower latency compared to FTR in this traffic pattern. For example, for 0.0095 MIR i-FTR has less than 116 AMD whereas FTR has more than 130 AMD. Moreover, i-FTR algorithm has higher saturation point in comparison with FTR algorithm in local traffic pattern.

Additionally, the next parameter, average message waiting in source nodes (AMWS), clarified this improvement. As we can see in fig. 7 a large portion of delays caused by AMWS which is lower for i-FTR.

The last parameter we consider is AMD over utilization in local traffic pattern to show the performance of i-FTR. As an example i-FTR algorithm has higher utilization for 130 AMD, 17%, compared to 15.97% for FTR which demonstrates 6.5% performance improvement in this case at 100% traffic load.

VI. CONCLUSION

Designing a deadlock-free routing algorithm that can tolerate unlimited number of faults is a great challenge. Faulty blocks are extended, by disabling good nodes, to be solid faults in existing fiction to assist the designing of deadlock-free routing algorithms for 2-D mesh networks. The simulation results show that up to 70% improvement of network delays in hotspot traffic pattern and over 12% in local traffic

pattern which are needed to work with rectangular faults can be recovered if the number of original faulty links is less than 10% of the total network links. Furthermore, we showed 65% and 6.5% improvement for utilization in hotspot and local traffic pattern, respectively.

In this paper, for the purpose of improving performance, we evaluated a method to shrink these block faults by using the same virtual channels as primitive algorithm.

The deterministic algorithm is enhanced from a non-adaptive supporter by utilizing the virtual channels that are not used in the non-faulty conditions. The method we used for enhancing the i-FTR algorithm is simple, easy and its principle is similar to the previous algorithm, FTR. There is no restriction on the number of faults tolerated in the proposed algorithm.

ACKNOWLEDGEMENT

This work is supported by Sama Technical and Vocational Training College, Islamic Azad University, Ahvaz Branch. The authors would like to thank them for their funding on this research project.

REFERENCES

1. Jian Wu, Zhen Zhang, and Chris Myers. A Fault-Tolerant Routing Algorithm for a Network-on-Chip Using a Link Fault Model. 32nd International Symposium on Reliable Distributed Systems; 2013 Sept. 30 -Oct. 3; Braga, Portugal. IEEE; 2013. P. 1-9.
2. Amit Zinzuwadia, Parag Parandkar, Renu Verma, Sumant Katiyal. An Efficient Deadlock-free NARCO based Fault Tolerant Routing Algorithm in NoC Architecture. International Journal of Emerging Technology and Advanced Engineering. 2012; 2(2): 227-234.
3. Chaochao Feng, Zhonghai Lu, Axel Jantsch, Minxuan Zhang, and Zuocheng Xing. Addressing Transient and Permanent Faults in NoC with Efficient Fault-Tolerant Deflection Router. IEEE Transactions On Very Large Scale Integration (VLSI) Systems. 2013; 21(6): 1053-1066.
4. Chris Jackson, Simon J. Hollis. A deadlock-free routing algorithm for dynamically reconfigurable Networks-on-Chip. Journal of microprocessors and microsystems. 2011; 35(2): 139-151.
5. Anita Tino, Gul N. Khan. Designing power and performance optimal application-specific Network-on-Chip architectures. Journal of micro-processors and microsystems. 2011; 35(6): 523-534.
6. Teijo Lehtonen, Pasi Liljeberg, and Juha Plosila. Analysis of Fault Tolerant Deadlock-free Routing Algorithms for Mesh NoCs. 3rd Workshop on Diagnostic Services in Network-on-Chips; 2009 April 24; Nice, France. 2009. p.54-57.
7. Junhui Wanga, Huaxi Gu, Yintang Yang, Kun Wang. An energy- and buffer-aware fully adaptive routing algorithm for Network-on-Chip. Microelectronics Journal. 2013; 44(2): 137–144.
8. P. Lotfi-Kamran, A.M. Rahmani, M. Daneshtalab, A. Afzali-Kusha, Z. Navabi. EDXY – A low cost congestion-aware routing algorithm for network-on-chips. Journal of Systems Architecture. 2010; 56(7): 256–264.
9. Wen-Chung Tsai, Kuo-Chih Chu, Yu-Hen Hu, Sao-Jie Chen. A scalable and fault-tolerant network routing scheme for many-core and multi-chip systems. J. Parallel Distrib. Comput. 2012; 72(11): 1433–1441.
10. Rahmati, H. Sarbazi-Azad, Sh. Hessabi, A. Eslami Kiasari. Power-efficient deterministic and adaptive routing in torus networks-on-chip. Micro-processors and Microsystems. 2012; 36(7): 571–585.
11. A. Patooghy, S. Gh. Miremadi. Complement routing: A methodology to design reliable routing algorithm for

Network on Chips. Microprocessors and Microsystems. 2010; 34(6): 163–173.
12. F. Safaei, M. ValadBeigi. An efficient routing methodology to tolerate static and dynamic faults in 2-D mesh networks-on-chip. Microprocessors and Microsystems. 2012; 36(7): 531–542.
13. Debora Matos, Caroline Concatto, Anelise Kologeski, Luigi Carro, Marcio Kreutz, Fernanda Kastensmidt, Altamiro Susin. A NOC closed-loop performance monitor and adapter. Microprocessors and Microsystems. 2013; 37(6-7): 661–671.
14. Feiyang Liu, Huaxi Gu, Yintang Yang. DTBR: Adynamic thermal-balance routing algorithm for Network-on-Chip. Computers and Electrical Engineering. 2012; 38(2): 270–281.
15. S. Chalasani, R.V. Boppana. Communication in Multicomputers with Nonconvex Faults. IEEE Trans. on Cmputers. 1997; 46(5): 616-622.
16. M. Mohtashamzadeh, L. Momeni, A. Rezazadeh. An Innovative Fault-Tolerant Method for 2-D Mesh-Based Network-on-Chip Routing. UKSim 5th European Symposium on Computer Modeling and Simulation; 2011 Nov. 16-18; Madrid, Spain. IEEE; 2011. p. 339-343.
17. A. Rezazadeh, M. Fathy, Gh. Rahnavard. An Enhanced Fault-Tolerant Routing Algorithm for Mesh Network-on-Chip. Int. Conf. on Embedded Software and Systems; 2009 May 25-27; Zhejiang, China. IEEE; 2009. p. 505-510.
18. L. Momeni, A. Rezazadeh, D. Abednejad. Improved-XY: A High Performance Wormhole-Switched Routing Algorithm for Irregular 2-D Mesh NoC. In: S. Fong et al. 3th International Conference on Networked Digital Technologies; Communications in Computer and Information Science; 2011 July 11-13; Macau, China. Springer; 2011. p. 93-104.
19. Arshin Rezazadeh and Mahmood Fathy. Throughput Considerations of Fault-Tolerant Routing in Network-on-Chip. In: S. Ranka et al. Second International Conference; 2009 August 17-19; Noida, India. Springer; 2009. p. 81–92.
20. Arshin Rezazadeh, Ladan Momeni, and Mahmood Fathy. Performance Evaluation of a Wormhole-Routed Algorithm for Irregular Mesh NoC Interconnect. In: K. Kant et al. 11th International Conference, ICDCN 2010; 2010 January 3-6; Kolkata, India. Springer; 2010. p. 365–375.
21. J. Duato, S. Yalamanchili, and L.M. Ni. Interconnection Networks: An Engineering Approach. Morgan Kauffman, 2003.
22. J. Duato, S. Yalamanchili and L. Ni, "Interconnection networks: An Engineering approach," Published by Morgan Kaufmann, 2003.
23. Samad Rostampour, Ladan Momeni, Arshin Rezazadeh. A Numerical Solution for Throughput Improvement of On-Chip Irregular Mesh Interconnection Network. UKSim 5th European Symposium on Computer Modeling and Simulation; 2011 Nov. 16-18; Madrid, Spain. IEEE; 2011. P. 323 – 328.

How to cite this article: Momeni L, Rezazadeh A. Performance evaluation of fault-tolerant routing for network-on-chip in hotspot and local traffic. Int J Res Rev. 2015; 2(6):355-364.

****