2009 International Conference on Embedded Software and Systems

An Enhanced Fault-Tolerant Routing Algorithm for Mesh Network-on-Chip

Arshin Rezazadeh[1], Mahmood Fathy[1], Gholamali Rahnavard[2]
[1] Department of Computer Engineering
[1] Iran University of Science and Technology, Tehran, Iran
rezazadeh@comp.iust.ac.ir, mahfathy@iust.ac.ir
[2] School of Computer Engineering
[2] Shahid Chamran University of Ahvaz, Ahvaz, Iran
rahnavard@scu.ac.ir

***Abstract*—Fault-tolerant routing is the ability to survive failure of individual components and usually uses several virtual channels (VCs) to overcome faulty nodes or links. A well-known wormhole-switched routing algorithm for 2-D mesh interconnection network called f-cube3 uses three virtual channels to pass faulty regions, while only one virtual channel is used when a message does not encounter any fault. One of the integral stages of designing Network-on-Chips (NoCs) is the development of an efficient communication system in order to provide low latency networks. We have proposed a new fault-tolerant routing algorithm based on f-cube3 as a solution to reduce the delay of network packets which uses less number of VCs in comparison with f-cube3. Moreover, in this method we have improved the use of VCs per each physical link by reducing required channels to two. Furthermore, simulations of both f-cube3 and our algorithm based on same conditions have been presented.**

Keywords-Network-on-Chip, virtual channel, deterministic routing, wormhole switching, delay, fault-tolerant

I. INTRODUCTION

Interconnection networks have become a popular means for interconnecting components of parallel computers. In these networks, nodes are connected to only a few nodes, its neighbors, according to the topology of the network and communicate with each other by passing messages. The 2-dimensional (2-D) mesh network is currently one of the most popular topologies for interconnection systems. Low-dimensional mesh networks, due to their low node degree, are more popular than the high dimensional mesh networks.

A possible approach for getting over the limiting factor in future system-on-a-chip designs is to use an on-chip interconnection network instead of a global wiring [8]. On-chip networks relate closely to interconnection networks for parallel computers, in which each processor is an individual chip [18]. The tile-based network-on-chip architecture is known as a suitable solution for overcoming communication problems in future VLSI circuits [8] [11] [13]. Such chips are composed of many tiles regularly positioned in a grid where each tile can be, for example, an embedded memory, or processor, connected to its adjacent tiles through routers [8][14]. Each tile has two segments to operate in communication and computation modes separately [10].

This enables us to use packets for transferring information between tiles without requiring dedicated wirings. A NoC is a regular/irregular set of routers that are connected to each other on a point to point link in order to provide a communication backbone to the cores of a SoC. The most common template for NoC is a 2-D mesh where each resource or set of resources is connected with a router [1]. In brief, NoCs present the scalable performance needed by systems which grow with each new generation [3]. They allow the wiring energy consumption to be reduced by avoiding the use of long global wires. Furthermore, NoCs are reusable templates and aid to reduce design productivity gap. Finally, bus architecture will not meet all the requirements of future SoCs, as NoCs assure to do. In on-chip networks, information of routing is distributed, and the determination of the network status is distributed among the nodes which exchange information with each other. This type of algorithm is used in the large-scale networks.

The wormhole switching technique proposed by Dally and Seitz [6] has been widely used in the interconnections such as [12]. In this technique, a packet is divided into a sequence of fixed-size units of data, called *flits*, and transmitted from source to destination in asynchronous pipelined manner. If a communication channel transmits the *header flit* of a message, it must transmit all the remaining flits of the same message. As the header flit containing routing and control information moves forward along a specific route; the sequential flits follow it in a pipelined fashion [5]. When the header flit is blocked due to lack of output channels, all of the flits wait at their current nodes for available channels.

Routing algorithms can be generally classified as adaptive routing and deterministic routing. In the former models, there are many paths between the source and the destination. The adaptive nature of this type of routing algorithms makes them very attractive [15]. In the latter, however, the path between the source and the destination of a packet is determined by the source, and it benefits from its simplicity in router design. Since adaptive algorithms are complex for NoCs, a flexible deterministic algorithm is a suitable one due to simple router implementation [7] [8].

Routing is the process of transmitting data from one node to another in a given system. To avoid deadlocks among

978-0-7695-3678-1/09 $25.00 © 2009 IEEE
DOI 10.1109/ICESS.2009.89

messages, multiple virtual channels (VC) are simulated on each physical channel [4]. Each unidirectional virtual channel is realized by an independently managed pair of message buffers [9]. The tradeoff design of NoCs can be achieved by varying the following parameters: topology, width of physical links, buffer allocation, switching techniques and routing algorithms [2]. Subsequently, in this paper we introduce a fault-tolerant routing algorithm based on Sui-Wang's [19] deterministic algorithms.

The fault model is a block fault model, called *f*-ring and *f*-chain. In the block fault model, each connected set of faults has a convex shape (for example, rectangular in 2-D meshes) [4]. Our approach in this paper is to demonstrate techniques which enhance known *f*-cube3 fault-tolerant routing algorithm to provide communication under faults by better latencies. To illustrate this, we apply our technique to the deterministic e-cube for meshes with rectangular faults.

We suppose that if a message has not reached its destination and is blocked due to busy channels, then it will continue to hold the channels it has already obtained and not yet released. Consequently, deadlocks can occur because of cyclic dependencies on channels. To avoid deadlocks, multiple virtual channels are simulated on each physical channel and they are allocated to messages systematically [6]. When faults occur, the dependencies are even more common, and more virtual channels may need to be used or the use of channels may have to be restricted further. Using extra buffers, multiple virtual channels can be simulated on a physical channel in a demand time-multiplexed manner to overcome the problem. We specify the number of virtual channels on per physical channel basis and denote the *i*th virtual channel on a physical channel with c_i.

This paper presents a fault-tolerant routing algorithm for 2-D mesh Network-on-Chips base that enhances a previously proposed technique. The primary distinction between the previous method and the method presented in this paper is in the use of virtual channels in the network. The algorithm considers both node and link faults. Simulation results show that delay of communication by *f*-cube3 algorithm is worse than the improved one, *if*-cube2 in the network. We simulate two algorithms for 3.5, 7, and 10.5% of all links faulty with uniform and hot spot traffic and different packet lengths. Results for all situations show that our algorithm has lower latency and can work in higher message injection rates, with higher saturation point.

The rest of the paper is organized as follows. In section II discusses some deterministic-based routing algorithms are discussed. In section III explains the new *if-cube2* fault-tolerant routing algorithm is explained followed by Section IV in which discusses our experimental result are discussed. Finally, Section V summarizes and concludes the work.

II. DETERMINISTIC ROUTING ALGORITHMS

Routing is the act of moving information from a source to a destination following the shortest possible path. Packet based communication has been brought to NoCs from the Internet. Currently, most of the proposals for routing in NoCs are based upon deterministic routing mechanisms – XY routing algorithm [8]. The remaining of this section discusses such algorithms that use deterministic routing even in the presence of nodes/links failure.

A. Routing Architecture

All routers have five ports, where one is used for its processor/core and the other four ports are used for communication channel between other switches. Each input port has a controller for handshaking and an input buffer. As a consequence of insensitivity to distance, pipelined flow of messages, and small buffer requirements, we have used wormhole technique for the switching [4]. After receiving the packet header, first the routing unit determines which output should be used for routing this packet according to its destination and then the arbiter requests for a grant to inject the packet to a proper output using the crossbar switch. For example we can see a 3 x 3 2-D mesh NoC in figure 1 with four pair links for neighbor links and one pair link for core link.

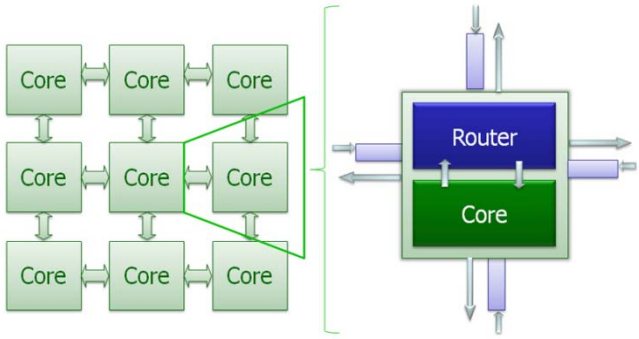

Figure 1. An Example of Regular 3 x 3 2-D Mesh Network-on-Chip

B. Deterministic-Based Fault-Tolerant Routing Algorithms

Some fault-tolerant algorithms use adaptive routing and some of them use deterministic routing. In deterministic ones a message route through network by XY-routing until the message encounters fault. At this point, several papers propose numerous methods for routing the blocked message to reach the destination. One of these algorithms was proposed by Boppana and Chalasani [4] that used deterministic e-cube and implemented by four virtual channels. Another deterministic-based algorithm introduced by Sui and Wang [19] that improved the Boppana and Chalasani's algorithm by using one less virtual channel. We modify this algorithm by using one less virtual channel by reducing VCs to two, twice. The mechanism which we use in the paper is described in the next section.

III. THE IF-CUBE2: A MODIFIED FAULT-TOLERANT ROUTING ALGORITHM

This section shows how we enhance an existing technique for fault-tolerant wormhole routing in Network-on-Chips 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 a faulty node or link is encountered and a given flit cannot be routed along its normal e-cube route, it changes direction as given by a set of

rules and is re-routed along a fault chain or fault ring around the faulty nodes/links. These rules give by Sui and Wang in [19]. The main idea describes in the rest of this section.

A. *f-cube3: Primitive Algorithm*

The algorithm presented by Sui and Wang [19], *f*-cube3, uses one less VC than the algorithm is discussed in [4]. Since each node in this algorithm needs fewer buffers, the area used by buffers of a chip would be reduced. Such an algorithm is able to pass faulty ring and faulty chain even with overlapped faulty regions [19]. Each message is injected into the network as a row message and its direction is set to null. Messages are routed along their deterministic e-cube hop if they are blocked by faults. When faults are encountered, depending on the message type and the relative position of the destination nodes to the source nodes, the direction of messages are set to clockwise or counter-clockwise by use of Set-Direction(M) procedure as shown in [19]. Messages are routed on faulty rings or faulty chains according to the specified directions. The direction of a message which is passed the faulty region would be set to null again. When an end point of fault chain is reached, messages take a u-turn and their directions are reversed.

B. *if-cube2: Modification to Routing Algorithm*

First, we show how to enhance the well-known *f*-cube3 routing 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 to be taken by the message. The message is said to be blocked by a fault, if its e-cube hop is on a faulty link. The proposed modification uses only two virtual channels, c_1 and c_2 on each physical channel and tolerates multiple block faults with overlapped *f*-rings. An entire column/row fault disconnects meshes and is not considered. An example of *f*-ring (F2) and *f*-chain (F1 and F3) showed in figure 2 and showed by bold lines.

To route messages around *f*-rings or *f*-chains, messages 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 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 depending on its direction of travel along the column. Thus, EW and WE messages can become NS or SN messages; however, NS and SN messages cannot change their types. EW and WE messages are collectively known as row messages and NS and SN as column messages [19]. As new algorithm is similar to previous one, *f*-cube3, additional information and usage of virtual channels on *f*-regions can be found at [19].

The technique presented in this paper has one primary advantage over the one presented in the previous work. According to [19], as long as no fault occurs, a flit always uses a fixed virtual channel (channel 0). When faults are encountered and a flit is re-routed, it uses two other possible virtual channels depending on a pre-defined set of rules that explained in [19]. However, in the current paper, a flit is allowed to use all virtual channels instead of just one fixed virtual channel. Using this modification, simulations are performed to evaluate the performance of the enhanced algorithms in comparison to the algorithms proposed in prior work. Simulation results indicate an improvement in the average message delay and average message wait times (both at the source and en-route) for different fault rates, different traffics, and different message lengths. Furthermore, the enhanced approach can handle higher message injection rates (i.e., it has a higher saturation rate). This modification allows us to use only two virtual channels instead of three because when a message routes on a faulty condition, it uses predefined virtual channels mentioned in [19], and while routed in non-faulty hops, it uses that two virtual channels. We show an example in the rest of this section. Since this algorithm is based on [19], deadlock/live-lock freeness features of this fault-tolerant routing algorithm are proven in prior work [19].

C. *Example*

We now consider the example of routing message M from (3, 7) to (7, 3) in figure 2. The path taken by M is also shown in figure 2. M is routed as an EW message from (3, 7) to (3, 6). At (3, 6), its next e-cube hop is faulty and its direction is set to clockwise. At (4, 6), its direction is reset to null and M is routed along its e-cube hop to (4, 3). At (4, 3), M becomes an NS message and travels from (4, 3) to (5, 3). At node (5, 3), due to the fact that its next e-cube hop is faulty, M travels in the counter-clockwise direction to (5, 0). At (5, 0), M takes a u-turn and its direction is reversed to clockwise, since an end node is encountered. M travels along the *f*-chain of F2 in the clockwise direction from (5, 0) to (6, 3). Direction of M is reset to null again at (6, 3) and M is routed along its e-cube hop to destination node (7, 3).

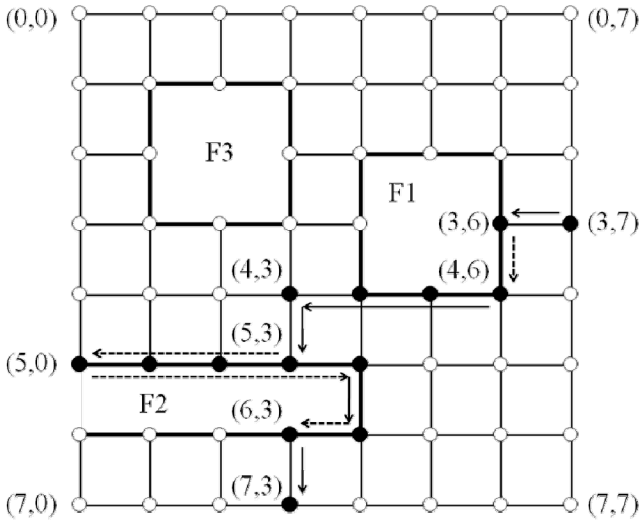

Figure 2. Example of routing message from (3,7) to (7,3). F2 is an instance of *f*-chain, and F1 and F3 are cases of *f*-ring.

Steps 1, 3, 4 and 5 of the path use virtual channels hc_1^- or hc_2^- because we can use both two virtual channels in

modified method, step 2 uses vc_1^- , steps 6, 14, and 16 use vc_1^- or vc_2^- according to previous notes, steps 7, 8, 9, and 15 use hc_a^1 , and steps 10, 11, 12, and 13 use hc_b^2. In steps 2, 7, 8, 9, 10, 11, 12, 13, and 15 only one virtual channel was used as explained in [19] and showed by square-dot arrows. Steps 1, 3, 4, 5, 6, 14, and 16 all two virtual channels can be used and demonstrated by solid arrows.

IV. Results and Discussions

In this section, we describe how we perform the simulation and acquire results from simulator. Furthermore, we show the improvements of the primitive algorithm by our modification.

A. Simulation Methodology

In order to model the interconnection network, an object-oriented simulator was developed base on [17]. The simulator is structured so that classes, such as the routing algorithm or message traffic, can be changed without any changes to the 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, i.e. one clock cycle. Our study is performed for different fault rates: 3.5%, 7.0%, and 10.5% of all links faulty. In our simulation studies, we assume message length to be equal to 32 and 48 flits and we use an 8 x 8 2-D mesh network. Two different traffic patterns are simulated:

- *Uniform* – The source node sends messages to any other node with equal probability.
- *Hotspot* – Messages are destined to a specific node with a certain probability and are otherwise uniformly distributed.

The number of messages generated for each simulation result, depends on the network size and traffic distribution, and is between 200,000 to 1000,000 messages. The simulator has three phases: start-up, steady-state, and termination. The start-up phase is used to ensure the network is in steady-state before measuring message latency. For this reason we do not gather the statistics for the first 10% of generated messages. All measures are obtained from the remaining of messages generated in steady-state phase. Messages generated during the termination phase are also not included in the results. The termination phase continues until all the messages generated during steady-state phase have been delivered [17].

Finally, in the remaining of this section, we study the effect of using two VCs on the performance of deterministic routing in the mesh network. We perform this analysis under a different traffic distribution pattern. It is noted that due to lack of space, only parts of simulation results are presented in this paper.

B. Uniform and Hotspot Traffic

Figures 3 and 4 show the simulation results for three different fault cases, 3.5, 7, and 10.5 percent, with uniform and hotspot *(p=10%)* traffic. Uniform traffic is the most used traffic model in the performance analysis of interconnection networks [17]. Figure 3(a), 3(b), 3(c), 3(d), 3(e), and 3(f) display the effect of the improvement on the performance of 2-D mesh network for this traffic pattern.

Figure 3(a), and 3(b) show the average message delay (AMD) over the message injection rate (MIR) for all fault rates with 32 and 48 flit messages on 8 x 8 mesh network. This delay 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. As we can see, the network which uses *f*-cube3 algorithm is saturated with low MIR while the *if*-cube2 algorithm has a higher saturation point, even with one less virtual channel. As an example in 10.5% case of *f*-cube3 with 32 flits, the AMD for 0.007 MIR is over 290 cycles, yet the other algorithm, *if*-cube2, works normally even for 0.0085 MIR. In fact our fault-tolerant routing algorithm has lower AMD for higher MIRs.

The next parameter we have examined is the average message waiting in source node (AMWS) which illustrates average number of cycles that a message waited to inject into the network because no buffer is available. As it is shown in figure 3(a), and 3(c), a large portion of delays which messages are encountered by, is the delay of waiting for an empty buffer in source nodes. For instance, comparing figure 3(c) and 3(e) it is clear that over 128 cycles of 297 cycles of AMD in 0.007 MIR are caused by waiting in source nodes. This condition is repeated for the other fault cases shown in figures with different message lengths.

The average message waiting in middle nodes (AMWM) is the last parameter simulated for comparing the power of *if*-cube2 algorithm and *f*-cube3 algorithm to work in a faulty condition. As figures 3(e) and 3(f) illustrate, the *f*-cube3 algorithm cannot route messages neither does *if*-cube2 algorithm because of delays in middle nodes. These results show that by using the new approach on routing, we could achieve higher performance with less virtual channels.

In order to generate hotspot traffic we used a model proposed in [16] [17]. 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.

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%.

As mentioned above, the hotspot traffic is a form of uniform traffic, so most of the results attained for uniform traffic also hold for low hotspot traffic rates. Fig. 4(a), 4(b), 4(c), 4(d), 4(e), and 4(f) illustrate the effect of the hotspot traffic on the performance of 2-D mesh, for the case with a hotspot node with 10% hotspot rate. Simulation results reveal that in the presence of hotspot traffic in high traffic rates, the network immediately saturates.

All of the abovementioned observations hold for this traffic pattern showed in figure 4. By comparing the results of mesh networks, it can be concluded that the effect that changing the message lengths and different traffics have on performance in low traffic loads is great for mesh networks.

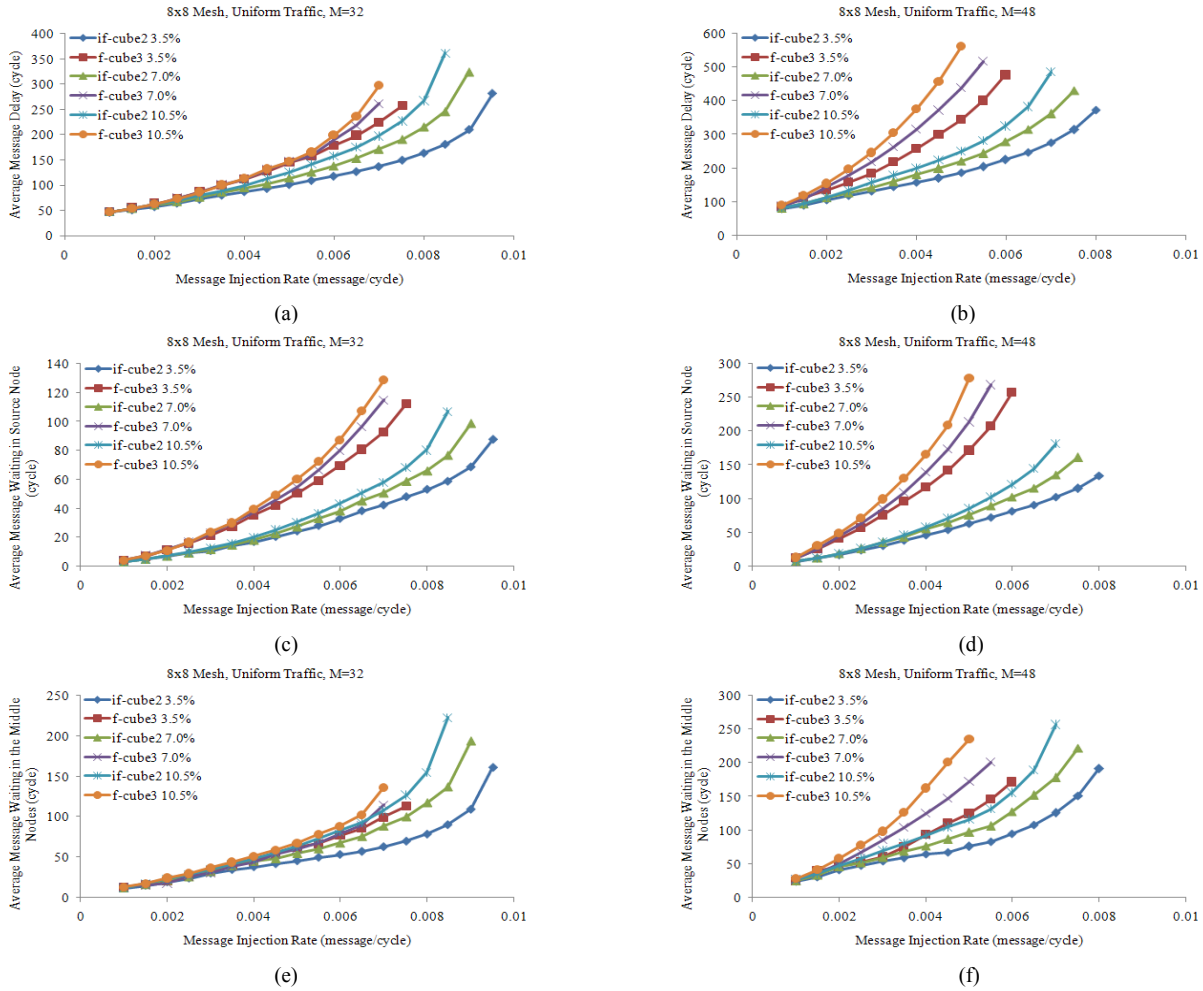

Figure 3. The Average Message Delay (AMD), a) 32 flits message, b) 48 flits message, Average Message Waiting in Source Node (AMWS), c) 32 flits message, d) 48 flits message, and Average Message Waiting in the Middle Nodes (AMWM) e) 32 flits message, f) 48 flits message; as a function of Message Injection Rate (MIR) in 2-D mesh with uniform traffic.

V. CONCLUSION

Designing a deadlock-free routing algorithm that can tolerate unlimited number of faults with two virtual channels is not an easy job. Faulty blocks are expanded, by disabling good nodes, to be rectangular faults in existing literature to facilitate the designing of deadlock-free routing algorithms for 2-D mesh networks. The simulation results show that up to 60% improvement of network latencies, 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.

In this paper, for the purpose of reducing the number of virtual channels, we proposed a method to shrink, by using two virtual channels, these block faults.

We also showed that in various traffics and different message lengths these block faults can be handled. The deterministic algorithm is enhanced from the non-adaptive counterpart by utilizing the virtual channels that are not used in the non-faulty conditions. The method we used for enhancing the *if*-cube2 algorithm is simple, easy and its principle is similar to the previous algorithm, *f*-cube3. There is no restriction on the number of faults tolerated and only two virtual channels per physical channel are needed in the proposed algorithm.

ACKNOWLEDGMENT

This research was supported in part by Shahid Chamran University of Ahvaz grant 2009. We would like to thank Dr. Hamid Sarbazi-Azad for his interesting consultations.

REFERENCES

[1] M. Ali, M. Welzl, M. Zwicknagl, S. Hellebrand, "Considerations for fault-tolerant network on chips," The 17th International Conference on Microelectronics, pp. 178-182, 13-15 Dec. 2005

[2] N. Banerjee, P. Vellanki, K.S. Chatha, "A Power and Performance Model for Network-on-Chip Architectures," Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'04), vol. 2, pp. 1250 - 1255, 16-20 Feb. 2004

[3] L. Benini, G. De Micheli, "Networks on chips: A new SoC paradigm," IEEE Computer, pp. 70-78, Jan. 2002

[4] R.V. Boppana, S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," IEEE Trans. Computers, vol. 44, no. 7, pp. 848-864, July 1995

[5] B.V. Dao, J. Duato, S. Yalamanchili, "Dynamically configurable message flow control for fault-tolerant routing," IEEE Transactions on Parallel and Distributed Systems vol.10, pp. 7–22, 1999

[6] W.J. Dally, C.L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," IEEE Trans. Computers, vol. 36, no. 5, pp. 547-553, 1987

[7] W.J. Dally, B. Towles, Principles and practices of interconnection networks, Morgan Kaufman Publishers, 2004

[8] W.J. Dally, B. Towles, "Route packets, not wires: On-chip interconnection networks," Proceedings. Design Automation Conference, pp. 684-689, Las Vegas, NV, USA, 18-21 Jun 2001

[9] J. Duato, S. Yalamanchili, L. Ni, Interconnection networks: An engineering approach, Morgan Kaufmann Publishers, 2003

[10] P. Guerrier, A. Greiner, "A generic architecture for on-chip packet-switched interconnections," Proceedings. Design Automation and Test in Europe Conference and Exhibition, pp. 250-256, Paris, France, 27-30 Mar. 2000

[11] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, and D. Lindqvist. "Network on chip: An architecture for billion transistor era," In Proc. of the IEEE NorChip Conf., pp. 120-124, Nov. 2000.

[12] A.E. Kiasari, H. Sarbazi-Azad, "Analytic performance comparison of hypercubes and star graphs with implementation constraints," Journal of Computer and System Sciences, vol. 74, iss. 6, pp. 1000-1012, Sep. 2008

[13] S. Kumar, A. Jantsch, M. Millberg, J. Oberg, J. Soininen, M. Forsell, K. Tiensyrj, and A. Hemani. "A network on chip architecture and design methodology," In Proc. Symposium on VLSI, pp. 117-124, April 2002.

[14] H. Matsutani, M. Koibuchi, Y. Yamada, A. Jouraku, H. Amano, "Non-minimal routing strategy for application-specific networks-on-chips," ICPP 2005, International Conference Workshops on Parallel Processing, pp. 273-280, 14-17 June 2005

[15] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," IEEE Tran. on Computers, vol. 26, pp. 62-76, Feb. 1993.

[16] G. Pfister, V. Norton "Hotspot contention and combining in multistage interconnection networks," IEEE Trans. Computers, vol. 34, no. 10, pp. 943-948, 1985.

[17] M. Rezazad, H. Sarbazi-Azad, "The Effect of Virtual Channel Organization on the Performance of Interconnection Networks," Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), 4-8 April 2005

[18] K. Srinivasan, K.S. Chatha, "A technique for low energy mapping and routing in network-on-chip architectures," ISLPED'05, pp. 387-392, San Diego, California, USA, 8-10 Aug. 2005

[19] P.H. Sui, S.D. Wang, "An improved algorithm for fault-tolerant wormhole routing in meshes," IEEE Trans. on Computers, Vol. 46, NO. 9, pp. 1040-1042, Sept. 1997

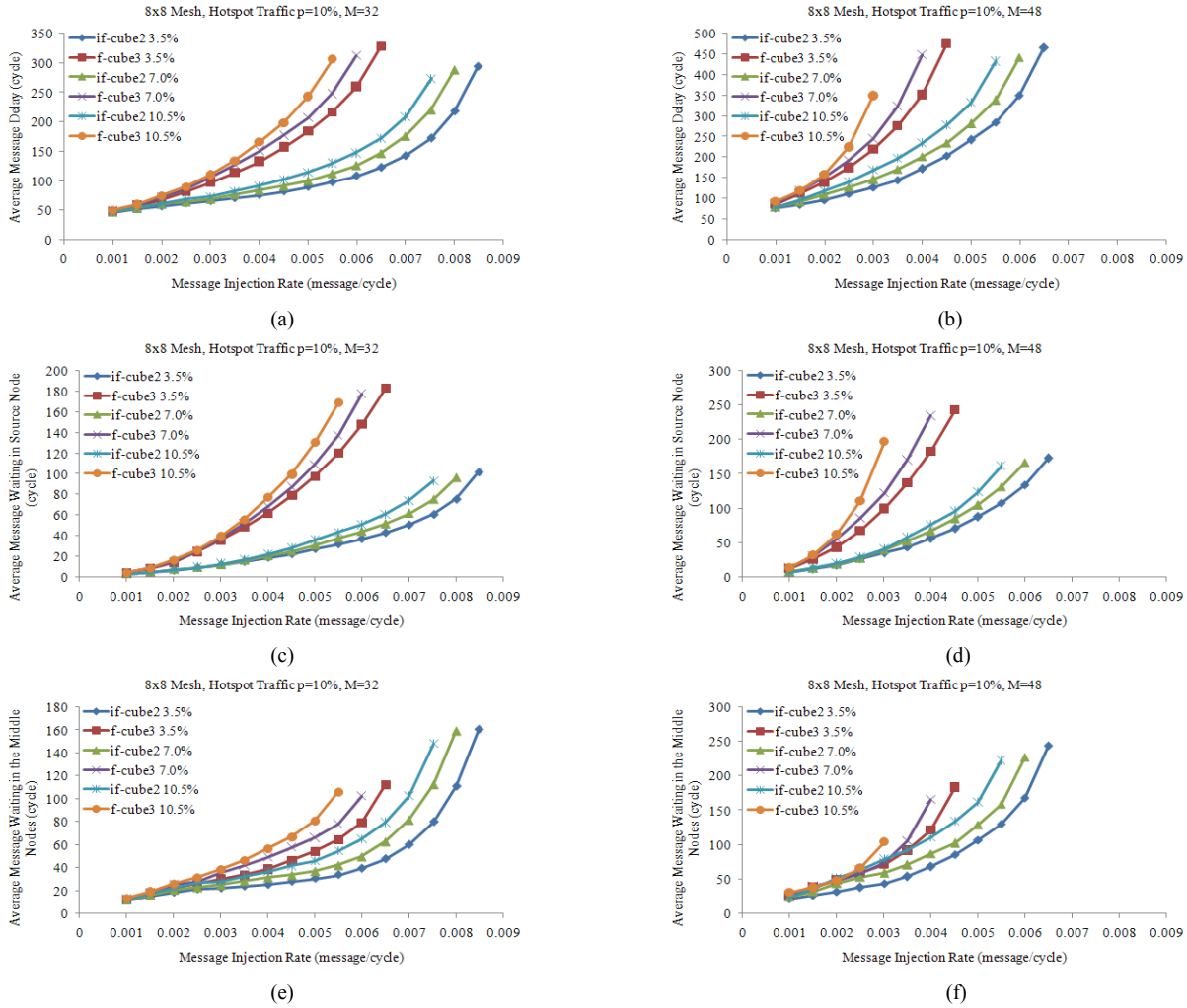

Figure 4. The Average Message Delay (AMD), a) 32 flits message, b) 48 flits message, Average Message Waiting in Source Node (AMWS), c) 32 flits message, d) 48 flits message, and Average Message Waiting in the Middle Nodes (AMWM) e) 32 flits message, f) 48 flits message; as a function of Message Injection Rate (MIR) in 2-D mesh in presence of hotspot traffic with p=10%.