IJICT International Journal of Information & Communication Technology

ITRC

Volume 2- Number 1- May 2010 (pp.45-51)

A Novel Scheme for Fault-Tolerant And Higher Capacity Network on Chip

Maryam Raiyat Aliabadi
IRAN Telecom Research Center
Tehran, IRAN
Mary_ra@itrc.ac.ir

Ahmad Khadem zadeh
IRAN Telecom Research Center,
Tehran, IRAN
Zadeh@itrc.ac.ir

Mohammad Raiyat Aliabadi
Semiconductor Research Center,
Tehran, IRAN
Raiyat2002@yahoo.com

Received: April 18, 2009- Accepted: February 8, 2010

Abstract- **As CMOS technology scales down, NoC (Network on Chip) gradually becomes the mainstream of on-chip communication. In this paper we present a methodology to design fault-tolerant routing algorithms for regular direct interconnection networks. It supports fully adaptive routing, does not degrade performance in the absence of faults, and supports a reasonably large number of faults without significantly degrading performance. Consequently, this work examines fault tolerant communication algorithms for use in the Communication Networks including NoC domain. Before two different flooding algorithms, a random walk algorithm and an Intermediate Node Algorithm have been investigated. The first three algorithms have an exceedingly high communication overhead and cause huge congestion in usual traffics. The fourth one which is Intermediate Node algorithm is a static fault-tolerant algorithm which focuses on the faults knowing in advance where they are located. We have developed a new dynamic algorithm based on intermediate node concept and stress value concept to overcome all of blind sides of mentioned algorithms. We have designed a switch/router base on this algorithm and simulated by MAX PLUS II tool and verified it on a mesh NoC in Xilinx environment.**

I. INTRODUCTION

A vast number of heterogeneous cores with varied communication requirements can be integrated into a single SoC (System on Chip) [9]. On-chip switching networks seem to be the scalable global interconnection solution to overcome the complexity of these systems [2][3][4]. Moving towards reconfigurability, scalability and efficient resource sharing on redundant platforms will increase the importance of fault tolerance on system-on-chip design [10][14]. A number of different fault tolerant methods of communication for large-scale systems have been proposed [5][12]. Previous work in this area is extremely limited as Network on Chip (NoC) design is still in its infancy. Work has begun on fault tolerant algorithms specifically for the NoC space, with most consisting of various forms of gossip algorithms [4]. DUMITRAS proposed a probabilistic flooding scheme based upon well known gossip techniques as a possible fault tolerant solution [6][7]. He have focused on stochastic communication which has been shown to cope well with random faults, requires no large lookup tables or reconfiguration, and requires very few retransmissions overall. Flooding is an effective fault tolerant technique because it is highly fault tolerant. If just one path exists to the destination, a message will almost certainly arrive. In practice, however, this level of fault tolerance may not be necessary. Resilience to a much smaller number of faults still offers increased chip yields, as well as resistance to transient failures, but also reduces unnecessary packet transmissions [10]. Lowering the number of unnecessary packet transmissions allows for higher network throughput, and a more efficient use of the interconnect .This paper investigates the previously proposed fault tolerance algorithms, then introduces a new one base on Intermediate Node concept in computer networks, and evaluates the

performance of the network to get the most efficient and fault-tolerant algorithm. To more accurately gauge the effectiveness of these algorithms in the NoC design space, we designed the components in VHDL, and used synthesis tools to determine the area requirements of the hardware. The remainder of this paper begins by discussing the algorithms and their implementation. We then compare the performance of the algorithms using a NoC and show that previous algorithms are not amenable for NoC implementation due to network congestion and transmission overhead especially in high packet rates.

II.PREVIOUS FAULT-TOLERANT ALGORITHMS

A: Probabilistic Flooding

The probabilistic flooding algorithm is a variant on a simple flooding protocol [4]. In this algorithm, every time a message is generated it will be passed to all of its neighbors with probability *p*, and will be dropped with probability *1- p*. *p* must be carefully chosen to obtain near flooding performance without sending as many redundant copies of the message [9]. Given enough time every node in the mesh will have received multiple copies of the message, with a high probability. This mode of communication is most susceptible to failure during its initial phase of transmission, when only a limited amount of nodes need not pass on the message. To be certain that the message does not die off during this period, an initial flooding phase of two cycles is performed to ensure that all functional nodes within two hops receive a copy of the message [6][7].

B: Directed Flooding

The probabilistic flooding algorithm is destination ignorant in that packets get routed irrespective of where they are in the network. The directed flooding algorithm makes use of a highly regular structure of NoC to direct a flood towards the destination [4]. In this algorithm, the probability *p* of passing a message to each outgoing link is not fixed, but instead varies based on the destination of the packet. More exactly the probabilities of a message being sent out on a given output ports are calculated as follows: First, the Manhattan distance Dc between the current node and the destination node is computed. This distance is also computed for all nodes adjacent to the current node, giving DN, DS, DW, and DE. A *multiplicative factor* MX is set to 1 for any direction where DX is greater than DC. For the remaining nodes the multiplicative factor MX is equal to the min (DX, 4). This min function prevents the scenario where a distant destination weights the multiplicative factors heavily, resulting in all packets following similar paths. Using a maximum multiplicative factor of 4 allows the most heavily favored ports to be followed more often and results in lower latencies, while still forwarding the packet on less favored ports an appreciable amount of the time. Finally, each of the multiplicative factors MX is multiplied by the *gossip rate, G* to find the probability of transmission to that particular link. *G* is a user-defined con constant that can be varied to control the level of performance versus fault-tolerance. This algorithm, like the probabilistic flood, has a statistical chance of not replicating a given message received. Therefore, there is an initial two cycle flood to seed the area, resulting in all functional nodes within two hops receiving a copy of the message [6][7].

C: Redundant Random Walk

The random walk algorithm sends a finite, predetermined number of copies of a message into the network. These messages follow a nondeterministic path, where every node that receives one of these messages must forward it to one of its output links. To simplify the creation of these additional redundant packets, an initial flooding stage can be used, just as in the other proposed algorithms. This initial flooding stage allows for a finite number of packets to be generated easily, while limiting the impact of any local faults. The choice of output link is determined by a set of random probabilities PN, PS, PW, PE, where the sum of all probabilities is 1. Therefore, every incoming message is paired with exactly one outgoing message, as in a standard network. Though normalization is often considered to be a large and slow operation, due to the limited number of possible values for the multiplicative factors, a reasonable approximation can be fashioned out of combinatorial hardware. The values for all probabilities (PN etc.) are computed as follows. First, the multiplicative factors MX are computed as in the discussion of directed flooding above. The multiplicative factors are then normalized to create the probabilities PN, PS, PW, PE. A random number is then generated, choosing one output port of the four to follow. Only one random decision unit is required per input port, rather than the multiple required for the directed flooding algorithm. This reduction in random decision units helps compensate for the additional logic required for the normalization process. In general, each message in the network will tend towards the same destination, but will follow slightly different paths to reach it. Thus, even if a few links on the way are broken, at least one message should arrive intact [4][11].

D: Intermediate Node

Intermediate Node Algorithm has been introduced in computer networks. It proposes a static fault-tolerant model and only focuses on the computation of the routing information for every source-destination pair, knowing in advance where the failures are located [1][16]. To introduce this algorithm, in what follows, we will assume a k*n network with minimal adaptive routing. An intermediate node I is used only if there exists at least one fault that can be encountered when routing packets from S (Source) to D (Destination) using adaptive routing. By appropriate selection of the intermediate node (I), packets will be routed from S to and then from I to D without encountering fault(s). Packets sent through intermediate nodes contain two destinations in their header. The first one corresponds

to the intermediate node and is removed there. The second one is the final destination of the packet.

When minimal adaptive routing is used, the intermediate node I should have the following properties so that the fault(s) Fi is (are) avoided when routing from S via I to D:

1) ∇i, Fi is not on any shortest path from S to I.
2) ∇i, Fi is not on any shortest path from I to D.
3) I is on at least one shortest path from S to D.

Let T0 be the set of nodes that can be traversed on any shortest path from S to D. Furthermore, let TSF be the set of nodes that can be traversed on any shortest path from S to Fi (for all i) and let TFD be the set of nodes that can be traversed on any shortest path from Fi (for all i) to D.

Theorem 1: A node N fulfills all three requirements if and only if it is in the set T0\ (TSF ∪ TFD) [1] [16]. Thus, when the set T0\ (TSF ∪ TFD) is non-empty, a suitable intermediate node can be selected among its members. By selecting the intermediate node this way, it is guaranteed that the path S-I-D yields minimal path routing from S to D and that adaptive routing can be used on each sub path without encountering a fault. If the set contains more than one node, the intermediate node can be selected randomly or based on some other criteria such as traffic balancing or routing flexibility.

A deterministic minimal routing function uses a subset of the paths returned by an adaptive minimal routing function [8]. Therefore, a node from the set T0\ (TSF ∪ TFD) can also be used as intermediate node when deterministic routing is used. However, there are scenarios where the set T0\ (TSF ∪ TFD) is empty but where it is still possible to find a suitable intermediate node if routing is restricted to a deterministic route, and (if required) by the use of non-minimal paths from S to D. In this way, nodes that could not be used as intermediate node with adaptive routing may be used as intermediate node with deterministic routing [1][16][8]. When fault-tolerance through intermediate nodes is applied in combination with deterministic routing we are looking for an intermediate node I such that:

1) ∇ i, Fi is not on the S - I deterministic path.
2) ∇ i, Fi is not on the I - D deterministic path.
3) There is no I' giving a shorter path than I.

Let TDRS be the set of nodes reachable through deterministic routing from S and TDD the set of nodes that have a valid deterministic route to D. Furthermore, let l(x, y) be the length of the minimal path from x to y in the fault free case. We then generalize the definition of T0, to Tj (for j>= 0), in the following way: A node N is in Tj if and only if l(S, N) + l (N, D) = l(S, D) + j. This way, Tj defines non-overlapping sets of nodes that can easily be identified by starting with T0 and continuing outwards.

Theorem 2: Let j' be the smallest j for which Tj ∩ TDRS ∩ TDD is non-empty. A node N fulfills all three requirements if and only if N∈Tj' ∩ TDRS ∩ TDD. This way, we start considering the shortest paths (j = 0), and then if necessary non-minimal paths (j > 0), to avoid the faulty link(s) [1][16].

Even when it is not possible to use minimal adaptive routing all the way from S via I to D (i.e. when the set T0 ∩ TDRS ∩ TDD is empty), it may still be possible to use adaptive routing from S to I or from I to D. In addition, when the intermediate node is selected outside T0, it may be possible to use adaptive routing both from S to I and from I to D. To identify these cases, let TRS ⊆ TDRS be the set of nodes reachable through any shortest path from S (i.e. without possibly encountering any Fi) and Td⊆ TDD the set of nodes that reach D through any shortest path. If the intermediate node is selected from Tj ∩ TRS ∩ TDD, adaptive routing can be used from S to I, whereas deterministic routing must be used from I to D. Similarly, if the intermediate node is selected from Tj ∩ TDRS ∩ Td, deterministic routing must be used from S to I, whereas adaptive routing can be used from I to D. If the intermediate node is selected from the set Tj ∩ TRS ∩ Td, adaptive routing can be used both from S to I and I to D [1] [16].

III. THE PROPOSED ALGORITHM

The first three algorithms result in significant extraneous transmission overhead, and as a result, limit overall network performance and throughput due to high congestion. The intermediate node algorithm is a static fault-tolerant one, so it has an important constraint against dynamic fault events [1][14][16]. As mentioned before, in intermediate node algorithm once a fault is detected, all the processes in the network are halted. To handle these problems and the other requirements (performance and load-balancing) we have proposed a new fault-tolerant methodology which has been developed based on two major concepts to meet above requirements. The basic concept is intermediate Node concept , but we developed this concept as a dynamic fault-tolerant model as it covers the faults occur at any point of time as well without halting the network and degrading the performance. The second concept is stress value which is used to remove congestion and balance the loads in the network. To make the load distribution more uniform, information to help the switches in their routing decision is sent between the switches. This informative value is called stress value or simply loads status. The simplest implementation of stress value is to count the number of packets switched and transmit the result to all the neighboring switches [15]. Assume that one switch is heavy loaded at a given time; the switch will send this stress value to the adjacent switches. The surrounding switches will then hesitate to transmit to the first switch during the next clock cycle. The stress value will then decrease to zero. In proposed algorithm the least congested path will be selected base on SV vector.

TABLE I- SV VECTOR EXPLENATION

SV vector	Meaning
00	no traffic
01	traffic is low
10	traffic is noticeable
11	traffic is high

Our proposed algorithm to fault-tolerant NoC design involves two major sections which have been implemented by three separated procedures shown in figure I.

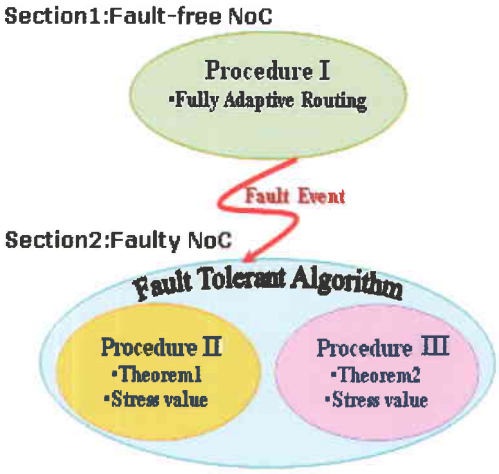

Figure I: The proposed NoC Routing Algorithm.

A. *Fault- Free NoC*

In fault free case (section 1), the fully adaptive routing procedure is running to transmit the messages through the shortest and least congested paths between S and D. To find the best options, Procedure I, operates on three criteria's which are SV vector, Traffic pattern type and path length. List of possible shortest paths from node X to node Y are in X-memory. After identifying the shortest paths it finds the least congested path using SV vector. Then it transmits the message (traffic pattern) which has higher priority via identified path.

B. *Faulty NoC*

If faults appear, fault-tolerant algorithm will be activated. It consists of two procedures which operate depending on the number of faults. Mentioned procedures use both intermediate node and stress value concepts as follows:

- If there are low faults in the network (as shown in figure II) which means the set of T0\ (TSF ∪ TFD) is non empty, then the procedure II will be activated. This procedure which is base on theorem1 of intermediate node concept, searches possible intermediate nodes in the mentioned set, selects one of them base on SV vector, then transmits the message through S-I and I-D sub-paths using fully adaptive routing methodology in each sub-path. So Procedure II transmits the messages via shortest & least congested paths which necessarily are fault free.

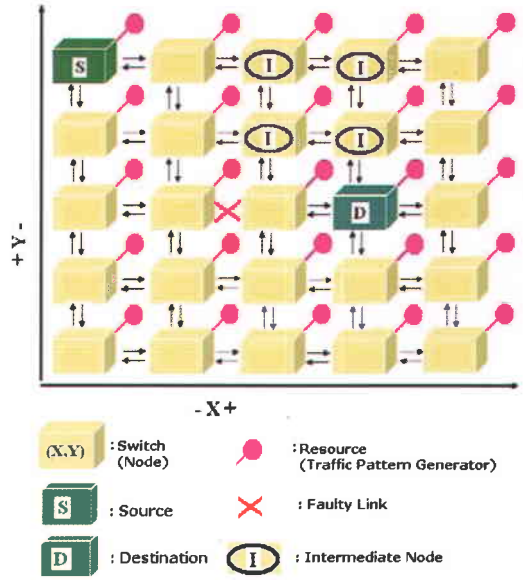

Figure II: Simple Scenario with low Faults (activates procedure II)

- If there are High faults in the network (as shown in figure III) which means T0\ (TSF ∪ TFD) = 0, the procedure III will be activated. This procedure which is based on theorem2 of intermediate node concept and stress value concept, finds possible intermediate nodes in the set of Tj' ∩ TDRS ∩ TDD, selects one of them base on SV vector, then transmits the message in S-I and I-D sub paths using X-Y deterministic routing in each sub path. So Procedure III transmits the messages via least congested paths. In this case since all the shortest paths are faulty the packets will be transmitted through minimal fault free paths.

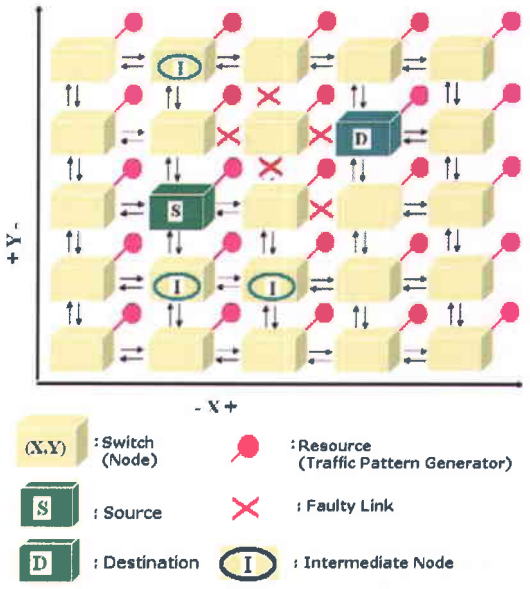

Figure III: More complex Scenario with high Faults (Activates procedure III)

IV. EXPREMENTAL RESULTS

We have designed a switch/router described in VHDL code and put it in a 5*5 mesh network. Each switch is connected to a resource which generates typical traffic patterns with different injection rates. We mention that the kinds of traffic patterns in a typical NoC with different appropriate packet length & different priority which we have used in this research are listed in TABLE II [12].

Implementation and evaluation of the network performance is done in XILINX environment. To evaluate the performance of the algorithms, we have used three separate metrics: fault tolerance, latency and area. For all simulations of previous algorithms, an injection rate (i.e. the proportion of nodes that generate messages per iteration) of 0.05% *packets/cycle/node* was used. The injection rate was chosen to be this low to allow for meaningful results for the flooding algorithms [14]. Higher injection rates quickly saturate the network when high levels of redundancy and flooding algorithms are used. But for proposed algorithm we increased injection rates up to 4.42% without encountering saturation.

A. *Fault Tolerance*

Fault tolerance evaluates how reliably each algorithm can route messages in despite transient and permanent faults. Transient faults are considered to have a percentage chance of disrupting a message during each hop, while permanent faults indicate nodes that are unable to send any packets [4]. The proposed algorithm focuses on permanent faults without knowing where they located, so it covers all of faults when they occur dynamically. We compared the fault tolerance and saturation threshold of four algorithms. Table II shows how percentage each algorithm can tolerate faults at 0.05% injection rate.

Table III compares the saturation threshold of three previous algorithms. The Probabilistic flooding algorithm saturates the network in injection rates higher than 0.05%. Directed flooding algorithm saturates the network in injection rates higher than 0.07%.Random walk algorithm saturates the network in injection rates higher than 0.2% while the proposed algorithm is still fault tolerant via resisting congestion. The other point is that the breakdown point of the proposed algorithm is at 18% permanent faults.

TABLE II- TRAFFIC PATTERN TYPE PRIORITIES

Pattern Type	Priority
Signaling	00
Real Time	01
RD/WR	10
Block transfer	11

TABLE II- FAULT TOLERANCE (%) For 0.05% Injection Rate base on the number of successful transfers at different permanent faults.

Algorithms	Permanent fault (%)					
	2	4	6	8	18	21
Probabilistic Flooding	96	96	94	92	82	80
Directed Flooding	96	96	96	95	84	80
Random Walk	90	80	72	66	72	38
Proposed algorithm	99.9	99.9	99.9	99.9	99.9	84.6

TABLE III- FAULT TOLERANCE (%) for 18% faults base on the number of successful transfers at different injection rates.

Algorithms	Injection rate				
	0.05	0.07	0.09	0.2	4.42
Probabilistic Flooding	82	sat	sat	sat	sat
Directed Flooding	84	69	sat	sat	sat
Random Walk	62	54	48	sat	sat
Proposed algorithm	99.9	99.9	98.2	94.6	68

Sat: saturation

B. *Latency*

Latency measures how quickly a message arrives at its destination. This depends on the number of hops that are required for a message to reach its destination, queue length and online switch computations [13]. The proposed algorithm choose the least congested ones, so queue length time is very low comparison to the others, but online switch computation is more time consuming than the flooding algorithms because of searching not only a suitable intermediate node but also least congested paths. The overall latency of algorithm is minimal in low injection rates but increases by increasing injection rate. Since the number of packets each switch should send increase, so packet queues become longer and the overall latency becomes more, while the other three algorithms have saturated the network. Our results indicate that latency mainly depends on path length, so we measured the mean path length for all algorithms in a range of meaningful injection rates and at the presence of 2% permanent faults. According to figure V the latency of probabilistic algorithm around 0.05% packet/cycle/node is averagely 17 and increases impulsively with higher injection rates. This situation is repeated for directed flooding and random walk algorithms in injection rates higher than 0.07% and 0.09% respectively, while the latency of proposed algorithm is still acceptable until in much higher injection rates (about 4.42%) which it tends to infinity.

TABLE IV
LATENCY BASE ON MEAN PATH LENGTH

Algorithms	*Mean path length*
Probabilistic Flooding(gos60)	17
Directed Flooding(gos60)	16
Random Walk	14
Proposed algorithm	13

C. *Area*

This metric evaluates the viability of the three algorithms based upon the implementation cost of each node [5]. Table V shows the relative area cost of each type of routing algorithm in a 70nm technology. In despite of high fault tolerance and low latency of the proposed algorithm, we have to pay the penalty of higher overhead area than the other ones because of truth tables stored in memory of control unit of each switch.

V. CONCLUSION

In this work, we compared four different fault tolerant communication algorithms for the Network on Chip domain. Our results indicate that flooding protocols, while simple and lightweight, rely on too many message replications to obtain their fault tolerance. A much more drastic savings in overhead can be obtained by using the proposed algorithm which is designed base on intermediate node and stress value concepts. We have designed an intelligent switch base on the proposed algorithm. In order to evaluating of the functionality of the switch and the performance of related algorithm we have implemented four mentioned fault-tolerant algorithms in a 5*5 mesh network and computed three critical parameters; fault-tolerance, latency and area. Our researches shows when the injection rate of entire packets is low the performance of four algorithms are closer, but by increasing the injection rate, three previous algorithms are saturated quickly because of redundant packets and their latencies tend to be infinite, while the proposed algorithm is still fault-tolerant and its latency is acceptable, so the proposed algorithm has the most capacity to transfer injected packets with the least congestion and without encountering saturation. It is also notable that the proposed algorithm can 99.96%≈100% tolerates 18% permanent faults.

TABLE V
AREA COMPARISON ACROSS ALGORITHMS

Algorithms	*Area per switch in 70nm*
Probabilistic Flooding(gos60)	0.121(sq.10)
Directed Flooding(gos60)	0.127(sq.10)
Random Walk	0.131(sq.10)
Proposed Algorithm	0.157(sq.10)

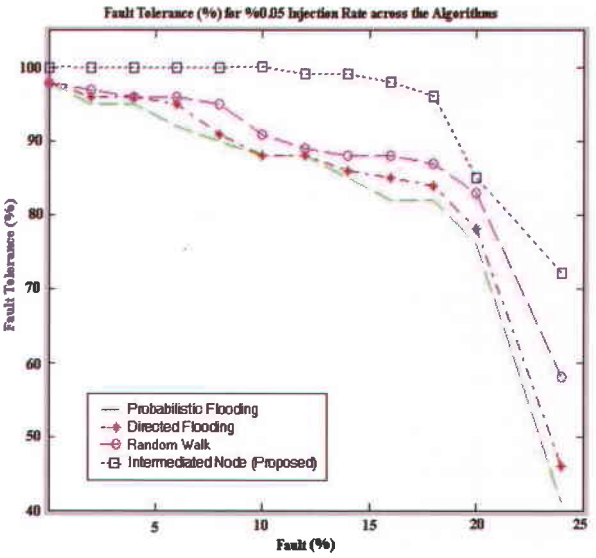

Figure VIII: Comparison of Fault Tolerance across the Algorithms

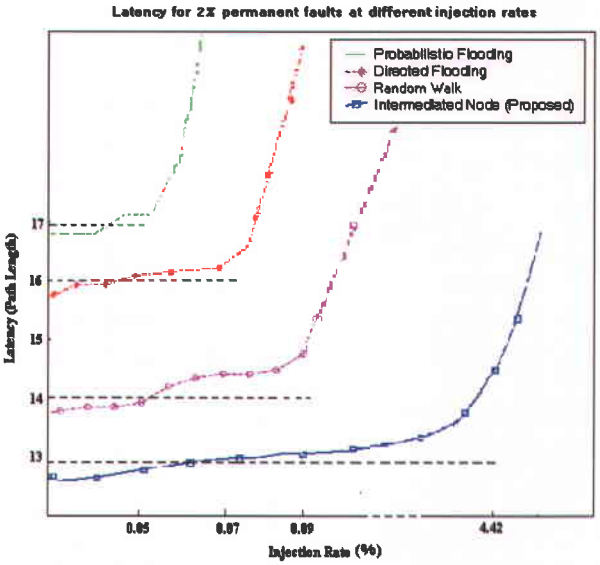

Figure V: Comparison of Latency across the Algorithms

REFERENCES

[1] M.E. Gomez, J. Duato, J. Flich, P. Lopez, and A. Robles "A Routing Methodology for Dynamic Fault Tolerance in Meshes and Tori", Springer Berlin / Heidelberg, High Performance Computing journal, Volume 4873/2007.

[2] L. Benini and G. D. Michele." Networks on chip: a new Paradigm for systems on chip design". In Proceedings Design, Automation and Test in Europe Conference and Exhibition, pages 418 –419, 2002.

[3] "The International technology roadmap for semiconductors", Semiconductor Industry Association, 2007.

[4] T. Dumitras, S. Kerner, and R. Marculescu." Towards on chip fault-tolerant communication". In Proc. Asia and South Pacific Design Automation Conference, 2003.

[5] Y. Hatanaka, M. Nakamura, Y. Kakuda, and T. Kikuno. "A synthesis method for fault-tolerant and flexible multipath routing protocols". pages 96–105, 1997.

[6] S. M. Hedetniemi, T. Hedetniemi, and A. L. Liestman." A survey of gossiping and broadcasting in communication networks" .NETWORKS, 18:319–349, 2000.

[7] D.W. Kru10e, G. Cybenko, and K. N. Venkataraman. "Gossiping
in minimal time". SIAM J. Comput., 21(1):111–139,2002.

[8] LI Xiaohui1, CAO Yang1, 2, WANG Liwei1, CAI Tian1," Fault-Tolerant Routing Algorithm for Network-on-Chip Based on

Dynamic XY Routing", State Key Laboratory of Software Engineering, Wuhan University, Journal of natural Sciences, Vol. 14 No. 4, 343-348, 2009.
[9] Kuei-Chung Chang," Reliable network-on-chip design for multi-core system-on-chip", The Journal of Supercomputing, Springer Netherlands, 2009.
[10] "A New Approach to Single Event Effect Tolerance Based on Asynchronous Circuit Technique" , Journal of Electronic Testing , Springer Netherlands, Volume 24, Numbers 1-3 / June, 2008
[11] A. Papoulis and S. U. Pillai. "Probability, Random Variable sand Stochastic Processes". McGraw Hill, 2002.
[12] Sridhara S R, Shanbhag N R.," Coding for reliable on-chip buses: A class of fundamental bounds and practical codes", IEEE Trans. CAD, 26(5): 977–982, 2007.

[13] M. Pirretti, G. M. Link, R. R. Brooks, N. Vijaykrishnan, M. Kandemir, M. J. Irwin,"Fault Tolerant Algorithms for Network-On-Chip Interconnect", the proceedings of ISVLSI, February 2004.
[14] Praveen Vellanki, Nilanjan Banerjee, Karam S. Chatha ," Quality-of-service and error control techniques for mesh-based network-on-chip architectures" ,Department of CSE, Arizona State University, the VLSI journal 38 (2005) 353–382,2005.
[15] Erland Nilsson, Mikael Millberg, Johnny O¨ berg, and Axel Jantsch, "Load distribution with the Proximity Congestion Awareness in a Network on Chip ", Laboratory of Electronics and Computer Systems / Royal Institute of Technology (KTH), 2007
[16] M.E. G´omez, J. Flich, P. L´opez, A. Robles, and J. Duato, "An Effective Fault-Tolerant Routing Methodology for Direct Networks" Dept. of Computer Engineering Universidad Polit´ecnica de Valencia Camino de Vera, 14, 46071–Valencia, Spain, 2003.

Maryam Raiyat Aliabadi was born in Kashan, Iran, in 1978. She received the B.Sc degree from Shahid Beheshti University (SBU), Tehran-Iran, and the M.Sc degree from Islamic Azad University, Tehran-Iran, in 2001 and 2005 respectively both in electronic engineering. She has been working as a researcher in Iran Telecom Research Center since 2001 in different fields. Her research interests are Computer Aided Design, Network on Chip Reliability, Virtualization Technology, Data Center Design, Enterprise Architecture and Information Technology Applications.

Ahmad Khadem Zadeh was born in Meshed, Iran, in 1943. He received the B.Sc. degree in applied physics from Ferdowsi University, Meshed, Iran, in 1969 and the M.Sc., Ph.D. degrees respectively in Digital Communication and Information Theory & Error Control Coding from the University of Kent, Canterbury, U.K. He is currently the Head of Education & National Scientifiic and International Scientific Cooperation Department at Iran Telecom Research Center (ITRC) He was the head of Test Engineering Group and the director of Computer and Communication Department at ITRC. He is also a Lecturer at Tehran Universities & he is a committee member of the Iranian Electrical Engineering Conference Permanent Committee. Dr. Khadem Zadeh has been received four distinguished national and international awards including Kharazmi International Award, and has been selected as the National outstanding researcher of the Iran Ministry of Information and Communication Technology.

Mohammad Raiyat Aliabadi received the B.Sc. degree in Electrical and Electronics Engineering from the university of Kashan , Kashan, Iran , 2004 and M.Sc. degree in Electronic Engineering from Iran University of Science & Technology (IUST), Tehran , Iran , 2008. His research interests include System on Chip Implementation, Design Automation, Surface Acoustic Wave sensor design and Semiconductors Manufacturing Technologies.