© 2018 IEEE.Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses,in any current or future media, including reprinting/republishing this material for advertising or promotional purposes,creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other Works. DOI: 10.1109/TENCON.2018.8650056

A Novel Fault-Tolerant Routing Technique for Mesh-of-Tree based Network-on-Chip Design

Mohit Upadhyay*, Monil Shah*, P. Veda Bhanu*, Soumya J*, Linga Reddy Cenkarmaddi†, and Henning Idsøe†
*Department of EEE, Birla Institue of Technology and Science-Pilani, Hyderabad, Telangana, India - 500078
*{mupadhyay09@gmail.com, shahmonil1996@gmail.com, vedabhanuiit2010, soumyatkgp}@gmail.com
†Department of Information and Communication Technology, University of Agder, Norway
†{linga.cenkeramaddi, henning.idsoe}@uia.no

***Abstract*—Due to the increase in the number of processing elements in System-on-Chips (SoCs), communication between the cores is becoming complex. A solution to this issue in SoCs gave rise to a new paradigm called Network-on-Chips (NoCs). In NoCs, communication between different cores is achieved using packet based switching techniques. In the deep sub-micron technology, NoCs are more susceptible to different kinds of faults which can be transient, intermittent and permanent. These faults can occur at any component of NoCs. This paper presents a novel Fault-Tolerant Routing (FTR) technique for Mesh-of-Tree (MoT) topology in the presence of router faults. The proposed technique is compared with routing technique without any faults. The results show improvements interms of the number of data packets reaching to any given destination node from any source node in MoT network in presence of faults.**

***Keywords*—System-on-Chip, Network-on-Chip, Fault-Tolerance, Mesh-of-Tree Topology, Routing.**

I. Introduction

Over the years according to Moore's law, the number of transistors on a chip are increasing exponentially [1]. Due to this, the number of Integrated Circuits (ICs) integrated on a single chip are increasing. This has led to increased communication complexities in a System-on-Chip (SoC) [2] communication architecture. Besides the communication infrastructure, there are many challenges faced during SoC design such as signal integrity, increase in delay due to coupling capacitances, cross talk effects [3]. This has motivated many researchers in the industry and academia to find the communication backbone of many-core based SoCs to meet the inter-core communication demands. Network-on-Chip (NoC) [4] has been found to be a viable alternative for designing modular and scalable communication architecture. NoC consists of three components namely Network Interfaces (NIs), Routers or Switches and Links. In NoC, the communication between different IP cores is achieved through packet-based switching technique [5].

In the deep submicron technology, ICs are always limited by random fabrication defects which are impossible to eliminate even in the best manufacturing process. These defects lead to three different kinds of faults namely intermittent, transient and permanent faults [6]. In this work, we have designed a reliable and efficient fault-tolerant NoC while considering permanent faults occurred in routers. In [7], they have reported about different topologies used in an NoC and their advantages. In [8], 3-D based topologies have been presented which increases the parallelism. Most of the routing techniques reported in the literature are for standard topologies like Mesh and Torus. A review of different routing algorithms in NoC with advantages and disadvantages have been summarized in [9]. The Mesh-of-Tree (MoT) based topology has many advantages like small diameter, small router degree, large bisection width along with a symmetric and recursive structure when compared to the direct network topologies like mesh or torus [10]. This has motivated us to design an efficient fault-tolerant routing technique for MoT topology based NoC. We have used the same addressing scheme reported in [11]. The rest of the paper is organised as follows. Section II gives a brief overview of the MoT topology and its addressing scheme. Section III describes the fault-free routing algorithm. Section IV describes the proposed fault-tolerant routing algorithm. Section V gives a comparison of both the algorithms. Section VI recites the experimental results followed by conclusion.

II. Overview

This section gives an idea about the structure of MoT topology and its advantages. The addressing scheme of the MoT structure is discussed in the section II.B.

A. Mesh-of-Tree Structure

The MoT topology is a hybrid interconnect network. The properties of MoT topology [9] have been described below.

Considering an *A* x *B* MoT structure (where *A* and *B* denotes the number of row trees and column trees respectively) has the following properties:

- Number of Routers = 3 * (A * B) - (A + B).
- Diameter = $2 * (log_2 A) + 2 * (log_2 B)$.
- Bisection width = min (A, B).
- Symmetric and recursive structure.

For an A x B MoT structure, the number of routers are as follows:

- Number of leaf router = A * B.
- Number of stem router = 2 * (A + B).
- Number of root router = A + B.

The Fig. 1 shows an 4 X 4 MoT structure, having 4 row trees and 4 column trees. The leaf routers are attached to both the trees. Two cores are connected to each of the leaf router.

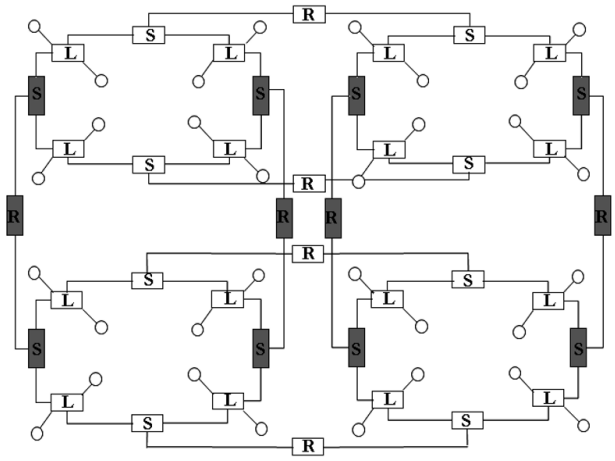

Fig. 1. 4x4 MoT Structure along with the cores attached

B. Addressing Scheme

In a MoT, the each router address consists of four different fields: (i) Row Number (R^N) (ii) Column Level (C^L) (iii) Column Number (C^N) (iv) Row Level (R^L). For every row tree, the value of R^N is fixed; thus, for a 4 X 4 MoT, R^N values are 00,01,10 and 11. The R^L values are gradually incremented by 1 from the leaf level to the root level of a row tree. In a row tree, C^L is 00 for all the routers. C^N is assigned as 00,01,10 and 11 for the column trees. For a column tree, the C^Lvalues are incremented by 1 from the leaf level to the column level. The R^L values for all column tree routers is equal to 00. A core has the same address as its associated router with one exception that an additional Core-ID bit is introduced for each core at the same leaf router.

III. Routing Algorithm

The routing algorithm follows a deterministic approach. Due to this, it is always ensured that the data packet reaches its destination through the specified shortest path. This ensures that the proposed network is always livelock free. This algorithm has been proposed in [11]. The following abbreviations are used to describe the algorithm.

- adrs_curr is used to denote current router address
- adrs_dest is used to denote the destination router address

Each leaf and stem router executes the algorithm reported in [11]. In root routers, no routing is performed and routers are replaced by First In First Out (FIFO) buffers. The routing algorithm is given as Algorithm 1.

In the Part I of the above algorithm, if there is a difference in the R^N of current and destination addresses it signifies that the current and the destination routers are at different row trees. Therefore the packet is to be routed to the root of the column tree so that R^N of the current router becomes equal to that of the destination router. For example, if router address of the source router and the destination router are 00-00-00-00 and 11-00-11-00. The path traversed by the packet is 00-00-00-00, then 0X-01-00-00 then XX-10-00-00. Therefore, after

Algorithm 1 Non-FTR Algorithm

Input: Address of the Source router (R^N, C^L, C^N, R^L), and the Destination router (R^N, C^L, C^N, R^L).
Output: Shortest Path from Source Router (R^N, C^L, C^N, R^L) to Destination Router (R^N, C^L, C^N, R^L).

```
if R^N of adrs_curr ≠ R^N of adrs_dest then
    Route to the Column Parent;                                  Part I
else if C^L of adrs_curr ≠ C^L of adrs_dest then
    Route to the Column Child having equal R^N as adrs_dest;     Part II
else if C^N of adrs_curr ≠ C^N of adrs_dest then
    Route to the Row Parent;                                     Part III
else if R^L of adrs_curr ≠ R^L of adrs_dest then
    Route to the Row child having equal C^N as adrs_dest;        Part IV
else if Destination Core-ID = 0 then
    Route to Core 1;
else
    Route to Core 2;                                             Part V
end if
```

Part I the packet will reach whose $R^N$is same as the destination router.

In *Part II* of the algorithm, if there is a difference in C^L of the current and destination addresses are found, it signifies that the current router is not at the leaf level, as all destination cores are at leaf level. Therefore, the packet is to be forwarded to the leaf level of the column tree for which R^N is equal to the destination router. For the above example, the path that is traversed by the packet is XX-10-00-00, then 1X-01-00-00 and then 11-00-00-00. So, after *Part II* the packet reaches a router whose R^N and C^L are same as those of destination router. So, the packet is at the same row tree as the destination router.

In *Part III*, a difference in C^N of the current and destination routers signifies that the current and destination routers are at different column tree levels. So, the packet should be forwarded towards the root of the row tree until C^N of current router is equal to that of the destination. For the above example, path traversed by the packet is 11-00-00-00, then 11-00-0X-01 and then 11-00-XX-10. So, after *Part III* the packet reaches a router whose first three fields are the same as the destination.

In *Part IV*, due to a difference in R^L of current and destination router, it is clear that the current router is not a leaf router. Therefore, the packet should be traversed towards the leaf level of the row tree whose C^N is same as that of the destination router. For the above given example, the path followed is 11-00-XX-10, then 11-00-1X-10 and then 11-00-11-00. So, after *Part IV* the packet reaches the destination router where the destination core is present.

In *Part V*, based on the Core-ID bit, the packet is forwarded to the destination core. So, in this way the proposed algorithm always governs the packet to reach the destination in a specified path.

This routing algorithm does not take faults into account. Hence, it fails when a fault is introduced in the system.

IV. Proposed Fault-Tolerant Routing Algorithm

The proposed fault-tolerant algorithm also follows the deterministic approach. The algorithm uses the same addressing scheme given in the section II.B. The proposed algorithm is given as Algorithm 2.

Algorithm 2 FTR Algorithm

Input: Address of the Source router (R^N, C^L, C^N, R^L), Address of the Destination router (R^N, C^L, C^N, R^L) and Address of the Faulty router (R^N, C^L, C^N, R^L)

Output: Path from Source router (R^N, C^L, C^N, R^L) to Destination router (R^N, C^L, C^N, R^L) .

```
if C^N of adrs_curr - C^N of adrs_dest ≤ 1 and C^N of adrs_curr = C^N of adrs_fault then
    check-even-odd (C^N);
    equate-row (adrs_curr, adrs_dest)
    equate-col (adrs_curr, adrs_dest)
else if R^N of adrs_curr - R^N of adrs_dest ≤ 1 and R^N of adrs_curr = R^N of adrs_fault then
    check-even-odd (R^N)
    equate-col (adrs_curr, adrs_dest)
    equate-row(adrs_curr,adrs_dest)
else if C^N of adrs_curr = C^N of adrs_dest then
    equate-col (adrs_curr, adrs_dest);
    equate-row (adrs_curr, adrs_dest);
else
    equate-row (adrs_curr, adrs_dest);
end if
if R^N of adrs_curr ≠ R^N of adrs_fault then
    equate-col (adrs_curr, adrs_dest);
else
    check-even-odd (R^N);
    equate-col (adrs_curr, adrs_dest);
    equate-row (adrs_curr, adrs_dest);
end if
if Destination Core-ID = 0 then
    Route to Core 1;
else
    Route to Core 2;
end if
```

In our Algorithm, there are three functions implemented they are check-even-odd (addr), equate-row (addr1, addr2) and equate-col (addr1, addr2). The function check-even-odd (addr) is a user defined function which checks whether argument addr is even or odd, then it increments the argument if it is even and decrements if it is odd. The function equate-row (addr1, addr2) is equal to the combination of Part I and Part II mentioned in the algorithm 1. It brings the packet from the addr1 to the same row tree as addr2. The function equate-col (addr1, addr2) is equal to the combination of Part III and Part IV mentioned in algorithm 1. It brings the packet from the addr1 to the same column tree as addr2.

There are two main parts in the algorithm proposed in [11]. Initial part will equate row numbers (R^N) between current router and destination router. Similarly in the second part of the algorithm [11] it will equate column numbers (C^N) between current router and destination router. Our algorithm uses these two parts independently depending upon the case. In our algorithm, it checks whether the fault is in row tree or column tree. If the fault is in a column tree i.e., column tree as the current router, the algorithm first makes the column numbers equal and then it makes the row number equal. This simply means that the packet is routed to go through the row root first and later on to column root. If the fault is in a row tree, the packet is manually re-routed to the adjacent row tree and then the packet is passed to go through the adjacent row root.

In the next section, we give a brief comparison between Algorithm proposed in [11] and our Algorithm. We will demonstrate the working of our algorithm by using a few examples.

V. Comparison of Proposed Algorithm with the Previous Algorithm

This section does a comparison of Algorithm proposed in [11] and our Algorithm by using few examples. The example shown in Fig. 2 is the case when the source router address is 00-00-00-00, the destination router address is 11-00-11-00. The fault address is taken to be XX-10-00-00, then the C^N of the current/source router and the fault router are the same. So, the packet will now go through the row tree and then the column tree. So, the path traversed by the packet will be 00-00-00-00, 00-00-0X-01, 00-00-XX-10, 00-00-1X-01,00-00-11-00, 0X-01-11-00, XX-10-11-00, 1X-01-11-00, 11-00-11-00.

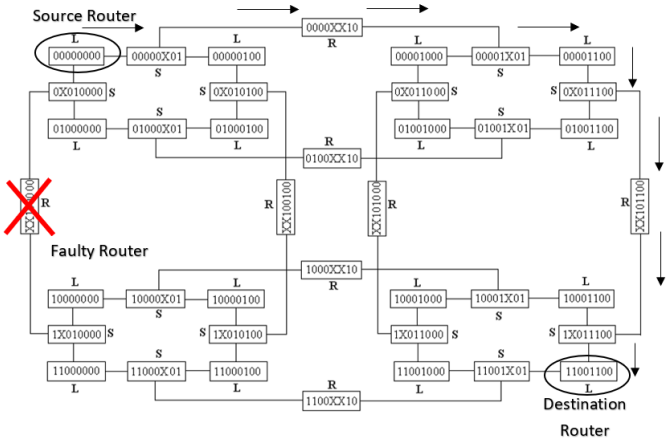

Fig. 2. Source Router at 00000000, Destination Router at 11001100 and Fault Router at XX100000

Algorithm proposed in [11] which is non fault-tolerant (non-FTR) fails at the stem level. According to the algorithm, the packet will first start at the leaf router 00-00-00-00, and then it will move on to the stem whose address is 0X-01-00-00. After the packet reaches the stem, the algorithm will try to

route the packet through the root whose address is XX-10-00-00. But, in this example this particular root is faulty and so, it cannot be used for communication. Hence it fails to reach the destination.

We take another example shown in Fig. 3 where the source, destination and the fault router addresses are 00-00-00-00, 11-00-11-00 and 11-00-XX-10 respectively. In this case according to our Algorithm, the packet follows the path 00-00-00-00, 0X-01-00-00, XX-10-00-10, 1X-01-00-00, 10-00-00-00, 10-00-0X-01, 10-00-XX-10, 10-00-1X-01, 10-00-11-00, 1X-01-11-00, 11-00-11-00.

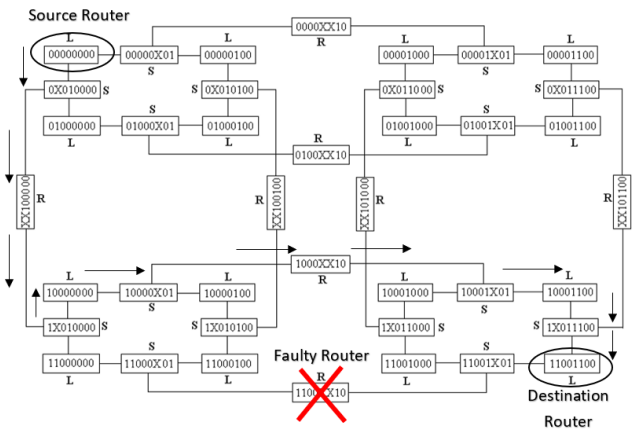

Fig. 3. Source Router at 00000000, Destination Router at 11001100 and Fault Router at 1100XX10

In the example shown in Fig. 3, we can see that the Algorithm proposed in [11] will fail after it reaches the row tree root. According to the algorithm, the packet starts at the router 00-00-00-00, then it moves to 0X-01-00-00, then to XX-10-00-00 and then to 1X-01-00-00 and moves on to 11-00-00-00. Here the algorithm reported fails as it will try to move the packet towards to 11-00-0X-01 and then to the row root 11-00-XX-10, but this is not possible as this row root is faulty. Hence, the routing algorithm fails at the root level.

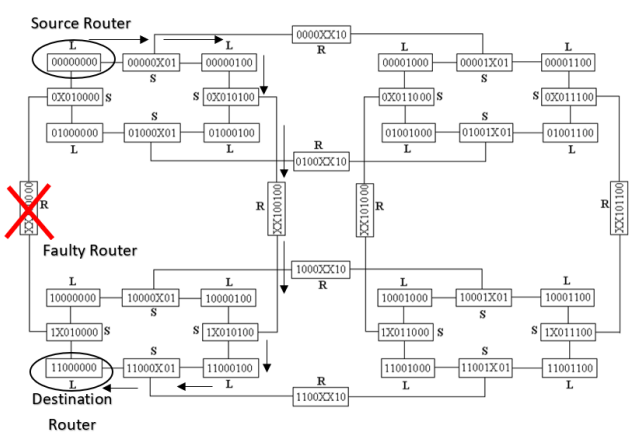

Fig. 4. Source Router at 00000000, Destination Router at 11000000 and Fault Router at XX100000

Similarly, the example shown in Fig. 4 is the case when the source router address is 00-00-00-00, the destination router address is 11-00-00-00. The fault address is taken to be XX-10-00-00, then the C^N of the current/source router and the fault router are the same. So, the packet will now go through the adjacent column tree. So, the path traversed by the packet will be 00-00-00-00, 00-00-0X-01, 00-00-01-00, 0X-01-01-00,XX-10-01-00, 1X-01-01-00, 11-00-01-00, 11-00-0X-01, 11-00-11-00.

For this example, algorithm proposed in [11] fails at the column root itself. The algorithm will route the packet to 00-00-00-00, and then to 0X-01-00-00 and then to the column root XX-10-00-00. But, in this case the column root is faulty. Hence, the routing is incorrect. Our proposed algorithm routes the packet as shown in Fig. 4.

As we can observe from all the examples (a,b and c) the packet routes the packets successfully to their respective destination routers.

VI. Experimental Results

In this section, we compare both the algorithms by varying the size of the network and also by varying the percentage of faults.

Here, we have assumed that the fault can occur only in root routers. However, stem faults and leaf faults will be considered as futurescope of the work. We have compared algorithm [11] and our algorithm by varying the percentage of faults in the MoT network and the percentage of packets reaching the destination incase of faults. In all those cases algorithm proposed in [11] simply stalls due to a fault in its way, most of those cases are taken care of when the packet is routed using our Algorithm. In case of our proposed algorithm, the faulty router is completely bypassed by taking another path. As we can observe from Fig. 5, 6 and 7 even when the number of faults are increased, more packets have reached their respective destination routers when they are routed using our algorithm.

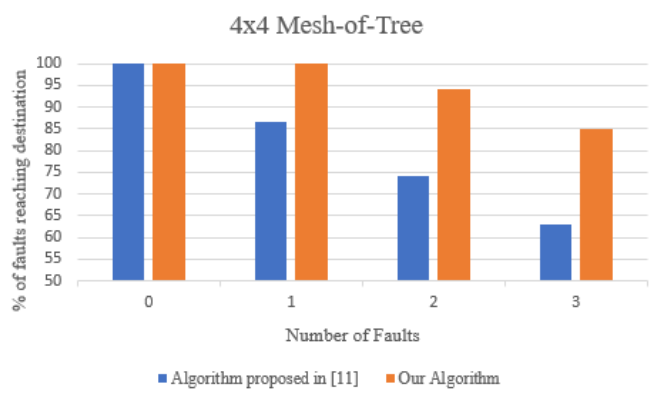

Fig. 5. Performance Variation on varying the number of faults in 4x4 Mesh-of-Tree Structure

According to the graphs shown in Fig. 5, 6 and 7, the number of faults were increased in both the cases, percentage of packets reaching their respective destination routers were higher when they were routed according to our algorithm

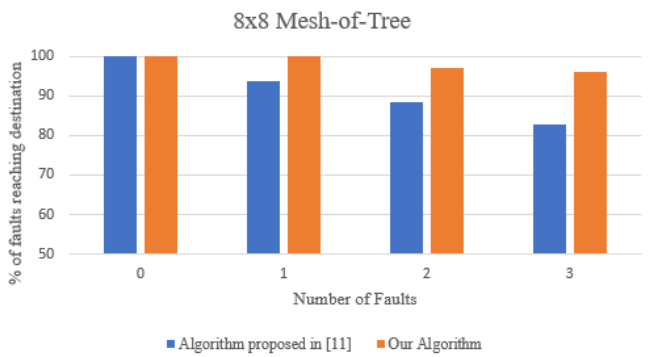

Fig. 6. Performance Variation on varying the number of faults in 8x8 Mesh-of-Tree Structure

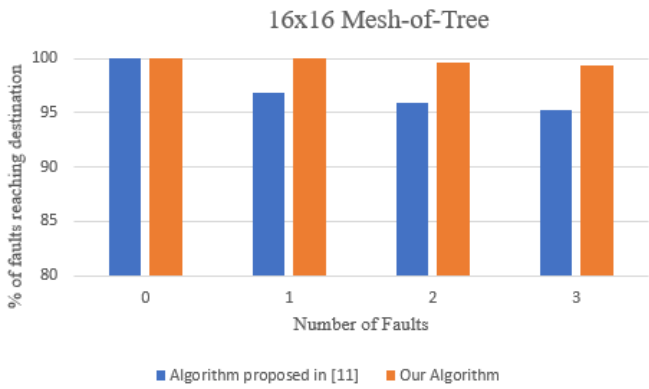

Fig. 7. Performance Variation on varying the number of faults in 16x16 Mesh-of-Tree Structure

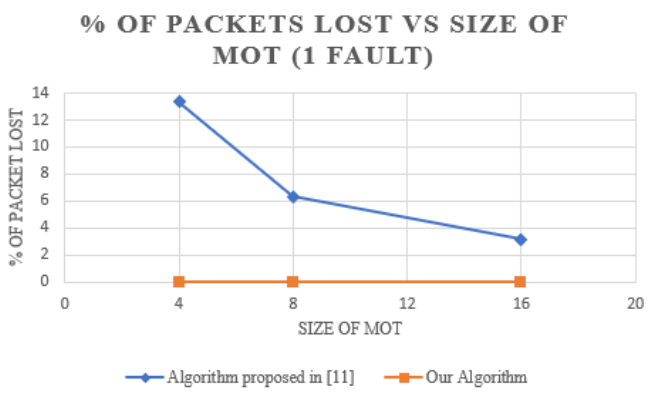

Fig. 8. Performance Variation with respect to Size of Network

in comparison to when they were routed using algorithm proposed in [11].

The scalability of our Algorithm is demonstrated in Fig. 5,6 and 7, where it can be seen that the algorithm can be scaled up to 16x16, 8x8 structure and besides the implementation in the original 4x4 MoT structure.

Using these results, it can also be seen that our algorithm is able to route data packets even when there are faults in the root level, and also that the scalability of the algorithm has

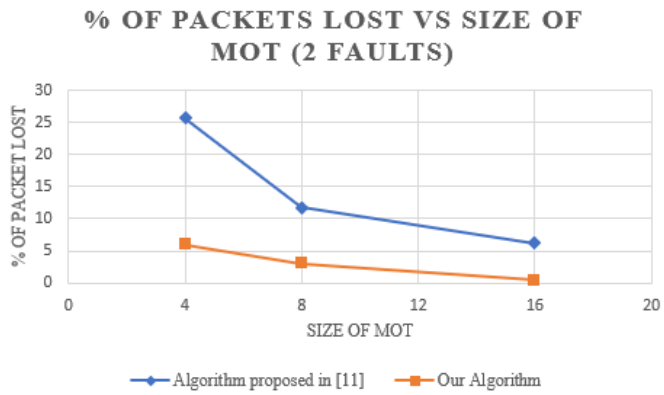

Fig. 9. Performance Variation with respect to Size of Network

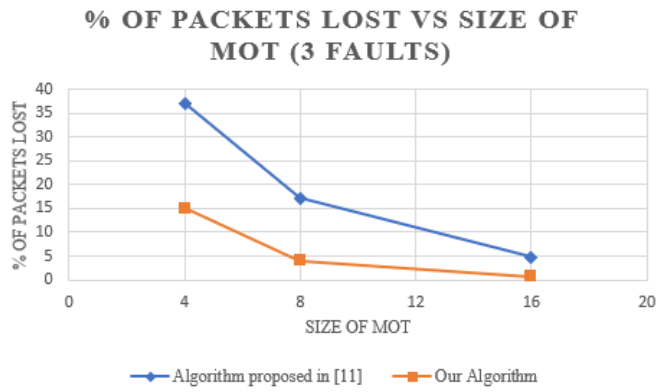

Fig. 10. Performance Variation with respect to Size of Network

also been tested here.

According to Fig. 8, it can be seen that in the case when onle one router fails in the network, there is no loss of packets in this case. This shows the superiority of our algorithm compared to the algortihm proposed in [11]. In the Fig. 8, 9 and 10, it can be seen that when the packets are routed using our routing algorithm loss of packets is significantly lower in comparison to the packets which are routed using the routing algorithm proposed in [11].

According to Fig. 8,9, and 10, it can be seen that percentage of packets lost reduces as the size of the network increases. This is due to the fact that faults are assumed only at the root level. As the size of the network is increased, the communication between the leaves increases as faults are not assumed at the lower levels, so this leads to increase in the communication between the leaves because the packets can be transmitted using the lower level routers, rather than the root router where the faults are assumed to occur. This is the reason why the packet loss reduces with increase in network size.

VII. Conclusion

We have implemented a fault-tolerant routing algorithm for MoT topology based NoC. The results show significant

improvement in sending the packets to the respective destination routing when using our routing algorithm compared to the routing algorithm reported in the literature. Our future work includes extending our algorithm to faulty stem and leaf routers.

ACKNOWLEDGEMENTS

This work is partially supported by the research project No. ECR/2016/001389, Dt. 06/03/2017, sponsored by the SERB, Govt. of India.

REFERENCES

[1] G. E. Moore, "Cramming more components onto integrated circuits, reprinted from electronics, volume 38, number 8, april 19, 1965, pp.114 ff." *IEEE Solid-State Circuits Society Newsletter*, vol. 11, no. 3, pp. 33–35, Sept 2006.

[2] "International technology roadmap for semiconductors," Tech. Rep., 2015.

[3] Y.-H. H. Wen-Chung Tsai, Ying-Cherng Lan and S.-J. Chen, "Networks on chips: Structure and design methodologies," *Journal of Electrical and Computer Engineering*, Oct. 2011.

[4] L. Benini and G. D. Micheli, "Networks on chips: a new soc paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, Jan 2002.

[5] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, 2001, pp. 684–689.

[6] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Comput. Surv.*, vol. 46, no. 1, pp. 8:1–8:38, Jul. 2013. [Online]. Available: http://doi.acm.org/10.1145/2522968.2522976

[7] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, Aug 2005.

[8] H. Naghibi Jouybari and K. Mohammadi, "A low overhead, fault tolerant and congestion aware routing algorithm for 3d mesh-based network-on-chips," *Microprocess. Microsyst.*, vol. 38, no. 8, pp. 991–999, Nov. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.micpro.2014.09.005

[9] V. Rantala, T. Lehtonen, and J. Plosila, "Network on chip routing algorithms," 2006.

[10] F. T. Leighton, *Introduction to Parallel Algorithms and Architectures*, 1st ed. San Mateo, CA: Morgan Kaufmann, 1992.

[11] S. Kundu and S. Chattopadhyay, "Network-on-chip architecture design based on mesh-of-tree deterministic routing topology," *Int. J. High Perform. Syst. Archit.*, vol. 1, no. 3, pp. 163–182, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1504/IJHPSA.2008.021797