Fault-Tolerant Network-on-Chip Design for Mesh-of-Tree topology using Particle Swarm Optimization

P. Veda Bhanu*, Pranav Kulkarni*, Sarthak Jain*, Soumya J*, Linga Reddy Cenkarmaddi†, and Henning Idsøe†
*Department of EEE, Birla Institute of Technology and Science-Pilani, Hyderabad, Telangana, India - 500078
*{vedabhanuiit2010, kulkarni.pranav2, jainsarthak3, soumyatkgp}@gmail.com
†Department of Information and Communication Technology, University of Agder, Norway
†{linga.cenkeramaddi, henning.idsoe}@uia.no

***Abstract*—As the size of the chip is scaling down the density of Intellectual Property (IP) cores integrated on a chip has been increased rapidly. The communication between these IP cores on a chip is highly challenging. To overcome this issue, Network-on-Chip (NoC) has been proposed to provide an efficient and a scalable communication architecture. In the deep sub-micron level NoCs are prone to faults which can occur in any component of NoC. To build a reliable and robust systems, it is necessary to apply efficient fault-tolerant techniques. In this paper, we present a flexible spare core placement in Mesh-of-Tree (MoT) topology using Particle Swarm Optimization (PSO) by considering IP core failures in NoC. We have experimented by considering several application benchmarks reported in the literature. Comparisons have been carried out, (i) by varying the percentage of faults in the MoT network with fixed network size and (ii) by taking the failed core as an input from the user. The results show limited overhead in communication cost while providing fault-tolerance.**

***Index Terms*—Network-on-Chip, Mesh-of-Tree Topology, Fault-Tolerance, Communication cost, Spare core.**

I. Introduction

The recent developments in the field of VLSI gave rise to drastic reduction in the size of the components being fabricated on a chip, which in turn leads to an increase in the integration density of the components on a chip. The traditional bus based communication architecture on a chip is not scalable and can no longer efficiently handle the high inter-core data rates. To overcome these limitations, Network-on-Chip (NoC) has been proposed in [1]. In NoC the communication among different IP cores is achieved through packet based switching technique. The major components of an NoC are Network Interfaces (NIs), Switches or Routers and Interconnection links. As the size of the chip is being reduced, the NoCs are prone to faults which degrades the system performance and makes the chip unreliable. According to [2], faults in VLSI can be classified into Transient, Intermittent and Permanent. The temporary interference like cross talk, voltage noises can lead to transient faults. Due to marginal hardware the faults can occur repeatedly at one location or often in bursts are called as intermittent faults. The faults occurred due to incorrect logic, setup time or hold time violations are considered as permanent faults.

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

In this paper, permanent faults occurred in application cores have been considered while providing fault-tolerance using spare cores. Application mapping in NoC is a NP-Hard problem [3]. The NoC network consumes 30-40% of the total power consumed by the chip which is quite significant [4]. This necessitates for the deployment of proper application mapping and routing algorithms to reduce the power consumption. This has motivated us to provide the flexibility in placing a spare core in the Mesh-of-Tree (MoT) topology to take care of failed ones using Particle Swarm Optimization (PSO). Since there are no approaches reported in the literature focusing on the core faults in MoT network, therefore we have compared our technique with native fault free approach reported in [5] and communication cost is calculated. In our approach we have considered only core faults, whereas router and link faults are beyond the scope of our work. The paper is organised as follows. Section II deals with the literature survey. Section III gives brief overview of MoT topology. Section IV explains about the problem definition Section V describes the PSO formulation. Section VI recites experimental results followed by the conclusion.

II. Related Works

There have been various strategies proposed for mapping of application core graphs onto NoC based architectures. NMAP mapping technique has been proposed in [6] to minimize the communication delay in a NoC. This technique uses split traffic routing technique to satisfy the bandwidth constraints of the links. A branch and bound algorithm has been proposed in [7] to map the application core graphs onto tile based NoC architectures to minimize the energy consumption while satisfying the bandwidth constraint of the links. An application for automatic topology selection and mapping namely SUNMAP has been proposed in [8]. The final mapped solution can be generated subject to different criteria like minimizing area, average communication delay and power dissipation.

Specific to the MoT topology, a mapping technique based on Kernighan-Lin (KL) partitioning has been proposed in [9]. The KL partitioning identifies the closely related cores and a heuristic which tries to map these cores onto the given MoT topology subject to minimizing the communication cost. One of the shortcomings of the KL partitioning is the solution may get stuck at a local minima instead of the global best solution. To overcome this shortcoming, a KL_GA mapping technique has been proposed in [9] for MoT topology which is based on KL partitioning and a Genetic Algorithm (GA). A MoT mapping technique based on Discrete Particle Swarm Optimization (DPSO) has been proposed in [5]. Most of the works reported in the literature have not considered the core failures in the MoT topology based NoC design. In our proposed work, we have addressed core failures in MoT using DPSO based technique [5] to map the application cores along with a spare core to make the system reliable and robust.

III. Overview of Mesh-of-Tree structure

The basic design of a *M x N* MoT structure has *M* number of row trees and *N* number of column trees. A *M x N* MoT has $3 * (M * N) - (M + N)$ nodes with a diameter of $2log_2M + 2log_2N$. The bisection width of a MoT is $min(M, N)$. The Fig. 1 shows a 4x4 MoT structure where there are 4 row trees and 4 column trees. There are three different type of routers namely leaf routers (L), stem routers (S) and root routers (R). Leaf routers are connected to both row tree and column tree. Each leaf router can accommodate two cores. These leaf routers are connected to stem routers and stem routers are connected to root routers.

For an M x N MoT structure, the number of routers will be as follows:

- Number of leaf routers = M * N.
- Number of stem routers = 2 * (M + N).
- Number of root routers = M + N.

Therefore for 4x4 MoT structure the number of leaf routers, stem routers, root routers are 16, 16, 8 respectively.

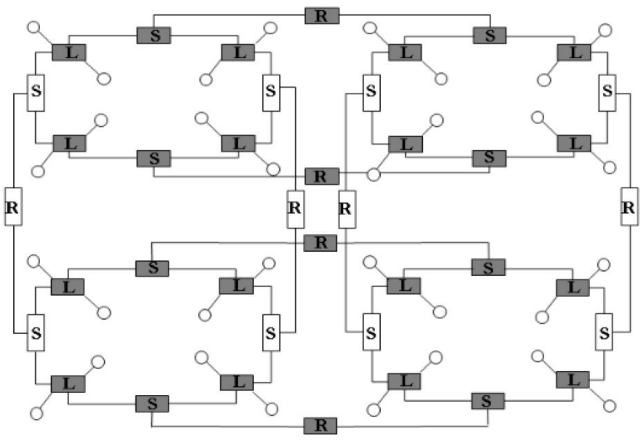

Fig. 1: 4x4 MoT structure

IV. Problem Definition

An application constitutes processing elements or cores which can implement set of tasks with required communication bandwidths. These can be represented in the form of core graphs and it is defined as follows.

Definition 1: The directed graph *G(C,E)* corresponds to communication cores (*C*) and edges (*E*) of an input application.

Let us consider the core graph of MPEG application shown in Fig. 2. It contains 12 cores represented as *C0 - C11* with the communication bandwidth represented on their edges (*E*). Core *C0* is communicating with core *C4* with a bandwidth of 190 mega bits per second. Similarly other cores in the application core graph communicate with the bandwidth mentioned on their edges. As we can observe core *C4* is the most communicating core and has high probability of failure [10]. Hence, we consider *C4* as the failed core. However, we have also considered different core failures and computed communication cost, defined next.

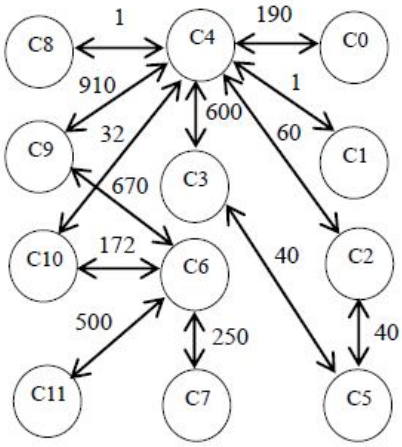

Fig. 2: MPEG application

Definition 2: The MoT network size of M x N with available number of routers to place the cores of an application core graph.

In the application core graph we have considered core failures due to permanent faults and a spare core is included which can take care of communications associated with failed core. By considering the definitions 1 and 2, the problem statement can be minimizing the communication cost by efficient placement of the cores (including spare core) in the MoT network. The communication cost is calculated as the product of hop distance between source core (C_i) and destination core (C_j) and the respective bandwidth (*BW*) between them.

$$Communication\ cost = \sum_{\forall Edges} (Hop\ distance * Bandwidth) \quad (1)$$

To address this problem we have proposed a methodology based on Particle Swarm Optimization.

V. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) [11] is a population based stochastic technique designed and developed by Eberhart and Kennedy in 1995, inspired by social behaviour of bird flocking or fish schooling in search of food. In a swarm of birds, each bird is referred to as particle which is known as solution. Each individual particle adjust its flying according to its own experience and experience of its neighbouring particles. The quality of individual particle is defined by its fitness value. Several researchers were successful in using PSO for continuous domain. This has motivated us to apply it for discrete domain as well. We have used the same formulation for Discrete PSO proposed in [12], and applied it to our problem. The detailed explanation is given below.

A. Particle Structure

A particle structure is an array of size *n*, where *n* represents the numbers of cores (including spare core) present in the core graph. The index of an array represents the routers present in the MoT network. If the number of cores in the core graph is less than the number of routers present in the MoT network, then dummy cores are added with zero communication cost. In a MoT topology each leaf level router can accommodate two cores present in the core graph. Hence two consecutive entries in the particle structure constitutes to one router. Further the particle structure is explained in detailed with an example.

Consider an example of a core graph having 6 cores which have to be placed in a 2x2 MoT topology. The number of available router positions are 8. The particle structure for the core graph having 6 cores numbered from 0-5 is shown in Fig. 3.

The particle *P:*

Core number	C4	C3	C5	C1	**C7**	C2	C6	C0
Router number	0	0	1	1	2	2	3	3
Core position	0	1	0	1	0	1	0	1

Fig. 3: Particle structure

In the particle *P*, *C7* represents the spare core which takes the communication associated with failed core *C4*. The index number represents the leaf router to which the cores are attached. Since, two cores can be attached to one leaf router, therefore core position can be differentiated between two cores by either 0 or 1. It can be noted from the particle *P*, core 3 and 4 have been mapped to the same leaf router i.e router 0 but they can be distinguished by the core position which is 0 & 1 respectively.

B. Fitness function

The quality of the particle is defined by its fitness function. Fitness of a particle is equal to the communication cost due to association of cores (which includes spare core) of an application on to leaf routers in MoT structure. As we have already mention in section IV, the spare core placement is performed by considering core failures. In an event of core failure, spare core takes care of failed core without compromising on communication cost while providing fault tolerance.

C. Local best

In a problem of search space every particle will have one set of core positions that leads to minimum fitness value which is known as *local best* or *pbest*. During the evolution process the *pbest* value can be updated if the particle encounters minimum fitness value than the previous one.

D. Global best

In the process of evolution of generations, each generation will have minimum communication cost. For a particular generation, the particle resulting in minimum communication cost is known as *global best* or *gbest*. It controls the evolution of particles and it is modified if the value in the current iteration is less than the previous iteration.

E. Evolution of Generation

New particles are created over the generations which give results closer to the optimum value. Initially particles are created randomly and fitness for each particle is evaluated. For the initial particle local best and global best will be the same. Further generations are created by exchanging entries of the particle using a *swap operator*. The sequence of swap operators is known as *swap sequence*.

1) Swap Operator: Swap operator will swaps the entries inside the particle to create a new particle. It is given by SO(a,b) where a and b are the entries inside a particle that are to be swapped.

For the particle *P* shown in Fig. 3, if we apply swap operator SO(2_0, 3_1) then cores at position 2_0 and 3_1 will be swapped. A new particle P_N is created which is shown in Fig. 4.
New particle P_N:

Core number	C4	C3	C5	C1	**C0**	C2	C6	**C7**
Router number	0	0	1	1	2	2	3	3
Core position	0	1	0	1	0	1	0	1

Fig. 4: Swap operator

2) Swap sequence: The series of swap operations applied on a particle is known as swap sequence. It is denoted by SS = [SO(a,b), SO(c,d)] where a, b, c and d are the different entries in a particle. During the swap sequence SO(a,b) will create an intermediate particle P_I on which SO(c,d) has to be applied.

For example a swap sequence SS = [SO(2_0,3_1), SO(1_0,2_1)] has been applied on the above defined particle *P* shown in Fig. 3. It creates new particle P_N in two steps.
In step 1, the swap operator SO(2_0, 3_1) is applied. The intermediate particle P_I is same as shown in Fig. 4. In the second step a swap operator SO(1_0, 2_1) is applied on a intermediate particle P_I. Fig. 5 shows the new particle P_N obtained after applying swap sequence.

Each particle tries to move towards the local best and the global best. After all particles have undergone the evolution, a new generation gets created. The best fitness of this generation gives the global best for the population.

Core number	C4	C3	**C2**	C1	C0	**C5**	C6	C7
Router number	0	0	1	1	2	2	3	3
Core position	0	1	0	1	0	1	0	1

Fig. 5: New particle

VI. Experimental Results

In this section we present the results obtained by performing spare core placement using DPSO technique for several benchmark applications reported in the literature and the applications generated using TGFF tool [13] by

- Considering no core has been failed.
- Considering most communicating core has been failed.
- Varying the percentage of faults while considering most communicating core failure.
- Taking failed core as an input from the user.
- Scaling the MoT network size

A. Communication cost results for zero core failures:

In this experiment we have considered different application benchmarks reported in the literature. In all the applications we have assumed no core has been failed and calculated the communication cost using eq. 1. Table. I, shows the communication cost [14] for each application.

TABLE I: Communication cost for zero core failures

Application	No.of cores	Communication Cost
MPEG	12	5752
MWD	12	2048
263Enc	12	33.19
Mp3Enc	13	22.02
263Dec	14	25.33
VOPD	16	6318

The results reported in the Table. I will act as a reference to compare with our approach for different core failures.

B. Communication cost results for most communicating core failure:

In this experiment we have assumed most communicating core has the highest probability of failure [15]. However, we have also experimented with other core failures in the application. Since most communicating core has the highest communication bandwidth compared to all other cores in the application. Hence the communication cost overhead is more compared to other core failures. In the event of core failure, the spare core will take the communications associated with the failed core. In-spite of being fixed position of spare core, our approach provides the flexibility in placing the spare core in 4x4 MoT. This will leads to minimum overhead in communication cost.

As we can observe from the Table. II, on comparison with fault free communication cost reported in Table. I the average percentage of overhead in communication cost is 4.48%. This shows the efficiency of our approach in placing the spare core in best suitable position in the network.

TABLE II: Communication cost for most communicating core failed

Application	No. of cores	Failed core	Communication cost	% Overhead
MPEG	12	4	6380	10.91%
MWD	12	4	2240	0.93%
263Encoder	12	0	36.99	11.44%
MP3Encoder	13	0	22.32	1.31%
263Decoder	14	2	25.39	0.23%
VOPD	16	7	6505	2.87%
			Average % overhead	4.48%

C. Varying percentage of faults in 4X4 MoT network with most communicating core failure

Fault percentage in the M x N MoT network can be termed as the number of leaf routers are not available for mapping the cores of an application. By considering the most communicating core failure, we have increased the fault-percentage to 15%, 30% and 45% and the results are reported in the Table. III. While increasing the fault-percentage in the network, the number of positions available for placement of a spare core decreases. Within the limited search space our approach will find the best position in the MxN network to reduce the communication cost.

TABLE III: Communication cost while varying percentage of faults in the 4x4 MoT network.

Application	Communication Cost		
	15% faults	30% faults	45% faults
MPEG	6132	6980	7540
MWD	2240	2272	2496
263Enc	37.01	41.93	41.96
Mp3Enc	22.34	24.502	24.47
263Dec	25.41	26.61	26.85
VOPD	6404	6721	6909

From Fig. 6 we can observe that percentage overhead in communication cost increases with increase in fault-percentage. It may be noted that the communication cost overhead is independent for each application and it varies different applications. For example if we observe Fig. 6, the overhead of communication cost for MPEG application for 15%, 30% and 45% are 5.2%, 20.3% and 30.1% respectively. Whereas the communication cost overhead for 263Decoder for 15%, 30% and 45% are 0.02%, 5.01% and 6.12% respectively. The overhead in communication cost for MPEG and MWD applications are different. This is due to the different communication behavior of an application.

This shows that with respect to each application and fault percentage in the MoT network, our approach tries to find the best position for the spare core in the network. On an average for all application benchmarks reported in Table. III, we could achieve 4.68%, 13.33%, 16.5% overhead in communication cost for 15%, 30% and 45% faults in the 4x4 MoT network respectively.

D. Communication cost results by taking failed core as user input

This section presents a spare core placement for different core failures in several application benchmark reported in the

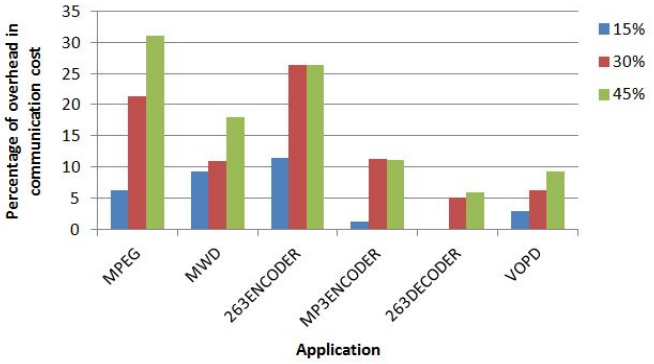

Fig. 6: Percentage of faults

literature. For a MPEG application, the spare core placement has been performed by considering different core failures and results are shown in Fig. 7. The X-axis represents the failed core of an application while Y-axis represents the communication cost obtained for corresponding core failure. The cores on the X-axis are arranged in a decreasing order of the communication bandwidth. The general trend of the graph indicates that the communication overhead by introducing a spare core decreases along the x-axis reaching fault-free communication cost. As we can observe from the Fig. 7, the communication cost decreases from most communicating core failure to least communicating core. Similarly we can observe the same trend in decrease of communication cost for MWD, 263Encoder, Mp3Encoder, 263Decoder, VOPD applications and the results are shown in Fig. 8 to Fig. 12 respectively. This decrease in the communication cost is due to the decrease in communication bandwidth associated with failed core.

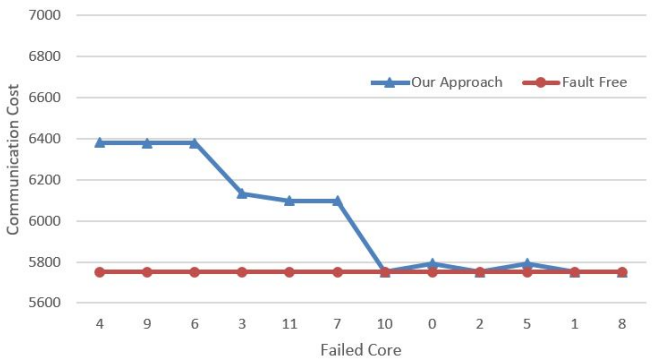

Fig. 7: Communication cost for core failure in MPEG application

From Fig. 13 we can observe that the average overhead in the communication cost for MPEG, MWD, 263Encoder, Mp3Encoder, 263Decoder, VOPD applications are 4.4%, 3.3%, 2.1%, 0.7%, 0.2% and 2.2% respectively. This shows that our approach is providing best position for the spare core in the MoT network for any one of the core has been failed in the application.

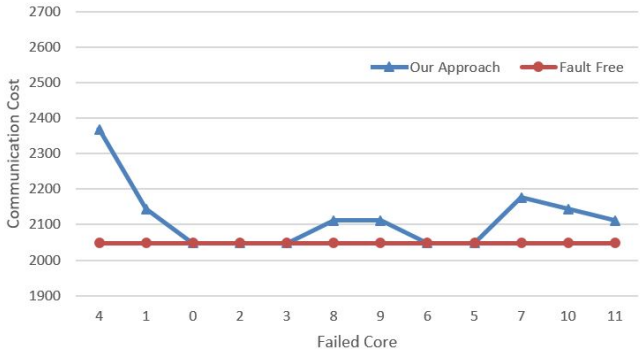

Fig. 8: Communication cost for core failure in MWD application

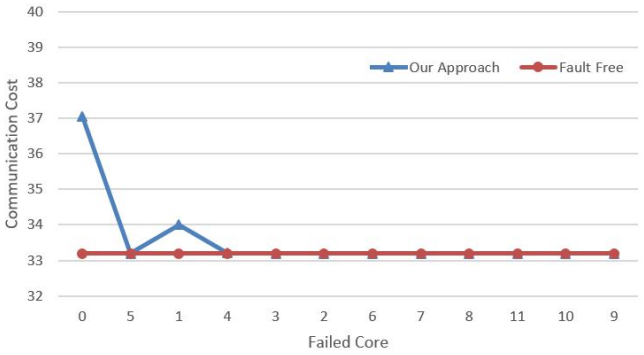

Fig. 9: Communication cost for core failure in 263ENCODER application

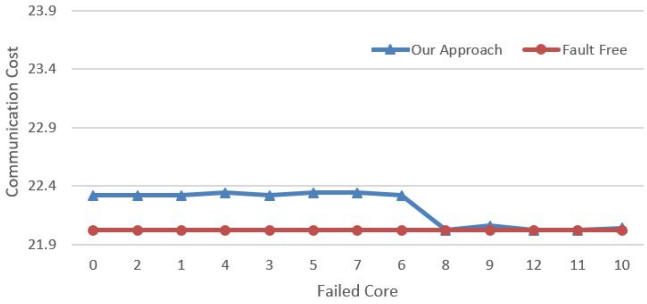

Fig. 10: Communication cost for core failure in MP3ENCODER application

VII. Scaling the MoT network size

In this experiment we scale the MoT network size from 4x4 to 8x8 to check the scalability of our approach. The applications reported in the literature have less number of cores i.e., 16 in case of VOPD.

Therefore we have used TGFF tool [13] to generate application core graphs up to 96 cores. Table IV shows the communication cost for spare core having more than 32 cores in an application. On an average we have achieved an optimal

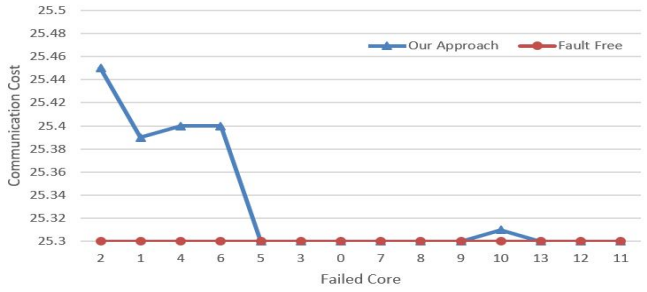

Fig. 11: Communication cost for core failure in 263DECODER application

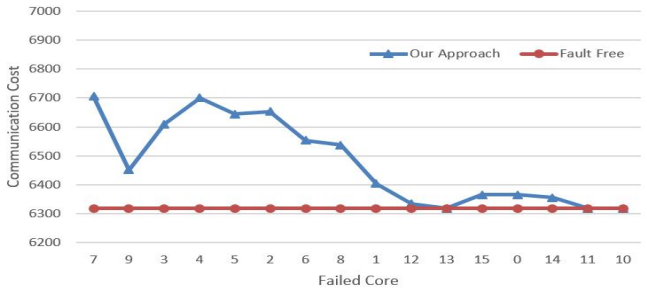

Fig. 12: Communication cost for core failure in VOPD application

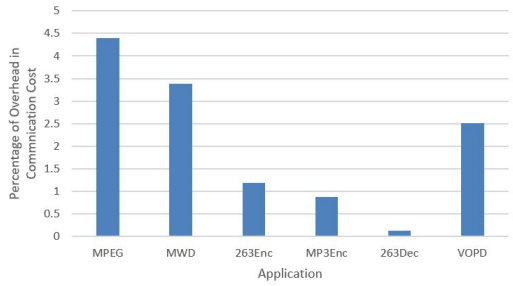

Fig. 13: Percentage of overhead in communication cost for different applications

overhead of 5.32% in communication cost for greater number of cores. This is due to the efficient placement of spare core in the given MoT network. This shows the scalability of our approach by providing best position for the spare core in 8x8 MoT network.

VIII. Conclusion

In this paper we have presented a PSO based meta-heuristic technique to determine the flexible placement of spare core in a Mesh-of-Tree based NoC architecture. It has been shown that the flexible placement of the spare core has resulted in a less communication overhead while providing a degree of fault tolerance in the network. Our proposed algorithm works efficiently in case of varying fault percentage in the network, by varying different core failures in the application and also by scaling the MoT network size. The future work includes multiple spare core placement in MoT network and proposing exact methods like Integer Linear Programming.

TABLE IV: Communication cost for applications having higher number of cores

Application	Cores	Fault-free	Our approach	% Overhead
G1	32	136519.51	143080	4.80
G2	48	182852.40	196369	7.39
G3	64	166032.73	174078.1	4.84
G4	80	189388.01	204172.17	7.81
G5	96	349265	355466.43	1.77
			Average % overhead	5.32

References

[1] 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.

[2] 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

[3] W. E. Donath, "Complexity theory and design automation," in *Proceedings of the 17th Design Automation Conference*, ser. DAC '80. New York, NY, USA: ACM, 1980, pp. 412–419. [Online]. Available: http://doi.acm.org/10.1145/800139.804563

[4] J. Fang, L. Yu, S. Liu, J. Lu, and T. Chen, "Kl_ga: an application mapping algorithm for mesh-of-tree (mot) architecture in network-on-chip design," *The Journal of Supercomputing*, vol. 71, no. 11, pp. 4056–4071, Nov 2015. [Online]. Available: https://doi.org/10.1007/s11227-015-1504-y

[5] P. K. Sahu, A. Sharma, and S. Chattopadhyay, "Application mapping onto mesh-of-tree based network-on-chip using discrete particle swarm optimization," in *2012 International Symposium on Electronic System Design (ISED)*, Dec 2012, pp. 172–176.

[6] S. Murali and G. D. Micheli, "Bandwidth-constrained mapping of cores onto noc architectures," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, vol. 2, Feb 2004, pp. 896–901 Vol.2.

[7] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based noc architectures under performance constraints," in *Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference, 2003.*, Jan 2003, pp. 233–239.

[8] S. Murali and G. D. Micheli, "Sunmap: a tool for automatic topology selection and generation for nocs," in *Proceedings. 41st Design Automation Conference, 2004.*, July 2004, pp. 914–919.

[9] P. K. Sahu, N. Shah, K. Manna, and S. Chattopadhyay, "A new application mapping strategy for mesh-of-tree based network-on-chip," in *2011 International Conference on Emerging Trends in Electrical and Computer Technology*, March 2011, pp. 518–523.

[10] F. Khalili and H. R. Zarandi, "A fault-tolerant low-energy multi-application mapping onto noc-based multiprocessors," in *2012 IEEE 15th International Conference on Computational Science and Engineering*, Dec 2012, pp. 421–428.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, Nov 1995, pp. 1942–1948 vol.4.

[12] S. J., A. Sharma, and S. Chattopadhyay, "Multi-application network-on-chip design using global mapping and local reconfiguration," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 2, pp. 7:1–7:24, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2556944

[13] R. P. Dick, D. L. Rhodes, and W. Wolf, "Tgff: task graphs for free," in *Hardware/Software Codesign, 1998. (CODES/CASHE '98) Proceedings of the Sixth International Workshop on*, Mar 1998, pp. 97–101.

[14] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *J. Syst. Archit.*, vol. 59, no. 1, pp. 60–76, Jan. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.sysarc.2012.10.004

[15] C. L. Chou and R. Marculescu, "Farm: Fault-aware resource management in noc-based multiprocessor platforms," in *2011 Design, Automation Test in Europe*, March 2011, pp. 1–6.