Hyperloop: An Efficient Linearly Extensible Network

[1]Shivam Chaudhary, [2]Gaurav Raj and [3]N. K. Gupta,
[1]Pre-final Year student, [2,3]Assistant Professor,
[1,2,3]Computer Science & IT, Faculty of Engineering & Technology, Jaypee University, Anoopshahr, India

Abstract-Quest for faster, higher and higher computing power leads to design high performance economical servers. A low cost high performance multiprocessor architecture has been proposed with only four nodes. It is a linearly extensible network with lesser diameter. And hence a less complex server has been developed. It has been tested for its performance by implementing various load balancing algorithms and it was found that it gives good performance and exhibits100% Degree of Parallelism (DOP).Therefore, it can be used as a server.

Keywords: *Server; Multiprocessor Architecture; DOP; Load Scheduling.*

I. INTRODUCTION

A multiprocessor system is a system which consists of a number of independent processors that work together to solve a given problem. One of the most important factor which must be considered for performance evaluation of any multiprocessor architecture is the interconnection network topology. The choice of interconnection network may affect several characteristics of the multiprocessor system. This is the reason why a suitable interconnection network is an integral part of any high performance parallel system. These multiprocessor architectures are often evaluated in terms of their properties such as diameter, degree, complexity and cost. The system should have the following properties to achieve higher performance: It should have small diameter, low degree, lesser complexity and low cost. A number of such parallel systems are reported in literature[1, 4, 7]. Among them binary n-cube which is also known as Hypercube (HC) network is one of the most famous and widely accepted parallel system (network topology).

Performance of a multiprocessor architecture does not completely depend upon its interconnection network but also on the load distribution among all the processors of the network[2, 3, 6,11-13]. The process of assigning tasks to the processors of a network on the basis of some parameters is known as task scheduling. In order to enhance the performance of such networks, a number of efficient task scheduling algorithms are implemented on the proposed network. An inefficient scheduling algorithm can lead to a load imbalance on various nodes which can significantly increase the response times of the tasks scheduled. There are two types of scheduling approaches which are used to schedule the tasks on multiprocessor systems namely Static and Dynamic approaches[6, 7]. In this paper, a new network topology is being proposed in order to achieve higher performance. In order to carry out the performance evaluation of the proposed network, a number of dynamic scheduling schemes have been implemented on the network and the performance is evaluated and compared with other similar networks. Performance evaluation of the proposed network is carried out on the basis of characteristics such as network diameter, degree, load imbalance factor (LIF), extensibility, number of nodes and degree of parallelism.

The rest of the paper is organized as follows: Section II presents the architectural details of the similar multiprocessor architectures with their topological properties. The proposed network topology is presented and discussed in section III. Comparative study of the proposed network with other networks is carried out in section IV. The performance evaluation of the proposed network topology is carried out in section V by applying different task scheduling schemes on it. In the end paper is concluded in section VI.

II. ARCHITECTURAL BACKGROUND

This section explains the topological features of the similar architectures to give a background of the proposed network topology[1, 4, 5]. It includes Hypercube (HC), Linearly Extensible Tree (LET) and Linearly Extensible Cube (LEC) networks.

A. Hypercube (HC)

It is a binary n-cube architecture and is considered as one of the most popular topology. In general, an n-cube consists of $N=2^n$ nodes spanning along n dimensions, with two nodes per dimension. A 3-cube network with 8 nodes is shown in figure 1. A 4-cube network can be formed by interconnecting the corresponding nodes of two 3-cube networks.

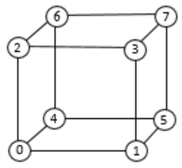

Figure 1: Hypercube network with 8 processors.

B. Linearly Extensible Tree (LET)

It is also a binary tree type network topology. Architecture of LET exhibits better connectivity, lesser number of nodes and linear extensibility over Hypercube network.It has low diameter. The LET network grows linearly in a tree like shape.

In a binary tree, the number of nodes at level j is 2j whereas in LET network, this number is equal to j+1. LET network is shown in figure 2.

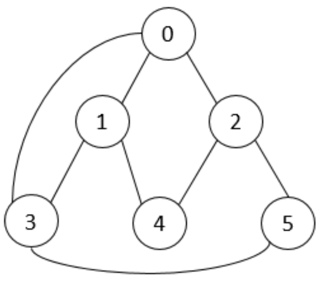

Figure 2: LET network with 6 processors.

C. Linearly Extensible Cube (LEC)

The LEC network grows linearly in a cube like shape. The number of processors N in the network is given by N=2*n. Here, n is the level or depth of the network ($n \in Z$, $n > 0$). For n=1, an LEC network of two processors is obtained. LEC network is shown in figure 3.

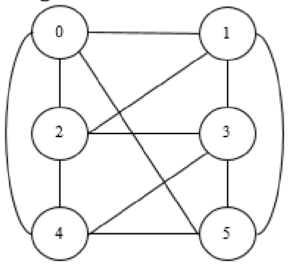

Figure 3: LEC network with 6 processors.

III. PROPOSED TOPOLOGY

In this section the architectural analysis of the proposed topology has been carried out. A new tree type architecture is defined and its topological properties are discussed in this section.

A. Proposed Architecture

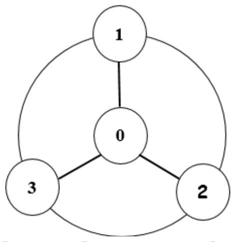

Figure 4: Proposed Hyperloop network with 4 processors.

The proposed network grows linearly in a looped fashion. The Hyperloop network itself is recursively connected and defined through the Link Functions depending upon the angular geometry. Let Q be the set of N identical processors represented as

$$Q= \{P_0, P_1, P_2, …., P_{N-1}\} \quad ... \text{ eq (1)}$$

$$Q= \{P_j\}$$

$$0<= j <=N-1 \quad … \text{ eq (2)}$$

where N is the total number of processors in the Hyperloop network and it is given by

$$N=3*n +1 \quad ... \text{ eq (3)}$$

where n ϵ Z, n > 0 (n is the level or depth of the network).

In Hyperloop (HL) network, there are three processors at each level which are connected with the root processor and with each other by making an angle of 120˚ with each other. With each increasing level upcoming 3 processors follow the same relationship with the root processor without disturbing the existing configuration. In order to define Link Functions for the proposed network, we denote three processors at each level (from 1st level) as P_{n1}, P_{n2}, P_{n3}, where n is the level or depth of the network.

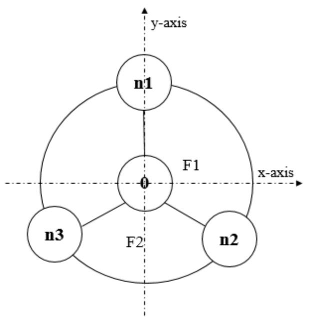

Figure 5: Formula based construction of Hyperloop architecture.

Let q be the set of designated processors of Q thus,

$$q= \{P_{ni}\} \quad … \text{ eq (4)}$$

, where n is level or depth of the network.

$$, 1<= i <=3 \quad … \text{ eq (5)}$$

The Link Functions F_1 and F_2 define the mapping from q to Q as

$$F_1 (P_{ni})= \angle P_{n1}P_0P_{n2}= 120^{\circ} \quad … \text{ eq (6)}$$

$$F_2 (P_{ni})= \angle P_{n2}\, P_0\, P_{n3}= 120^{\circ} \quad … \text{ eq (7)}$$

These two link functions F_1 and F_2 indicate the links between various processors in the network as shown in the figure 3.

B. Topological Properties

Following are the topological properties of proposed network topology:

Theorem 1: The total number of nodes in the Hyperloop architecture is 3*n+1.

Proof: The proposed architecture is an undirected graph, where at each level network has three nodes except at 0th level, therefore total number of nodes N is given as

N=3*n+1 e.g. {1, 4, 7, 10….} where n ϵ Z, n > 0

Theorem 2: The diameter of the Hyperloop architecture is equal to n (number of level or depth).

Proof: Diameter is defined as the maximum shortest path between the source and destination nodes inside any network. In proposed architecture the diameter increases as the level or depth of the network increases. Diameter of the proposed network shows a linear trend.

Diameter= n (1, 2, 3…)

Theorem 3: The extensibility of the Hyperloop network is 3.

Proof: Extensibility is defined as the smallest increment by which the system can be expanded in useful way. In proposed architecture, the number of processors increases in a constant manner because each extension requires single layer of 3 nodes and no additional node is required at any extension and it is being depicted in the figure 6.

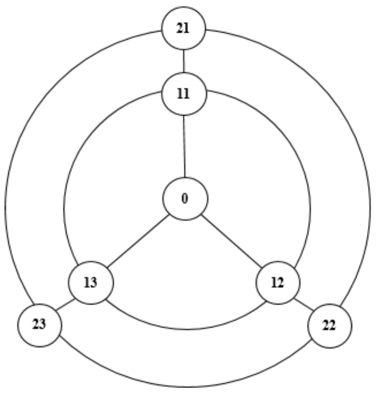

Figure 6: Extended version of the proposed Hyperloop network.

Table 1: Characteristics of the Hyperloop network.

Level	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Number of processors	1	4	**7**	10	13	16	19	22
Diameter	0	1	2	3	4	5	6	**7**
Degree	0	3	4	4	4	4	4	3

Theorem 4: The degree of parallelism (DOP) of the Hyperloop architecture is 100%.

Proof: DOP is defined as the ratio of the number of processors which are involved and utilized during the execution of a particular program to the total number of processors which are present in the multiprocessor architecture. It simply describes the involvement of network processors during the load balancing process at any given instant of time. DOP of various architectures is shown in table 2.

DOP= (degree (P_h) +1)*k^{-1} … eq (8)

where P_h= Processor with highest degree in the network and k= Number of processors present in the network.

Table 2: Degree of Parallelism of various architectures

Architecture	HC	LET	LEC	Hyperloop
DOP	50 %	66.66 %	83.33 %	100%

IV. COMPARATIVE STUDY

To draw a general conclusion on the proposed architecture, its comparative study with other similar networks have been carried out and is shown below in table 3. This comparative study is based on four parameters namely, number of processors, node degree, diameter and extensibility.

Table 3: Characteristics of different architectures.

Architecture	HC	LET	LEC	Hyperloop
Number of Processors	$N=2^n$	N=∑k k=1 to n	N=2* n	N=3*n+1
Degree	n	4	4	4
Extensibility	2^n	n+1	2	3
Diameter	$O(\log_2(n))$	O(√N)	$\lfloor N \rfloor$	n

The above comparative study has been also shown below in a graphical format. Graphs are plotted for each of the parameters for proposed architecture and for HC, LET, LEC networks as well. In Hypercube network, the number of processors rises exponentially with the rise in levels. It attains larger numerical values even at lesser number of levels there by showing an exponential trend which is depicted in the graph.

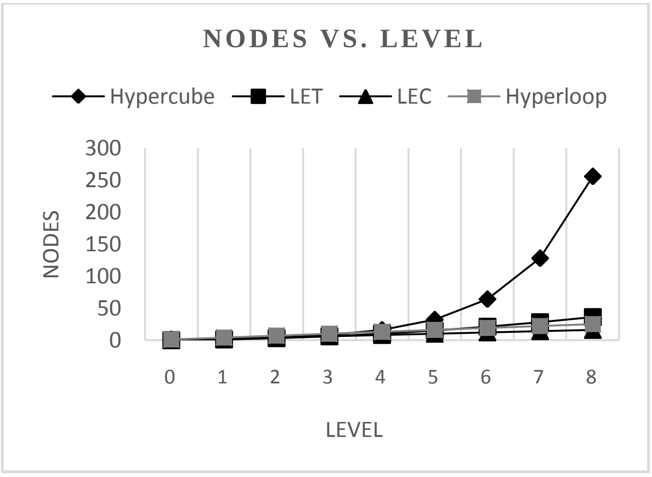

Figure 6: Number of nodes of different architectures.

In LET, the number of processors also rises at a faster rate with levels however, this increment is smaller than the Hypercube network. In case of Hyperloop (HL) architecture, there is an increment of 3 processors at each level, hence the number of nodes in case of proposed architecture rises linearly but this rise is smaller than LET and Hypercube networks. LEC also shows a linear trend of a constant increment in number of processors with levels. The graph of proposed architecture lies in between LET and LEC and is shown in figure 6.

The second important parameter which is discussed here is diameter. Generally, Hypercube network has the smallest diameter of all. LEC shows a linear trend in diameter with the rise in levels however, this increment is larger as compared to the Hypercube network. In case of LET network, its graph lies in between Hypercube and LEC. In case of proposed Hyperloop (HL) architecture, this trend is linear in nature and lies in between LEC and LET. Its diameter increases at a much slower rate as compared to the LEC due to which it has less intercommunication distance between the processors as compared to LEC network. Diameter of the proposed network increases by 1 at each level and this trend is being depicted in the figure 7.

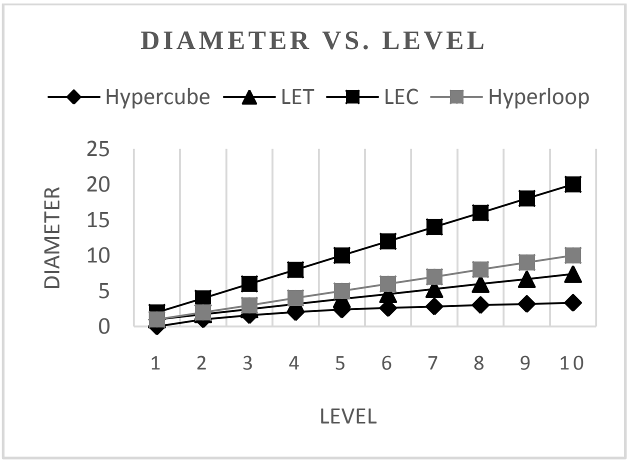

Figure 7: Diameter of different architectures.

The third parameter which is discussed here is the degree of nodes inside any network topology. The degree of a node in a network is defined as the total number of connections required at each node. It clearly determines the connectivity of any node inside any network. Higher the connectivity, higher will be the hardware complexity and the cost of the network. Therefore, degree must be kept as low as possible inside any network. The connectivity of the Hypercube network increases with size therefore it becomes more and more complex with increase in size. The connectivity of LEC network remains equal to 4 always. In case of proposed Hyperloop network it always remains less than 4. For first and last level it is equal to 3 and for other levels it is equal to 4. The degree of LET network also remains equal to 4 or less. These trends are being depicted in the figure 8.

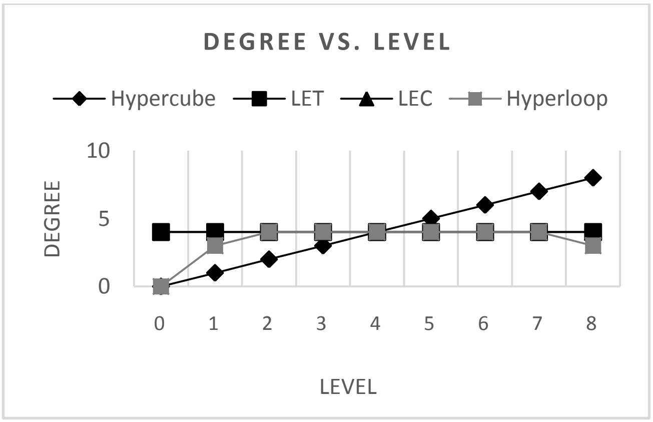

Figure 8: Node degree of different architectures.

The other important parameter which is discussed here is the extensibility. It is defined as the smallest increment by which the system can be expanded in a useful way. In proposed Hyperloop network, a single layer of 3 nodes is required at each extension and hence the extensibility of the network remains constant. In case of LET network, the extension complexity increases linearly because each extension requires addition of a single layer of (n+1) nodes. Therefore, at higher levels, the number of nodes may become large and hence the complexity of the network increases. Similarly in case of Hypercube network, the extension complexity increases exponentially by the power of 2. In case of LEC network, it requires a single layer of 2 nodes at each extension. The constant growth of the proposed network makes the extension less costly. The extensibility trends of various network topologies are being depicted in the figure 9.

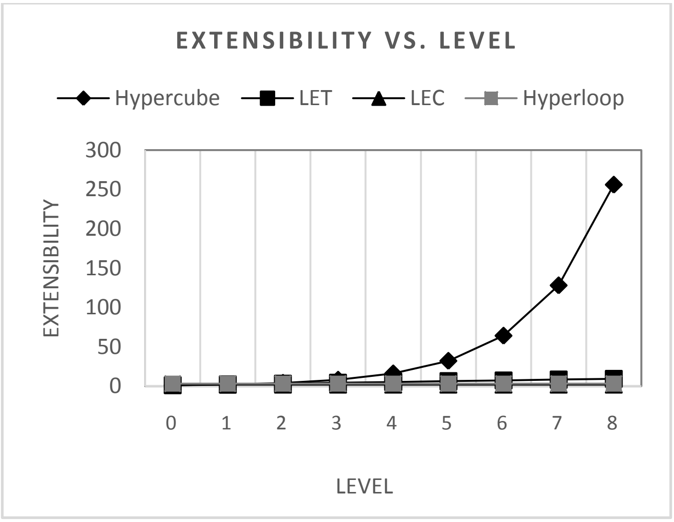

Figure 9: Extensibility of different architectures.

V. PERFORMANCE ANALYSIS

It involves the mapping of tasks on various network processors. The process of assigning tasks to the processors of a network on the basis of a particular parameter is known as task scheduling. There are several scheduling schemes which have been reported, which carry out the distribution of load on a multiprocessor network, both statically and dynamically. Dynamic load scheduling algorithms are often considered better than the static load scheduling algorithms because they carry out the load distribution in such a way that each and every processor has to do almost equal amount of work. In the proposed work, the performance of proposed Hyperloop architecture is tested by applying two scheduling algorithms namely Minimum Distance Scheduling (MDS) algorithm and Two Round Scheduling (TRS) algorithm. The MDS algorithm carries out the load balancing process on the basis of minimum distance property. To carry out the load balancing process, MDS algorithm calculates the value of ideal load (IL) at each stage of the load. IL is the value of load, a processor is having when the network is fully balanced. This algorithm identifies the underloaded (acceptor) processors and overloaded (donor) processors on the basis of IL. Migration of tasks can take place between the donor and acceptor processors only.The Two Round Scheduling (TRS) algorithm works as an extension of MDS algorithm. It follows the same procedure as MDS algorithm to carry out the load balancing but it takes into consideration of even those acceptor nodes which are not directly connected to the donor nodes. This algorithm has a constraint in the scheduling to consider only one intermediate node between donor and acceptor nodes. The load imbalance factor (LIF) for ith load stage is denoted as LIF_i. It is defined as:

$$LIF_i = [\max\{load\,(P_j)\} - \{IL_i\}] / \{IL_i\} \quad \dots \text{ eq (9)}$$

where $(IL)_i = [load_i(P_0) + load_i\,(P_1) + \dots\dots + load_i\,(P_{N-1})] / N$

$$\dots \text{ eq (10)}$$

and, $\max\{load_i(P_j)\}$ denotes the maximum load pertaining to the stage i on a processor P_j, $0 <= j <= (N-1)$ and $load_i(P_j)$ stands for the load on processor P_j due to the ith stage.

In order to carry out the comparison, same algorithm is also applied on Hypercube, LEC and LET networks. For simulation purpose, a simple problem is assumed in which the load is partitioned into a number of tasks. The performance has been evaluated by simulating artificial dynamic load on different networks. In simulation run, uniform dynamic load is generated and is mapped on to various nodes of the system and finally imbalance is obtained for various stages of the task structure. Graphs are plotted on the basis of average percent imbalance (LIF) against load stages. The results for MDS algorithm are depicted in figure 10 while the results for TRS algorithm are depicted in figure 11.

Algorithms:

Following are the steps of scheduling algorithms applied and corresponding comparative graphs are drawn:

(MDS)

1. Mapping of the load to the root processor in the network.
2. Calculate IL at any particular load stage.
3. Migrate the tasks to other processors of the network.
4. Create the subset of acceptors and donors on the basis of IL.
5. Transfer the load from donors to the acceptors on the basis of connectivity of the processors.
6. Repeat step v.
7. Repeat steps ii to v until each and every processor has the same load.

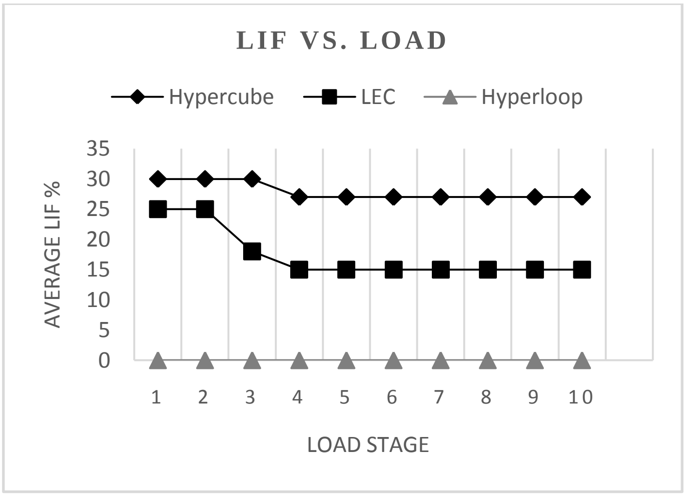

Figure 10: Implementation results of MDS algorithm.

It is evident from the above graph that both LEC and Hypercube networks are performing well for the given algorithm. They both have smaller values of %LIF starting from less than 40% and at a particular stage it becomes saturated. In case of Hyperloop architecture, the graph of %LIF remains saturated and equal to 0% throughout the load scheduling process.

(TRS)

1. Map the load at the root processor and calculate IL at a particular stage of the task structure (Load).
2. Transfer the load onto various available processors in the network. Check the load of each processor to identify the donors and acceptors (processors).
3. Check the connectivity of donors and acceptors and migrate tasks from donors to acceptors to make the connected processors balanced.
4. If no connection is available between donor and acceptor, then find the alternative path by considering only one intermediate node, and perform migrations to make the network fully balanced.
5. Repeat the above procedure for the next stage.

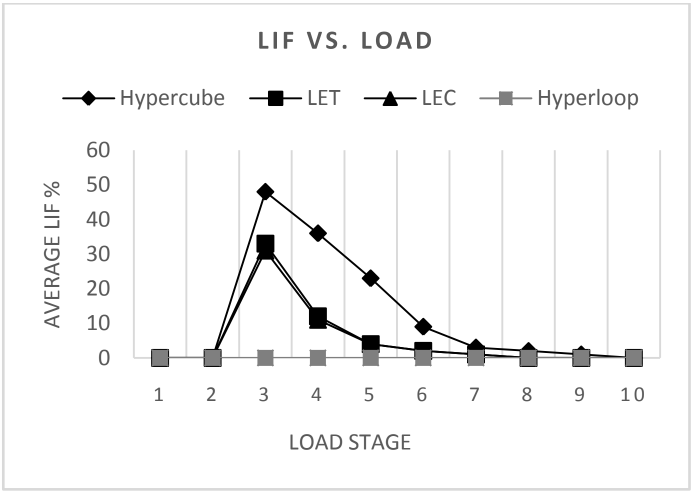

Figure 11: Implementation results of TRS algorithm.

When TRS algorithm is applied on the Hyperloop architecture, LEC, LET and Hypercube networks it is found that initially all of them start from 0%. Hypercube attains the highest peak of 50% but the graph of LEC and LET always remain less than 40%. In case of Hyperloop architecture the graph of %LIF remains saturated and equal to 0% throughout the load scheduling process.

CONCLUSION

The characteristics of the proposed architecture have been discussed and its performance has been evaluated and compared with other existing multiprocessor architectures. The proposed low cost multiprocessor architecture was found to be a comparative network to work as a low cost efficient server. Number of tables have been shown for various parameters of different multiprocessor architectures and it was found that our proposed architecture is better in all respects.

To carry out the performance evaluation, a number of scheduling algorithms have been implemented on our proposed network and other existing networks as depicted in the graph. It is found that our network is performing consistently better showing lesser performance metrics, load imbalance factor (LIF). The reason behind lower LIF is the higher degree of parallelism (DOP). Higher the degree of parallelism (DOP) lower will be the LIF and vice- versa. Lesser LIF means better and higher performance, as shown in the graphs for the proposed architecture. Therefore, it can be said that the proposed Hyperloop network can perform better than the other architectures on any scheduling scheme while having lesser number of processors with lesser cost and space. Therefore it can be concluded that the proposed Hyperloop (HL) architecture can be used as an efficient server.

Acknowledgement

We are grateful and obliged to our director Professor Rajeev Saxena for his help, encouragement and advises throughout the work. We are also thankful to Professor Mohd. Qasim Rafiq for his help, technical discussions and suggestions, manya times whenever we were in need. Last, but not the least, we are very much grateful to Jaypee University for providing all the facilities and a congenial environment to complete this work.

References

[1] A. Samad, M. Q. Rafiq and O. Farooq, “LEC: An Efficient Scalable Parallel Interconnection Network,” In proceedings of International Conference on Emerging Trends in Computer Science, Communication, and Information Technology, India, pp. 453-458, 2010.

[2] A. Samad, M. Q. Rafiq and O. Farooq, “Two Round Scheduling (TRS) Scheme for Linearly Extensible Multiprocessor Systems,” International Journal of Computer Applications (0975-8887) vol. 38, no. 10, pp. 34-40, 2012.

[3] A.Samad, M. Q. Rafiq and O. Farooq, “A novel Algorithm for fast Retrieval of Information from A Multiprocessor Server,” in Proceedings of 7th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS ’08), University of Cambridge, UK, pages 68-73.

[4] A. Samad, M. Q. Rafiq and O. Farooq (2005), “A Novel Server Architecture for Networking,” in proceedings of International Conference on Robotics, Vision, Information and Signal Processing, (ROVISP2005), University Sains, Malaysia, pages 1029-1032.

[5] Z. A. Khan, J. Siddiqui, A. Samad, “A Novel Multiprocessor Architecture for Massively Parallel System,” International Conference on Parallel, Distributed and Grid Computing, 2014.

[6] M. Alam, Z. A. Khan, “Issues and Challenges of Load Balancing Algorithm in Cloud Computing Environment,” Indian Journal of Science of Technology, vol. 10(25),pp. 1=12, July 2017.

[7] A. Samad, “Performance evaluation of linearly extensible multiprocessor architectures for networking,” PhD thesis, AMU, 2009.

[8] S. Kim and A. V. Veidenbaum, “Interconnection network organization and its impact on performance and cost in shared memory multiprocessors,” Journals of parallel computing, vol. 25, pp. 283-309, 1999.

[9] N. Adhikari and C. R. Tripathy, “On a New Multicomputer Interconnection Topology for Massively Parallel Systems,” International Journal of Distributed and Parallel Systems (IJDPS), vol. 2, no. 4, 2011.

[10] Z. A. Khan, J. Siddiqui and A. Samad, “Topological Evaluation of Variants Hypercube Network,” Asian Journal of Computer Science and Information Technology, vol. 3, no. 9, pp. 125-128, 2013.

[11] K. Efe, “The Crossed Cube Architecture for Parallel Computation,” IEEE Transactions on Parallel and Distributed Systems, vol. 3, no. 5, 513-524, 1992.

[12] S. B. Akers, D. Harel and B. Krishnamurthy, “The Star Graph: an Attractive Alternative to the n- Cube,” International Conference on Parallel Processing, pp. 393, 1987.

[13] N. Adhikari and C. R. Tripathy, “Extended Crossed Cube: An Improved Fault Tolerant Interconnection Network,” Proceedings of Fifth International Joint Conference on INC, IMS and IDC, 2009.

[14] Y. Saad and M. H. Schultz, “Topological Properties of Hypercubes, ” IEEE Trans. Computer, vol. 37, no. 7, pp. 867-872, 1988.

[15] K. K. Peter, W. J. Hsu, and Y. Pan, “The Exchanged Hypercube,” IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 9, pp. 866-874, 2005.