Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

A Cluster-Based Dynamic Algorithm For Application Mapping On The Noc Architectures

Savita Gautam[1*], Abdus Samad[1], and M. Sarosh Umar[2]

[1*] *University Women's Polytechnic*
[1,2] *Department of Computer Engineering*
Aligarh Muslim University
Aligarh, India
[1*] *savvin2003@yahoo.co.in*

The effectiveness of Network on Chip (NoC) is highly dependent on how well the real-time tasks are mapped on NoC communication infrastructure. In this paper an efficient dynamic scheduling algorithm is proposed and implemented on standard Mesh topology as well as on similar recently proposed LECΔ topology. The performance of proposed algorithm is evaluated in terms of Load Imbalance Factor (LIF), makespan, communication cost and system throughput. The results are evaluated with different task distribution profile using Java-based simulator. The behavior of load distribution is estimated and compared with different performance metrices on two considered topologies. The simulation results show an improvement of 32.57% in LIF at lesser execution time. When comparing the performance of two networks in terms of makespan, the results show a reduction of 43.7% in case of 4 x 4 LECΔ topology as compared to 4 x 4 Mesh topology. The comparative study in terms of LIF, makespan and system throughput is carried out by implementing other existing techniques on both the considered NoC architectures which shows that the proposed approach is an optimized solution for task mapping at considerable reduced cost on considered NoC architectures. The analysis of simulation results shows that the proposed approach may be considered a better choice and implemented on similar NoC architectures as well as other than mesh-based topologies.

Keywords: Network on Chip, Application Mapping, Load Imbalance Factor, Mesh, Makespan, Throughput.

1 Introduction

Multicore systems on a single chip are the basic bone in advanced embedded system design. The performance of such systems is highly dependent on the efficient and advanced processes that run on multicore systems with appropriate interconnection topologies used in the system. Very Large Scale Integration (VLSI) has enabled to make use of various processor cores on a single system-on-chip (SoC). These systems are designed with communication links in the form of interconnection network for exchanging data between the interconnected cores. As the number of cores increases the complexity of the layout of interconnection topologies also increases. Therefore, choice of well-organized structure of SoC is critical and a improper selection leads to inefficient utilization of available cores in the system. The inefficient utilization of cores will ultimately effect system throughput and efficiency of complete organizational model.

NoC based architectures have been extensively studied and research is carried out on the performance improvement of topological based network models in which various cores are connected and work in co-operation on a single chip [1]. The NoC architecture accomplish the requirements of high-performance system by having desirable properties like link bandwidth, latency, throughput etc. A number of interconnection topologies has been developed based on network interconnection. Mesh is the most common topology used for design and implementation of multicore and manycore systems. The major advantage of mesh topology is having regular and scalable properties, however, when implemented for larger systems leads to complex SoCs. The longer diameter results high latency and average hop count which ultimately effects the overall system throughput. Variants of mesh have been introduced by incorporating additional virtual links or by introducing diagonal links [2-3]. Concentrated mesh (CMesh), Diagonal Mesh (DMesh), Partially Diagonal Mesh (PDMesh) and ZMesh are few examples of variant of mesh which are designed with improved characteristics in order to reduce power and area overheads [4-7].

Mesh-based NoC topology has an advantage of reliability, however, they suffer from complex routing mechanism and high energy consumption. The diagonal mesh or partial-diagonal mesh topologies have improved the performance of mesh in terms of link utilization, latency and throughput. The power and area overhead are still the issues that need to be considered while selecting the NoC architecture. The diagonal mesh topologies for NoC architecture inherit a number of desirable properties which are crucial for having promising solution for task allocation and scheduling the computational load at runtime. These topologies also bring a tradeoff in terms of link utilization, load imbalance, communication cost in the form of latency and processor idle time. Therefore, a highly parallel computation requires efficient task mapping with high data rate so that efficiency of parallel architecture could be improved. There are number of such systems which require moderate to high connectivity requirement with distributed communication in the task graph.

To improve the performance metrics, it is required that an efficient task mapping mechanism must be evolved to ensure better communication and reliability along with efficient execution of tasks on the selected core of underlying interconnection architecture of the multicore system. The present work is based on a new task mapping strategy that can be applied for computation and communication workloads on a NoC based multicore system. The key contribution of the present research are as follows:

- A triangle-based topology called Linearly extensible Cube Triangle (LECΔ) topology that exhibits desirable topological characteristics with 16-cores has been considered us underlying architecture [24].
- An efficient application mapping for 4 x4 mesh and considered LECΔ networks is proposed.
- The proposed scheme makes the task mapping optimal by applying a cluster-based approach that reduces the overall computational overhead.
- The computational overhead is evaluated and analyzed in terms of LIF, makespan and

system throughput. The simulation results show a notable improvement in performance metrices on considered NoC architectures.

- System throughput and communication cost is also evaluated which show significant improvement.

The remaining part of the paper is organized in seven sections. Section 2 presents the existing work on application mapping for NoC architectures. Section 3 gives a brief overview of underlying NoC architecture. Section 4 describes the performance analysis model, performance metrices and problem formation. The proposed approach is described in section 5. The experimental results are discussed and a comparative analysis is carried out in section 6. Section 7 concludes the paper.

2 Related Work

The goal of researchers has always been to investigate different mapping techniques that can be employed to achieve optimal performance. Different optimized solutions have been obtained for different types of NoC architectures. However, a majority of these solutions suffer from high computational overhead due to their complex nature. Application mapping is categorized based on the action time and are known as static or dynamic. In static approach, the mapping decision is taken during design stage in which computational cores are well defined. Dynamic approach is widely studied and implemented to map the tasks to the available cores of NoC architectures in which decision of action is taken at run time. Both static and dynamic approaches have their own merits and demerits. The static approaches provide better solution in terms of energy consumption and computational overhead. On the other hand, dynamic approaches are more flexible and considered better solution as decisions are taken on fly, however, they suffer from high complex computations. Research is continued to obtain simpler solutions while using dynamic approach of task mapping.
A mathematical-based linear programming is a conventional approach to solve different optimization problems. For instance, an integer linear programming (ILP) method is proposed for optimal solution which is purely a conventional approach [8]. The solution is fine but the complexity of computation increases exponentially with the increase in NoC system size. Some improvement has been made to make the computation complexity simpler by introducing a cluster-based approach [9]. This approach gives a near-optimal solution for specific NoC based systems. Another approach to minimize the computational complexity at larger data set is reported which is based on segmented brute-force mapping (SBMAP) [10]. In this approach application is divided into different modules in such a way that a systematic search could be applied to each segment or module. The segmentation technique considerably reduces the complexity of computation particularly for large data set. The approach is further improved by applying amalgam optimization technique in which initial mapping is done with a fast BB algorithm and then modular concept is incorporated [11]. The modular approach helps to optimize the initial mapping and provide better solution under certain conditions. However, due to more power consumption these approaches could not be considered as generalized solution for optimal task mapping.
With a view to improve the execution time a communication-aware list scheduling was introduced which selects the target processor while considering the execution time of task. The earliest available time and effective transmission time is taken into account for final allocation

of core in a NoC architecture [12]. The execution time is improved; however, the algorithm fails to address the deadline requirement of tasks and hence not suitable for mapping real time applications. Another approach similar to communication-aware algorithm is reported named as contention-awareness algorithm [13]. The proposed scheme is a heuristic scheduling approach that applies prediction techniques for traffic pattern and utilizes link accordingly in order to minimize the communication delay. The main objective of contention-aware algorithm is to reduce makespan for non-real time applications and does not care to meet the deadline satisfaction of tasks. Therefore, communication-aware and contention-aware algorithms though reduce the overall congestion, however, not fit to map real time applications. A similar approach without considering the deadline requirements is proposed in which the author projected an optimized solution in which communicating tasks are allocated in close vicinity of each other [14]. The communication load is scheduled on NoC systems while considering future link utilization to reduce communication cost. In this approach both internal and external congestion is taken into consideration so as to minimize the occurrences of delay which ultimately reduces the communication cost.

Communication-aware run-time mapping heuristics are proposed which are capable of improving Network-on-Chip (NoC) congestion bottlenecks and provide good solution for the efficient mapping of multiple applications onto an MPSoC platform [15]. In the presented mapping heuristics, the available resources are identified prior to recommending the adjacent communicating tasks on to the same core. The placement for a task is found by looking at previously mapped tasks onto a core in the NoC system. Furthermore, the proposed heuristics give priority to the tasks of an application in close proximity so as to additionally minimize the communication overhead. The method of mapping adjacent communicating tasks that belong to same application on to the same node whenever possible has played a substantial role in the overall reduction in the communication overhead. When implemented on 8 x 8 mesh network the author claimed a consistent reduction in communication overhead. Unlike the method used in [15] in which the previously mapped tasks were taken into account, a new approach namely Dynamic Task Mapping with Congestion Speculation (DTMCS) algorithm is reported in which decision of utilization of link is made based on upcoming traffic pattern. The DTMCS can reduce the overall congestion by applying link utilization-based speculation. Three types of benchmarks, namely random benchmark, embedded system benchmark, and multimedia benchmark are used to make the DTMCS effective [16].

To map real time applications and to meet the deadline requirements contention-awareness approach was further improved by jointly allocating the task on on-chip links and scheduling of tasks for efficient utilization of processor core [17]. In this approach the task mapping mechanism dynamically selects a route for extenuating network contention during the execution of allocated tasks present in the real time application. The proposed approach minimizes the communication latency as well as satisfies the deadline requirements of real time applications. However, the proposed approach considers a tile-based NoC platform in which a communication of processing element, router and network interface are configured to form the system. Moreover, tiles are uniform in nature having same architectural configuration and not applicable for heterogeneous system architecture.

Application mapping techniques are applied using genetic algorithm to minimize area and power requirements. A two-step task mapping using quadratic binary problem for NoC was presented in [18]. The algorithm was implemented to assign the voltage and frequency levels to each task in order to minimize the overall energy consumption. The algorithm performs well in terms of power reduction, however, increases the computational overhead. For improving accuracy and energy reduction an approximate computing approach by selective data dropping for reducing network congestion on NoC is reported [19]. The presented approach uses an online algorithm for accuracy and congestion-aware dynamic control of traffic. With the aim of obtaining low latency and energy consumption another approach that utilizes hardware-based allocation manager is reported in the literature [20]. It incorporates a dedicated manager unit using an efficient trellis search algorithm and explores contention-free routes with smaller latency between source and destination nodes in a NoC architecture. The main objective of the scheme is a guaranteed low latency and improved task allocation policy. However, this scheme has an additional overhead of hardware implementation.

From the above literature survey, it is observed that emphasis is given on the problem of computation workload assignment and lesser attention is drawn to the inter-processor communication and scheduling of application at run time. In the present work an efficient dynamic allocation and scheduling for computation and communication workloads on a NoC based multicore system is proposed.

3 NoC based Target Architecture

Topology is nothing but the connection of computing cores through which we can obtain the shortest path for communication. The topology of NoC concerns the placement and interconnection of network-on-chip nodes. Protocols of a topology specify how these nodes and links work. There are different forms of topology, including regular forms and irregular forms. Topology is the first dimension of research which is focused on the choice of communication infrastructure. The communication infrastructure design essentially points to the design of underlying hardware, acting as the backbone for the on-chip communication network. Selection of a network topology, design of router architecture, determining inter-router link and layout design are the key design aspects of this dimension. The NoC design methodology provides a flexible user-oriented network topology that declines the complexity of interconnection, separates computation and communication units and supports modularity in NoC-based systems.

There are numerous topologies that have been used to employ on-chip interconnection networks. The most prominent topologies that support QoS with effective traffic engineering are mesh topology [21], cube-based topology [22], tree topology [23] and hybrid topology [24]. An individual topology has its own advantages and disadvantages in terms of NoC parameters. Mesh topology is often used in the fabrication of numerous NoCs due to its simple VLSI implementation and therefore considered for evaluation. A 4 x 4 mesh topology used for performance evaluation is shown in Figure 1(a). Hybrid topology used in this work is a scalable interconnection network which exhibits the desirable characteristics of cube-based networks and having lesser complexity at higher level of architecture [24]. For basic 4 x 4 LECΔ topology, the arrangement of 16 cores with 4 nodes is shown in Figure 1(b). Similar pattern could be extended for higher levels of architecture. The simulation is carried out for n x n system where n = 4 representing the number of cores on a particular node N. The total number of cores in the system is 16. Tasks are generated on Head Node $C_{0,0,0}$ core among a set of four such nodes. As a result of task generation, the communication associated between $C_{0,0,0}$ through $C_{0,1,0}$, $C_{0,2,1}$ and $C_{0,3,2}$ by means of available links. Each directed link represents a communication path between different available nodes where each node consists of four cores thus formulating a 16-core system. Each directed link represents a communication path between different available nodes where each node consists of four cores thus formulating a 16-core system. Similar approach is applied for n x n mesh network in which NoC architecture Graph is represented as a directed graph G (C, E), where C = {C_0, C_1, …. C_n} is a set of vertices of n x n mesh architecture and E denotes the number of links in n-core forming a cluster.

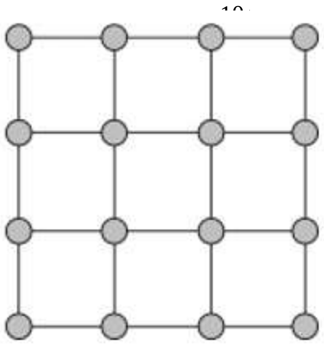
Fig. 1(a). 4 x 4 Mesh Topology

C000, C010, C021, C032 C100, C110, C121, C132
C200, C210, C221, C232 C300, C310, C321, C332

Fig. 1(b). Arrangement of Cores in 4 x 4 LECΔ Topology

The dimension and size of the network depends upon the total number of links in any NoC topology. At base level with direct link, the latency in LECΔ is equivalent to the minimum distance. Therefore, the considered topology gives optimal performance at base level. The considered topology has a constant node degree equal to four, lesser diameter and smaller average distance which makes the network efficient for mapping run-time application. For higher levels the LECΔ scales uniformly while preserving the topological properties.

4 Performance Analysis Model

4.1 Problem Formulation

Application mapping is done by dividing the NoC graph into four clusters. Tasks are generated in an unpredictable manner. The initial population is handled by node $C_{0,1}$ where balancing of tasks is carried out among available number of cores. The fitness of each task graph is evaluated based on a threshold value know as ideal load (IL). Communication to the directly connected node takes place only when all the tasks on the source node have been received and evenly distributed among available cores. The decision of migration of task is based on a threshold value know as ideal load (IL). Ideal load is defined as the average load on a particular node. There are direct links available between different nodes (clusters) through which the communication packets are routed based on the value of IL. The ideal load depends upon total load on network and the number of available cores (n) for each cluster.

$$IL_i = [loadk(C_0)+loadk(C_1)+\ldots+loadk(C_{n--1})] / n \quad (1)$$

After mapping the tasks to all the available cores in a cluster the load imbalance is evaluated as Load Imbalance Factor (LIF) at a particular stage of task structure. LIF is calculated as.

$$\text{LIF} = [\max\{\text{load}_i(Ci)\}-(IL)i] / (IL)i \qquad (2)$$

Maximum load denoted as $\max\{\text{load}_i\}$ is the value of maximum load on a particular core C_i, where, C_i ,$0 \le i \le n-1$. For the same stage of task structure, the makespan is evaluated. The makespan is the total time required for task mapping to a cluster that produces an acceptable value of the LIF.

4.2 Performance Evaluation Metrices

The following mathematical models are used for the assessment of load Imbalance factor, communication cost, throughput to evaluate the performance of 4*4 NoC architecture.

The average LIF of the network is represented by

$$\text{LIF}_{av} = [\ [\ \textstyle\sum_{j=1}^{n} M_j\ ] - IL\ ] \qquad (3)$$

$$M_{j=} \max\ [CL\ (1...n)]$$

(4)

The communication cost is calculated as

$$\text{COST}=\textstyle\sum_{i=1}^{N}\sum_{j=1}^{N} B\ pi.pj * D\ pi.pj$$

(5)

The throughput is represented by

$$\text{Throughput} = \frac{1}{T\ S}\textstyle\sum_{I=1}^{N} DP$$

(6)

5 Proposed Approach

The proposed strategy considers task graph in the form of low to moderate load. Task generated in the range of 0 to 5000 are considered as low volume and tasks greater than 5000 are considered as moderate. The algorithm selects a segment of underline architectures as a cluster and task mapping progress dynamically from cluster to cluster covering the entire network. This is achieved by (i) selecting a cluster for executing the communicating tasks while considering the IL for underlying network, (ii) scheduling the allocated task on different cores of the cluster, (iii) determining a route for the data communication using active and able links and (iv) scheduling the task on adjacent clusters covering the entire network links.

The whole algorithm is developed using Java platform and described an Algorithm 1 and Algorithm in the subsequent section. Table 1 depicts the notation used for the representation of different criterion used in algorithms.

Table 1. Symbols for algorithm

Notation	Description
LS	Server Load
NC	Number of Cores
AD_c	Average Distribution of load on cluster
IL	Ideal Load
L_c	Load on particular core
FL_c	Final Distribution of load on core
LIF	Load Imbalance Tactor
T_{sim}	Total Simulation Time
NC_L	Number of cluster
DP_{NW}	Load on network
L_H	Head Node within a cluster
$P_i.P_j$	i and j are designated cores

5.1 Allocation Strategy

Algorithm 1: Cluster Balancing

```
        Input  : NC , $L_c$ , $DP_{NW}$
    Output : $FL_C$ , Lif. $T_{sim}$
  Calculate quantity of IL using value of $DP_{NW}$
        IdealLoad IL = $AD_c$ / NC
IF $L_c$ is less than $_I$IL
     $L_c$= $L_c$ +1
     $L_H$ =$L_H$ -1
Repeat above 2 steps until $L_H$ is greater than IL and
                $L_c$ is less than  IL
Task Migration from Heavy burden to idle core
Calculate LIF using equation 3
Calculate Simulation Time
```

Algorithm 2: Calculate and update status **Task Generation**

```
  Generate task and Calculate $DP_{NW}$
     For(i=1;i<=NC;i++)
```

```
        Check for active and able links

    For(i=1;i<=NC;i++)
        Generate connectivity status $N_L$

Calculate the distribution of Tasks on cluster ADc
    $AD_c = DP_{NW}//NC_L$
```

6 Result and Discussion

The initial population of task graph is generated and mapped on the cluster of four cores. As the population of task increases, the connectivity of other nodes forming a cluster is checked and task migration takes place in the subsequent steps thus mapping the whole task graph. In the following sub-section, the test setup along with the simulation results for evaluating the performance of the proposed algorithm are described. Three performance parameters namely LIF, Makespan and communication cost are evaluated for standard mesh and LECΔ networks. Analysis of throughput is also carried out after simulation.

6.1 Simulation Setup

Experiments are performed with java simulation platform developed for mapping and scheduling on NoC architectures. For simulation purpose, applications are selected randomly from the set of generated applications. However, for simplicity we have considered uniform task structures that consists of independent tasks and can be run on any available core. We use an Intel Core i7-10750H system with 6-cores operated on 2.66 GHz of clock frequency.

6.2 Performance Evaluation on LIF

Firstly, the application task based on computational model for 512 task and 16-cores is generated randomly using the following specification. The task graph is divided into two possible groups from low to moderate load. One task graph at low level consists of around 512 uniform tasks. The Load Imbalance Factor is evaluated for 4 x4 Mesh and LECΔ networks and curves are plotted as average LIF against three task graphs. Experimental results are first evaluated for proposed algorithm by applying on standard 4 x 4 mesh network as well as with two well-known algorithms namely Communication-Aware Nearest Neighbour (CA-NN) and Dynamic Task Mapping with Congestion Speculation (DTMCS) algorithms. The results of the comparison for LIF are demonstrated in figure 2(a). In the first experiment, task sizes ranging from 500-1000-10000 are considered for task allocation and the percentage of the allocated tasks satisfying their balancing conditions are evaluated. We observe from figure 2(a) that for a given size of task graph the value of LIF initially is same for mesh network, however, the value of LIF drastically reduces with the increase in task structure. The value of LIF for more than 5000 tasks is quite less and sustainably remain lesser for high volume of task structure which is a significant improvement over CA-NN and DTMCs algorithms.

104

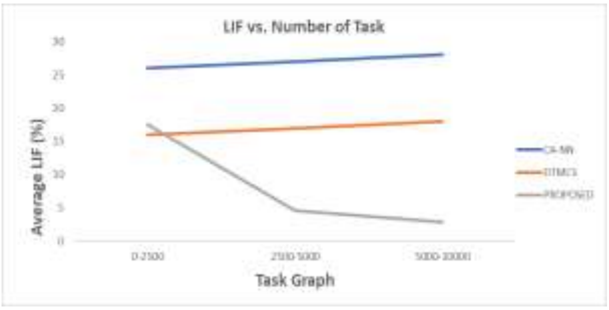

Fig. 2 (a). Performance of Proposed Algorithm in terms of LIF

In the second experiment, similar results are obtained for LECΔ network by applying proposed algorithm and are compared with standard 4 x 4 mesh and the same are demonstrated in figure 2 (b). It is observed from the curves that the average number of allocated/scheduled tasks when mapped on LECΔ network topology by applying the proposed method improves the LIF by 32.57% as compared to conventional mesh network implementation.

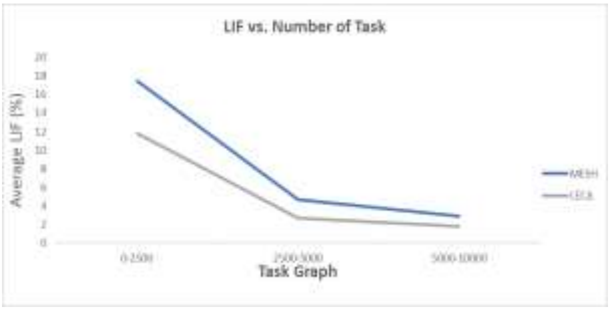

Fig. 2(b). Performance of Proposed Algorithm in terms of LIF

6.3 Performance Evaluation on Makespan

Efficiency of a NoC architecture depends on optimal migration of tasks by finding out the best execution sequence. In order to minimize the makespan of overall execution, the CA-NN and DTMCS algorithms and the proposed technique are applied and the results are evaluated in msec. The main objective of these algorithms is to find the allocation method that minimizes the execution time while satisfying the load balancing criterion. The value of makespan is measured for both the considered networks in analogy to LIF and demonstrated in figure 3(a) and figure 3(b) respectively.

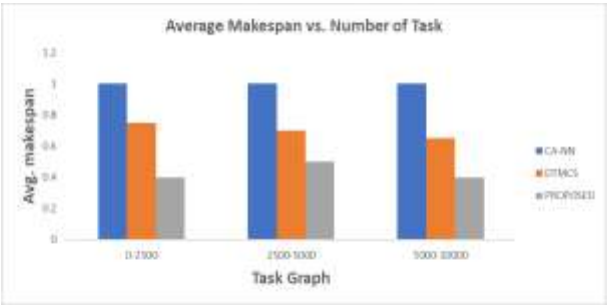

Fig. 3 (a). Performance of Proposed Algorithm in terms of makespan

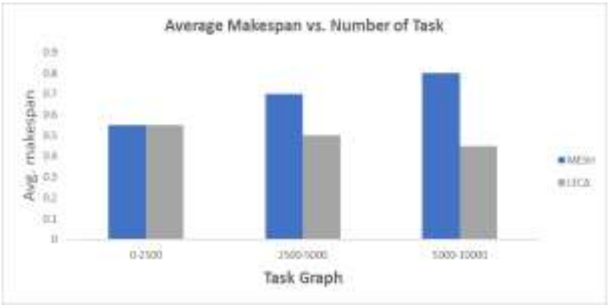

Fig. 3(b). Performance of Proposed Algorithm in terms of makespan

Makespan is the time taken consumed during the execution of all the applications to be mapped on the NoC platforms. The computation of a packet corresponding to a task mapped on a cluster of cores starts just after its receiving and the computation is finished before receiving the next packet for the same task. It is required that computation time of smaller tasks should not absorbed with communication time. Therefore, computation time should always be greater than the communication time. It is observed from the figure 3(a) that makespan with proposed algorithm is significantly smaller as compared to other considered approaches. All the considered algorithms consuming almost in similar time with smaller number of tasks in application. However, in cases with high volume of task structure the results of LECΔ network reporting a reduction of 43.7% as compared to mesh network when proposed scheme is implemented. Results for different task graphs for both the considered networks are demonstrated in figure 3(b).

6.4 Performance Evaluation on Communication Cost

Communication cost can be evaluated as the accumulative cost when executing a particular application on the NoC platform after mapping on a network. Comparison of communication cost of the proposed approach is presented in figure 4. The cluster-based technique helps to map the initial population largely on highly communicating cores which are at minimum distance to each other. As the process repeats iteratively, it generates new quality solutions.

The proposed scheme provides a base for fast convergence of the algorithm with a variety of task structures, making it more efficient. It can be concluded from the results shown in figure 4 that the proposed approach outcomes a cost saving of 42.8% for LECΔ network as compared to conventional mesh network.

10

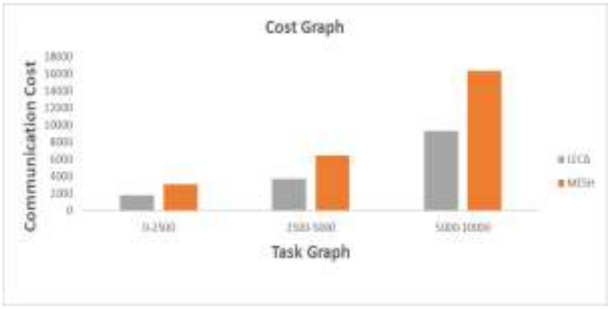

Fig. 4. Performance of Proposed Algorithm in terms of Communication Cost

6.5 Throughput Performance

Throughput is measured as the total number of packets received by the network in a unit of time. Throughput is also considered a measure constraint while evaluating the performance of a NoC system. An efficient mapping method is a foundation for getting high throughput value. Like makespan, throughput is also dependent on the task communication and the execution path a packet follows. The longer path a packet travels will always lead a low system throughput. The proposed approach maps heavily communicating tasks with minimum distance, which ultimately reduces the packet traversed path in the network. Hence, it delivers high average network throughput. When applied on LECΔ and mesh networks the proposed approach provides almost similar results with some improvements in some cases. Figure 5 shows the average network throughput for three set of task structures. The result indicates the effectiveness of the proposed algorithm for mesh-based systems as well.

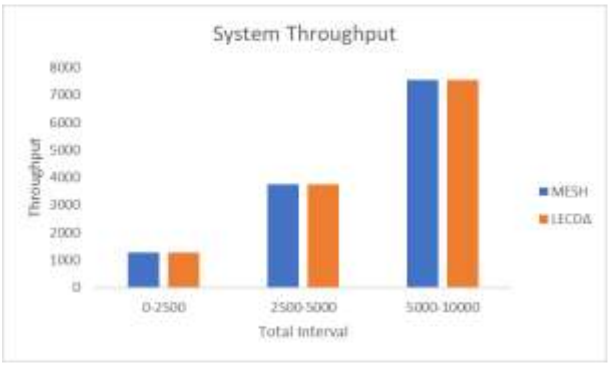

Fig. 4. Performance of Proposed Algorithm in terms of Communication Cost

7 Conclusion

In the presented work, an improved approach for dynamic task allocation and scheduling for NoC based multicore systems is proposed. The proposed algorithm is a cluster-based approach which utilizes the available links of the target multicore platform while selecting an appropriate core for task execution. It is dynamic in the sense that it selects a route for underlying network during the execution of real-time applications. The performance of the proposed algorithm is evaluated and compared with other dynamic task and communication mapping and scheduling algorithms namely Communication aware Nearest Neighbour (CA-NN) and Dynamic task mapping with Congestion Speculation (DTMCS). The proposed algorithm along with the two considered algorithms are applied on standard 4 x 4 mesh and recently reported Linearly Extensible Cube Triangle (LECΔ) networks. Experimental results demonstrate the effectiveness of the developed approach in terms of Load imbalance factor (LIF), makespan and throughput. The proposed algorithm is performing better and producing a 32.6% improvement in LIF when implemented on 4 x 4 LECΔ network. Communication cost is also evaluated with all the considered approaches and a reduction of 43.7% is reported for considered network. The proposed algorithm When applied on LECΔ and mesh networks the proposed approach provides almost similar results in terms of throughput. Hence, it can be concluded that the proposed approach a good solution for task mapping in NoC based systems.

References

1. Marculescu, R., Ogras, Ü.Y., Peh, L., Jerger, N.D., & Hoskote, Y.V. (2009). Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp 28, 3-21. doi:10.1109/TCAD.2008.2010691
2. Varanasi, L.K., Srinivasarao, B.K.N. (2022) Design and evaluation of an energy efficient DiamondMesh topology for on-chip interconnection networks. Des Autom Embed Syst 26, pp 161–187. https://doi.org/10.1007/s10617-022-09266-0
3. Tatas, Konstantinos & Siozios, Kostas & Soudris, Dimitrios & Jantsch, Axel. (2014). Designing 2D and 3D Network-on-Chip Architectures. 10.1007/978-1-4614-4274-5
4. Xu, T.C., Leppänen, V., Liljeberg, P., Plosila,j., and Tenhunen, H. (2015) PDNOC: Partially Diagonal Network-on-Chip for high efficiency multicore systems, Concurr Comput.: Pract. Exper. 27 (4). 1054–1067. http://dx.doi.org/10.1002/cpe.3364.
5. Prasad, N. & Mukherjee, P. & Chattopadhyay, S., and Chakrabarti, I. (2017). Design and Evaluation of ZMesh Topology for On-Chip Interconnection Networks. Journal of Parallel and Distributed Computing. pp. 17-36. 113. 10.1016/j.jpdc.2017.10.011.
6. Wang C., and Bagherzadeh, N. (2012) Design and Evaluation of a High Throughput QoS-Aware and Congestion-Aware Router Architecture for Network-on-Chip," 2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing, Munich, Germany, pp. 457-464, doi: 10.1109/PDP.2012.20.
7. Kundu, S. & Chattopadhyay, S. (2014). Network-on-chip: The next generation of system-on-chip integration. 10.1201/b17748.
8. Tosun, S., Ozturk O., Ozen, M. (2009) An ILP formulation for application mapping onto Network-on-Chips. International Conference on Application of Information and Communication Technologies, Baku, Azerbaijan. pp. 1-5, doi: 10.1109/ICAICT.2009.5372524

9. Tosun, S. (2011) Cluster-based application mapping method for Network-on-Chip. Advances in Engineering Software. 42. 868-874. 10.1016/j.advengsoft.2011.06.005.
10. Khan, S., Anjum, S., Gulzari, U.A., Umer, T., and Kim, B. S. (2018) Bandwidth-Constrained Multi-Objective Segmented Brute-Force Algorithm for Efficient Mapping of Embedded Applications on NoC Architecture. IEEE Access, vol. 6, pp. 11242-11254. doi: 10.1109/ACCESS.2017.2778340.
11. Khan, S., Anjum, S., Gulzari, U.A., Afzal, M.K., Umer, T., and Ishmanov, F. (2018) An Efficient Algorithm for Mapping Real Time Embedded Applications on NoC Architecture. IEEE Access, vol. 6, pp. 16324-16335. doi: 10.1109/ACCESS.2018.2811716.
12. Chao, H.L., Chen, Y.R., Tung, S.Y., Hsiung, P.A., and Chen, S.J. (2012) Congestion-aware scheduling for NoC-based reconfigurable systems. Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany. pp. 1561-1566, doi: 10.1109/DATE.2012.6176721. a
13. Singh, Amit & Jigang, Wu & Kumar, Akash & Srikanthan, Thambipillai. (2010). Run-time mapping of multiple communicating tasks on MPSoC platforms. Procedia Computer Science. 26. 1019-1026. 10.1016/j.procs.2010.04.113.
14. Fattah, M., Ramirez, M., Daneshtalab, M., Liljeberg, P., and Plosila, J. (2012) CoNA: Dynamic application mapping for congestion reduction in many-core systems. IEEE 30th International Conference on Computer Design (ICCD), Montreal, QC, Canada, pp. 364-370. doi: 10.1109/ICCD.2012.6378665.
15. Singh, Amit & Srikanthan, Thambipillai & Kumar, Akash & Jigang, Wu. (2010). Communication-aware heuristics for run-time task mapping on NoC-based MPSoC platforms. Journal of Systems Architecture. 56. 242-255. 10.1016/j.sysarc.2010.04.007
16. Chao, H., Tung, S., & Hsiung, P. (2016). Dynamic Task Mapping with Congestion Speculation for Reconfigurable Network-on-Chip. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 10, pp. 1-25. doi.org/10.1145/2892633
17. Suraj, P., Chatterjee, N., and Ghosal, P. (2021). Dynamic Task Allocation and Scheduling with Contention-Awareness for Network-on-Chip based Multicore Systems. Journal of Systems Architecture. 115. 10.1016/j.sysarc.2021.102020.
18. Li, D. and Wu, J. (2016) Energy-efficient contention-aware application mapping and scheduling on NoC-based MPSoCs. Journal of Parallel and Distributed Computing, vol. 96, pp. 1-11. doi.org/10.1016/j.jpdc.2016.04.006
19. Xiao, S., Wang, X., Palesi, M., Singh, A.K., T. Mak, T., and Acdc. (2019): An accuracy- and congestion-aware dynamic traffic control method for networks-on-chip, Design, Automation Test in Europe Conference Exhibition (DATE), pp. 630–633, http://dx.doi.org/10.23919/DATE.2019.8715189
20. Chen, Y., Matus, E., Moriam, S., and Fettweis, G.P. (2020). High performance dynamic resource allocation for guaranteed service in network-on-chips, IEEE Trans. Emerg. Top. Comput. Vol. 8, no. 2, pp. 503–516.
21. M. Khari, R. Kumar, D.-N. Le, and J. M. Chatterjee, “Interconnect Network on Chip Topology in Multi-core Processors: A Comparative Study,” Int. J. Comput. Netw. Inf. Secur., vol. 9, no. 11, pp. 52–62, Nov. 2017, doi: 10.5815/ijcnis.2017.11.06.
22. A. Samad, J. Siddiqui, and Z. A. Khan, “Properties and Performance of Cube-Based Multiprocessor Architectures,” Int. J. Appl. Evol. Comput., vol. 7, no. 1, pp. 63–78, Jan. 2016, doi: 10.4018/IJAEC.2016010105.
23. S. Gautam and A. Samad, “Properties and performance of linearly extensible multiprocessor interconnection networks,” Commun. Comput. Inf. Sci., vol. 839, pp. 3–12, 2019, doi: 10.1007/978-981-13-2372-0_1.
24. Gautam, S. Umar, M.S., and Samad, A. (2021) Linearly Extensile System based Interconnection

Network for Multi-core Multi-Processor Systems. In Proc. 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom), pp. 870-874. doi: 10.1109/INDIACom51348.2021.00156.