Mapping Algorithms for Network-on-Chip

Project Report submitted in partial fulfillment of the requirement
for the degree of

Master of Technology
in
Computer Science and Engineering

Submitted By

Arvind Kumar (132222)

Under the Supervision of

Dr. Vivek Kumar Sehgal

Department of Computer Science and Engineering
Jaypee University of Information and Technology
Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled **Mapping Algorithms for Network-on-chip**, submitted by **Arvind Kumar** in partial fulfillment for the award of degree of Master of Technology in Computer Science and Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

DATE:

SUPERVISOR:
Dr.Vivek Kumar Sehgal
Associate Professor
Department of CSE and IT

Acknowledgement

I **Arvind Kumar** would like to thank **Prof. Dr. RMK Sinha (Dean CSE and IT)** and **Prof. Dr. S.P Ghrera (Head, Dept. of CSE)** for all the support and guidance they have provided to me during my M.Tech programme. I am thankful for his continuous motivation and encouragement provided to me at every point in time.

I would like to thank **Dr. Vivek Kumar Sehgal (Associate professor, Department of CSE and IT)** for offering me the opportunity to do my post graduation project under his supervision at Jaypee University of Information Technology. In the meetings and discussions, he always gave me right advice and he directed my research in the right direction.

I would also like to thank **Dr. Pardeep Kumar (Associate M.Tech Research Coordinator)** for his valuable guidance and motivation. I am thankful to my seniors and friends for their support and help provided during the completion of the thesis. I would sincerely like to thank all the librarian for their support and help by providing me the study material.

Finally, I would like to express my profound thanks to my parents for teaching me how to soar. I would not have made it this so far without their unbounded love, guidance, support and most importantly their prayers.

DATE:

ARVIND KUMAR(132222)
M.Tech C.S.E. 2ndyr

Declaration

I hereby declare that the thesis entitled **”Mapping Algorithms for Network-on-Chip”** submitted by me for the award of degree of Master of Technology in Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat is original and it has not submitted previously to this or any other university for any degree or diploma. Signature of

Student :
Name of Student :
Date :

Abstract

Network on chip is gaining popularity as the time is going on, due to increasing communication. As the number of transistors growing the network on chip is becoming complex. To reduce the complexity, 3D NoC was there but still improvements are required as more the number of wires more is the complexity and lesser is the speed involved. So the aim of this research is to find such an mapping algorithm which can map the cores or task to a topology in such a way that less energy is consumed, less time for communication, faster speed of accessing, cores with maximum communication are brought closer making the network less complex. To achieve these properties, the existing mapping approaches are mentioned in this report. Keeping the drawbacks of existing approaches in mind my aim is to find such a mapping technique which suits all the QoS requirements and is best among all the existing approaches.

Contents

Certificate **i**

Acknowledgement **ii**

Declaration **iii**

Abstract **iv**

1 Introduction **1**

1.1 Structure of Router in NoC 4

1.2 XY Routing Algorithm 4

1.3 Communication Task Graph 7

1.4 Types of graphs used in NoC 7

1.4.1 Application characterization graph (APCG) 7

1.4.2 Architecture Characterization Graph(ACG) 8

2 Literature Review **10**

2.1 Research Papers 10

2.1.1 An Energy-Aware Methodology for Mapping and Scheduling of Concurrent Applications in MPSoC Architectures 10

2.1.2 Spiral: A heuristic mapping algorithm for network on chip 11

2.1.3 Bandwidth-aware application mapping for NoC based MPSoCs 12

2.1.4 GA-MMAS: An energy aware mapping algorithm for 2D network on chip 13

2.1.5 A3MAP: Architecture-aware analytic mapping for network on chip . . 14

2.1.6 Heuristic for routing and spiral run-time task mapping in NoC based heterogeneous MPSoCs 17

3 Problem Statement **18**

4 Existing Mapping Algorithm **22**

4.1 Random Mapping Algorithm 22

5 Proposed Approach **25**

5.1 Horological Algorithm 25

5.2 Rotational Algorithm 29

5.3 Divide and Conquer Mapping Algorithm 30

5.4 Energy Model 34

5.5 Latency Model 36

6 Simulator Tool **39**

6.1 OMNeT++ 39

6.2 Orion 2.0 40

7 Design Analysis **43**

8 Experimental Results **45**

9 Conclusion **65**

10 Future Work **66**

Bibliography **67**

List of Publications **70**

List of Figures

1.1 Moore’s Law for increasing number of transistors 2

1.2 Networks on Chip Architecture 3

1.3 2D router architecture 5

1.4 3D router architecture 6

1.5 Communication Task Graph 8

1.6 Application Characterization Graph 9

2.1 Spiral Mapping. 12

2.2 Cross-Over Order 14

2.3 A3MAP 15

2.4 A3MAP Matrices 15

2.5 A3MAP-SR 16

2.6 A3MAP-GA 16

2.7 Task placement with Spiral Strategy. 17

3.1 Logical Application Trace graph 19

3.2 NoC Architecture Characterization Graph 19

3.3 NoC Mapping Technique 20

3.4 Flowchart representation of application mapping onto topology 21

4.1 Load balancing by random mapping algorithm in 2D NoC 23

4.2 Load balancing by random mapping algorithm in 3D NoC 24

5.1 Horological mapping algorithm 27

5.2 Load balancing by horological mapping algorithm in 2D NoC 28

5.3 Load balancing by horological mapping algorithm in 3D NoC 28

5.4 Load balancing by rotational mapping algorithm in 2D NoC 30

5.5 Load balancing by rotational mapping algorithm in 3D NoC 31

5.6 Grid Divison into sub-grid 32

5.7 Divide and conquer mapping algorithm 33

5.8 Load balancing by divide and conquer mapping algorithm in 2D NoC 34

5.9 Load balancing by divide and conquer mapping algorithm in 3D NoC 35

5.10 NoC topology tile 37

5.11 Latency flow in single hop 37

5.12 Latency flow in two hop from source to destination core 38

6.1 OMNeT++ execution flowchart. 41

6.2 Orion execution flowchart. 42

7.1 Mesh Topology 43

7.2 3D Mesh Topology 44

8.1 Average latency (in ns) of mapping algorithms in mesh topology 46

8.2 Queuing Time in mesh topology in random mapping 46

8.3 Queuing time in mesh topology in horological mapping 47

8.4 Queuing time in mesh topology in rotational mapping 47

8.5 Queuing time in mesh topology in divide and conquer mapping 48

8.6 Total queuing time (in ns) of mapping algorithms in mesh topology 48

8.7 Service time in mesh topology in random mapping 49

8.8 Service time in mesh topology in horological mapping 49

8.9 Service time in mesh topology in rotational mapping 50

8.10 Service time in mesh topology in divide and conquer mapping 50

8.11 Total service time (in ns) of mapping algorithms in mesh topology 51

8.12 Energy consumption of cores in random mapping algorithm 52

8.13 Energy consumption of cores in sequential mapping algorithm 53

8.14 Energy consumption of cores in rotational mapping algorithm 53

8.15 Energy consumption of cores in divide conquer algorithm 54

8.16 Comparison of energy consumption (in pJ) of mapping algorithms 54

8.17 Average latency (in pJ) in mapping algorithm for 3D mesh topology 56

8.18 Queuing time (in ns) in 3D mesh topology in random mapping 57

8.19 Queuing time (in ns) in 3D mesh topology in horological mapping 57

8.20 Queuing time (in ns) in 3D mesh topology in rotational mapping 58

8.21 Queuing time (in ns) in 3D mesh topology in divide and conquer mapping . . . 58

8.22 Total queuing time (in ns) of mapping algorithms in 3D mesh topology 59

8.23 Service time (in ns) in 3D mesh topology in random mapping 59

8.24 Service time (in ns) in 3D mesh topology in horological mapping 60

8.25 Service time (in ns) in 3D mesh topology in rotational mapping 60

8.26 Service time (in ns) in 3D mesh topology in divide and conquer mapping 61

8.27 Total service time (in ns) of mapping algorithms in 3D mesh topology 61

8.28 Energy Consumption (in pJ) in Random mapping algorithm for 3D mesh topology 62

8.29 Energy Consumption (in pJ) in horological mapping algorithm for 3D mesh topology 62

8.30 Energy Consumption (in pJ) in rotational mapping algorithm for 3D mesh topology 63

8.31 Energy Consumption (in pJ) in divide and conquer mapping algorithm for 3D mesh topology 63

8.32 Comparison of energy consumption (in pJ) of mapping algorithms in 3D mesh topology 64

List of Tables

8.1 Energy of router (in pJ) at different load 51

8.2 Energy of link (in pJ) at different load and link length (in mm) 52

8.3 Comparison of average Latency, total queuing time and total service time of mapping algorithms (in ns) 55

Chapter 1

Introduction

Network on chip is concentrated over the routing of the packets instead of the wires. It is a communication network which is basically layered architecture [1]. Packets are the major source for NoC to transfer the data from the source node to the destination node. Processing elements are basically IP cores (Intellectual Property), processor, DSP(Digital Signal Processor), ASIC(Application Specific Integrated Circuit) etc. [2]. NoC is a tile based architecture and basically consists of processing element, routers and links which provide interconnection [3, 4, 5]. NoC basically concentrates on parallelism, concurrency and scalability. There are a large number of NoC applications and few among them are point-to-point signal wires, shared bus and segmented bus [6]. Traditional exchanges of data signal between IP cores are replaced by packets routed through the fabric router [7, 8].

Issue of bottleneck is the biggest challenge in front of the System-on-chip due to global interconnections. For achieving low latency, high bandwidth and scalability [9] simultaneously, the basic concentration of NoC is the shrinking of the size of the chip, and making it as small as possible. Bus based architecture [10] which was used traditionally dosen't proves to be reliable architecture due to improper scalability and lack of parallelism, high latency low power dissipation along with low throughput. Future system on chip has considered network on chip to be the best solution for all the issues occurring in it [11] [12]. A large number of processors can process the tasks parallel on them hence networks on chip provides scalability and parallelism to a great extent.

The problems which were avoided during the consideration of system on chip or are considered

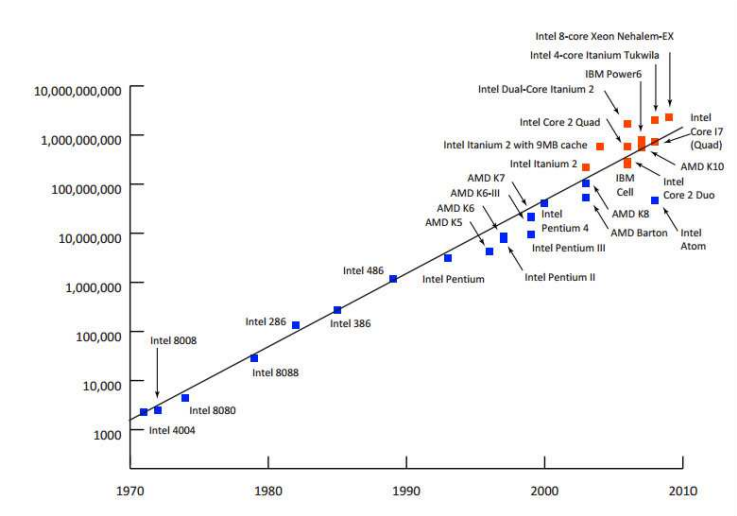

Figure 1.1: Moore's Law for increasing number of transistors

at the nano scale level are also tackled in networks on chip as an important constraint for performance evaluation. Researchers are developing new techniques to improve the performance of the mesh topology as well as they are concentrating on those architectures which are basically application specific and are build keeping in mind the requirements of the application. But there are some limitation in the NoC architecture too and even either providing a large scale of parallelism and scalability NoC cannot be proved to be the best solution. The issues related to NoC architecture are high power consumption, high cost communication,high energy consumption and low throughput. To resolve the issues of NoC researchers proposed the concept of 3D Networks on chip, which can resolve the issues of basic 2D NoC, as 3D NoC is capable of providing low power requirements [13], low energy consumption [14] [15] and high speed. From the Moores law as shown in Fig.1.1, in the past few decades a tremendous increase in the number of transistors over the chip is observed, which is continuously increasing with a rapid rate day by day, due to this shrinking the size of the chip without effecting the performance of the system is the biggest issue to be tackled [16]. As the number of components increase, to achieve scalability with such a large number of components it has become more complex to achieve the performance goals.

Due to this large size there are many issues being faced such as power dissipation, resource management [17] [18] etc. So interconnection networks have initiated the determining of the performance and also the power consumption of the entire chip, and is working on these parameters to enhance the performance of the system. Lack of scalability, concurrency / parallelism, high latency and power dissipation, and low throughput has made the traditional bus architec-

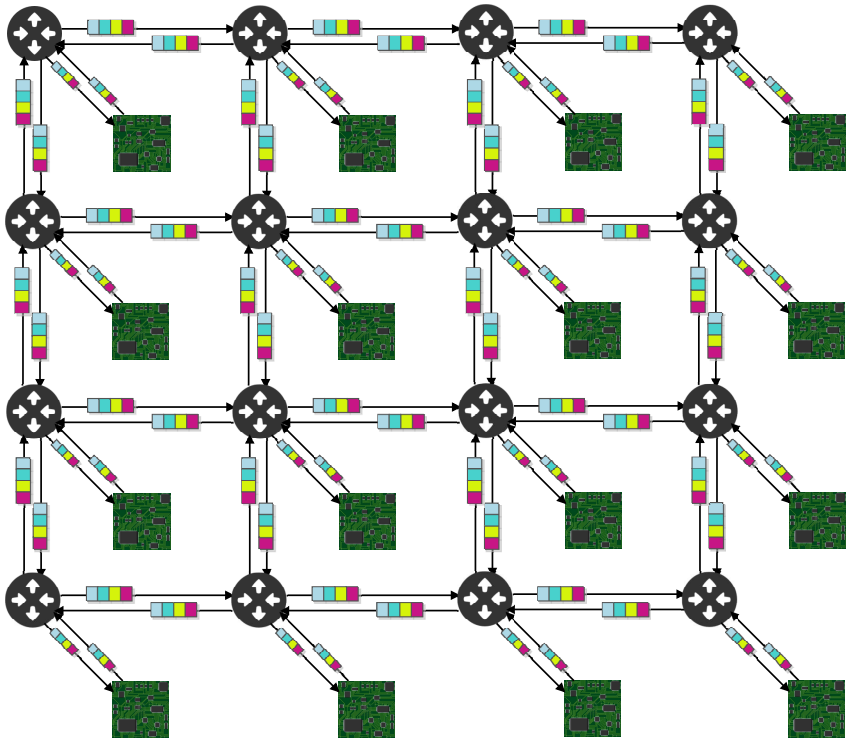

Figure 1.2: Networks on Chip Architecture

ture and point-to-point architecture a non reliable system on chip architecture. But all these issues are best resolved in the Network on chip, hence it proves to be more promising solution over system on chip [19] [20]. Hop-by-hop basis is the basis is considered for switching packets, whereas Networks on chip connects processors, memories and designs for providing customization and hence provide high bandwidth and high performance [21] [22]. As shown in Fig.1.2, NoC architectures are basically concentrated upon connection of the segment (or wires) and switching blocks. Switching techniques used in NoC are circuit switching, packet switching, wormhole switching, virtual cut-through (VCT) switching etc. [23, 24, 25, 26, 27, 28, 29, 30]. The main aim of this research is to map the task on the most suitable core for processing in both 2D and 3D. For mapping the task on to some core there is requirement of the various routing algorithms, which treat task in the form of packet and route the task to the particular core chosen for mapping [31] [32] [33] [34]. Task migrate over different routers and cores in order to be mapped on the suitable core. For 2D NoC a 5 port router [35] is basically used whereas for 3D NoC a 7 port router is used.

1.1 Structure of Router in NoC

In NoC different structure of router are used for both 2D NoC and 3D NoC architecture. For 2D NoC, routers used for routing purpose have 5 ports which include north_port, south_port, east_port, west_port and core_port. The north port is used by the router to send the packet if the core on which task has to be mapped lies on the north direction link of the router. Similarly for sending task on to the cores in the south, east and west direction the corresponding ports of the router are used. Fig. 1.3. In the Fig. 1.3 each port has two links one is incoming link to the port while other one is a outgoing link from the port. There is a 3D router which is used to route the task to the core on which the task has to be mapped using 7 ports. This router is basically designed for 3D NoC as shown in Fig. 1.4. It has all the 5 ports as was there in 2D routers for NoC, and along with these 5 routers there are 2 more ports which include top_port and a bottom_port. For convenience in the Fig. 1.4 top_port and bottom_port configuration are shown by different colors. Different colors doesn't imply that there is some difference in the port, but different colors are used to show the difference between the 2D and 3D router.

1.2 XY Routing Algorithm

The most commonly used routing algorithm is XY routing is proposed by the author in his paper [36]. Author has considered that packet consist of header, tail and data flits. Header flit contains the address of the destination node. For the purpose of implementation 2-D mesh topology along with wormhole switching technique are considered. The co-ordinates of each router are represented as (x,y),for the routing current address (C_x, C_y) is compare with the destination router address (D_x, D_y) of the packet ,depends up on the comparison output of routing algorithm router routes the packets. if $(D_x > C_x)$ head flit moves to East else it take West turn upto $(D_x C_x)$ become equal this passion is called as horizontal alignment .Now (D_y, C_y) undergoes compression , if it is found that $(D_y < C_y)$ then packet's header flit moves towered South else North upto $(D_y = C_y)$.For simulation an environment maintained as the packet size is 8 bytes with a random destination mode , the percentage load is 50% which mean that 50% of maximum bandwidth is used, the interval between successive flits is 2 clock cycles, the simulation runs 1000 clock cycles and the clock frequency is 1 GHz, Synthetic traffic generators generate traffic in the first 300 clock cycles with warm-up period of 5 clock cycles. After simulation authors define an average performance parameter P to evaluate the average performance of the algorithm that is P where P= Average Throughput of Network/ Average Latency per packet of Network

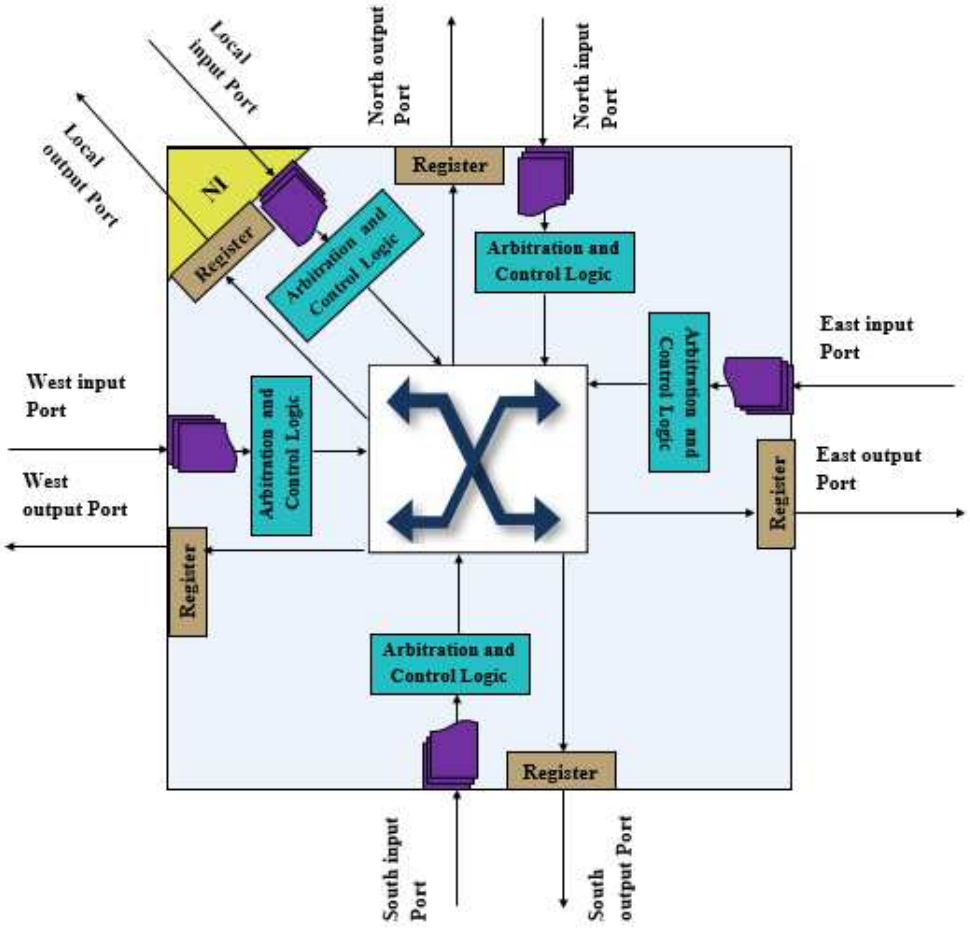

Figure 1.3: 2D router architecture

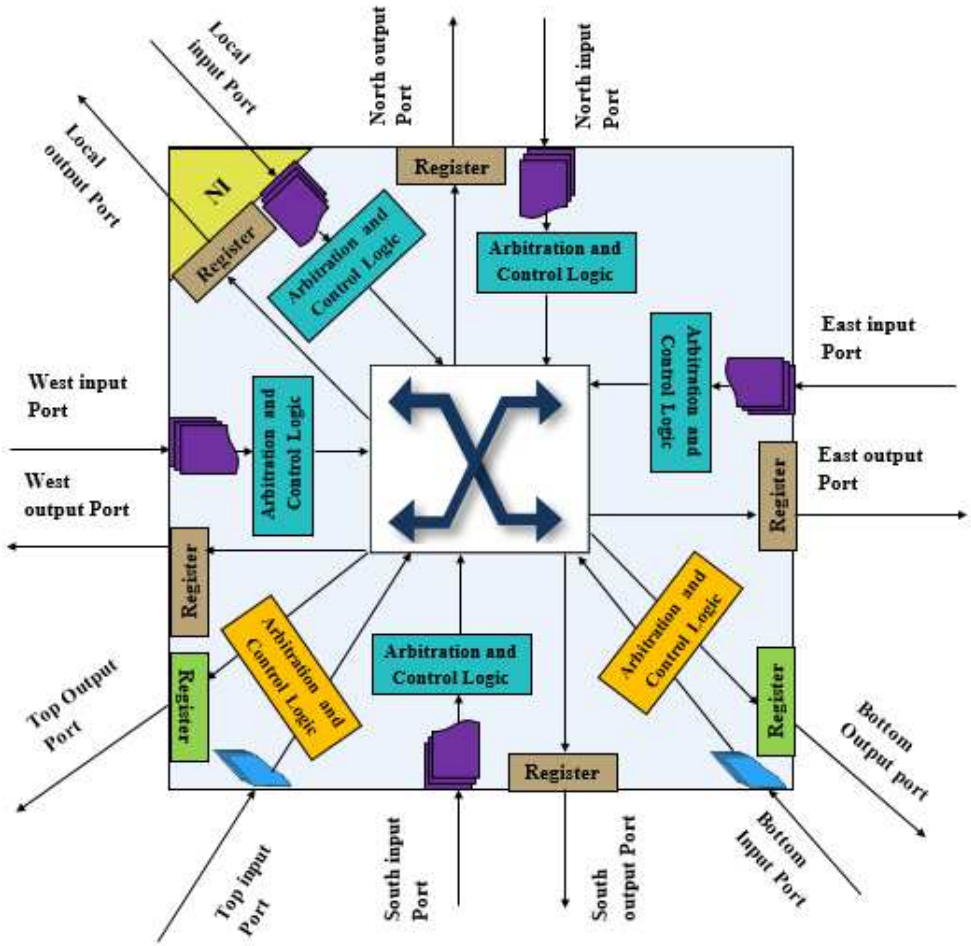

Figure 1.4: 3D router architecture

which is 0.86 for the XY algorithm. So they come to conclusion that the implement of XY routing algorithm is simple as well as the X-direction channel latency is averagely lager than Y-direction channel latency and X-direction channel throughput is averagely all square with Y direction throughput.

1.3 Communication Task Graph

Communication task graph or CTG is an running application on an system which is basically used to evaluate the performance of the system. There are two types of dependencies on which the task depends control and data dependencies [37, 38]. Both these dependencies can be explained as given below:

- Control Dependency:- In this a ask has to wait for other task to complete its job. So when some task is in executing phase then other task cannot start its execution till other tasks are there in the execution phase.
- Data Dependancy:- It implies that the tasks depends on each other for their execution and they depend on each other at the time of execution.

Communication task graph represented as (CTG), G' = G'(T,D), is a graph which is acyclic in its occurrence. The vertex of the graphs shows the computational model represented as task $t_i \in$T. Each task t_i has the relevant data which include execution time, energy consumption by the task and deadline of the task. Each directed arc $d_{i,j} \in$ D between tasks t_i and t_j gives the detailed information of either data or control dependencies. There is a value v($d_{i,j}$) which is associated with $d_{i,j}$ and the associated value v($d_{i,j}$) stands for communication volume in bits exchanged between tasks [38, 39, 40, 42].

1.4 Types of graphs used in NoC

1.4.1 Application characterization graph (APCG)

Application characterization graph is represented as G=(C,A) where vertex c_i is the selected IP/core and as it is a directed graph so directed arc $a_{i,j}$ defines the communication from core c_i

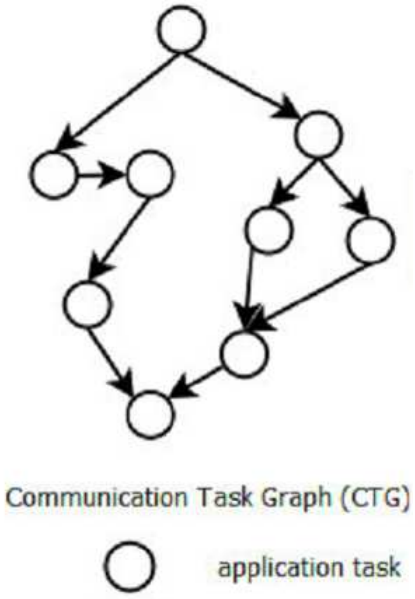

Figure 1.5: Communication Task Graph

to c_j.

Directed arc $a_{i,j}$ has some of the properties as mentioned here:

- v($a_{i,j}$) is the volume of the arc, here the arc is between the cores c_i to c_j. This volume is calculated in bits from core c_i to c_j.
- Arc bandwidth represented as b($a_{i,j}$) is represented as the arc between the vertex c_i to c_j, and it is the minimum bandwidth represented as bits per second, and it is assigned for achieving the performance constraints of the network[38, 41, 42].

1.4.2 Architecture Characterization Graph(ACG)

Architecture Characterization Graph(ACG):- G = G(T,R) is the graph representation for the architecture characterization graph. It is an undirected graph and a vertex t_i here represents a tile and there is a directed arc $r_{i,j}$ also which represents the routing parameters from the tile i to t_i. Following are the properties each $r_{i,j}$ considers for[38, 41, 42]:

- There are minimum paths between any two tiles and these minimum paths are called

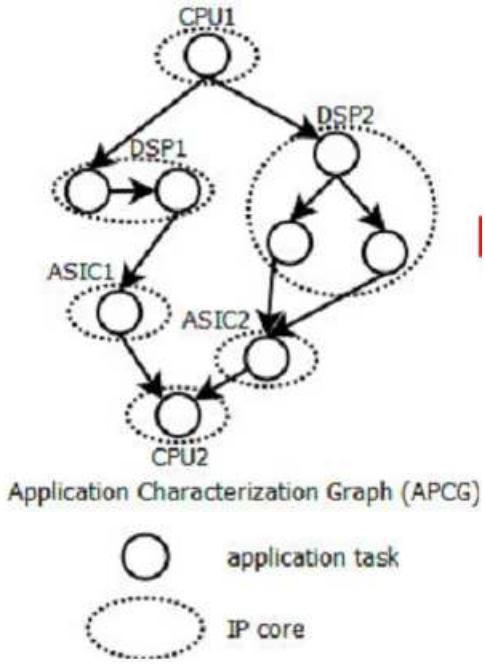

Figure 1.6: Application Characterization Graph

candidate paths. Path $P_{i,j}$ represents this set of the candidate paths from tile t_i to t_j, and there are links between these tiles which are used by $p_{i,j}$ represented as L($p_{i,j}$).

- The cost involved for any arc is represented as e($r_{i,j}$).This cost is determined in terms of the energy consumption calculated in joules involved for sending a single bit of the information from the tile t_i to tile t_j.

Chapter 2

Literature Review

2.1 Research Papers

In this section the basic idea is to give the explanation of some of the papers which are being reviewed by me for reference to mapping algorithm.

2.1.1 An Energy-Aware Methodology for Mapping and Scheduling of Concurrent Applications in MPSoC Architectures

Rajaei et al. [43] discusses about the EMAP for mapping which is based upon the energy consumption and proves to be best algorithm for mapping and scheduling of cocurrent applications. EAMS algorithm finds the mapping of task of the application to IP core in order to reduce the energy consumption without performance loss including task having deadline are satisfied.In this algorithm, we use deterministic routing which uses fewer buffers, resulting in less latency and energy consumption. Also, deterministic routing algorithms are deadlock and livelock free. NoC architecture consists of set of heterogeneous IP cores is considered. EAMS algorithm partitioned the tasks into critical tasks, (having probability of failing deadline) and non-critical task (not having probability of failing).

Algorithm for critical tasks given in the paper is as mentioned here. Arrange the critical tasks list. Calculate the execution time of tasks on all PE's. Task having large execution time has

higher priority. Choose the idle PE, assign highest priority task from list and continue untill all idle PE's finishes or all critical task are assigned to PE's for execution. Evaluate that tasks meet their deadlines, if not then either use busy PE or find an appropriate PE for task. If busy PE are used for execution of new task, then previous task will be added to task list.

Algorithm for non-critical tasks is as mentioned. Arrange the non-critical tasks list. Calculate the average value(Deviation) of tasks meeting deadline. Task having large deviation has higher priority. Choose the lowest power consuming idle PE, assign highest priority task from list for execution of tasks.

Xsimulator which is object oriented tool, to evaluate interconnection network and EDF-ASAP, EDF-ACAP two scheduling algorithms are used to evaluate energy saving, and performance(in terms of traffic) of algorithms.

2.1.2 Spiral: A heuristic mapping algorithm for network on chip

A heuristics core mapping algorithm for 2D mesh topologies called Spiral is proposed in this paper [44]. To measure the efficiency and speed, spiral mapping algorithm is compared with genetic algoirthm. Spiral algorithm improves reduction in energy consumption. In this algorithm, tasks are assigned to set of cores and tasks are maintained in task priority list(TPL). In mesh topology, high degree of connection is present at center as compared to the boundary and it forms platform priority list(PPL) which starts at center of mesh and ends at boundary switch or router in spiral fashion.

Priority assignment rules are as mentioned. Task having higher size of data transfer should be placed as close as possible to each other. The task which are tightly related should have least possible Manhattan Distance. Task which have high connection degree(maximum degree of task in task graph), should be placed at the center of mesh topology. Task having higher priority should be mapped spirally from center to boundary of mesh platform.

Techniques and tools used in this paper are:

- X-Y Routing algorithm.

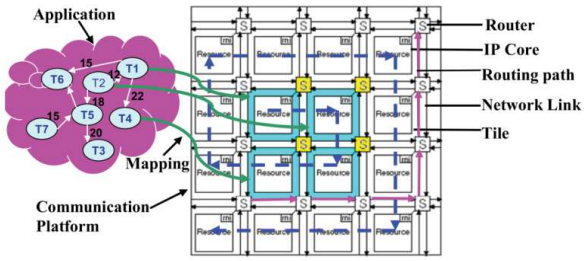

Figure 2.1: Spiral Mapping.

- MATLAB is used to evaluate the performance of spiral algorithm, which measures energy consumption.
- SMAP tool is capable of creating random graphs onto mesh platform.

2.1.3 Bandwidth-aware application mapping for NoC based MPSoCs

A mapping method is proposed to schedule and map the tasks of an application onto NoC architecture using ACO-based (Ant Colony Optimization) algorithm in order to minimize the bandwidth requirement of NoC [45]. With the help of the results, it can be evaluated that using X-Y routing algorithm, one can achieve reduction of around 48% in required bandwidth. The mapping algorithm is such that we get B = min ($\sum_{\forall i} b_l$) = min ($\sum_{\forall (i,j)} (c_{i,j} \times$ dist(i,j)))

Biological ants finds the way to reach their food from their colony. This is the basic idea, which author has concentrated upon, for finding the solution of NP-problems, i.e. non-deterministic polynomial problem.

- Initialize values of counter(NC), ants be numbered as k, size of network considered as S, the T be the number of tasks and the values of the pheromone τ_{ij}
  - While (NC $\gg$0)
  - for (ants from 1 to k)

– generate the tabu-search table Tabu

– for (each task j)

– generate the probability such that: $p_{ij}^k = \frac{\tau_{ij}}{\sum_{i \notin tabu} \tau_{ij}}$, otherwise $p_{ij}^k = 0$

– assign task j to processor i according to the probability p_{ij}^k.

– update the tabu-search table Tabu.

– end

– calculate the total bandwidth requirement B^k

– end

– search the best ant which minimizes the value of B^k ($B^{best} = \min(B^k)$)

– update the pheromone for the best ant

– end

– search the best ant and output the mapping results

2.1.4 GA-MMAS: An energy aware mapping algorithm for 2D network on chip

Fang et al.[46] proposed a mapping algorithm named GA-MMAS based on Genetic Algorithm (GA) combined with MAX-MIN Ant System Algorithm (MMAS), to optimize energy consumption for NoC. In proposed technique, pheromones are initialized for MMAS from the result of GA. GA in GA-MMS Algorithm is as discussed here. Generate a random population of valid solutions with the help of chromosomes which consists of series of genes.Each gene represents an IP core. With the help of cross-over method as shown in Fig. 2.2, next generation population is generated and it stops at 1000 threshold value. With the help of genes, pheromone network is created. From those network path matrix table is formed which shows mapping of tasks on IP core and initialized to MMAS.

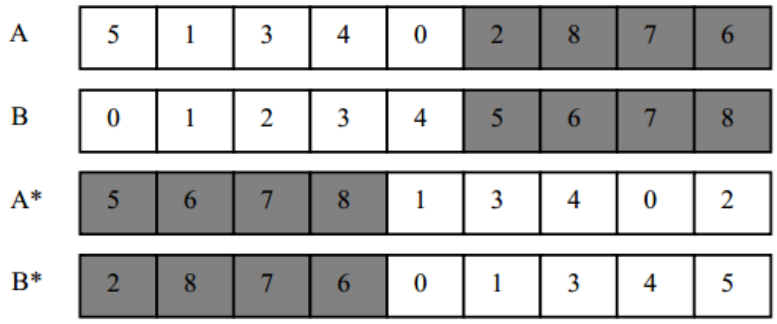

Figure 2.2: Cross-Over Order

MMAS in GA-MMAS : Each ant has one more table called stable table which consists of number of unassigned task to core. Task is assigned to IP core according to calculating probability having higher value. After assignment entry from stable table is deleted and updated into path table. Calculate the fitness function in terms of minimization of energy consumed and then update the pheromone for best ant.

The algorithm can save about 60% of energy consumption. This algorithms are compiled in C++ using 2D mesh platform on Window XP.

2.1.5 A3MAP: Architecture-aware analytic mapping for network on chip

WOOYOUNG JANG et al. propose Architecture-Aware Analytic Mapping (A3MAP) algorithms applied to Networks-on-Chip (NoCs) with homogeneous Processing Elements (PEs) on a regular mesh network and heterogeneous PEs on an irregular mesh network [47]. An application mapping problem is exactly formulated to Mixed Integer Quadratic Programming (MIQP). Since MIQP is NP-hard, author propose two effective heuristics, a successive relaxation algorithm providing short run time, called A3MAP-SR and a genetic algorithm achieving high mapping quality, called A3MAP-GA. A3MAP algorithms reduce total hop count and reduces traffic and communication delay between cores.

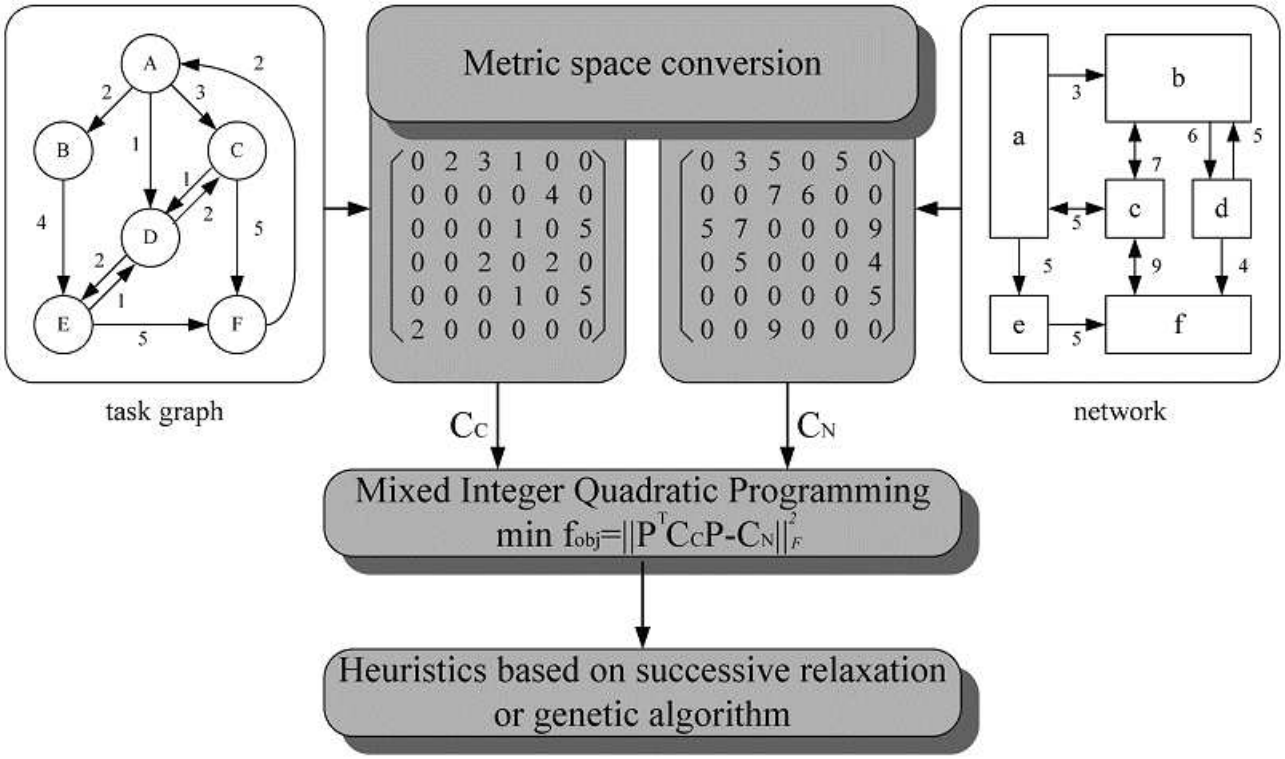

Figure 2.3: A3MAP

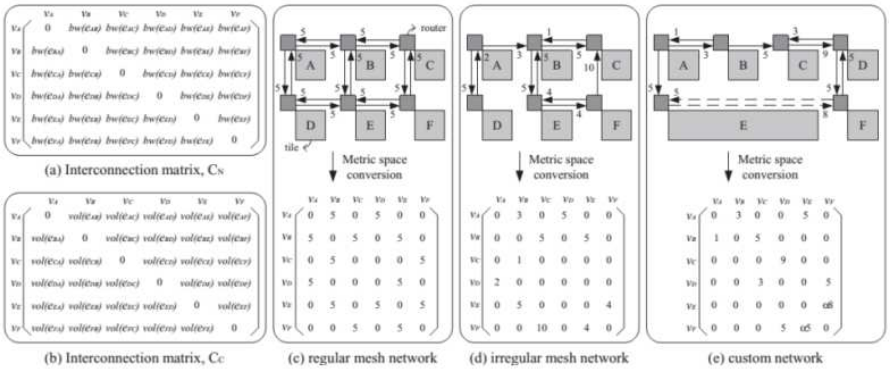

Figure 2.4: A3MAP Matrices

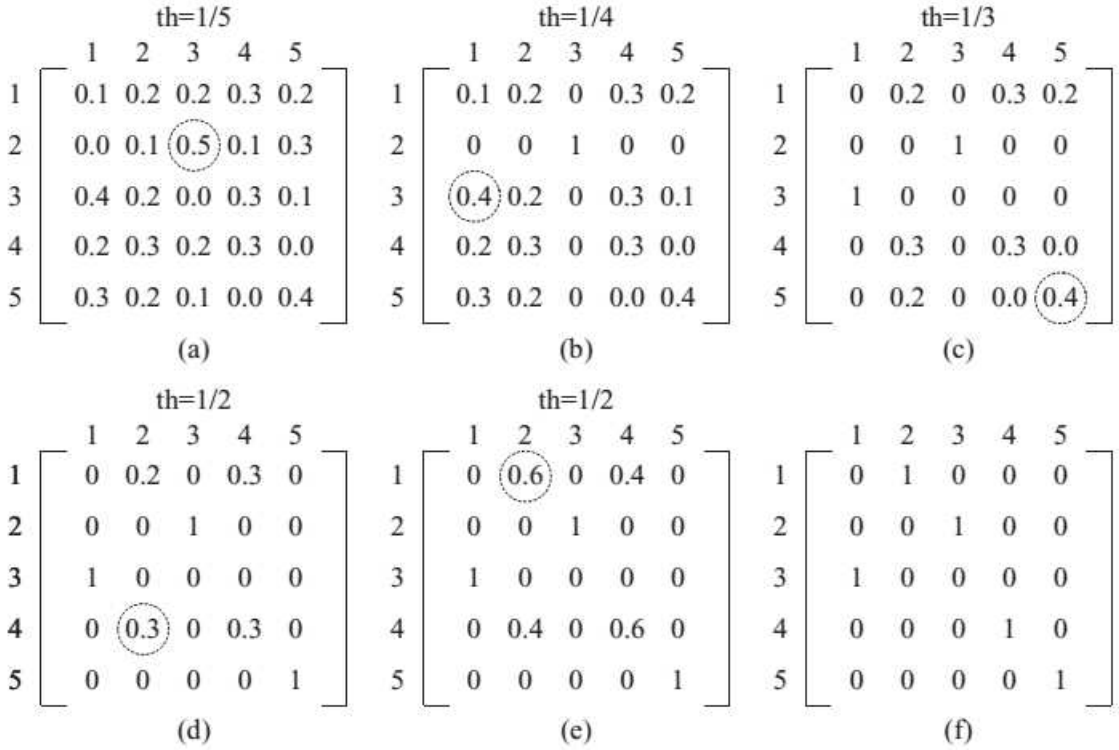

Figure 2.5: A3MAP-SR

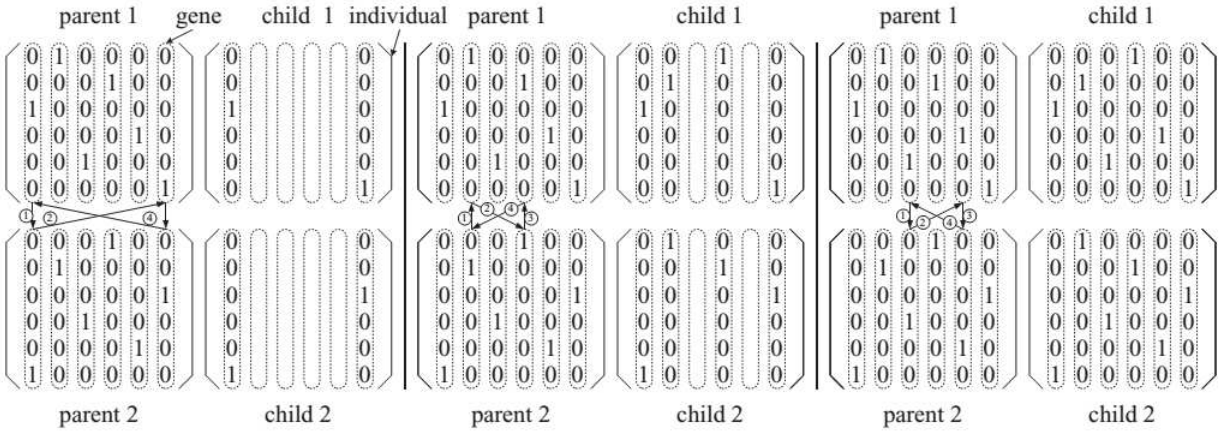

Figure 2.6: A3MAP-GA

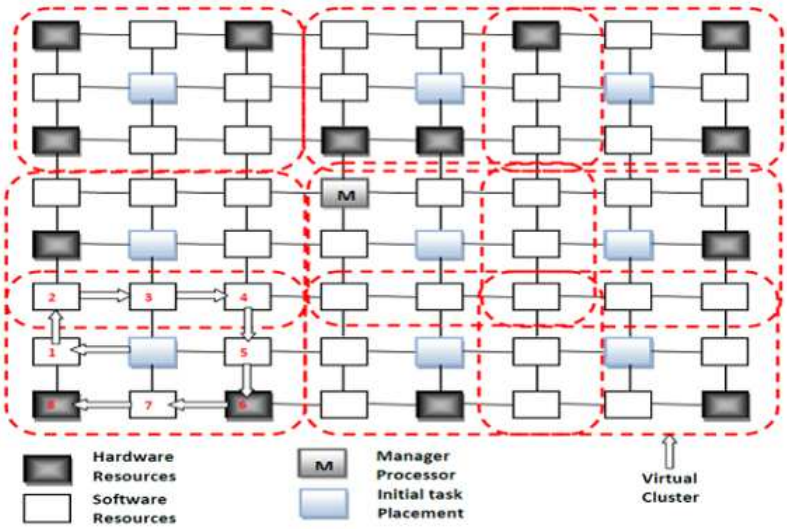

Figure 2.7: Task placement with Spiral Strategy.

2.1.6 Heuristic for routing and spiral run-time task mapping in NoC based heterogeneous MPSoCs

El Hasan et al.[49] describes a new Spiral Dynamic Task Mapping heuristic for mapping applications onto NoC-based Heterogeneous MPSoC and new modified dijkstra routing algorithm proposed are capable of reducing the total execution time and energy consumption of applications. Proposed spiral heuristic based on our Modified Dijikstra routing Algorithm.

Chapter 3

Problem Statement

Before formulating mapping problem, we assume that in order to perform mapping, we are given with application that is characterized by set of tasks which performs scheduling onto NoC cores. For appropriate understanding of mapping problem strategies, some important definition need to be explained.

Definition 1 *A Logical Application Trace Graph (LATG)* $G = (A_t, E_t)$ is an directed acyclic graph, where $a_t \in A_t$ represents task from list of application tasks and $c_{i,j} \in E_t$ is an directed arc between application tasks, that shows communication dependency between tasks a_{t1} and a_{t2}. Logical application trace graph is depicted in Fig. 3.1 Each directed edge or arc has one property:-

- $v(c_{i,j})$ represents volume bits transferred between from arc c_i to c_j.

Definition 2 *NoC Architecture Characterization Graph (NACG)* $G = (T, L_T)$ represents undirected graph as shown in Fig. 3.2, where vertex node $t_i, t_j \in T$ shows tiles in NoC architecture, whereas $l_k = l_{i,j} = (t_i, t_j) \in L_T$ represents routing path between t_i and t_j. Routing path in NACG consists of two properties :-

- $e(l_{i,j})$ is average energy consumption of task in bits from t_i and t_j.
- $Lat(l_{i,j})$ represents average latency of task from t_i and t_j.
- $band(l_{i,j})$ is defined as bandwidth of link between t_i and t_j.

Definition 3 A mapping function (Ω) is represented as $\Omega : A_t \rightarrow T$, that shows mapping of application tasks from LATG onto tiles available in NACG, where $a_t \in A_t$ and $\Omega(a_t) \in T$ and $\Omega(a_t)$ characterizes mapped tile in NACG. Fig. 3.3 shows mapping of application tasks onto NoC tile based architecture.

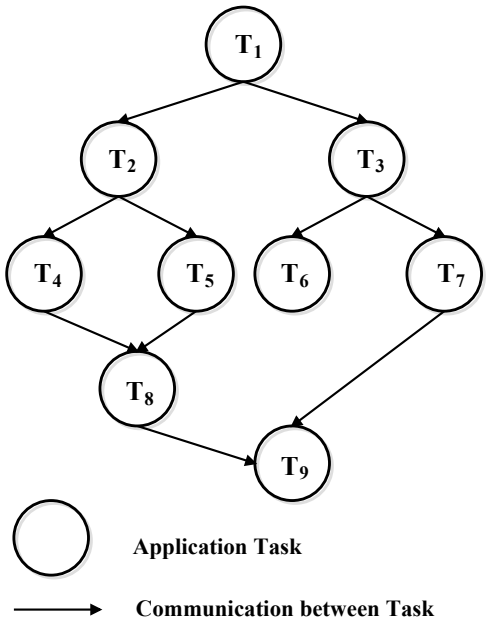

Figure 3.1: Logical Application Trace graph

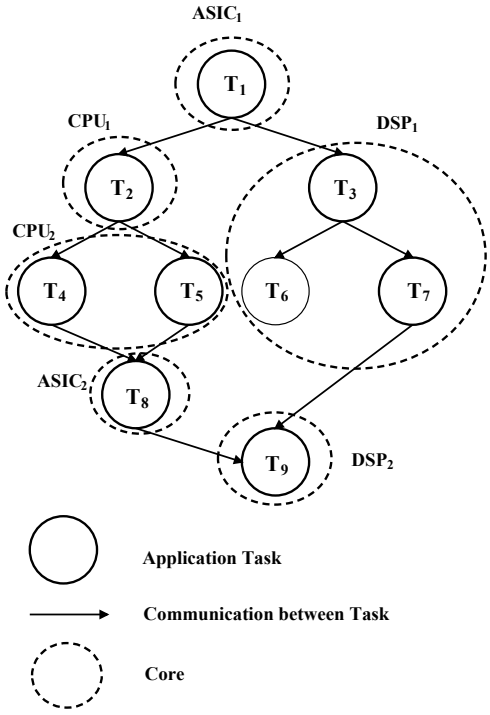

Figure 3.2: NoC Architecture Characterization Graph

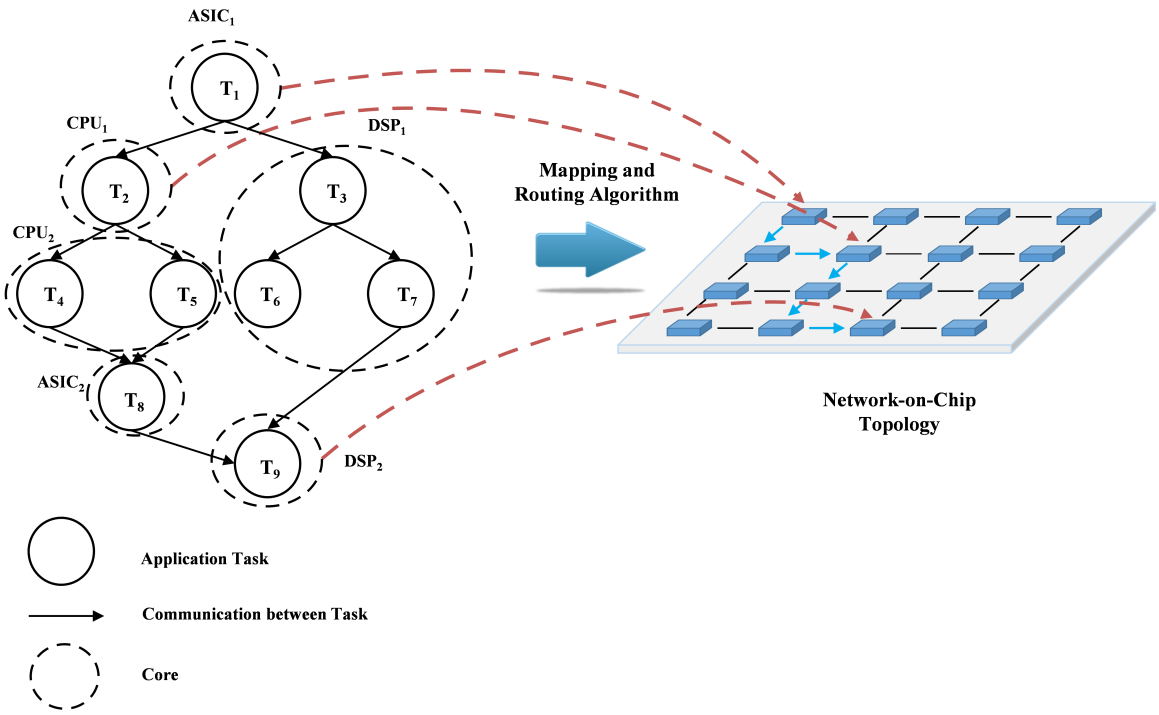

Figure 3.3: NoC Mapping Technique

Finally, the formulation of mapping problem is as follows:

Given: An LATG $G = (A_t, E_t)$ and NACG $G = (T, L_T)$,

Evaluate mapping function $\Omega : A_t \rightarrow T$, that maps task $a_t \in A_t$ in LATG to tile $t_i \in T$ in NACG,

such that energy consumption and average latency is minimized. Fig. 3.4 represents the flow of mapping of application task onto topology in order to get optimized results in terms of energy consumption and average latency.

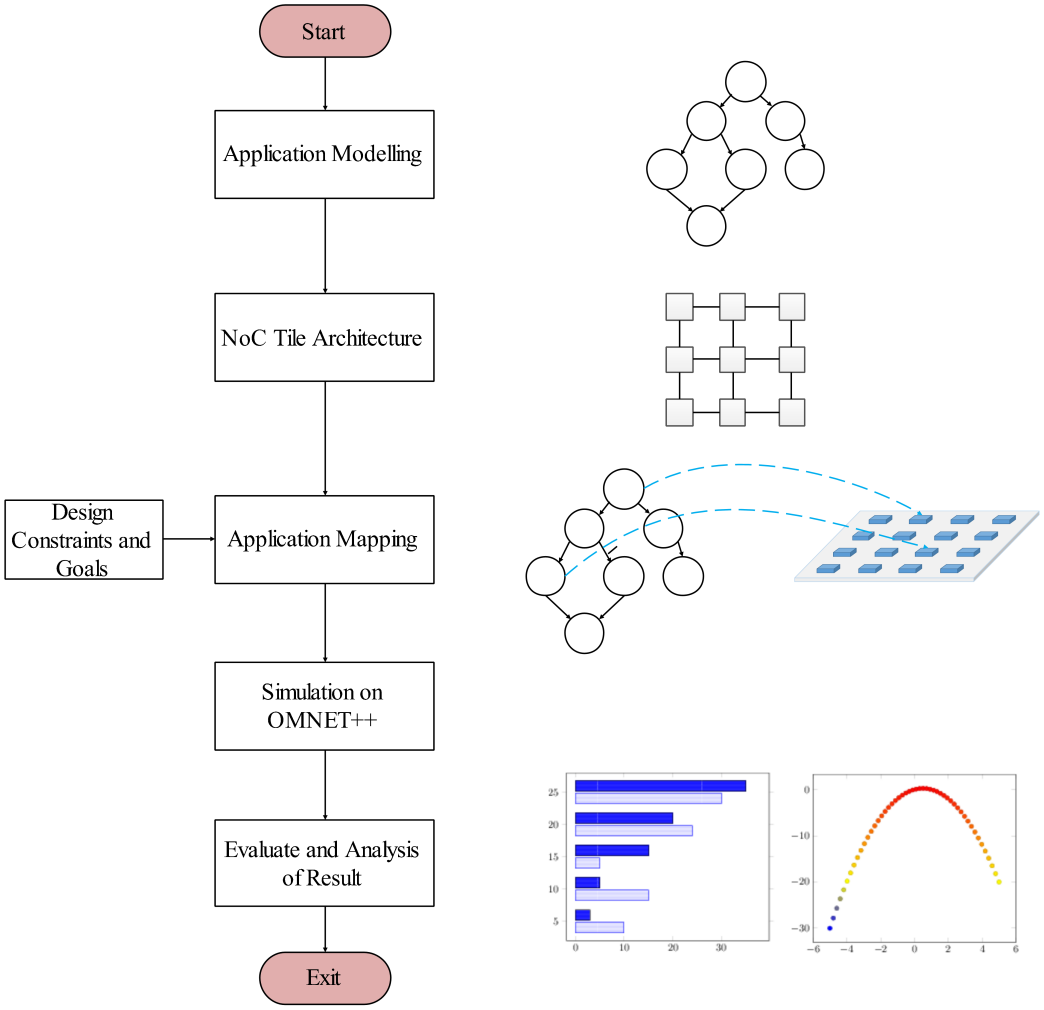

Figure 3.4: Flowchart representation of application mapping onto topology

Chapter 4

Existing Mapping Algorithm

4.1 Random Mapping Algorithm

Random mapping algorithm for NoC is most commonly used mapping algorithm by different researchers, but there are many issues involved in random algorithm. Issues such as load balancing as shown in Fig. 4.1, latency, service time and queuing time are not handled by random algorithm for NoC. In random algorithm, tasks are mapped on the cores randomly as discussed in Algorithm 1 . The worst case of the algorithm is, when every time the same core is chosen for mapping the task. As all tasks are mapped on the same core, so, the new tasks to be mapped will remain in the queue and wait for an infinite period of time till the core is not ready to process the new task. Once the core is available task is mapped on the core. In the best case of random algorithm for mapping, the randomly chosen cores will have an equal probability to be chosen, and task will be mapped on to these cores uniformly. There are rare chances to obtain the best case of the random algorithm. Let us consider a scenario that every time the last core of the grid is chosen to map the tasks. If such a case exist then latency involved to map the tasks on the cores will be very high. So mapping the task on to the cores in case of random algorithm consumes a large amount of latency, service time, queuing time and the energy consumption. To improve the performance of the mapping algorithm in this paper, the horological, rotational and divide and conquer mapping algorithms are proposed.

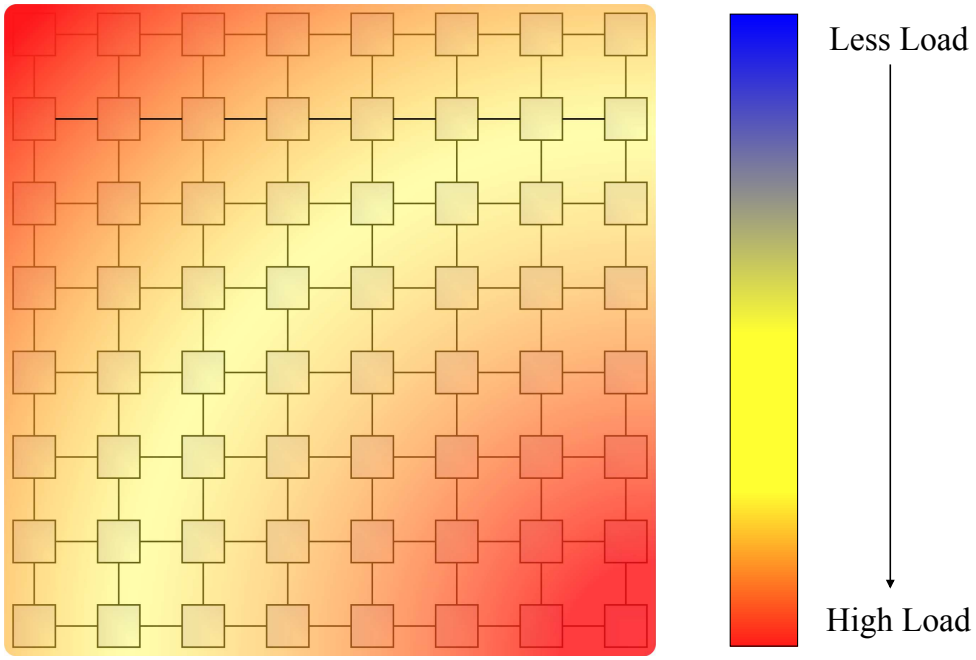

Figure 4.1: Load balancing by random mapping algorithm in 2D NoC

Algorithm 1: Random Mapping Algorithm

Data: *dst_core* as the destination core, id be the unique number assigned to each core and $id \in [0, n)$, n × n mesh topology is given having n^2 cores and t_n be the number of task.

Result: Task mapped on cores randomly

while $(t_n > 0)$ **do**

```
dst_core = (id + intuniform(1,n)) % n;
//returns a random core from n cores available.
Assign task to dst_core;
t_n–;
```

Random mapping algorithm for 3D NoC is similar to the random algorithm as discussed in 2D NoC mapping technique. Issues such as load balancing can be viewed in Fig. 4.2 where the red color shows the most energy consuming part of the NoC disk. In random algorithm, tasks are mapped on the cores randomly similar to that of the 2D NoC random mapping algorithm, as discussed in Algorithm 1.

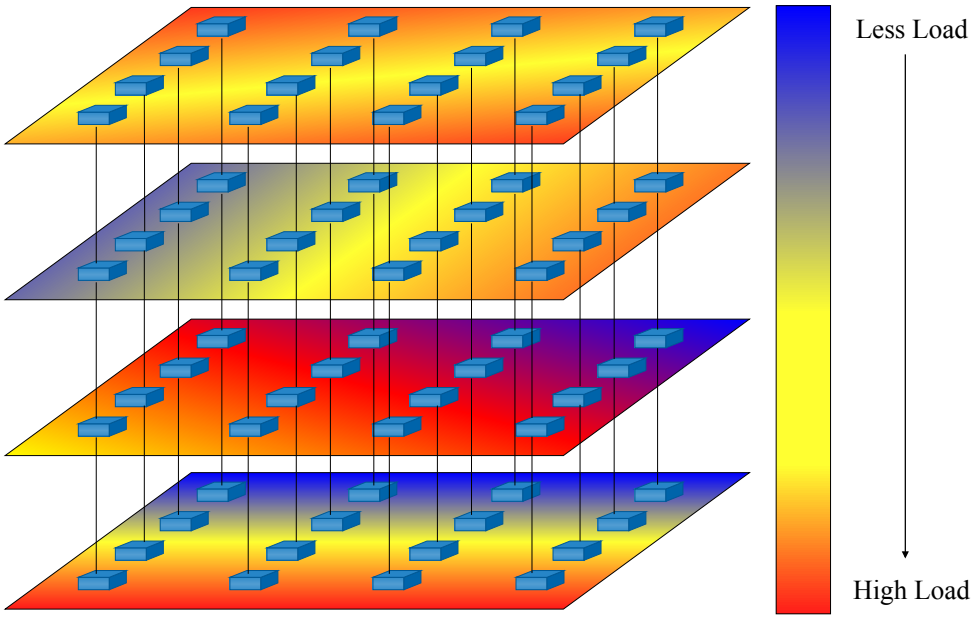

Figure 4.2: Load balancing by random mapping algorithm in 3D NoC

Chapter 5

Proposed Approach

In this section the three proposed approaches are discussed which proves to be better than the existing random mapping algorithm in terms of latency, load balancing and energy consumption. First approach discussed is horological mapping algorithm, in which the cores are visited one by one guaranteeing load balancing over the cores of the grid. Second approach is the rotational mapping algorithm. In this the task are assigned to the cores in rotation one by one guaranteeing the least latency involved during mapping of tasks. The third algorithm proposed in the paper is the divide and conquer mapping algorithm, which provides an assurity of load balancing on the grid.

5.1 Horological Algorithm

As the name suggest, in this mapping algorithm the tasks are mapped horologically on the cores one by one. As the task are assigned to the cores, then the core will process these task, and after the processing of task, the core gets ready to execute the next task in the queue. In this, the cores are allotted an core_id horologically. The first task in the queue is allocated to the first core, second task to the second core and so on. When the task on some core is completed, then a new task is allocated to this core. This algorithm produces good results in terms of load balancing on the cores, but the accessing time of the core increases as we moves towards the last core, with the last core having the maximum access time. So the accessing time of the cores is increased moving towards the last core. Fig. 5.1 shows the allocation of task on 8×8 mesh topology. For an instance, suppose there are 8 tasks which are to be mapped on the cores then even if the core with core_id 8 is closer to the queue the task will not be assigned to it, instead tasks will be assigned to the cores having core_id 0 to core_id 7. Horological mapping proves

to be better than the random mapping in terms of load balancing as shown in Fig. 5.2, queuing time and service time. It also resolve the issue of bottleneck existing in random mapping algorithm. Hence the horological mapping algorithm proves to be better over the random mapping algorithm. Horological mapping algorithm is given in Algorithm 2.

Algorithm 2: Horological Mapping Algorithm

Data: n $\times$ n Mesh topology having n^2 cores represented as c[i][j], t_n be the number of task.
Result: Cores chosen horologically to map task.

```
while (t_n > 0) do
    for i:=0 to n-1 step 1 do
        for j:=0 to n-1 step 1 do
            Assign the task to core c[i][j];
            t_n − −;
```

Horological mapping algorithm for 3D Networks on Chip is similar to the 2D horological mapping algorithm. The load balancing in case of the horological mapping algorithm for 3D NoC is shown in the Fig. 5.3. For the 3D NoC horological mapping algorithm the same algorithm is followed which was there for 2D NoC horological mapping.

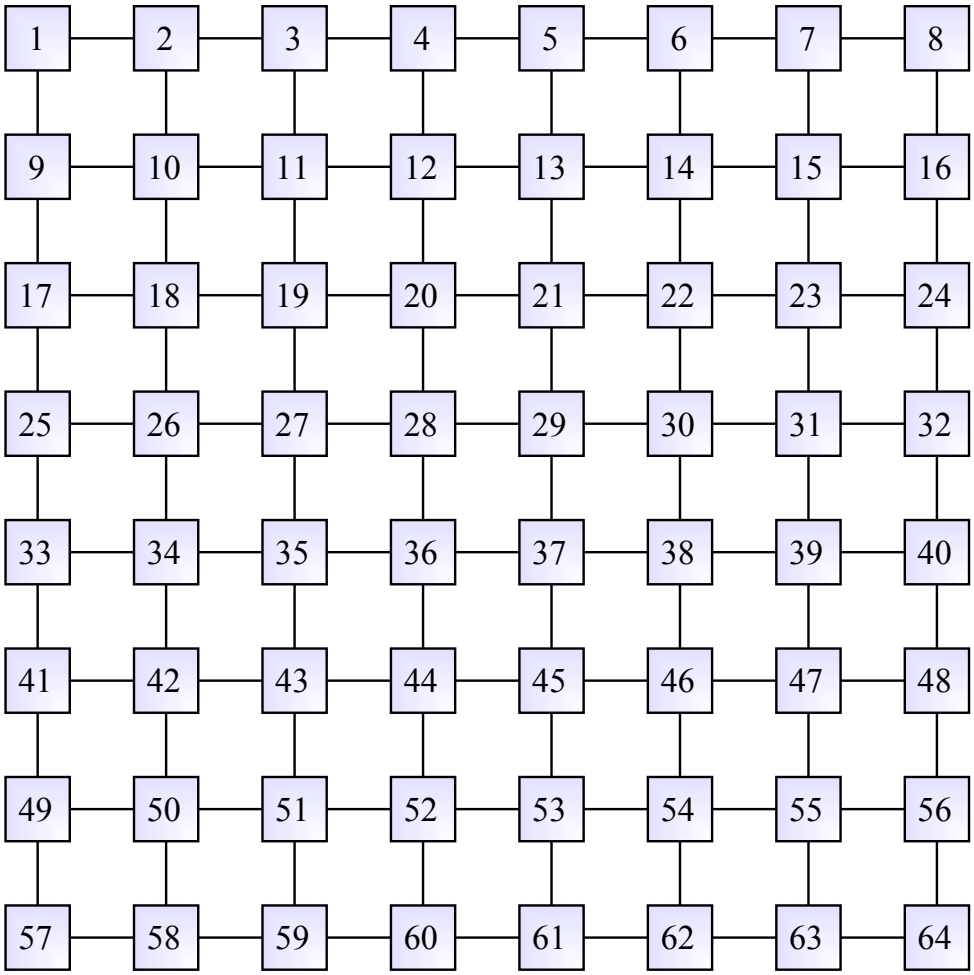

Figure 5.1: Horological mapping algorithm

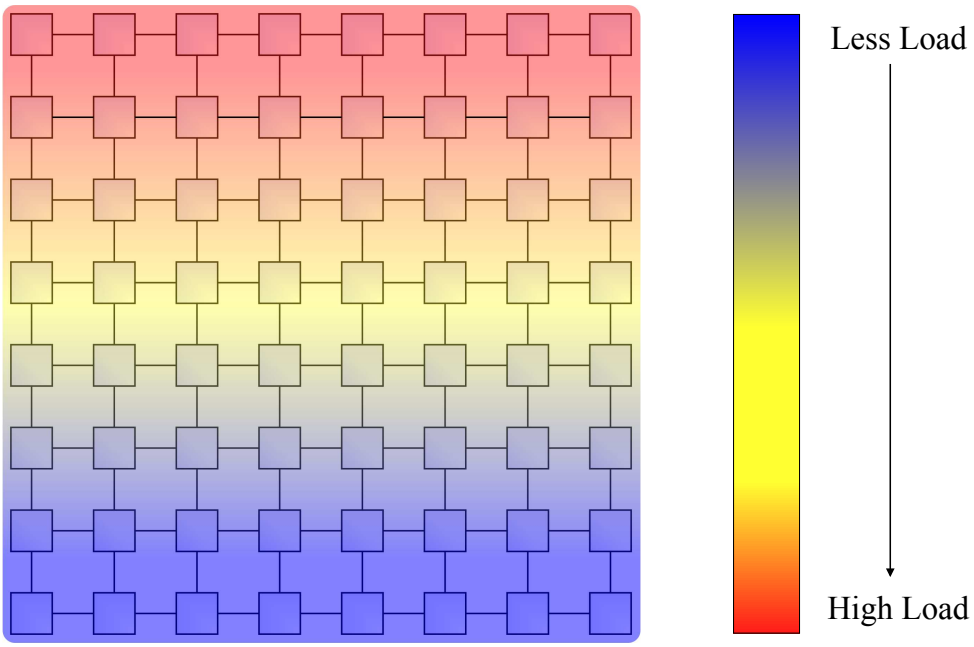

Figure 5.2: Load balancing by horological mapping algorithm in 2D NoC

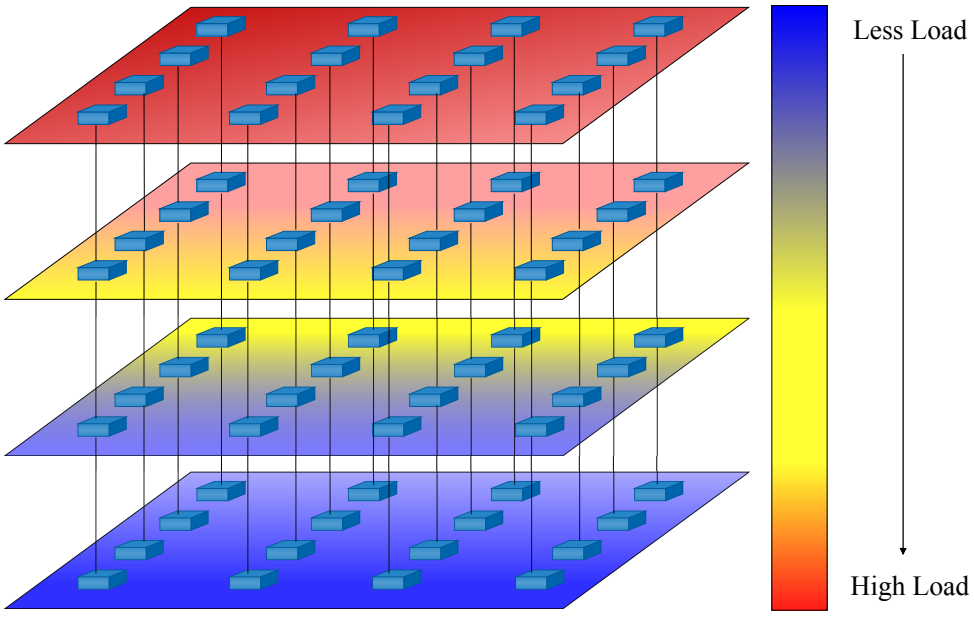

Figure 5.3: Load balancing by horological mapping algorithm in 3D NoC

5.2 Rotational Algorithm

Rotational mapping algorithm is proposed in this paper in order to minimize the latency involved during the mapping of the task on to the core, but there is no assurity of load balancing in this mapping algorithm as shown in Fig. 5.4. In rotational algorithm task are mapped on the cores in the rotational manner. Basic concern of the proposed approach is to reduce the amount of time required for mapping task on the core. In order to achieve the goal it is required to map the task on the core which is placed nearest to the task allocation queue, so whenever task has to be mapped, it is mapped on to the core which is nearest to the allocation queue and is in ready state, i.e. it is ready to accept the task for execution. For this purpose the ports of routers are considered to be very important. In this task to be mapped is routed on to the elements (routers or cores) attached to the ports of the router. For each router starting from port zero to the last port, task are passed to each port in an sequential order. Once all the ports are visited then this procedure repeats from first port of the router to the last port. In this way the algorithm is capable of mapping multiple task on the cores till the task allocation queue is not empty. As the procedure repeats for each router considering all the ports every time hence the algorithm is called as rotational algorithm. Rotational mapping algorithm is given in Algorithm 3.

Algorithm 3: Rotational Mapping Algorithm

Data: n × n Mesh topology having n^2 cores represented as c[i][j], t_n be the number of task, r[i][j] be the router corresponding to core c[i][j], P[i][j] be the number of ports associated to each router, variable assign to track the assignment of the task on the core and k[i][j]=0.

Result: Task mapped on cores by rotational Algorithm

while *($t_n > 0$)* **do**

    assign = 0; **while** *(assign$! = 1$)* **do**

        k[i][j] = k[i][j] % P[i][j]; //K[i][j] and P[i][j] are counters.

        if *c[i][j] is attached to port k[i][j]* **then**

            Assign the task to the c[i][j];

            assign = 1;

            k[i][j]++;

            $t_n - -$;

        else

            Pass the task on the router r[m][n] attached at port k[i][j];

            i = m;

            j = n;

            $k[i][j++]$;

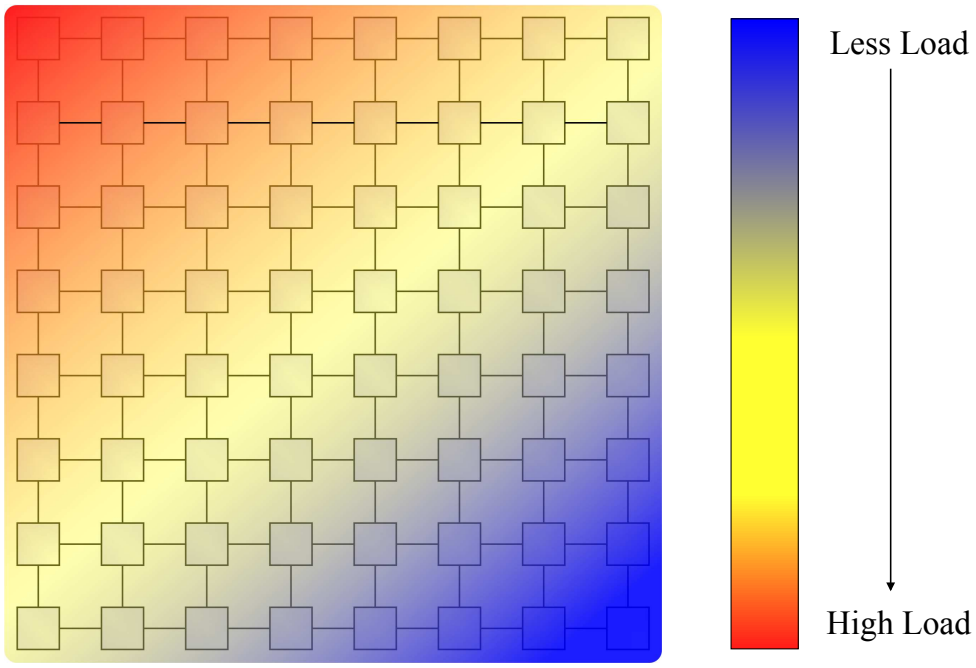

Figure 5.4: Load balancing by rotational mapping algorithm in 2D NoC

Rotational mapping algorithm in 3D networks on chip is same as that of the 2D networks on chip, as described above. The concept of load balancing is shown in the Fig. 5.5. The algorithm followed for mapping task on the cores using the 3D NoC rotational mapping is same as that of the one given above for 2D NoC.

5.3 Divide and Conquer Mapping Algorithm

In divide and conquer mapping algorithm, the main emphasis is on load balancing on n $\times$ n mesh topology. As the name suggest, in this algorithm first 2D Mesh topology is divided vertically into two (nearly equal) parts and then the division is carried out horizontally. After each vertical and horizontal division the topology is divided into 4 sub-grids of nearly equal dimensions(rows $\times$ columns) as shown in Fig. 5.6. Different tasks from task list, which are maintained in queue, are being mapped onto sub-grids in such a way that load is equally balanced on the mesh topology. For this purpose each time the task has to be mapped, the grids and sub-grids are further divided both vertically and horizontally. The task is assigned to the core belonging to that sub-grid in which there are least number of task mapped. In this way the task mapped on the cores of sub-grid are balanced, hence there is an assurity of load balancing during the mapping of the task to the cores. For an

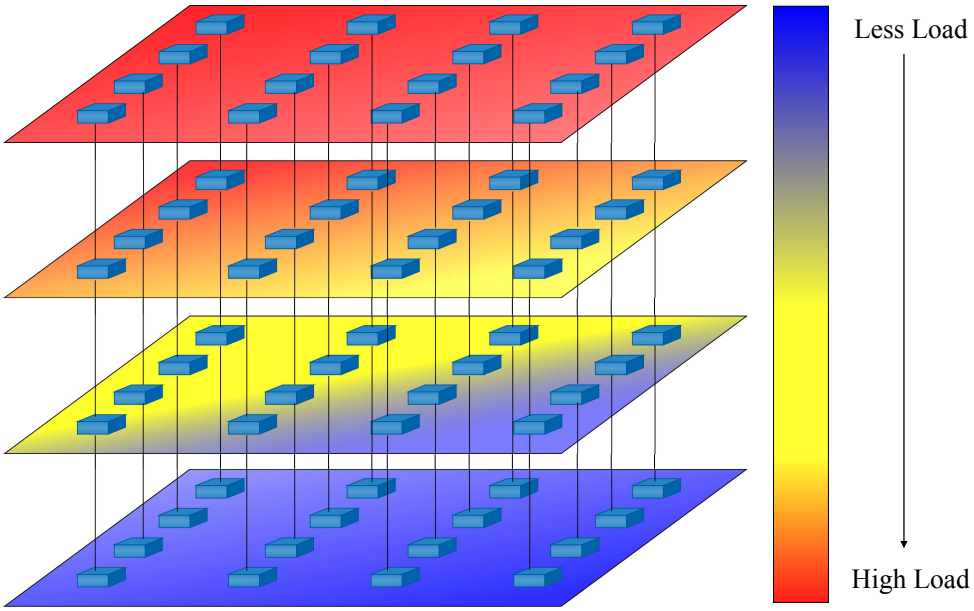

Figure 5.5: Load balancing by rotational mapping algorithm in 3D NoC

instance let us consider a simple scenario for mapping task on the 8 × 8 mesh topology, first task from task list is mapped onto first core of first sub-grids. Second task from task list, is mapped onto 5th core belonging to second sub-grids. In the similar way, 3rd task mapped onto 33th core belonging to third sub-grids and next task mapped onto 37th core which belongs to fourth sub-grids. So in this way, all task is mapped onto mesh topology as shown in Fig. 5.7, assuring the researcher to get a NoC architecture with complete load balancing in Fig. 5.8. Divide and conquer mapping algorithm is given in Algorithm 4.

Algorithm 4: Divide And Conquer Mapping Algorithm

Data: n × n mesh having n^2 cores and t_n number of tasks.

Step I : Divide grid vertically with one partition having $\lfloor \frac{n}{2} - 1 \rfloor$ cores and other partition having $\lfloor \frac{n}{2} \rfloor$ cores in each row.

Step II : Divide grid horizontally with one partition having $\lfloor \frac{n}{2} - 1 \rfloor$ cores and other partition having $\lfloor \frac{n}{2} \rfloor$ cores in each column.

Step III : Assign task to first core of each partition.

Step IV : Repeat above steps for each sub-grids obtained with vertical and horizontal line partitions till grid having exactly one core is obtained.

Step V : If all task are not assigned to the cores then repeat the full process given above considering the full n × n grid again.

The algorithm used for the divide and conquer mapping algorithm is same in the 3D

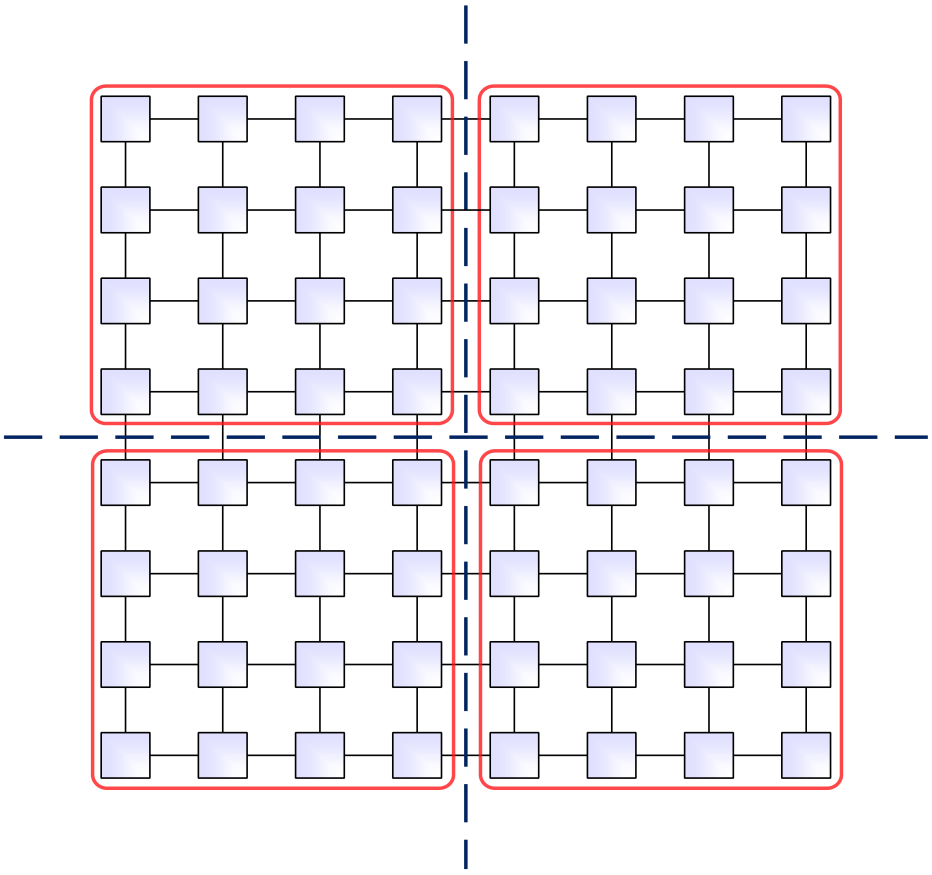

Figure 5.6: Grid Divison into sub-grid

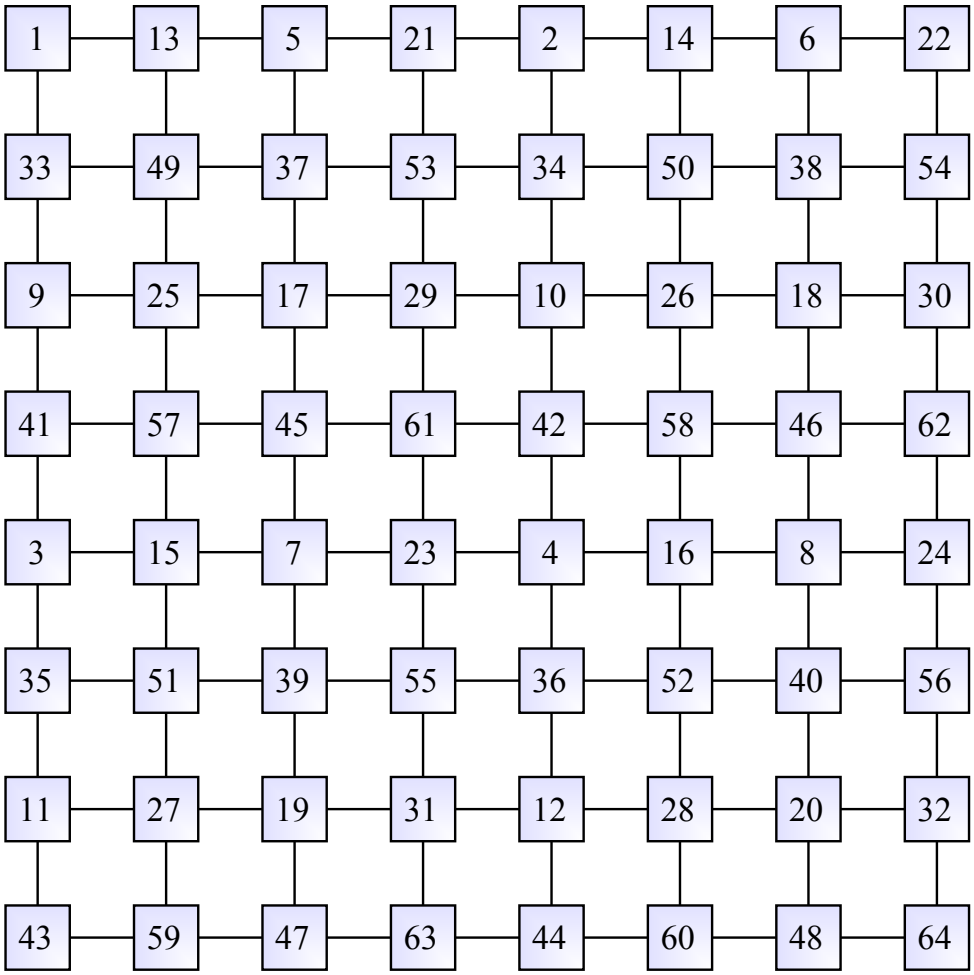

Figure 5.7: Divide and conquer mapping algorithm

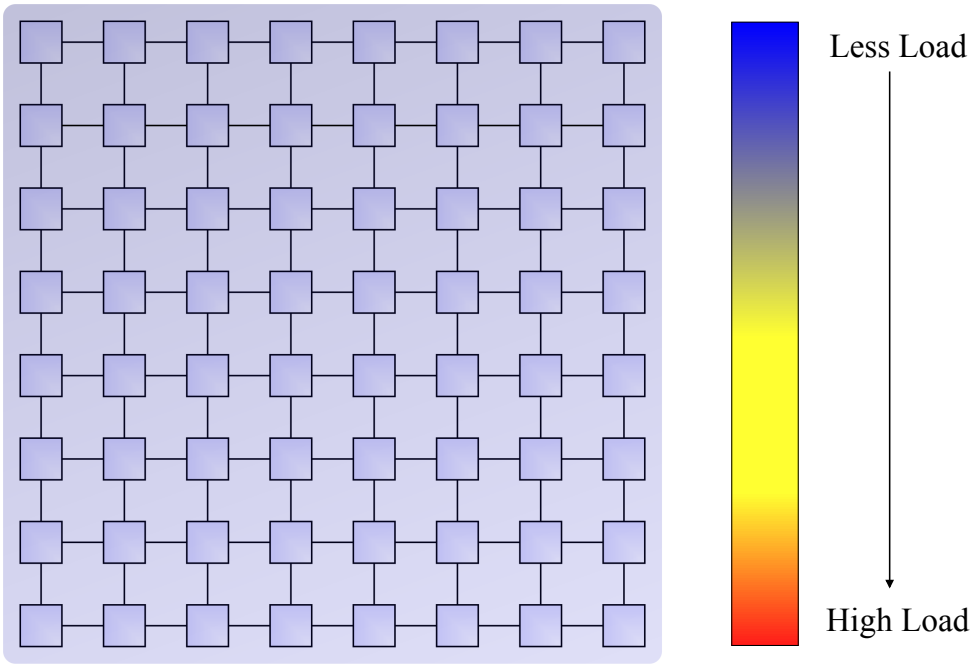

Figure 5.8: Load balancing by divide and conquer mapping algorithm in 2D NoC

NoC as that of the 2D NoC, discussed above. The concept of load balancing in case of the 3D networks on chip mapping algorithm is shown in the Fig. 5.9

5.4 Energy Model

The objective function is to minimize the energy consumption, which can be mathematically represented as:

$$\min\{\sum_{\forall a_t \in A_t} e_{\Omega(a_t)} + \sum_{\forall c_{i,j} \in E_t} v(c_{i,j}) \times \sum_{l_{i,j} \in R_{\Omega(a_{t1}),\Omega(a_{t2})}}^{|R_{\Omega(a_{t1}),\Omega(a_{t2})}|} e(R_{\Omega(a_{t1}),\Omega(a_{t2})})\} \tag{5.1}$$

satisfying conditions as

$$\forall a_t \in A_t, \quad \forall \Omega(a_t) \in T \tag{5.2}$$

$$\forall a_{t1} \neq a_{t2}, \quad \forall \Omega(a_{t1}) \neq \Omega(a_{t2}) \tag{5.3}$$

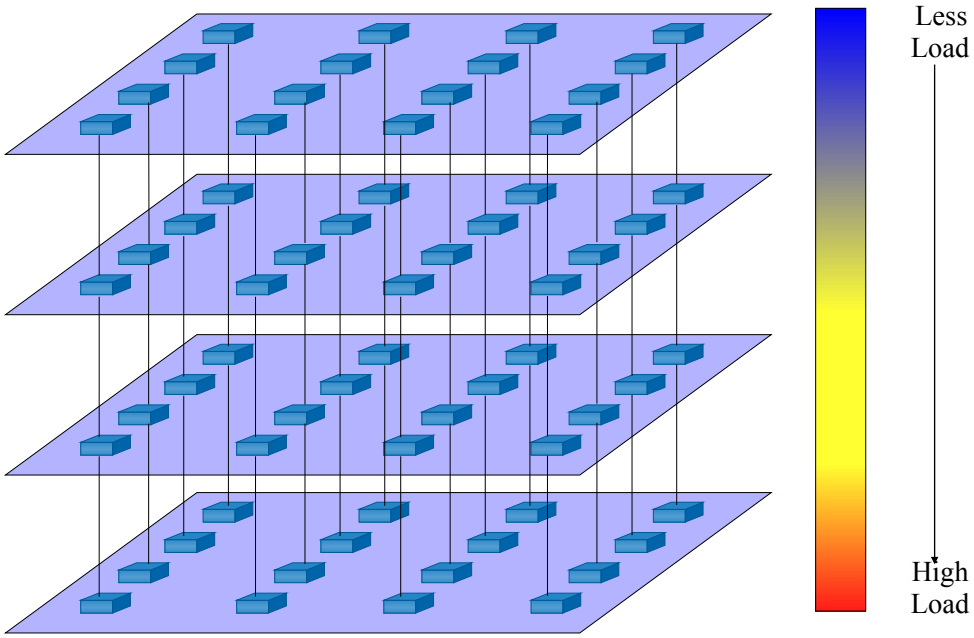

Figure 5.9: Load balancing by divide and conquer mapping algorithm in 3D NoC

The average energy consumption for transferring task from t_i to t_j can be represented as follows:

$$E_{task}^{t_i,t_j} = N \times num_{hops} \times E_{Link} + N \times (num_{hops} - 1) \times E_{Router} \tag{5.4}$$

where, E_{Link} and E_{Router} represents energy consumption of link and energy consumption of router. In order to compute E_{Router}, we have to compute energy consumption of buffer (E_{Buffer}), energy consumption of crossbar switch ($E_{Crossbar}$) and energy consumption of arbiter ($E_{Arbiter}$). $E_{Arbiter}$ is further divided into two parts : (i) $E_{Crossbar\_Allocation}$, energy consumption of switch allocation and (ii) $E_{VC\_Allocation}$, energy consumption of virtual channel allocation. E_{Link} can be computed as gievn in Equation 5.7. Energy consumption of topology is calculated for all N tasks is given in Equation 5.8.

$$E_{Router} = E_{Buffer} + E_{Crossbar} + E_{Arbiter} \tag{5.5}$$

$$E_{Arbiter} = E_{Crossbar\_Allocation} + E_{VC\_Allocation} \tag{5.6}$$

$$E_{Link} = \frac{P_{Link}}{Freq.} \tag{5.7}$$

$$E_{Total} = \sum_{i=1}^{N} E_{task_i} \tag{5.8}$$

5.5 Latency Model

The mapping function for minimization of average latency of topology can be mathematically formulated as:

$$\min\{\sum_{\forall a_t \in A_t} Lat_{\Omega(a_t)} + \sum_{\forall c_{i,j} \in E_t} v(c_{i,j}) \times \sum_{l_{i,j} \in R_{\Omega(a_{t1}),\Omega(a_{t2})}}^{|R_{\Omega(a_{t1}),\Omega(a_{t2})}|} Lat(R_{\Omega(a_{t1}),\Omega(a_{t2})})\} \tag{5.9}$$

satisfying conditions as

$$\forall a_t \in A_t, \quad \forall \Omega(a_t) \in T \tag{5.10}$$

$$\forall a_{t1} \neq a_{t2}, \quad \forall \Omega(a_{t1}) \neq \Omega(a_{t2}) \tag{5.11}$$

The latency from tile t_i to tile t_j can be computed according to Equation 5.12. The overall latency for all N tasks is calculated by Equation 5.13.

$$Lat_{task}^{t_i,t_j} = N \times num_{hops} \times Lat_{Link} + N \times (num_{hops} - 1) \times Lat_{Router} \tag{5.12}$$

$$Lat_{Total} = \sum_{i=1}^{N} Lat_{task_i} \tag{5.13}$$

Fig. 5.10 shows the 3×3 NoC topology in the form of tile, where each tile consist of cores (that can be IP core, DSP core etc.) and routers (consists of crossbar switch, routing algorithm and arbitration logic). Latency of single task to be transferred across channel are $D_{injection}$ and $D_{ejection}$ respectively and latency of a task across router are D_{switch}, $D_{routing}$ and $D_{waiting}$. In Fig. 5.11, we have considered link injection latency ($D_{injection}$), latency of first router ($D_{switch} + D_{routing}$), inter-tile latency ($D_{waiting}$), second router latency ($D_{routing} + D_{switch}$), and link ejection latency ($D_{ejection}$). Latency flow of single hop can be calculated according to Equation 5.14:

$$\begin{aligned} Latency\_single\_hop = D_{injection} + (D_{routing} + D_{switch}) + D_{waiting} \\ + (D_{routing} + D_{switch}) + D_{ejection} \end{aligned} \tag{5.14}$$

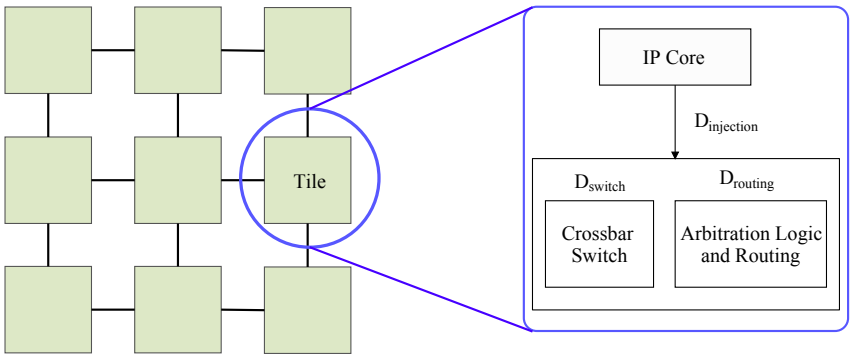

Figure 5.10: NoC topology tile

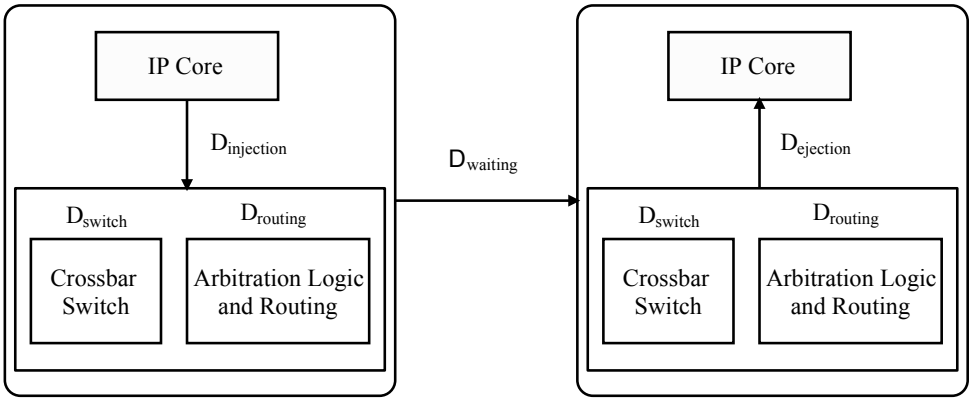

Figure 5.11: Latency flow in single hop

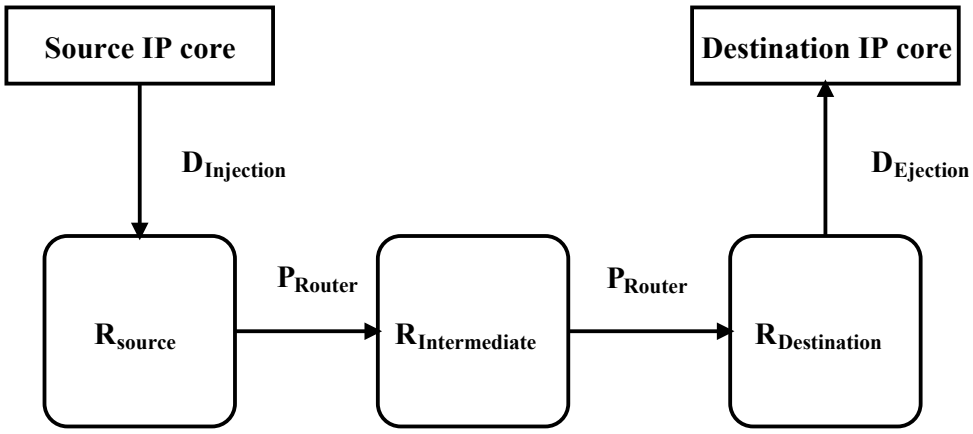

Figure 5.12: Latency flow in two hop from source to destination core

In order to calculate the latency from source to destination core, we have assumed that as task reaches to destination core, then the task is immediately accessible by destination core. In Fig. 5.12, the latency involved, is considered from source IP core to destination IP core passing through routers are $R_{source}, R_{intermediate}$ and $R_{Destination}$. The latency of task having two hops between source and destination core ($L_{source \to destination}$) is calculated as given in Equation 5.15, where W^{source}, $W^{destination}$ and $W^{intermediate}$ represents the waiting time in routers. The average latency of task (L) can be calculated in Equation 5.16, where $P_{source \to destination}$ is probability of task to be generated.

$$\begin{aligned} L_{source \to destination} = {} & D_{injection} + (D_{routing} + W^{source}_{inj \to port} + D_{switch}) \\ & + D_{waiting} + (D_{routing} + W^{intermediate}_{port \to port} + D_{switch}) \\ & + D_{waiting} + (D_{routing} + W^{destination}_{port \to ejc} + D_{switch}) \\ & + D_{ejection} + (m-1)(D_{switch} + D_{waiting}) \end{aligned} \tag{5.15}$$

$$L = \sum_{source} \sum_{destination} P_{source \to destination} \times L_{source \to destination} \tag{5.16}$$

Chapter 6

Simulator Tool

6.1 OMNeT++

OMNeT++ is an object-oriented modular discrete event network simulation framework. OMNeT++ itself is not a simulator of anything concrete, but rather provides infrastructure and tools for writing simulations. One of the fundamental ingredients of this infrastructure is a component architecture for simulation models. OMNeT++ simulations can be run under various user interfaces. Graphical, animating user interfaces are highly useful for demonstration and debugging purposes, and command-line user interfaces are best for batch execution. The simulator as well as user interfaces and tools are highly portable. They are tested on the most common operating systems (Linux, Mac OS/X, Windows), and they can be compiled out of the box or after trivial modifications on most Unix-like operating systems. OMNeT++ also supports parallel distributed simulation. OMNeT++ can use several mechanisms for communication between partitions of a parallel distributed simulation.

An OMNeT++ model consists of modules that communicate with message passing. The active modules are termed simple modules; they are written in C++, using the simulation class library. Simple modules can be grouped into compound modules. OMNeT++ execution has different phases and these phases are shown in the form of the flowchart as shown in the Fig. 6.1 NED files specify the structure of any component in the simulator. There are different types of NED files: Network and Module files. The Network NED file contains all elements of a simulation (i.e., all hosts, routers, connections, etc.) Any simulation will have 1 Network, defined in a NED file. Module NED files detail the

structure of any simulation component. They can be defined recursively. For example, a Router module has submodules such as a RoutingTable, Point-to-Point interfaces, and an IP-Layer. A module may have parameters specified in the NED file, which serve as inputs to the module at runtime. Since any module in the simulation is a submodule of another module or a submodule in a network, any module in the simulation has a Path.

6.2 Orion 2.0

Orion, interconnection network simulator is used to determine the a power-performance which can provide the power characteristics in full descriptive manner, along with these performance characteristics, it also enables a power-performance of the system at a architecture level. This capability is provided within a general framework that builds a simulator starting from a micro architectural specification of the interconnection network. A key component of this construction is the architectural-level parametrized power models that we have derived as part of this effort. Using component power models and a synthesized efficient power (and performance) simulator, a micro architect can rapidly explore the design space. As case studies, we demonstrate the use of Orion in determining optimal system parameters, in examining the effect of diverse traffic conditions, as well as evaluating new network micro architectures. Fig. 6.2 shows the flowchart for the execution in the Orion Simulator. In each of the above, the ability to simultaneously monitor power and performance is key in determining suitable micro architectures [50].

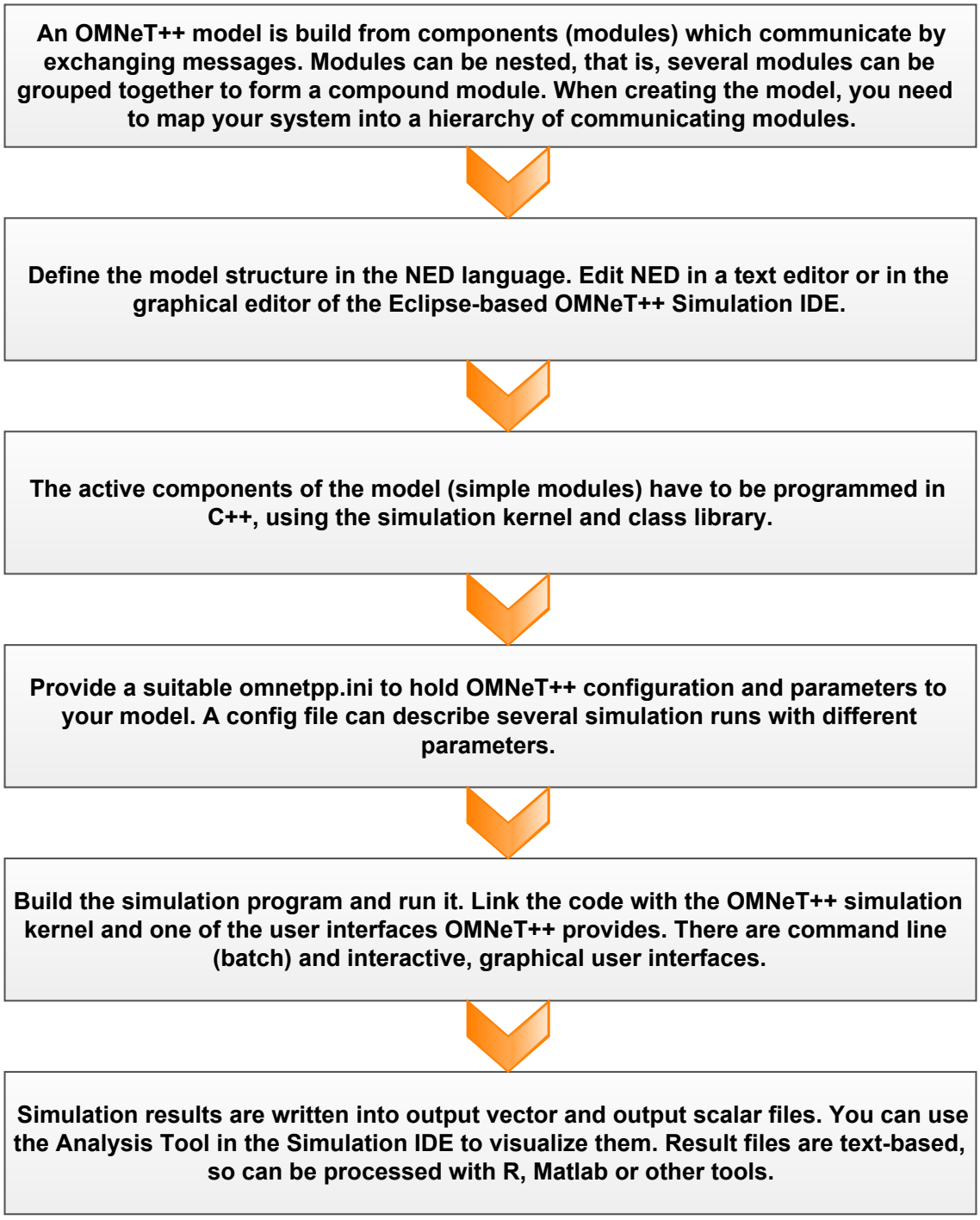

Figure 6.1: OMNeT++ execution flowchart.

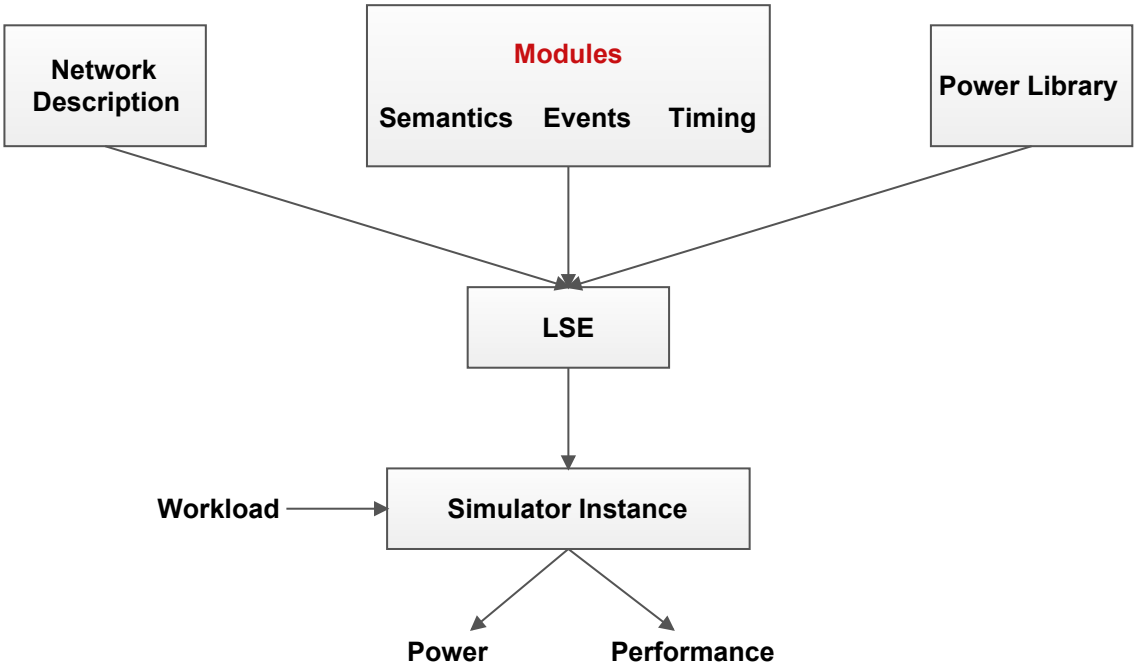

Figure 6.2: Orion execution flowchart.

Chapter 7

Design Analysis

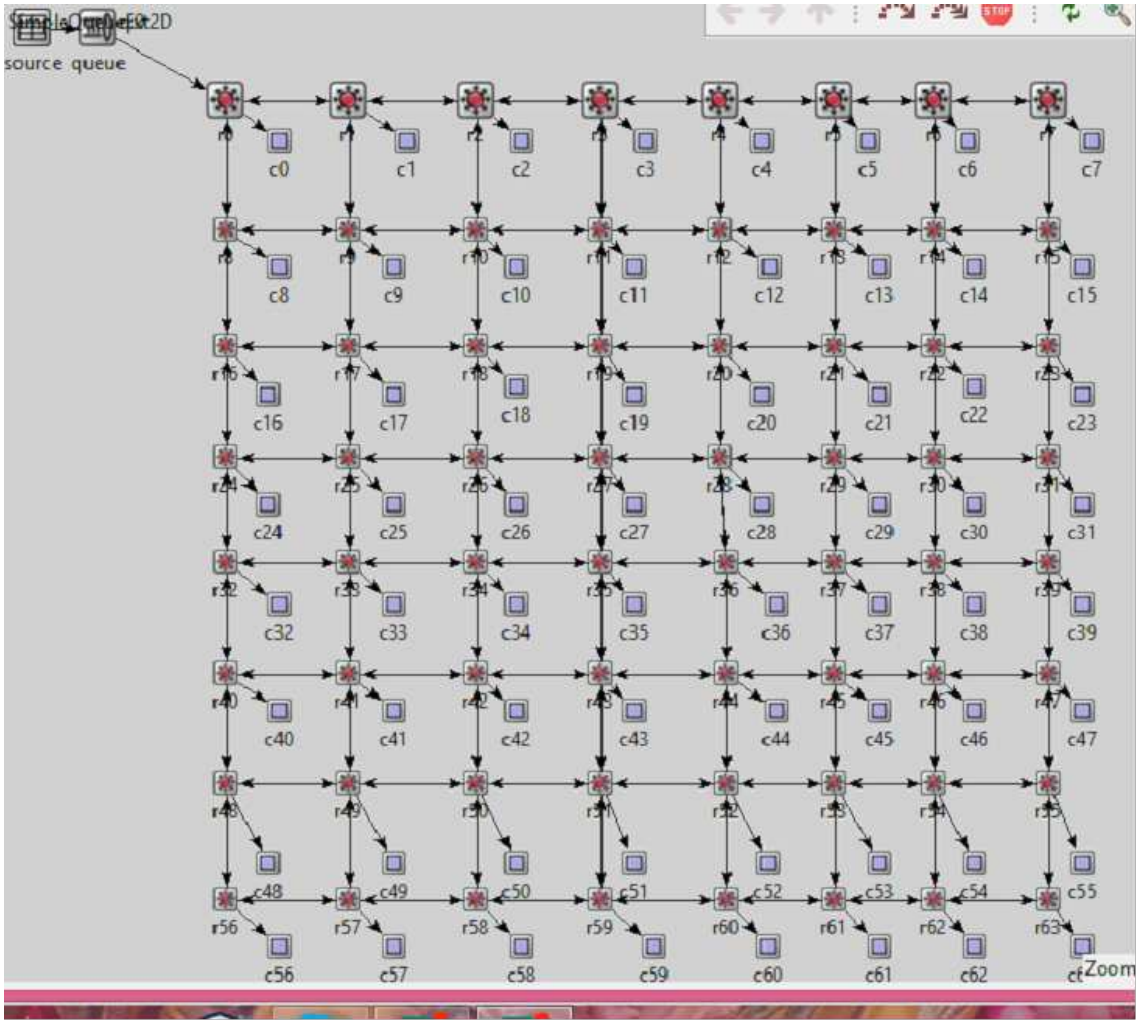

Figure 7.1: Mesh Topology

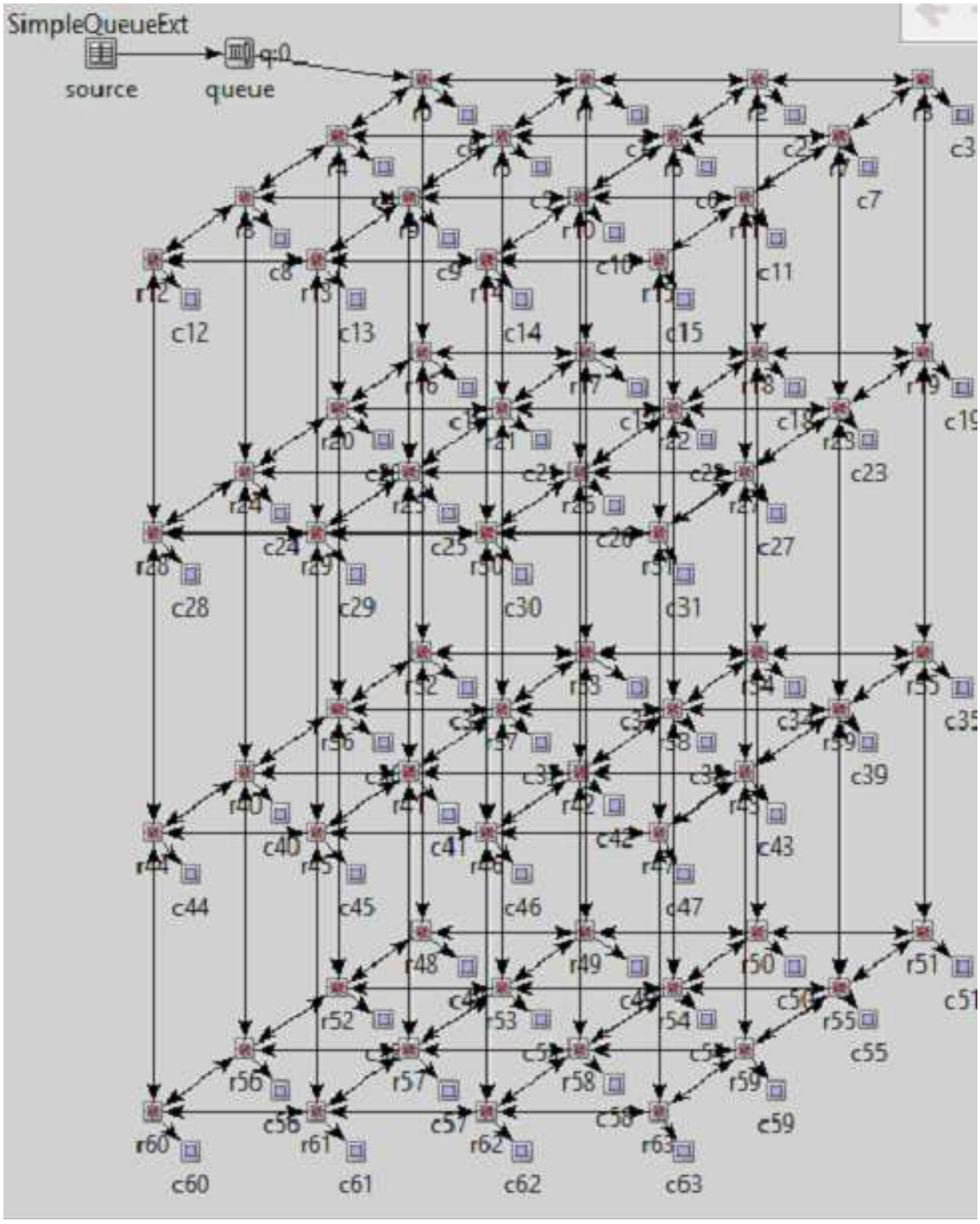

Figure 7.2: 3D Mesh Topology

Chapter 8

Experimental Results

For implementation purpose, we have used OMNeT++ simulator along with the use of in-built mapping package. In order to implement proposed mapping algorithms, we have considered the 2-dimensional 8 $\times$ 8 mesh topology for NoC. Initially, the application tasks are maintained in the task list, which can be the queued. From that task list, tasks are mapped on the cores, following the proposed mapping algorithms as mentioned in section 4. We have perform simulation varying the number of tasks from 64 to 128 and compared the results in terms of latency, queuing time, service time and energy consumption. Fig. 8.1 shows average latency of proposed mapping algorithms for mesh topology, and results are compared with random mapping algorithm.

Fig. 8.2-8.5 gives graphical analysis of queuing time for random and proposed mapping algorithms, and comparison of total queuing time is given in Fig. 8.6. Results obtained for service time required by each task, using OMNeT++ simulator, are shown in Fig. 8.7-8.10. Best mapping algorithm, in terms of total service time can be obtained by the comparative analysis of mapping algorithms as shown in Fig. 8.11.

In order to compute the energy consumption of topology, we have used Orion 2.0 simulator. With the help of orion simulator, we calculate the energy consumption of link represented as E_{Link} and energy consumption of router represented as E_{Router}. Table 8.1 shows the energy of router at different loads. Table 8.2 represents the energy consumption of link at different link length and different load. With the help of Equation 5.4, we compute the energy consumption of individual core in mesh topology using random mapping and proposed mapping algorithm as shown in Fig. 8.12 -8.15. Comparative analysis of energy consumption using mapping algorithms are given in Fig. 8.16 Table 8.3 shows the comparison of proposed and random mapping algorithm in terms of average latency, total queuing time and total service time for mesh topology.

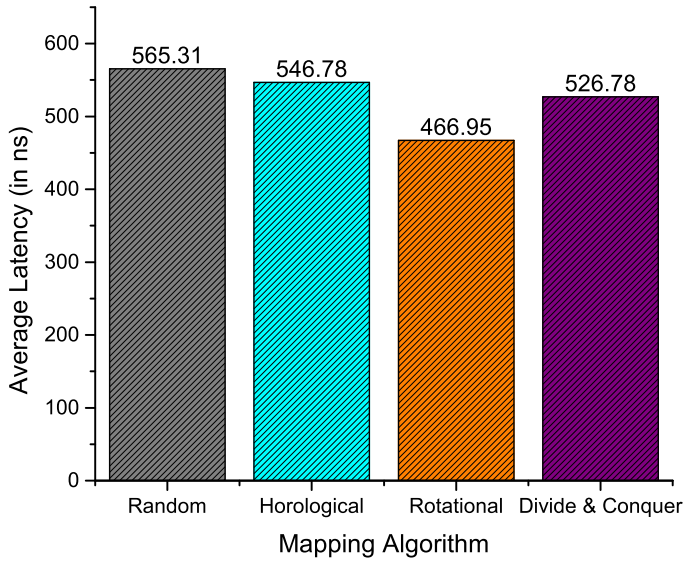

Figure 8.1: Average latency (in ns) of mapping algorithms in mesh topology

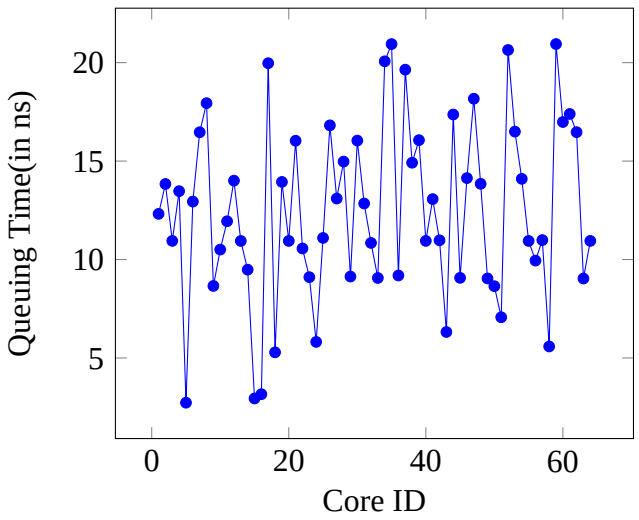

Figure 8.2: Queuing Time in mesh topology in random mapping

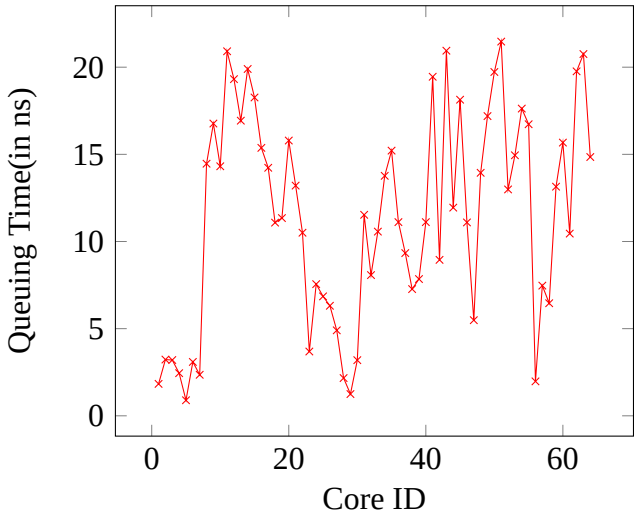

Figure 8.3: Queuing time in mesh topology in horological mapping

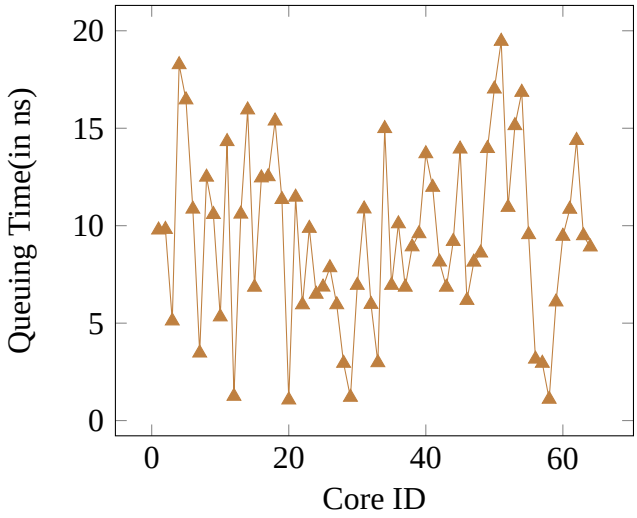

Figure 8.4: Queuing time in mesh topology in rotational mapping

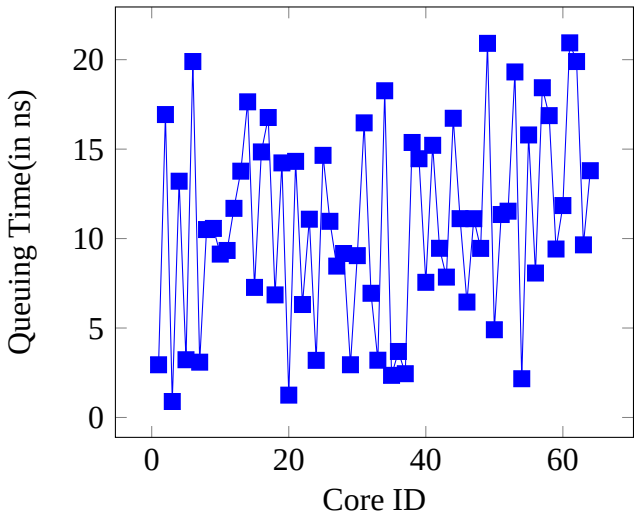

Figure 8.5: Queuing time in mesh topology in divide and conquer mapping

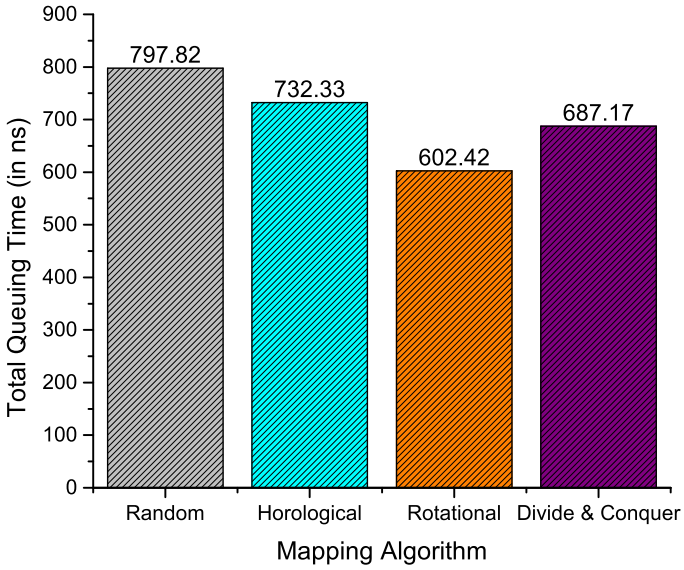

Figure 8.6: Total queuing time (in ns) of mapping algorithms in mesh topology

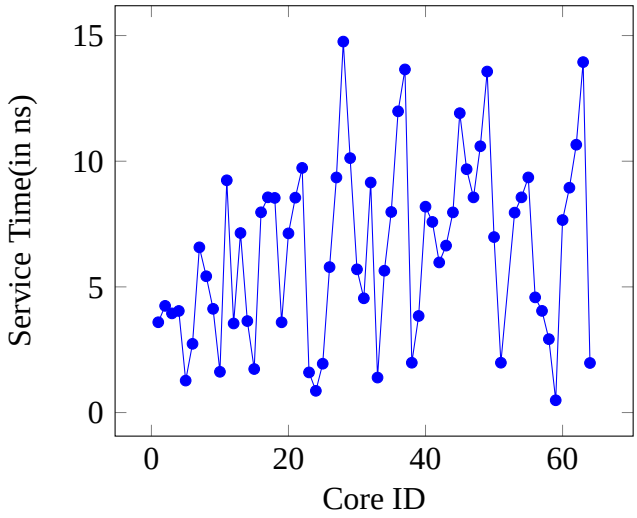

Figure 8.7: Service time in mesh topology in random mapping

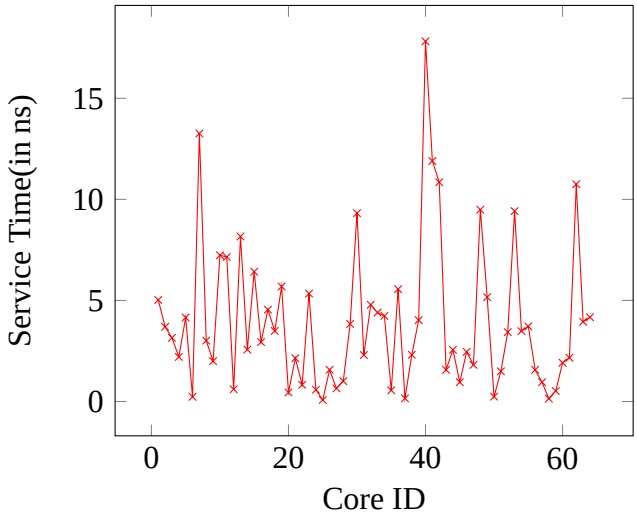

Figure 8.8: Service time in mesh topology in horological mapping

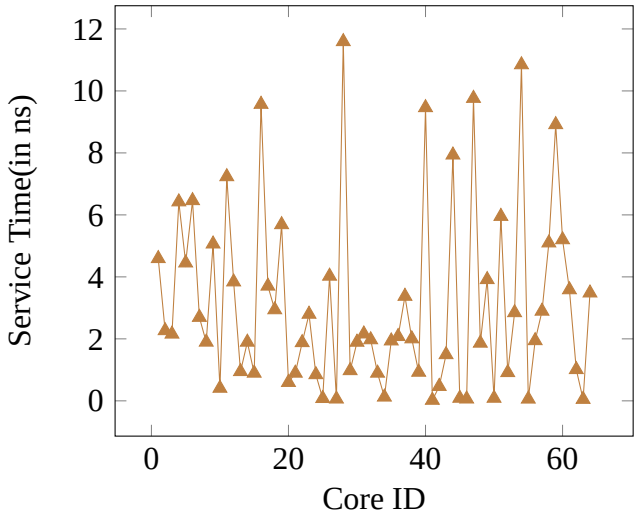

Figure 8.9: Service time in mesh topology in rotational mapping

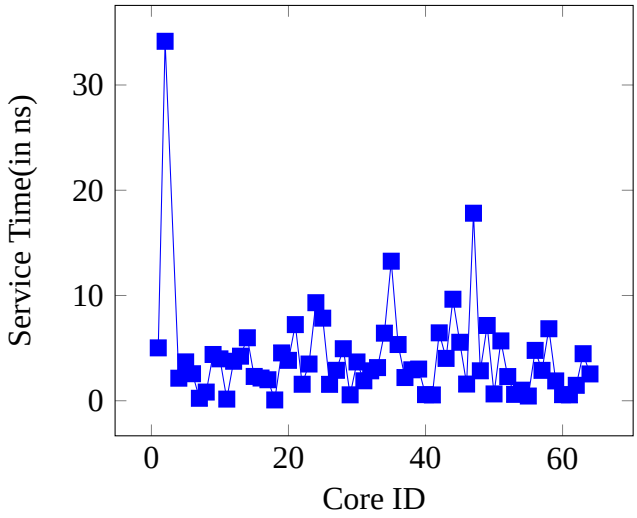

Figure 8.10: Service time in mesh topology in divide and conquer mapping

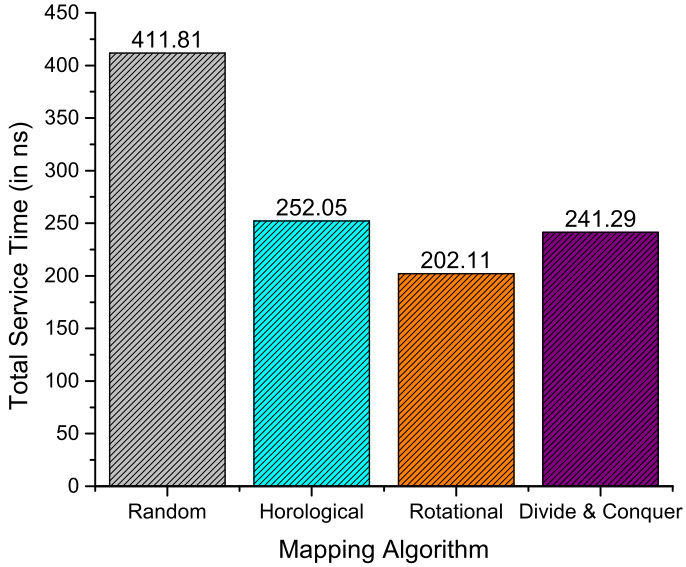

Figure 8.11: Total service time (in ns) of mapping algorithms in mesh topology

Table 8.1: Energy of router (in pJ) at different load

S. No.	Load	Energy of Router (in pJ)
1	0.2	16.8
2	0.4	27.138
3	0.6	37.46
4	0.8	47.78
5	1	58.09

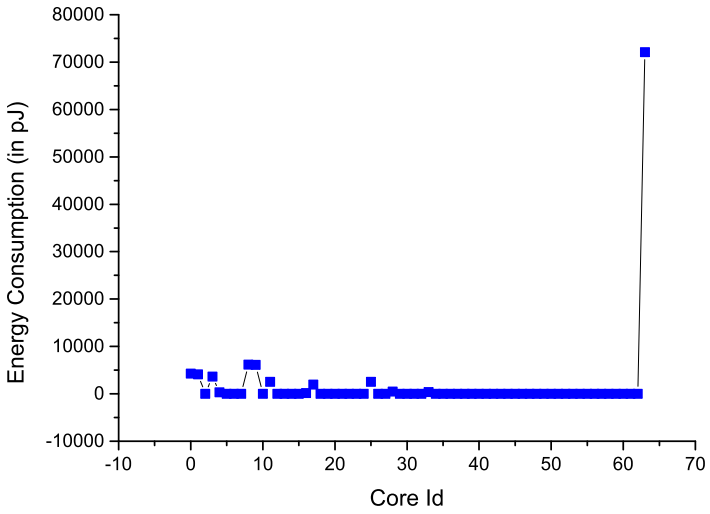

Figure 8.12: Energy consumption of cores in random mapping algorithm

Table 8.2: Energy of link (in pJ) at different load and link length (in mm)

	Link Length					
Load	1 mm	2 mm	3 mm	4 mm	5 mm	6 mm
0.2	7.65	15.31	22.97	30.63	38.28	45.94
0.4	12.10	24.20	36.30	48.40	60.50	72.60
0.6	16.54	33.08	49.62	66.17	82.71	99.20
0.8	20.98	41.97	62.95	83.94	104.93	125.91
1	25.42	50.085	76.28	101.71	127.14	152.57

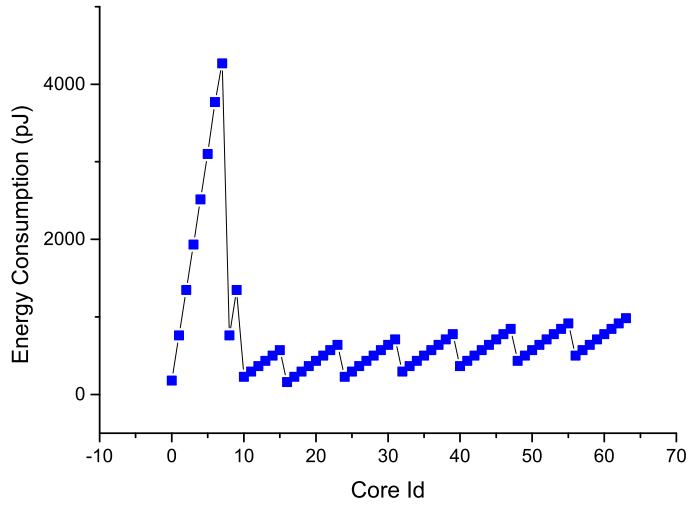

Figure 8.13: Energy consumption of cores in sequential mapping algorithm

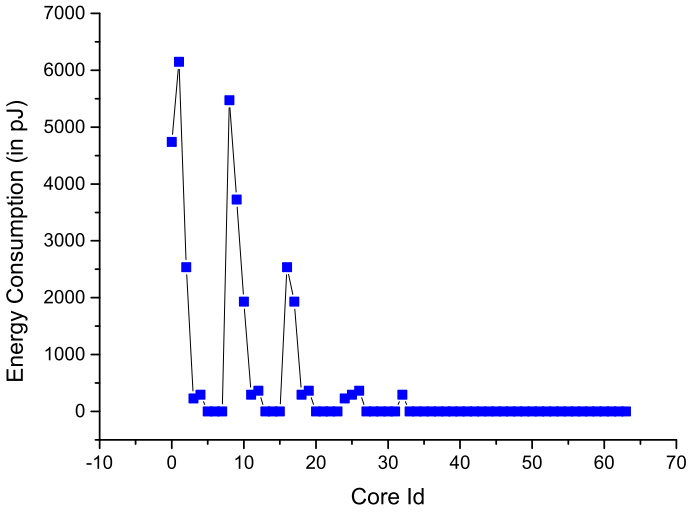

Figure 8.14: Energy consumption of cores in rotational mapping algorithm

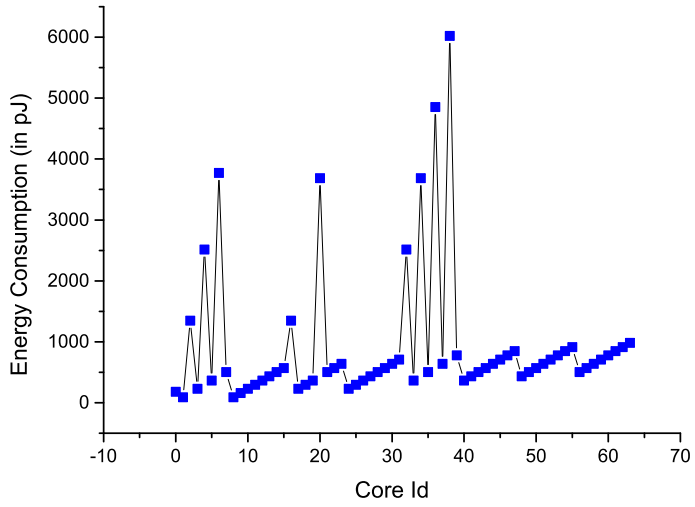

Figure 8.15: Energy consumption of cores in divide conquer algorithm

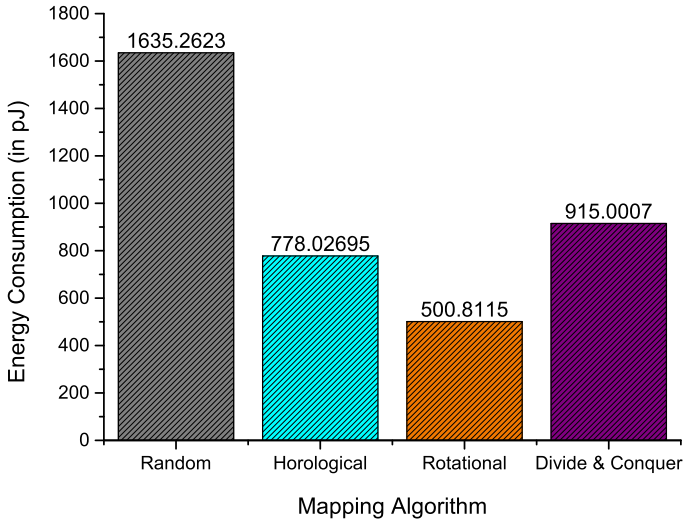

Figure 8.16: Comparison of energy consumption (in pJ) of mapping algorithms

Table 8.3: Comparison of average Latency, total queuing time and total service time of mapping algorithms (in ns)

S. No.	Mapping Algorithms	Average Latency	Total Queuing Time	Total Service Time
1	Random	565.31	797.82	411.81
2	Horological	546.78	732.33	252.05
3	Rotational	466.95	602.42	202.11
4	Divide and Conquer	526.78	687.17	241.29

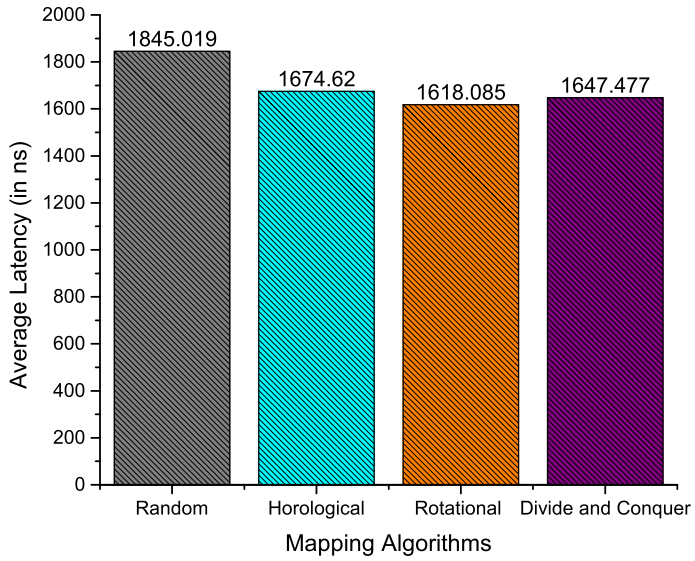

Figure 8.17: Average latency (in pJ) in mapping algorithm for 3D mesh topology

After implementing the proposed mapping algorithms on 2D mesh topology, we have also implemented on 3D mesh topology. For simulation purpose, we have considered $4 \times 4 \times 4$ 3D mesh topology. Fig. 8.17 shows average latency comparison in random mapping and proposed mapping algorithms. Queuing time taken by random mapping algorithm is shown in Fig. 8.18, whereas queuing time using proposed algorithms are given in Fig. 8.19-8.21. Comparison of total queuing time of mapping algorithms are plotted in Fig. 8.22. Service time in 3D mesh topology in case of random mapping is given in Fig. 8.23 and Fig. 8.24-8.26 shows service time of proposed mapping algorithms. The overall comparison of total service time of mapping algorithm in given in Fig. 8.27. Energy consumption in case of the random mapping algorithm is shown in the Fig. 8.28. For the description of the results obtained in case of the horological mapping algorithms for 3D mesh topology, Fig. 8.29 can be consulted. The results of the energy consumption in case of the rotational mapping algorithm are represented in Fig. 8.30. In Fig. 8.31, the energy consumption of the divide and conquer mapping algorithm for 3D mesh topology is shown. Fig. 8.32 is the comparative analysis of the energy consumption for the random, horological, rotational and divide and conquer mapping algorithm.

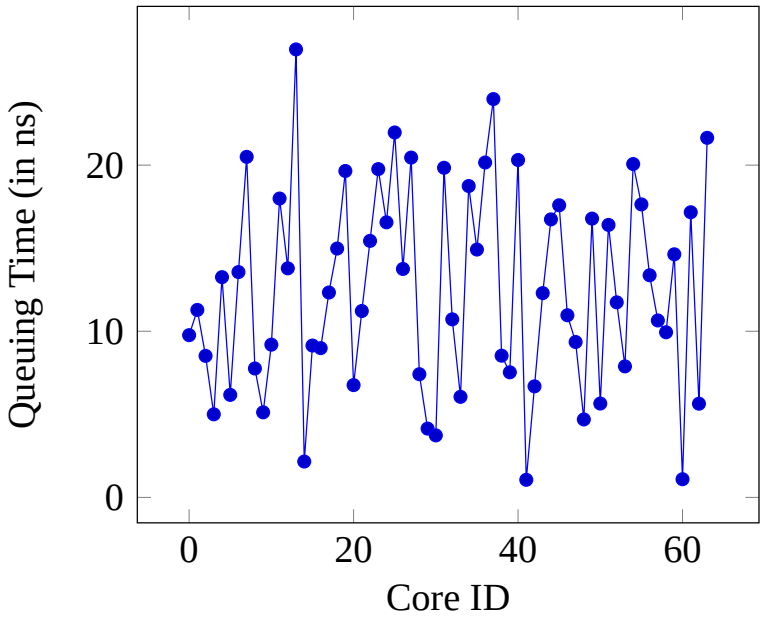

Figure 8.18: Queuing time (in ns) in 3D mesh topology in random mapping

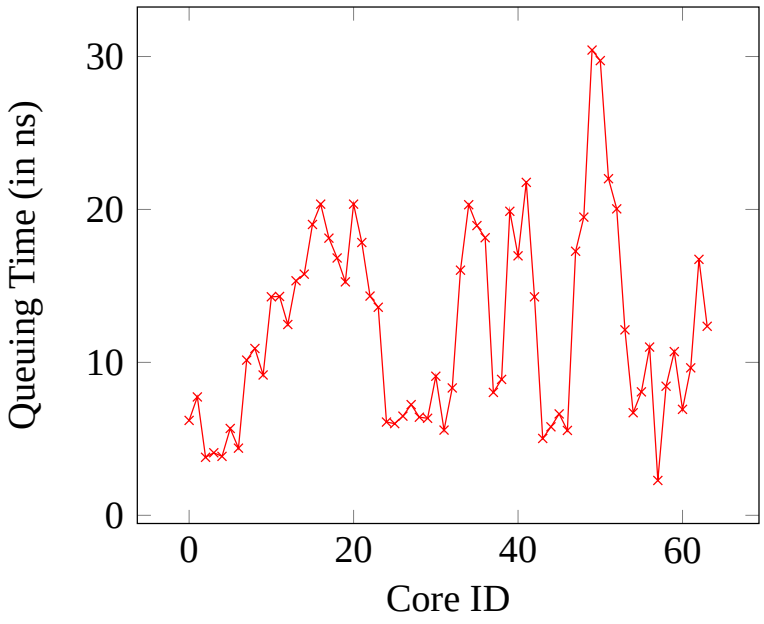

Figure 8.19: Queuing time (in ns) in 3D mesh topology in horological mapping

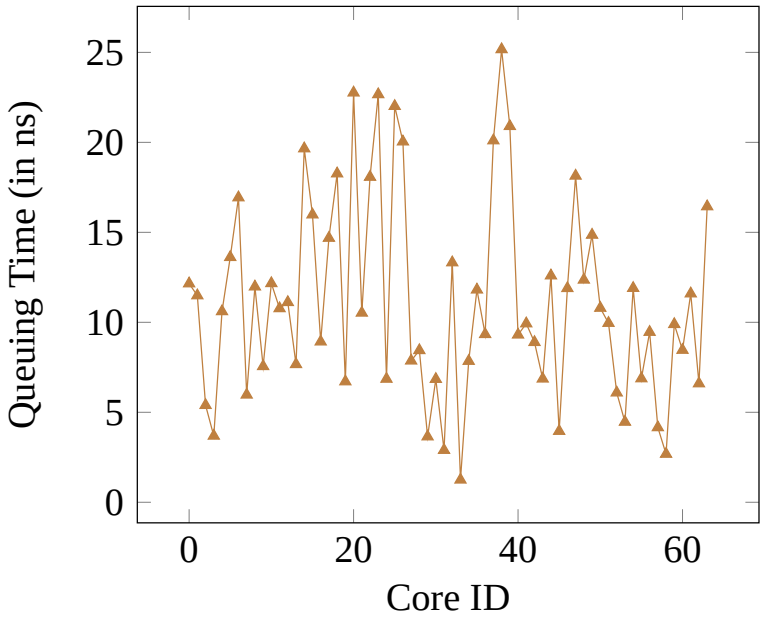

Figure 8.20: Queuing time (in ns) in 3D mesh topology in rotational mapping

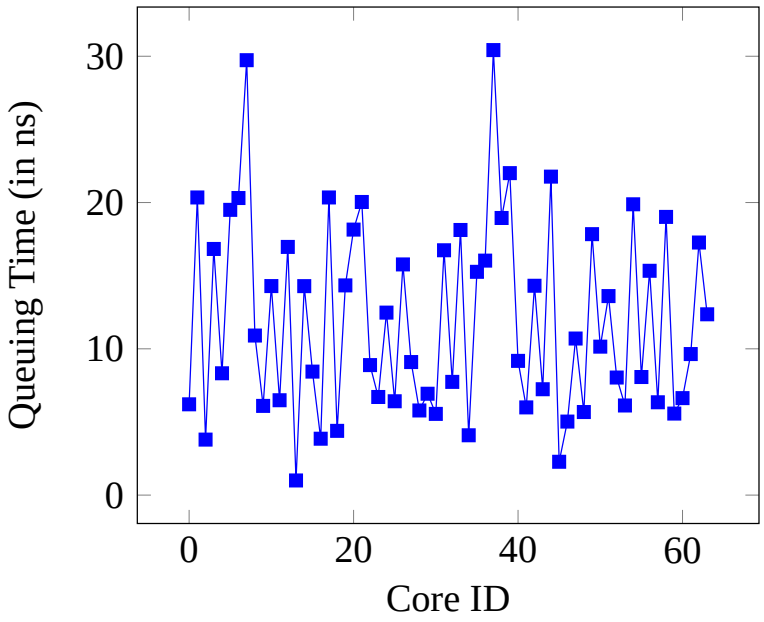

Figure 8.21: Queuing time (in ns) in 3D mesh topology in divide and conquer mapping

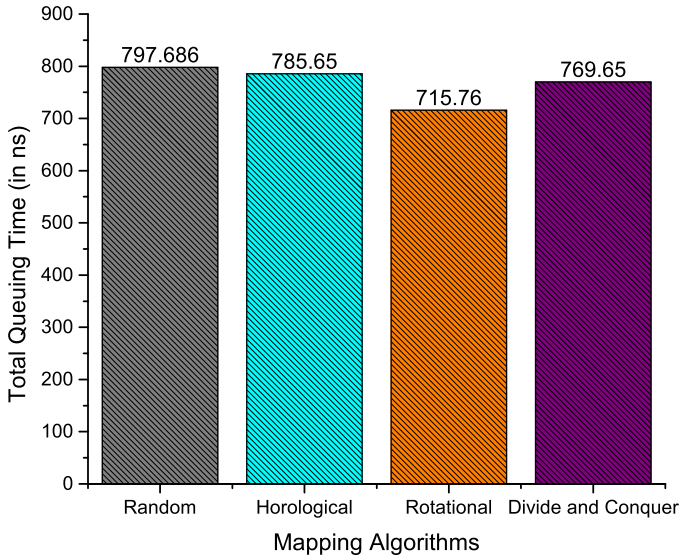

Figure 8.22: Total queuing time (in ns) of mapping algorithms in 3D mesh topology

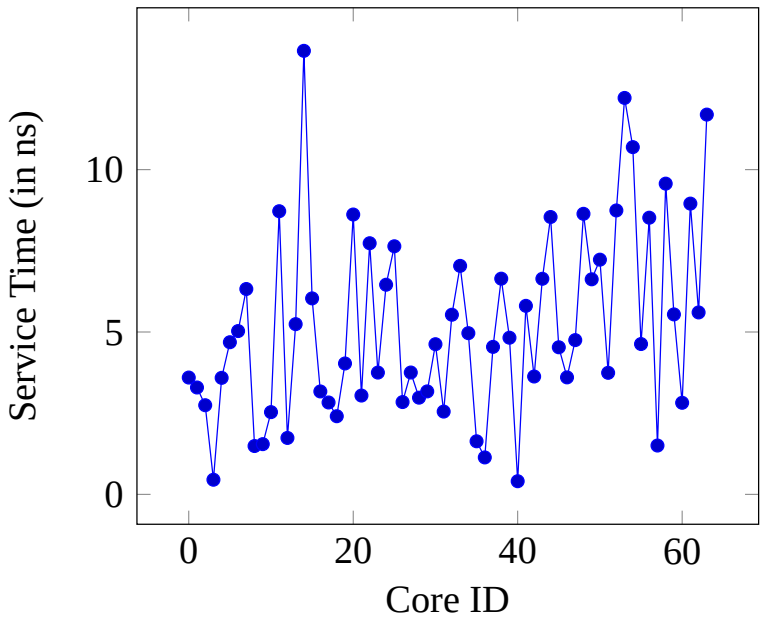

Figure 8.23: Service time (in ns) in 3D mesh topology in random mapping

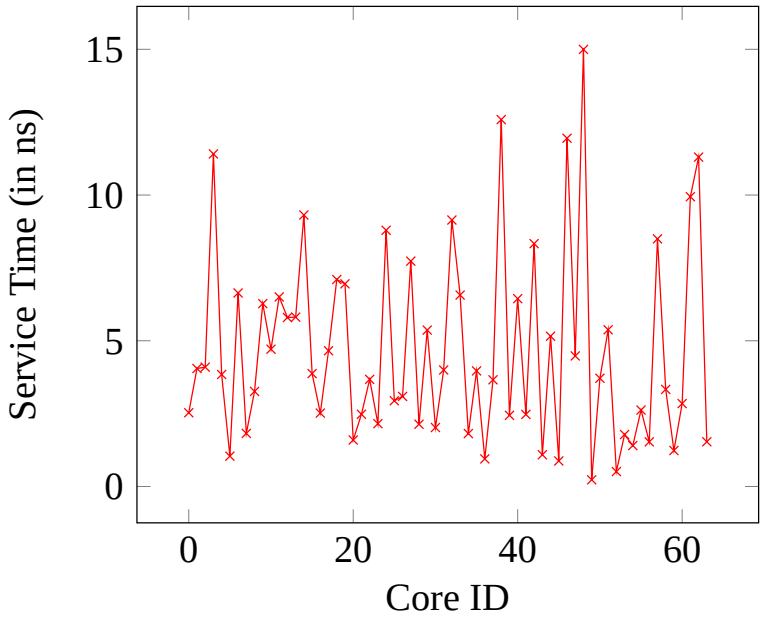

Figure 8.24: Service time (in ns) in 3D mesh topology in horological mapping

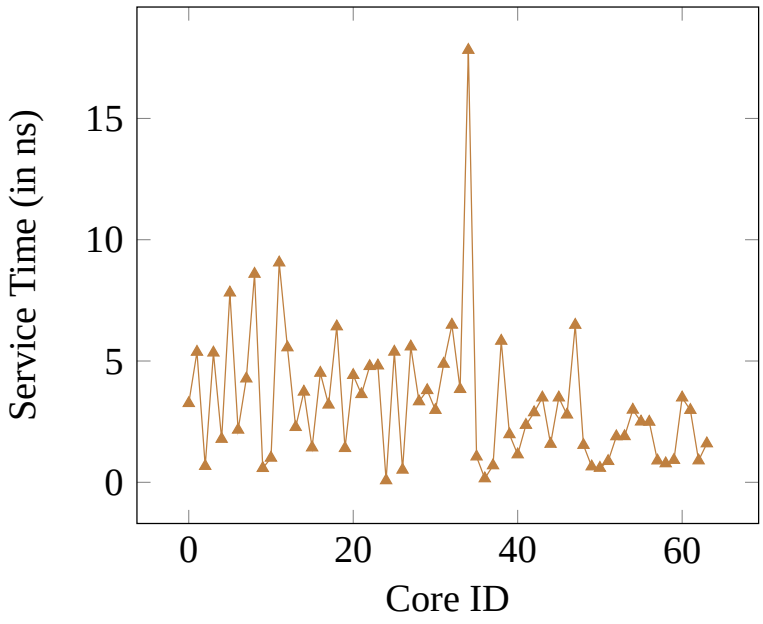

Figure 8.25: Service time (in ns) in 3D mesh topology in rotational mapping

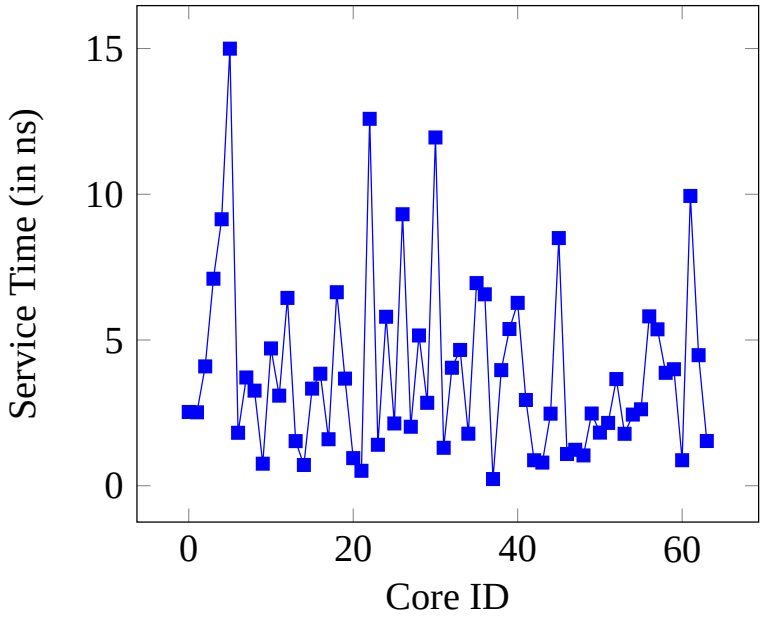

Figure 8.26: Service time (in ns) in 3D mesh topology in divide and conquer mapping

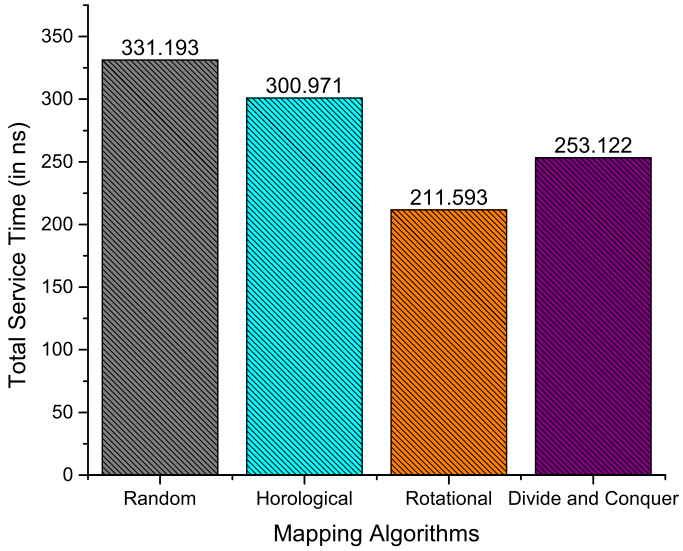

Figure 8.27: Total service time (in ns) of mapping algorithms in 3D mesh topology

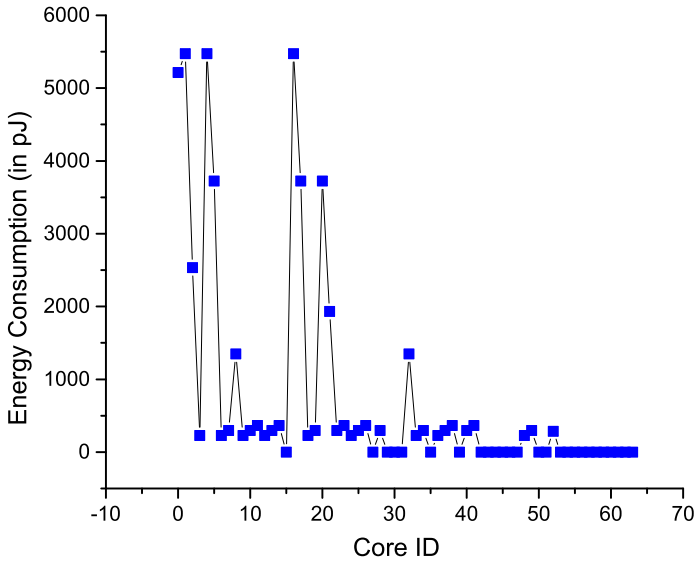

Figure 8.28: Energy Consumption (in pJ) in Random mapping algorithm for 3D mesh topology

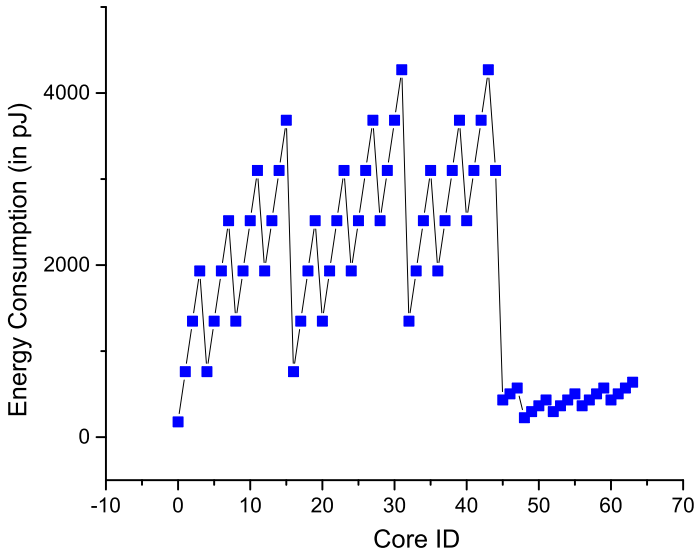

Figure 8.29: Energy Consumption (in pJ) in horological mapping algorithm for 3D mesh topology

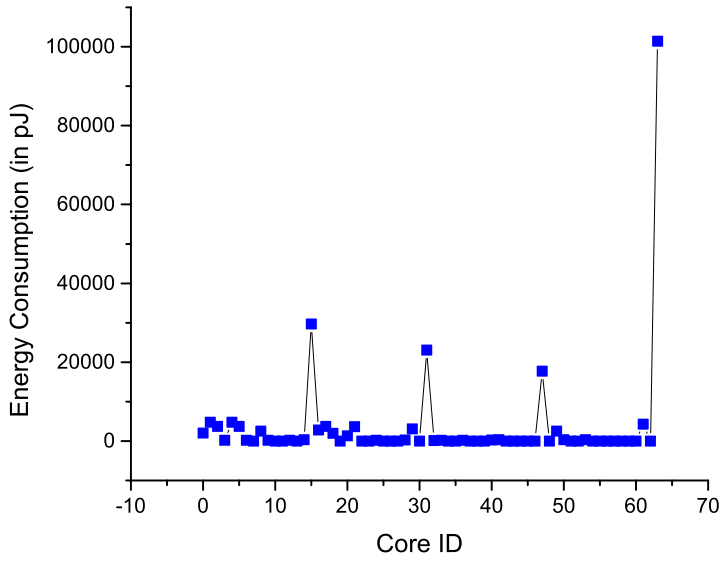

Figure 8.30: Energy Consumption (in pJ) in rotational mapping algorithm for 3D mesh topology

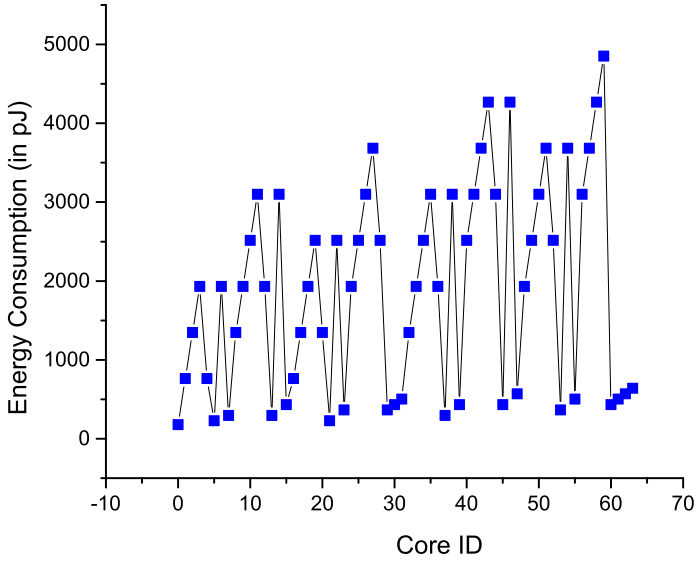

Figure 8.31: Energy Consumption (in pJ) in divide and conquer mapping algorithm for 3D mesh topology

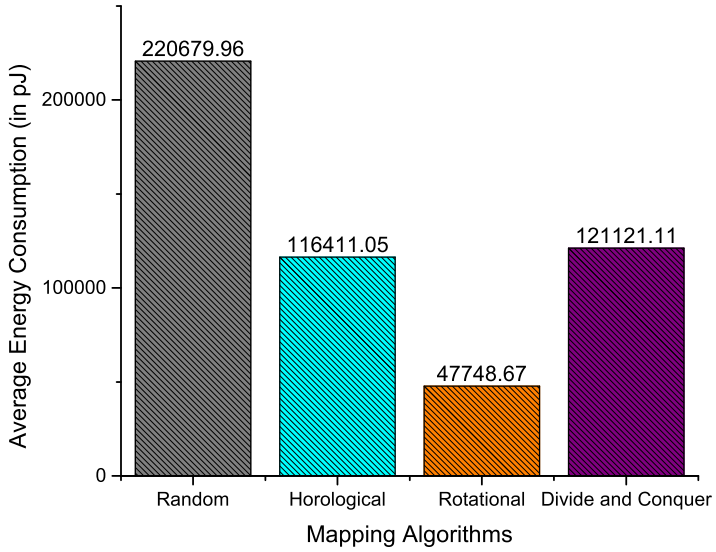

Figure 8.32: Comparison of energy consumption (in pJ) of mapping algorithms in 3D mesh topology

Chapter 9

Conclusion

Mapping algorithms need to be mapped on the most suitable cores such that the latency, service time, queuing time and the energy consumption are minimized. Different mapping algorithms are provided by different researchers but all these algorithm consider different parameters. So our main emphasis is to propose such a mapping algorithm which is best suitable in terms of latency, service time, queuing time and energy consumption. Along with this, the mapping algorithm should map the tasks onto the cores in such a way that load is balanced on to the grid to avoid problem of overheating. All the algorithms being discussed are generalized and hence can be implemented on the 3D mesh topology for NoC. We have simulated the mapping algorithms for 2D as well as 3D mesh topology.

Chapter 10

Future Work

Working on the arbitrator based IP cores in the mesh topology. Formalized the conceptual idea about the arbitrator based core mapping algorithm. As a future extension to this work i will simulate the concept of arbitrator based core mapping in OMNeT++ and will compare the results with other mapping algorithm.

Bibliography

[1] Naveen Choudhary. *"Migration of On-Chip Networks from 2 Dimensional Plane to 3 Dimensional Plane"*, International Journal of Engineering and Advanced Technology, Vol. 2, No. 4, pp. 516-519, April 2013.

[2] Paulo Santos, Jonathan Martinelli, Cezar Reinbrecht, Débora Matos, Altamiro Susin, *"Efficient Processing Element Unit for MPSoC NoC-based"*, 26th South Symposium on Microelectronics, pp. 153-156, April 2011.

[3] S. Kumar, A. Jantsch, J.P.Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani , *"A Network on Chip Architecture and Design Methodology"*, IEEE Computer Society Annual Symposium on VLSI, Pittsburgh, PA, pp. 105-112, 2002.

[4] Ankur Agarwal, Cyril Iskander, Ravi Shankar, *"Survey of Network on Chip (NoC) Architectures and Contributions"*, Journal of Engineering, Computing and Architecture, Vol. 3, No. 1, pp. 1-15, 2009.

[5] Dr. Srinivasan Murali, *"Designing Reliable and Efficient Networks on Chips"*, Lecture Notes in Electrical Engineering, Springer Netherlands, Vol. 34, 2009.

[6] Tobias Bjerregaard, Shankar Mahadevan , *"A Survey of Research and Practices of Network-on-Chip"*, ACM Computing Surveys (CSUR), Vol. 38, No. 1, Article No. 1, April 2006.

[7] L. Benini and G. D. Micheli, *"Networks on Chips: A New SoC Paradigm"*, IEEE Computer, Vol. 35, No. 1, pp. 70–78, Jan. 2002.

[8] W. J. Dally and B. Towles , *"Route Packets, Not Wires: On-chip Interconnection Networks"*, Design Automation Conference, pp. 684–689, 2001.

[9] A.Y. Weldezion, M. Grange, D. Pamunuwa, Lu Zhonghai, *"Scalability of Network-on-Chip Communication Architecture for 3-D meshes"*, 3rd ACM/IEEE International Symposium on Networks-on-Chip, San Diego, pp. 114 - 123, May 2009.

[10] Ling Wang, Jianye Hao, Feixuan Wang, *"Bus-Based and NoC Infrastructure Performance Emulation and Comparison"*, Sixth International Conference on Information Technology: New Generations, Las Vegas, pp. 855 - 858, April 2009.

[11] Wang Zhang, Wuchen Wu, Kuanglun Wang , *"Network on Chip and Key Research Problems"*, International Conference on E-Product, E-Service and E-Entertainment, Henan, pp. 1-5, Nov. 2013.

[12] R. Marculescu, Hu Jingcao, U.Y. Ogras, *"Key Research Problems in NoC Design: A Holistic Perspective"*, Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Jersey City, USA, pp. 69 - 74, Sept. 2005.

[13] Haytham Elmiligi, Fayez Gebali, M. Watheq El-Kharashi, *"Power-aware Mapping for 3D-NoC Designs Using Genetic Algorithms"*, 11th International Conference on Mobile Systems and Pervasive Computing, Vol. 32, pp. 538–543, 2014.

[14] Cai Jueping, Jiang Peng, Yao Lei, Hao Yue, Li Zan, *"Through-silicon via (TSV) Capacitance Modeling for 3D NoC Energy Consumption Estimation"*, 10th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), Shanghai, pp. 815 - 817, Nov. 2010.

[15] Jingcao Hu, Radu Marculescu, *"Energy-aware Mapping for Tile-based NoC Architectures Under Performance Constraints"*, Proceedings of the 2003 Asia and South Pacific Design Automation Conference, New York, pp. 233-239, 2003.

[16] Avi Kolodny, *"Networks on Chips: keeping up with Rent's Rule and Moore's Law"*, SLIP '07 Proceedings of the 2007 International Workshop on System Level Interconnect Prediction, pp. 55-56, 2007.

[17] G. Castilhos, M. Mandelli, G. Madalozzo, F. Moraes, *"Distributed Resource Management in NoC-based MPSoCs with Dynamic Cluster Sizes"*, 2013 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), Natal, pp. 153 - 158, Aug 2013.

[18] Chou Chen-Ling, R. Marculescu, *"FARM: Fault-Aware Resource Management in NoC-based Multiprocessor Platforms"*, Design, Automation and Test in Europe Conference Exhibition (DATE), Grenoble, pp. 1-6, March 2011.

[19] B. Osterloh, H. Michalik, B. Fiethe, K. Kotarowski, *"SoCWire: A Network-on-Chip Approach for Reconfigurable System-on-Chip Designs in Space Applications"*, NASA/ESA Conference on Adaptive Hardware and Systems, Noordwijk, pp 51 - 56 , June 2008.

[20] J.P. Diguet, S. Evain, R. Vaslin, G. Gogniat, *"NOC-Centric Security of Reconfigurable SoC"*, First International Symposium on Networks-on-Chip, Princeton, pp 223 - 232, May 2007.

[21] Kevin Chang,Sujay Deb, Amlan Ganguly, Xinmin Yu, Suman Prasad Sah, Pratha Pratim Pande, Benjamin Belzer and Deukhyoun Heo, *"Performance Evaluation and Design Trade-Offs for Wireless Network-on-Chip Architectures"*, ACM Journal on Emerging Technologies in Computing Systems, Vol. 8, No. 3, Article 23, August 2012.

[22] S. Umamaheswari, J. Rajapaul Perinbam, K. Monisha, J. Jahir Ali , *"Comparing the Performance Parameters of Network on Chip with Regular and Irregular Topologies"*, Trends in Network and Communications, Vol. 197, pp 177-186, 2011.

[23] Ankur Agarwal, Cyril Iskander, Ravi Shankar, *"Survey of Network on Chip (NoC) Architectures and Contributions"*, Journal of Engineering, Computing and Architecture, Vol. 3, No. 1, pp. 1-15, 2009.

[24] Liu Jian, L.R. Zheng, H. Tenhunen, *"A Circuit-Switched Network Architecture for Network-on-Chip"*, IEEE International SOC Conference, pp. 55-58, Sept. 2004.

[25] Angelo Kuti Lusala, Jean-Didier Legat, *"Combining SDM-Based Circuit Switching with Packet Switching in a Router for On-Chip Networks"*, International Journal of Reconfigurable Computing, Vol. 2012, Article ID 474765, pp. 1-16, 2012.

[26] Leonel P. Tedesco, Ney Calazans, Fernando Moraes, *"Buffer Sizing for Multimedia Flows in Packet-Switching NoCs"*, Journal Integrated Circuits and System, Vol. 3, No. 1, pp. 46-56, 2008.

[27] T. Pionteck, C. Albrecht, R. Koch, *"A Dynamically Reconfigurable Packet-Switched Network-on-Chip"*, Design Automation and Test in Europe, Munich, Vol. 1, March 2006.

[28] Yogita A. Sadawarte, Mahendra A. Gaikwad, Rajendra M. Patrikar, *"Implementation of Virtual Cut-Through Algorithm For Network on Chip Architecture"*, International Journal of Computer Applications, pp. 5-8, 2011.

[29] L.Rooban, S.Dhananjeyan, *"Design of Router Architecture Based on Wormhole Switching Mode for NoC"*, International Journal of Scientific and Engineering Research, Vol. 3, No. 3, pp. 1-5, March 2012.

[30] Faizal A. Samman, Thomas Hollstein, Manfred Glesner, *"Networks-On-Chip Based on Dynamic Wormhole Packet Identity Mapping Management"*, VLSI Design, Vol. 2009, Article ID 941701, pp. 1-15, January 2009.

[31] P. Ghosal , T.S. Das, *"Network-on-Chip Routing using Structural Diametrical 2D mesh architecture"*, Third International Conference on Emerging Applications of Information Technology (EAIT), Kolkata, pp 471 - 474, 2012.

[32] Young Bok Kim, Yong-Bin Kim, *"Fault Tolerant Source Routing for Network-on-chip"*, 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems, Rome, pp 12 - 20, Sept. 2007.

[33] Yongfeng Xu, Jianyang Zhou, Shunkui Liu, *"Research and Analysis of Routing Algorithms for NoC"*, 3rd International Conference on Computer Research and Development (ICCRD) , Shanghai, Vol. 2, pp 98 - 102, March 2011.

[34] A. Sharifi, M. Kandemir , *"Process Variation-Aware Routing in NoC based Multicores"*, Design Automation Conference (DAC), New York, pp 924 - 929, June 2011.

[35] S. Swapna, A.K. Swain, K.K. Mahapatra, *"Design and Analysis of Five Port Router for Network on Chip"*, Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics, Hyderabad, pp 51 - 55, Dec 2012.

[36] Wang Zhang, Ligang Hou, Jinhui Wang, Shuqin Geng , *"Comparison Research between XY and Odd-Even Routing Algorithm of a 2-Dimension 3X3 Mesh Topology Network-on-Chip"*, WRI Global Congress on Intelligent Systems, Xiamen, pp 329 - 333, May 2009.

[37] Lei Tang, S. Kumar, *"A Two-Step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture"*, Euromicro Symposium on Digital System Design, Belek-Antalya, Turkey, pp. 180-187, Sept. 2003.

[38] Jingcao Hu, Radu Marculescu, *"Energy and Performance-Aware Mapping for Regular NoC Architectures"*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 24, No. 4 , pp. 551-562, April 2005.

[39] Amit Kumar Singh, Thambipillai Srikanthan, Akash Kumar, Wu Jigang, *"Communication-Aware Heuristics for Run-time Task Mapping on NoC-Based MPSoC Platforms"*, Journal of Systems Architecture, Vol. 56, No. 7, pp. 242-255, July 2010.

[40] Yu-Kwong Kwok, Ishfaq Ahmad, *"Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors"*, ACM Computing Surveys, Vol. 31, No. 4, pp. 406-471, December 1999.

[41] Radu Marculescu, Paul Bogdan, *"The Chip is the Network: Toward a Science of Network-on-Chip Design"*, Electronic Design Automation, Vol. 2, No. 4, pp. 372-461, 2007

[42] Radu Marculescu, Umit Y. Ogras, Li-Shiuan Peh, Natalie Enright Jerger, Yatin Hoskote, *"Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives"*, IEEE

Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 28, No. 1, pp. 3-21, January 2009.

[43] Ramin Rajaei, Shaahin Hessabi, Bijan Vosoughi Vahdat, *"An Energy-Aware Methodology for Mapping and Scheduling of Concurrent Applications in MPSoC Architectures"*, 19th Iranian Conference on Electrical Engineering, Tehran, pp. 1, May 2011.

[44] Armin Mehran, Samira Saeidi, Ahmad Khademzadeh, Ali Afzali-Kusha, *"Spiral: A Heuristic Mapping Algorithm for Network-on-Chip"*, The Institute of Electronics, Information and Communication Engineers, Vol. 4, No. 15, pp. 478-484, 2007.

[45] JianWang, Yubai LI, Song Chai, Qicong Peng, *"Bandwidth-Aware Application Mapping for NoC Based MPSoCs"*, Journal of Computational Information Systems, Vol. 7, No. 1, pp. 152-159, 2011.

[46] Ning Wu, Yifeng Mu, Fang Zhou, Fen Ge, *"GA-MMAS: An Energy-aware Mapping Algorithm for 2D Network-on-Chip"*, Proceedings of the World Congress on Engineering and Computer Science, 2011.

[47] Wooyoung Jang, David Z. Pan, *"A3MAP: Architecture-Aware Analytic Mapping for Networks on-Chip"*, ACM Transactions on Design Automation of Electronic Systems, Vol. 17, No. 3, Article 26, 2012.

[48] Mohammad Behrouzian Nejad, Ebrahim Behrouzian Nejad, Aref Sayahi, Sayed Mohsen Hashemi, Javad Chaharlang, *"Mapping and Scheduling Techniques for Network-on-Chip Architecture"*, International Journal of Basic Sciences Applied Research, Vol. 2, No. 7, pp. 686-690, 2013.

[49] Abbou El Hassen Benyamina, Mohammed kamel Benhaoua, Pierre Boulet, *"Heuristics for Routing and Spiral Run-time Task Mapping in NoC-based Heterogeneous MPSOCs"*, International Journal of Computer Science Issues, Vol. 10, No. 4, 2013.

[50] Andrew B. Kahng, Bin Li, Li-Shiuan Peh, Kambiz Samadi, *"ORION 2.0: A Power-Area Simulator for Interconnection Networks"*, IEEE Transactions on Very Large Scale Integration, Vol. 20, No. 1, pp. 191 - 196, March 2011.

List of Publications

- Suchi Johari, **Arvind Kumar**, ***"Algorithmic Approach for applying Load Balancing During Task Migration in Multi-core System"***, International Conference on Parallel, Distributed and Grid Computing, pp. 27-32, December 2014 ***(Published).***

- **Arvind Kumar**, Suchi Johari, Vivek Kumar Sehgal, ***"Arbitrated IP Core Based Dynamic Task Mapping Algorithm for Networks-on-Chip"***, Seventh International Conference on Computational Intelligence, Communication Systems and Networks, Riga, Latvia, 2015 ***(Accepted).***

- Suchi Johari, **Arvind Kumar**, Vivek Kumar Sehgal, ***"Heterogeneous and Hybrid Clustered Topology for Networks-on-Chip"***, Seventh International Conference on Computational Intelligence, Communication Systems and Networks, Riga, Latvia, 2015 ***(Accepted).***