MACHINE LEARNING BASED DESIGN SPACE EXPLORATION OF NETWORKS-ON-CHIP

Thesis

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

Anil Kumar
(148005 CS14F05)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,
SURATHKAL, MANGALORE - 575025

AUGUST 2021

DECLARATION

by the Ph.D. Research Scholar

I hereby declare that the Research Thesis entitled **Machine Learning Based Design Space Exploration of Networks-on-Chip** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfilment of the requirements for the award of the Degree of **Doctor of Philosophy** in **Computer Science and Engineering** is a **bonafide report of the research work carried out by me**. The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.

(**148005 CS14F05**, **Anil Kumar**)

Department of Computer Science and Engineering

Place: NITK, Surathkal.

Date: August 21, 2021

CERTIFICATE

This is to *certify* that the Research Thesis entitled **Machine Learning Based Design Space Exploration of Networks-on-Chip** submitted by **ANIL KUMAR**, (Register Number: **148005 CS14F05**) as the record of the research work carried out by him, is *accepted as the Research Thesis submission* in partial fulfilment of the requirements for the award of degree of **Doctor of Philosophy**.

Dr. Basavaraj Talawar

Research Guide

Chairman - DRPC

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my research guide Dr. Basavaraj Talawar, for his guidance, support and encouragement throughout my research work at the Department of Computer Science and Engineering NITK, Surathkal.

I express heartfelt thanks to my Research Progress Assessment Committee (RPAC) members Dr. Deepu Vijayasenan, Associate Professor, E and C department and Dr. Jeny Rajan, Asst. Professor, Dept. of CSE, for their valuable suggestions and constant encouragement to improve my research work.

I sincerely thank all teaching, technical and administrative staff of the Computer Science and Engineering Department for their help during my research work. I also want to take the opportunity to thank all my teachers throughout the years. There are many whom I owe thanks.

I wish to express my love and gratitude to all my family members, especially my father and my mom for their encouragements and supports throughout all my studies from primary school to the current level.

Finally, I would like to express my gratitude to my wife Mahadevi and my friends for proofreading the papers submitted to conferences, journals and also the thesis.

Place: Surathkal Anil Kumar

Date: August 21, 2021

ABSTRACT

As hundreds to thousands of Processing Elements (PEs) are integrated into Multiprocessor Systems-on-Chip (MPSoCs) and Chip Multiprocessor (CMP) platforms, a scalable and modular interconnection solution is required. The Network-on-Chip (NoC) is an effective solution for communication among the On-Chip resources in MPSoCs and CMPs. Availability of fast and accurate modelling methodologies enable analysis, development, design space exploration through performance vs. cost tradeoff studies, and testing of large NoC designs quickly. Unfortunately, though being much more accurate than analytical modelling, conventional software simulators are too slow to simulate large-scale NoCs with hundreds to thousands of nodes.

Machine Learning (ML) approaches are employed to simulate NoCs to address the simulation speed problem in this thesis. A Machine Learning framework is proposed to predict performance, power and area for different NoC architectures. The framework provides chip designers with an efficient way to analyze NoC parameters. The framework is modelled using distinct ML regression algorithms to predict performance parameters of NoCs considering different synthetic traffic patterns. Because of the lack of trace data from large-scale NoC-based systems, the use of synthetic workloads is practically the only feasible approach for emulating large-scale NoCs with thousands of nodes. The ML-based NoC simulation framework enables a chip designer to explore and analyze various NoC architectures considering both 2D & 3D NoC architectures with various configuration parameters like virtual channels, buffer depth, injection rates and traffic pattern.

In this thesis, four frameworks have been presented which can be used to predict the design parameters of various NoC architectures. The first framework named Learning-Based Framework (LBF-NoC) which predicts the performance, power, area parameters of direct (mesh, torus, cmesh) and indirect (fat-tree, flatfly) topologies.

LBF-NoC was tested with various regression algorithms like Artificial Neural Networks with identity and relu activation functions, different generalized linear regression algorithms, i.e., lasso, lasso-lars, larsCV, bayesian-ridge, linear, ridge, elastic-net and Support Vector Regression (SVR) with *linear*, *Radial Basis Function*, *polynomial* kernels among these SVR provided the least error hence, it was selected for building the framework. The existing framework was enhanced by using multiprocessing scheme named Multiprocessing Regression Framework (MRF-NoC) to overcome the issue of simulating NoC architecture '*n*' number of times for 2D Mesh and 3D Mesh in the second framework. The third framework named Ensemble Learning-Based Accelerator (ELBA-NoC) is designed to predict worst-case latency analysis and to predict the design parameters of large scale architectures using the random forest algorithm. It was designed to predict results of five different NoC architectures which consist of both 2D (Mesh, Torus, Cmesh) and 3D (Mesh, Torus) architectures. Later the fourth framework named Knowledgeable Network-on-Chip Accelerator (K-NoC) is presented to predict two types of NoC architectures one with a fixed delay between the IPs and another with the accurate dealy and it was build using random forest algorithm.

The results obtained from the frameworks has been compared with the most widely software simulators like Booksim 2.0 and Orion. The LBF-NoC framework gave an error rate of 6% to 8% for both direct and indirect topologies. It also provided a speedup of 1000× for direct topologies and speedup of 5000× for indirect topologies. By using MRF-NoC all the various NoC configurations considered can be simulated in a single run. ELBA-NoC was able to predict the design parameters of five different architectures with an error rate of 4% to 6% and a minimum speedup 16000× when compared to the cycle-accurate simulator. later, K-NoC was able to predict both NoC architectures considered one with fixed delay and another with the accurate delay. It gave a speedup of 12000× and error rate less than 6% in both the cases.

Keyword: Network-on-Chip, 2D NoC, 3D NoC, Simulation, Performance modelling, Machine Learning, Prediction, Regression, Support Vector Regression, Ensemble Learning, Random Forest, Booksim, Performance; Power, Area, Router, Traffic Pattern.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

In the last few decades, there is an advancement in deep submicron technology which have given rise to the incorporation of thousands of Intellectual Property (IP) cores executing simultaneous processes on a single chip following Moore's law. It further continued to increase using Instruction Level Parallelism (ILP), using faster clock frequency and incrementing the number of transistors (Schaller [1997], Hennessy and Patterson [2011]). One of the remarkable characteristics of transistors which fuels their rapid growth is an increase in speed and cost decrease as their size is reduced. International Technology Roadmap for Semiconductors (ITRS) illustrates that the wiring delay is growing exponentially because of the increased capacitance caused by narrow channel width and increased crosstalk. Therefore, the wiring and consequently communication between cores is one of the main limiting factors to be concerned.

To overcome the problem of complexity, System-on-Chip (SoC) was proposed as an efficient architecture which was useful for simplifying and improving the performance of the chip. In SoCs, interconnect structures between IP cores traditionally use a shared bus design and a point-to-point design. As the number of cores continues to rise, neither conventional bus-based nor point-to-point architectures provide scalable solutions to meet the tight power and performance requirements of on-chip communication requirements. This problem causes a significant change in the architecture of microprocessors and therefore the current design approach needs to be changed from computation-based design to communication-based design. Depending on application domains and versatility, SoC can be classified into two categories: (1) general-purpose

multiprocessor SoC (MPSoC) and (2) application-specific SoC.

Networks-on-Chip (NoCs) have become the tangible on-chip communication technique as a viable alternative to design for the present and future generation of SoC designs (Dally and Towles [2001], Benini and De Micheli [2002], Lee et al. [2006]). Currently, NoCs are the most suitable interconnection architecture for modern many-core systems. The NoC provides scalable, flexible and parallel communication technology for the integration of complex logic cores in SoCs. As NoCs become the de facto on-chip communication standard, methodologies of performance evaluations emerge as critical components for validating new architectures of NoCs as well as their Design Space Exploration (DSE). Many industrial products use the NoC technology, such as Intel SCC (Vangal et al. [2008]), Polaris (Howard et al. [2010]), Tilera64 (Bell et al. [2008]).

NoC has become an emerging area of research as it has been shown that NoCs have the potential to be an effective and efficient way of communicating fabric in CMP and MPSoC systems. NoC is an essential part of system design especially in situations where the number of cores on the chip will increase in the near future. With thousand-core systems planned in the future, higher demand for connectivity and traffic will bring more pressure on NoCs and will consume more power. However, energy efficiency is already a major concern (Borkar [2007, 2010]) for researchers and designers, as NoCs consume considerable power in modern CMPs (Hoskote et al. [2007], Taylor et al. [2002]). Research and development of NoCs, therefore, have a key role to play in designing future large-scale architectures with hundreds to thousands of cores.

However, the lack of fast modelling methodologies which can provide a high degree of accuracy is a major obstacle to research and development of large-scale NoCs. Analytical models such as those proposed in (Peh and Dally [2001], Ogras et al. [2010]) are extremely fast, but in many cases can incur significant inaccuracy. Thus, NoC designers often rely on simulation to test their ideas and make design decisions, which will provide much more accurate evaluation results and insights into the designs.

Simulation is the de facto evaluation method not only in the analysis of NoC design but in general computer architecture as well. Simulation can provide much more accurate evaluation results than analytical simulation while providing relatively low

development costs compared to hardware prototyping. It also a key component to study the system design process. This process helps to handle the increasing modern many-core processors complexity at different levels. Simulation tool allows to perform large-scale DSE in software and it estimates performance cost, power, and reliability of the design, etc. Simulators serve the following purposes: 1) Evaluation of different hardware designs, without actual systems being implemented. 2) Creating opportunities for testing components or systems that do not exist. 3) Estimating various metrics like performance, power and area parameters of architecture. 4) Simulators can produce a large set of performance data through a single execution and debugging before system implementation. Upon detection of an error in a real system, it typically requires re-booting and re-running of the code.

1.1.1 Network-on-Chip Simulators

There are different evaluation tools and methodologies intended to support NoC research. Each developed tool attempts to cover one or more aspects of NoC DSE like:

- Configuration of nodes.
- Configuration of the NoC like topology, routing algorithms, virtual channels and others.
- Data communication requirements.
- Benchmarking and analysis of results.

Various NoC simulators were built for NoCs space exploration assessment and design. (Halavar et al. [2019], Achballah and Saoud [2013]) provides a list of NoC simulators and tools available for simulating and analyzing different NoC types. In this section, we explore some of the open-source NoC simulators.

Booksim2.0 (Jiang et al. [2013]) is an open-source, cycle-accurate simulator for NoCs. It offers a wide range of configurable NoC parameters such as topology, algorithm routing, flow control, traffic and injection rate. It supports 10 topologies such as mesh, torus, cmesh, fat tree and others. Various routing algorithms for the supported topologies can be configured to direct the packets. Booksim2.0 results are validated

against the Register Transfer Language (RTL) implementation of the NoC router for accuracy. Noxim (Catania et al. [2015]) is a NoC simulator developed in SystemC. It has a command-line interface to parametrize various components of NoC. In Noxim user can customize network size, buffer size, packet size, routing algorithm, injection rate, traffic pattern and it only supports mesh architecture. NoCTweak (Tran and Baas [2012]) is similar to Noxim simulator developed using SystemC. It supports only 2D mesh topology with a core and network interface consisting of each node. Topaz (Abad et al. [2012]) supports configuration parameters for different components of the network such as router, topology, and traffic. Users can integrate this simulator with full-system simulation tools such as GEM5 (Binkert et al. [2011]) for holistic performance evaluation and Orion (Kahng et al. [2009]) for power analysis. GARNET (Agarwal et al. [2009a]) is an NoC simulator built into the full system simulator GEM5. Orion2.0 (Kahng et al. [2009]) includes power and area model for accurate estimation of the power and area of network interconnection routers. These results can be used in early phases to obtain effective exploration of NoC design space.

Cycle accurate simulator and simulation with synthetic workloads has been considered in this thesis. Where cycle accuracy provides a level of simulation accuracy which simulates the target design on a cycle-by-cycle basis. With the same input, all the state elements in the simulation contain the same values at every clock cycle as those in a real hardware implementation. Thus, the evaluation results obtained from cycle-accurate simulations are completely reliable. In NoC research specifically in computer architecture research in general, cycle-accurate simulations are extremely important. Currently, due to the lack of trace data from large-scale NoC-based systems, the use of synthetic workloads is practically only the feasible approach for emulating large-scale NoCs with thousands of nodes. In real applications, synthetic workloads are those based on mathematical modelling of common traffic patterns. They are high in flexibility and easy to create. A set of carefully designed synthetic workloads can provide fairly thorough coverage of emulated NoCs characteristics.

Stand-alone NoC simulators, which often support much more accurate NoC models, are quicker compared to full-system simulators, but simulating a large-scale NoC often takes a significant amount of time (Jiang et al. [2013], Kumar and Talawar [2018], Angepat et al. [2014], Wang et al. [2012], Guo et al. [2015]). Which can be

seen in Figure 1.1 it shows the study of simulation time versus architecture sizes for 2D, 3D and 4D architectures. The x-axis specifies the topology sizes (n×n) and y-axis specifies the simulation time in hours. The experiments were done using Booksim simulator which is one of the most widely used simulator in NoC community(Chu et al. [2017], Van Chu [2015], Van Chu et al. [2015]). The experiment was conducted for Mesh topology with virtual channel 4, injection rate 0.005 and buffer depth 8 for uniform random traffic pattern. The results show that simulation time varies from 6 seconds to 18 days. Because of this, most of the previous studies are limited to NoCs with around 100 (10×10) nodes. This becomes very difficult for designers to analyze, test the MPSoCs and CMPs for next-generation systems. There is a need for a fast and accurate evaluation of the performance regarding various configurations exploring the design space of large NoC designs as thousands of cores are targeted in many-core architectures in the near future (Sanchez and Kozyrakis [2013], Borkar [2007], Kurian et al. [2010]). It is crucial to improve the simulation speed while maintaining the simulation accuracy to investigate novel designs with hundreds to thousands of nodes.

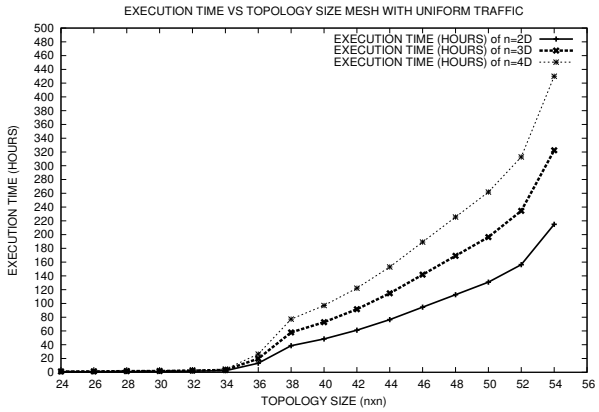


Figure 1.1: Booksim simulation time for k-ary 2, 3, 4-dimensional Mesh networks(k=2 to 56)

1.2 Problem Statement and Objectives

Simulation plays a key role in NoCs for analyzing and testing new architectures. To achieve the best performance vs. cost tradeoff, this is important for both the intercon-

nect designer as well as the system designer in the earlier stages of design. Software simulators are too slow for evaluating medium and large scale NoCs. Simulation for optimization purposes is therefore very difficult to use and these architectures need to be validated against discrete synthetic traffic patterns varying topology sizes. To investigate novel designs with hundreds to thousands of nodes, improving simulation speed while maintaining the accuracy of the simulation is crucial. To address the simulation speed problem, there is a need of fast and accurate model which provides the performance, power and area parameters of different NoC architectures inconsiderable amount of time which is done using machine learning algorithms.

1.2.1 Objectives

1. Creating a machine learning framework to predict design parameters of NoCs.
    - Comparing efficiency of various machine learning algorithms against cycle-accurate simulators and identifying the factors affecting prediction of NoC parameters.
2. Enhancement of machine learning framework to predict performance, area and power parameters of 3D NoC topologies using Multiprocessing.
    - Evaluation of enhanced framework by considering various synthetic traffic patterns and standard benchmarks.
3. Machine learning framework to analyse the maximum latency values of both 2D and 3D NoC architectures as they enter the saturation region.

1.3 Contributions

This thesis makes four contributions by developing a machine learning framework for evaluation of large scale NoC architectures considering various configurations and architectures.

- A highly configurable machine learning framework is created which provides the complete DSE of 2D and 3D NoCs architectures considering various design constraints. This is done by comparing various machine learning algorithms and selecting an algorithm which provided efficient results.

- Along with machine learning algorithms, multiprocessing scheme is used to overcome the issue of simulating NoC architecture multiple times. By using multiprocessing scheme various NoC configurations can be executed simultaneously.
- Enhancement of the framework to predict the worst-case latency analysis.
- Creating a unified framework which provides the performance, power and area parameters of two different scenarios, one with a fixed delay between the IPs and floorplan based (accurate) delay between the IPs of NoCs.

1.4 Organization of the Thesis

The rest of the part of the thesis is organized as follows:

- **Chapter 2: Literature review**: This chapter is structured into two sections. The first section gives a brief introduction of NoCs and the second section gives the survey of various works which have used machine learning in NoCs.
- **Chapter 3: Machine Learning based framework for NoCs**: This chapter presents comparision of various machine learning algorithms against cycle-accurate simulators and identifying the factors affecting prediction of NoC parameters. And, enhancing the machine learning framework using multiprocessing.
- **Chapter 4: Ensemble Learning-Based Accelerator**: This chapter presents a machine learning framework to analyse worst case latencies of both 2D and 3D NoC architectures along with design parameters.
- **Chapter 5: Floorplanned-Based Learning Framework**: This chapter presents framework which predicts design parameters of two different simulators one with fixed delay and another with floorplan based delay of 2D Mesh architecture.
- **Chapter 6: Summary and Conclusions**: The contributions of this thesis, along with some important conclusions, outlines for future research directions have been summarized.

Chapter 2

Literature Review

This chapter includes two sections. The first section gives a basic introduction to NoCs. The second section provides the various works done using machine learning on NoCs.

2.1 NoC Basics

NoC has emerged as a highly structured and efficient On-Chip reliable communication framework in CMPs and SoCs to achieve high-performance and scalability. It has appeared as an infrastructure for interconnection to reduce global wire delays by designing separate, flexible interconnection fabrics to enable high-speed communication between cores. NoC platforms will allow design productivity to develop as fast as technology capabilities, and eventually close to the design productivity gap (Jantsch et al. [2003]). Furthermore, NoCs have inherent redundancy which helps tolerate failures and communication bottlenecks (Dally and Towles [2001]). In NoC, data is generally transferred via wormhole switching through virtual channel-based switches. Data packets are broken down and transmitted in the form of flow control units or flits that signify the smallest amount of information in a packet that can be transmitted in one clock cycle between adjacent switches (Duato et al. [2002]).

A basic NoC architecture consists of various techniques and blocks that are connected to form interconnected architecture for an SoC. The main aspects of the NoC architecture are:

- Network topology
- Router

- Flow control
- Routing algorithm
- Network interface

2.1.1 Network Topology

Network topology is the interconnection and organization of a number of nodes and links in the network. In NoC, the topology relies on many factors, i.e. efficiency, scalability, power consumption, design complexity, etc (Moadeli et al. [2009]). In addition, network topology helps determine the number of nodes within NoC and determines the length of links between nodes. NoC topology can be divided into two main groups, i.e., direct and indirect topology (Jantsch et al. [2005]) which can be seen in Figure 2.1. In direct topology, each router is directly connected to its neighbouring routers and its local IP core (Cota et al. [2011], Benini and De Micheli [2002]). In indirect topology, however, not all routers are connected to the IP cores, since some of these routers are only used to relay packets inside the network. Furthermore, the number of routers is greater in indirect topology than the IP cores. In comparison, direct topology, the number of routers and cores will be the same. One of the most common examples of direct topology are Mesh, Torus, and Cmesh along with Fat-Tree and Flatfly topologies is one of the most common examples of indirect topology. These examples of direct and indirect topology are the commercial types most popular in NoC (Duato et al. [2003]). The most popular commercial types in NoC are these examples of direct and indirect topology, which will be explained in detail in the section below.

- **Mesh Topology:** Mesh topology is the simplest and most effective topology for NoC so far, because of its regular structure with uniform router design (Pande et al. [2005]). It is used in most recent multi-core chips, such as the Intel SCC 48-core (Howard et al. [2010]), TFlops 80-core (Vangal et al. [2007]), and Tilera 64-core (Bell et al. [2008]). The mesh topology consists of a grid of routers positioned alongside the routers with interconnecting nodes. Each router is linked to four neighbouring routers and one heart, except those at the edges. Thus, the mesh has a radix (number of ports) of five. Figure 2.2(a) shows the

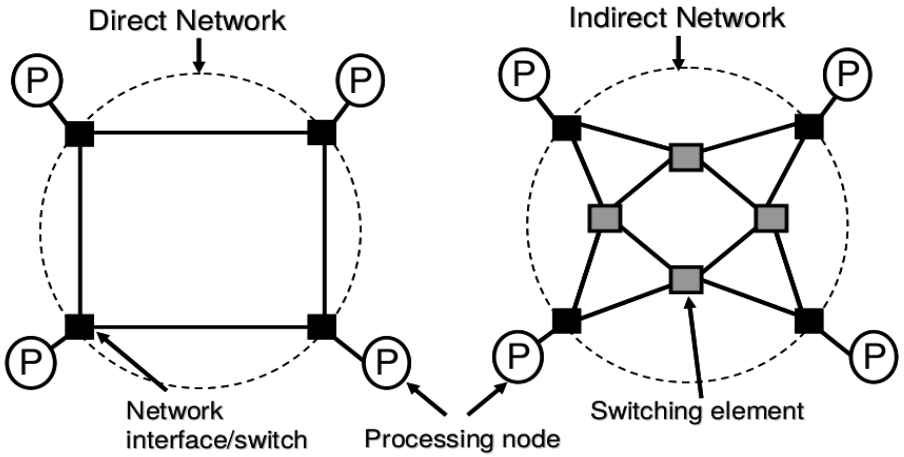


Figure 2.1: Direct and indirect networks (P: Intellectual Property) (Vangal [2007]).

layout for a mesh of (4 ×4) 16 nodes. The mesh topology presumes all links are of the same length. The area requirements for mesh topologies are easier to predict since area requirements grow almost linearly with an increase in the number of cores.

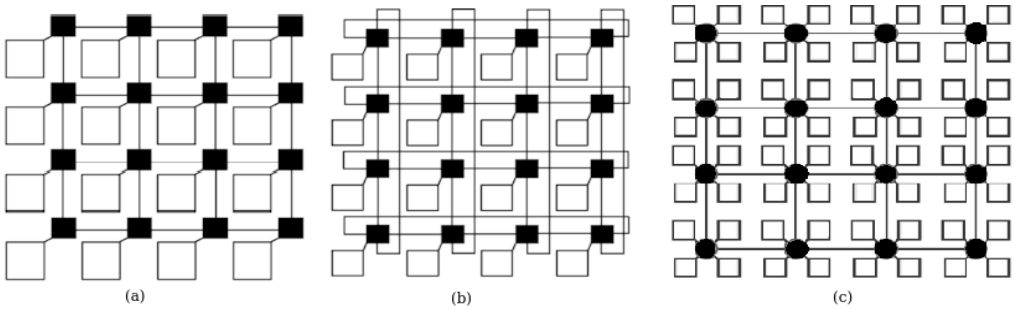


Figure 2.2: (a) Mesh Topology, (b) Torus Topology, (c) Cmesh Topology

- **Torus Topology:** Torus topology in terms of implementation is more complicated than mesh topology since it uses wrap-around links to connect the routers at the edges with the other routers at the opposite edges as shown in Figure 2.2(b). A major advantage of torus topology over mesh topology is the decreased diameter of the network which will reduce by half the maximum number of hops and it has a larger bisection width.

- **Concentrated Mesh Topology:** Balfour and Dally introduced the Concentrated mesh (Cmesh) topology (Balfour and Dally [2006]) to preserve the advantages of a mesh with a decreased diameter and it is obtained by increasing

the degree of the router and the concentration of a larger number of nodes at a single router which can be observed in Figure 2.2(c). Because of its lower hop count, it scales better than the 2D Mesh topology and consequently improved latency. In this work, we use a degree of four. Despite the larger number of ports, having a quarter of the number of 2D mesh routers leads to significant power consumption savings (Camacho et al. [2011]).

- **Fat Tree Topology:** This is indirect network topology and, it depicts a tree, as the name suggests, as shown in Figure 2.3. There is a root node in Fat-Tree network topology which expands into the branch nodes, also known as child leaf nodes. The IPs are connected only to the child leaf nodes. The number of links to the leaf nodes downwards is the same as the number of links upwards to the root node (Ansari et al. [2015]).

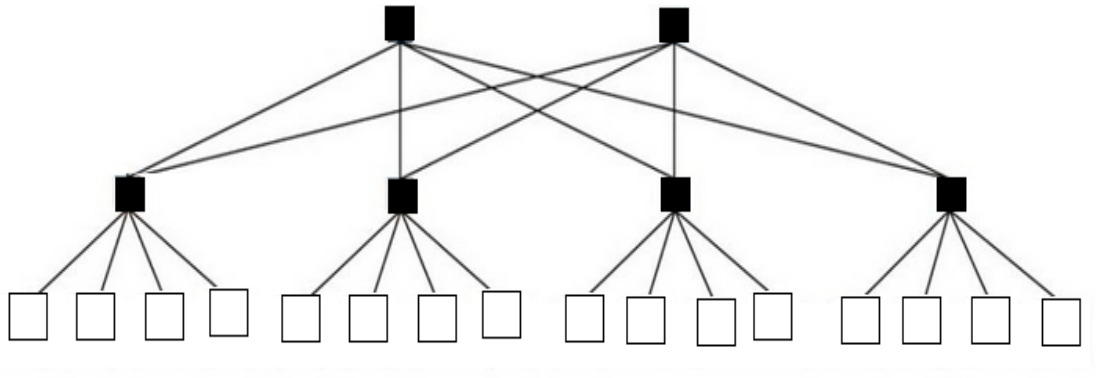


Figure 2.3: Fat Tree Topology.

- **Flattened Butterfly Topology:** Flattened Butterfly topology (Flatfly) (Kim et al. [2007]) is a cost-effective topology for use with high-radix routers. The Flatfly is obtained by combining (or flattening) the routers in each row of a conventional butterfly topology while retaining the links between the routers. It reduces the topology's wiring complexity significantly, allowing it to scale more efficiently.

2.1.2 Router

The router is the most critical part of any network and is the backbone of NoC communication. Routers in the centre nodes in NoC have five ports: East, West,

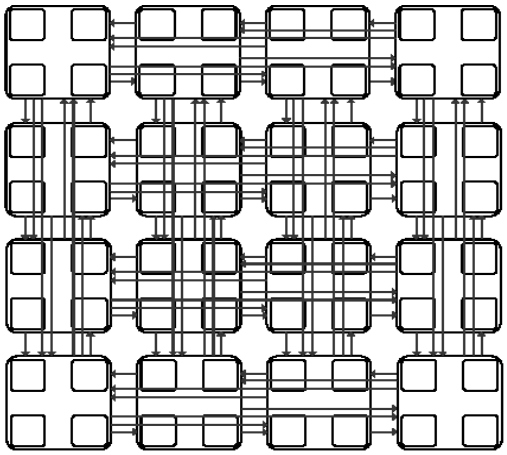


Figure 2.4: Flattened Butterfly Topology (Grot et al. [2009]).

South, North and a Local. The first four ports are linked to the neighbouring routers (East, West, North, and South), and the Local port is linked to the IP core. The router's main function in NoC is to receive packets and to decide the direction each packet should take to the destination (Jerger and Peh [2009]). This is determined by the use of routing protocols already implemented within the router. Figure 2.5 depicts the micro-architecture of the router (Pande et al. [2005]). The router's key components are virtual channel buffers, route compute logic, virtual channel allocator, switch allocator, and crossbar switch. Buffers hold the flit as it enters into the routers. Router logic would compute the next router port that the flit would have to traverse. The allocators decide the flits need to be selected and sent through the crossbar switch. The crossbar switch is responsible for physically transferring the flits from buffers to the output ports.

2.1.3 Flow Control

Flow control defines resource allocation for network buffers and links. This scheme regulates the way routers communicate with each other; In particular, it defines when packets or fixed-size parts of packets called flits can be forwarded from one router to the next in many practical implementations. Flow control thus controls resource utilization and thus has a major impact on performance. Furthermore, the buffer space requirements imposed by a given flow control scheme directly impact the cost of each router of implementation and power consumption.

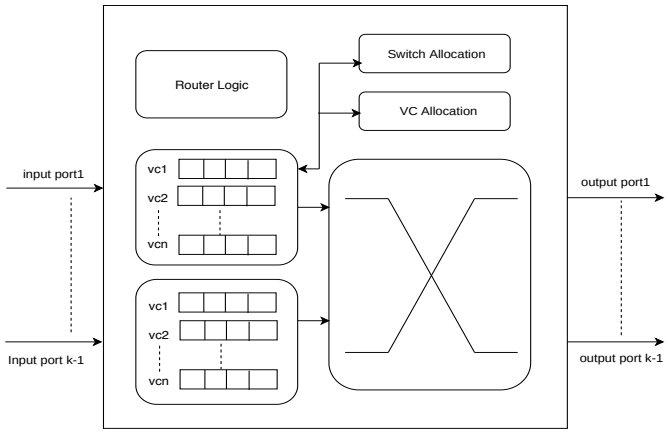


Figure 2.5: Router Micro-architecture (Pande et al. [2005])

2.1.4 Routing algorithm

A routing algorithm's purpose is to ensure low latency and high throughput in an NoC. It can be achieved by distributing the traffic, avoiding hot spots and reducing packet contention. Significant challenges in implementing routing algorithms are deadlock, livelock, starvation and distribution of traffic to achieve an improved quality of service (QoS). Deadlock occurs when two or more packets in a network waiting to be routed forward. Livelock occurs when a packet starts to rotate around its destination without ever reaching it. Starvation occurs when various priorities of lower priority packets are used, so there is often a higher priority packet that would never reach the destination. The issues listed in routing algorithms can be solved in the algorithm with certain techniques.

Different types of routing algorithms which are used in NoC connections that can be generally classified as follows: **Deterministic Routing, Source Routing, Adaptive Routing.**

- **Deterministic Routing**: The path is determined by the source and destination alone, in deterministic routing. The path from source to destination is always the same. The next hop is computed on each intermediate router. To decide on the next hop it requires only the destination address. One common deterministic routing scheme for NoC is XY routing (Bjerregaard and Mahadevan [2006], Agarwal et al. [2009b]). The XY routing occurs when the packets are routed

along the X-axis than the Y-axis to reach their destination (Zhang et al. [2009]).

- **Source Routing**: In source routing, the source core determines entire path to the destination. The path of all intermediate routers can be an ordered list of addresses. Usually the path is modified in the intermediate router to reflect the appropriate routing choice for the next router (Bjerregaard and Mahadevan [2006]).

- **Adaptive Routing**: The routing path is decided in adaptive routing on a per hop basis. The decision on every intermediate router is not entirely based on the destination address. The traffic information is also taken into account. This leads to more complicated router implementations but provides advantages such as dynamic load balancing (Bjerregaard and Mahadevan [2006], Duato et al. [2003]).

2.1.5 Network Interface

Network interfaces are modules that connect the routers to IPs. It also provides router path information based on route computation within the router. It can be split into two parts: front end and back end. The front portion handles the requests from the IP core, and it is not aware of the network's existence. The back end component is directly connected to the network and handles the network protocol, ordering and reordering packets, buffers and helps the router in terms of storage.

2.2 3D Networks-on-Chip

The previous section discussed 2-Dimensional (2D) NoCs. Over the last few years, 2D NoC architectures have been well researched and studied. A 3D NoC, however, is a very new research area with immense possibilities. Since they offer an appealing alternative to conventional 2D NoCs and it offers shorter global interconnects, higher packing density, lower interconnect power consumption and higher performance (Feero and Pande [2008], Xie et al. [2006], Topol et al. [2006]). A 3D Network-on-Chip is created by stacking layers of integrated chips and connecting the layers with vertical link interconnects.

2.3 Various Machine Learning Methodologies used in On-Chip Networks

Machine learning is a fast-growing area in computer science and refers to a collection of pattern recognition techniques that allow algorithms to recognize patterns and make predictions or decisions that are driven by data. Machine learning has been classified into three fields which include supervised learning, unsupervised learning, and reinforcement learning (Biship [2007]). In this section, various works are explained which used machine learning algorithms in NoCs.

In Farahnakian et al. [2014] proposed congestion-aware routing algorithm which was implemented using Q-learning approach for avoiding congested areas in the network.The routing algorithm which was presented divides the network into many clusters that each hold a CQ-table. This table stores information on local and global congestion about alternate routes for forwarding a packet to the destination cluster. It mainly concentrates on estimating the traffic status of the network. Based on information from CQ-table, each cluster will select the less congested output channel. The results showed a significant improvement in performance over traditional methods.

In Jeong et al. [2010] proposed a framework for modelling on-chip router power, area and performance using nonparametric regression algorithm multivariate adaptive regression splines (MARS). The framework offered a substantial estimate of error reductions between 85% and 89%, relative to ORION 2.0 (Kahng et al. [2010]) and parametric models on average.

In Ebrahimi et al. [2012] proposed an adaptable routing algorithm based on minimal and non-minimal paths for on-chip networks to predict the latency of the output channel using Q-learning. In the learning model, the switches maintain distributed tables to store the global congestion information form various regions of the network and thereby to choose a less congested channel to process the output.

In DiTomaso et al. [2015] proposed QORE a liable NoC architectural solution using converse channel buffers. Decision trees algorithms were used to forecast the traffic direction of connections to improve the weak connections. QORE showed an accelerated execution of 1.3× and better throughput by 2.3× on synthetic traffic.

In Jin et al. [2011] created a hybrid network named duo using different traffic

classes to identify similar behaviour in the network by applying a clustering method which provides a global view of communication behaviour. By using the results a hybrid network is designed which provided a reconfigurable spanning channels network allowing higher latency and energy consumption through global coordination for a large portion of long-distance communication.

In Kumar et al. [2018] a machine learning model was proposed and validated for various NoC topologies and synthesized on Virtex 5 FPGA. The model predicted all the synthesis results and design parameters with an accuracy of 98%. The primary benefit of using such models is that when there is a change in design requirements, HDL design is not required every time.

In Farahnakian et al. [2012] proposed a congestion- aware routing algorithm based on dual reinforcement Q- routing. It collects local and global congestion statistics and updates according to the traffic condition. A maximum and minimum threshold values are set to check the empty buffer slots at a particular intermission. Results showed the proposed method was more effective than other already existing routing methods.

In Carloni et al. [2009] proposed an estimation model for delay, power and area of global interconnects which used in the early phases of system-level exploration. The model was later integrated into COSI-OCC communication synthesis model (Pinto et al. [2007]), it was found that it improved the quality the NoC synthesis results.

In Feng et al. [2010] proposed a fault-tolerant deflection routing algorithm (FTDR) focused on local & global congestion information which was built using Q-learning method. It was used to reconfigure the routing table to avoid faults and to reduce the routing table size. Another optimized routing algorithm was proposed using hierarchical Q-learning (FTDR-H). Both the algorithms were compared and FTDR-H switch saved the area around 27%.

In DiTomaso et al. [2017] proposed LESSON (Learning Enabled Sleepy Storage Links in NoCs) to reduce both static and dynamic power consumption. Decision tree algorithm was used to predict traffic flow and link utilization. By the predictions made it can provide more bandwidth when required to improve the performance and it can accurately power-gate links and buffers to adjust the power dissipation. Results showed an improved total network power between 31.7% - 85.6% and improved packet

latency by up to 14%.

In Qian et al. [2013, 2015] proposed a framework named SVR-NoC which used the average channel waiting time and the traffic flow latency to predict the average packet latency of both synthetic traffic & real-time traffic with an average error less than 12% and speedup of more than 100×.

In Das et al. [2016] proposed a robust design development methodology to boost the energy efficiency of 3D NoC architectures. Online machine learning algorithm was utilized, combining the benefits of small-world networks and machine learning techniques (Boyan and Moore [2000]). The proposed model achieved a reduction of 35% Energy Delay Product (EDP) over conventional 3D Mesh.

In Park et al. [2017] presented the combination of machine learning and bayesian optimization for optimizing the electrical and thermal performance of 3D ICs and systems. This method has also demonstrated the ability to manage a large number of input parameters with fast convergence and flexibility. The results showed an average improvement of 4.4%, 31.1% for temperature gradient, and CPU time.

In Yin et al. [2018] demonstrated the efficacy of applying deep Q-learning to the design of NoC arbitration policies. Through online learning, the model is learning to make decisions that maximize long term NoC performance. Results from the experiments show that the DQL arbitration model is effective in reducing packet latency. While a complete DQL algorithm is not practical to implement directly in a real NoC.

In Rafie et al. [2014] presented a new mapping generator which uses metrics like obustness index, contention factor, and communication cost for unique mapping. The experimental results showed that, in comparison with the ordinary algorithm, the proposed algorithm has more than fifteen times performance and more supervision in finding the best mappings from numerous generated solutions.

In Shen et al. [2013] proposed Power-aware and Reliable Encoding Schemes Supported reconfigurable Network-on-Chip (PRESSNoC) for problematic issues, like crosstalk interferences and wire power consumption, in NoCs. It includes a novel reconfigurable NoC design, four data encoding strategies and an intelligent REasoning And Learning (REAL) encoding strategy selection framework. Experiments showed that PRESS-NoC induces a higher probability towards reduction of crosstalk interferences and

dynamic power consumption at the same overheads of performance and hardware resources.

In Tosun [2011] proposed a cluster-based method for application mapping onto NoC architectures. The proposed method was a superior version of Integer Linear Programming based methods because in very short time it can decide optimum or very close to optimum results.

In Dageleh and Jamali [2018] V-CastNet 3-D mapping method was proposed for graphs with large tasks for reducing power consumption and delay in NoCs. Initially clustered nodes of task graphs based on communication weights were created. Later CastNet 2-D method was used to map each cluster on a 2-D layer. As a result it chose the best network arrangement. V-CastNet as compared with the Tabu search-based mapping method it was able to reduce mapping execution time significantly.

In Aravindhan et al. [2016] presented a new cluster mapping method which was used to reduce communication cost and cut degree. The performace and efficiency of the proposed method was checked with the experiments carried out in NoC for various benchmarks. Experimental results showed a 3.8% reduction in communication costs for MPEG4, and a 3.0% reduction in communication costs for the PIP benchmark.

In Farahnakian et al. [2011] presented a congestion-aware routing algorithm named QCA: Q-learning based Congestion-aware Algorithm for NoCs. It mainly concentrated on estimating the traffic status of the network. Routers maintained congestion information which was updated by learning packets. This routing table information was used to pick a less congested path, and the packets forwarded in that direction.

In Juan and Marculescu [2012] author used semi-supervised reinforcement learning based approach was prposed for performing dynamic voltage and frequency scaling (DVFS). It was evaluted for controlling cores and uncores in synergy for NoC-based CMPs. Experimental results showed an 11% reduction on the program execution time.

Most of the existing works have considered only Mesh topology or any single topology for their works and considered individual parts of NoCs like local, global congestion, routing tables, the power consumption of different router components, latencies and experiments have been done by considering the architecture sizes less than 10×10 which is 100 nodes.

2.4 On-Chip simulations using FPGAs

Other than machine learning approaches there are other attempts have been made to address the simulation speed problem using Field-Programmable Gate Arrays (FPGAs) (Genko et al. [2005], Wolkotte et al. [2007], Krasteva et al. [2008], Lotlikar et al. [2011], Papamichael [2011], Papamichael et al. [2011], Wang et al. [2012], Drewes et al. [2017], Kamali and Hessabi [2016], Kamali et al. [2017]). But these NoC emulators suffer from the problem of scalability. Because of the FPGA logic and memory constraints they can not scale up to large NoCs. A recent study in Wang et al. [2012] has shown that even an extremely large FPGA doesn't have sufficient logic blocks to fit around 150 nodes in a moderately complex NoC design.

2.5 Machine Learning based NoC simulations in this Thesis

The main goal of this thesis is to create frameworks which can be used to predict the design space exploration of large-scale architectures, hence it becomes an very essential tool for the chip designers. It helps them in understanding the impact of various design parameters in the early stage thus reducing the actual development cost and simulation time. By using the proposed framework, chip designer can explore and analyze various NoC architectures like direct topologies (2D Mesh, Torus, Cmesh, 3D Mesh and Torus), indirect topologies (Fat-Tree, Flatfly) with various configuration parameters like virtual channels, buffer depth, injection rates and traffic pattern.

This thesis considers synthetic traffic patterns for conducting the experiments as they are the ones focused on mathematical modelling in real applications of common traffic patterns viz,. uniform, tornado, bitrev, bitcom, shuffle, transpose (Dally and Towles [2004]). These traffic patterns have a high degree of flexibility and easy to create. A collection of carefully designed synthetic workloads may provide reasonably detailed coverage of emulated NoCs characteristics. Other than synthetic workloads, there are trace-driven workloads which cannot be used in this thesis because of the lack of trace data from large-scale NoC-based systems, the use of synthetic workloads is practically the only feasible approach for emulating large-scale NoCs with thousands of nodes.

2.6 Summary

This chapter provides the basic concepts of NoCs and also reviews the efforts that have been made over the years on NoCs using machine learning. It also gives insight into various works done using FPGAs on NoCs. Later it provides the various architectures and configurations considered in this thesis and specifies why synthetic traffic patterns were considered for experimentation purpose.

Chapter 3

Machine Learning based Framework for NoCs

In this chapter, a machine learning-based framework named Learning-Based Framework (LBF-NoC) is presented to predict performance, power and area parameters of direct (Mesh, Torus, Cmesh) and indirect (Fat-Tree, Flatfly) NoC architectures. It is designed using supervised machine learning regression algorithms. Various machine learning algorithms were explored to predict the design parameters of NoC architectures and selecting the algorithm which provided efficient results. Later the framework is extended using multiprocessing scheme to overcome the issue of simulating NoC architectures 'n' number of times.

3.1 Learning-Based Framework (LBF-NoC)

LBF-NoC is a machine learning (ML) framework proposed by considering six synthetic traffic patterns viz, uniform, tornado, transpose, shuffle, bitrev and bitcom. It is designed to predict different performance, power and area parameters of five different NoC architectures. Figure 3.1 shows the overview of LBF-NoC where it has been divided into three phases pre-processing phase, training phase and testing phase.

3.1.1 Pre-processing phase

The Booksim2.0 (Jiang [(accessed 2012]) cycle-accurate simulator is used to generate reference data over various NoC configurations. Booksim provides the performance, power and area for each architecture. Figure 3.1 shows how the various configurations are provided to Booksim simulator. Results are accumulated from Booksim simula-

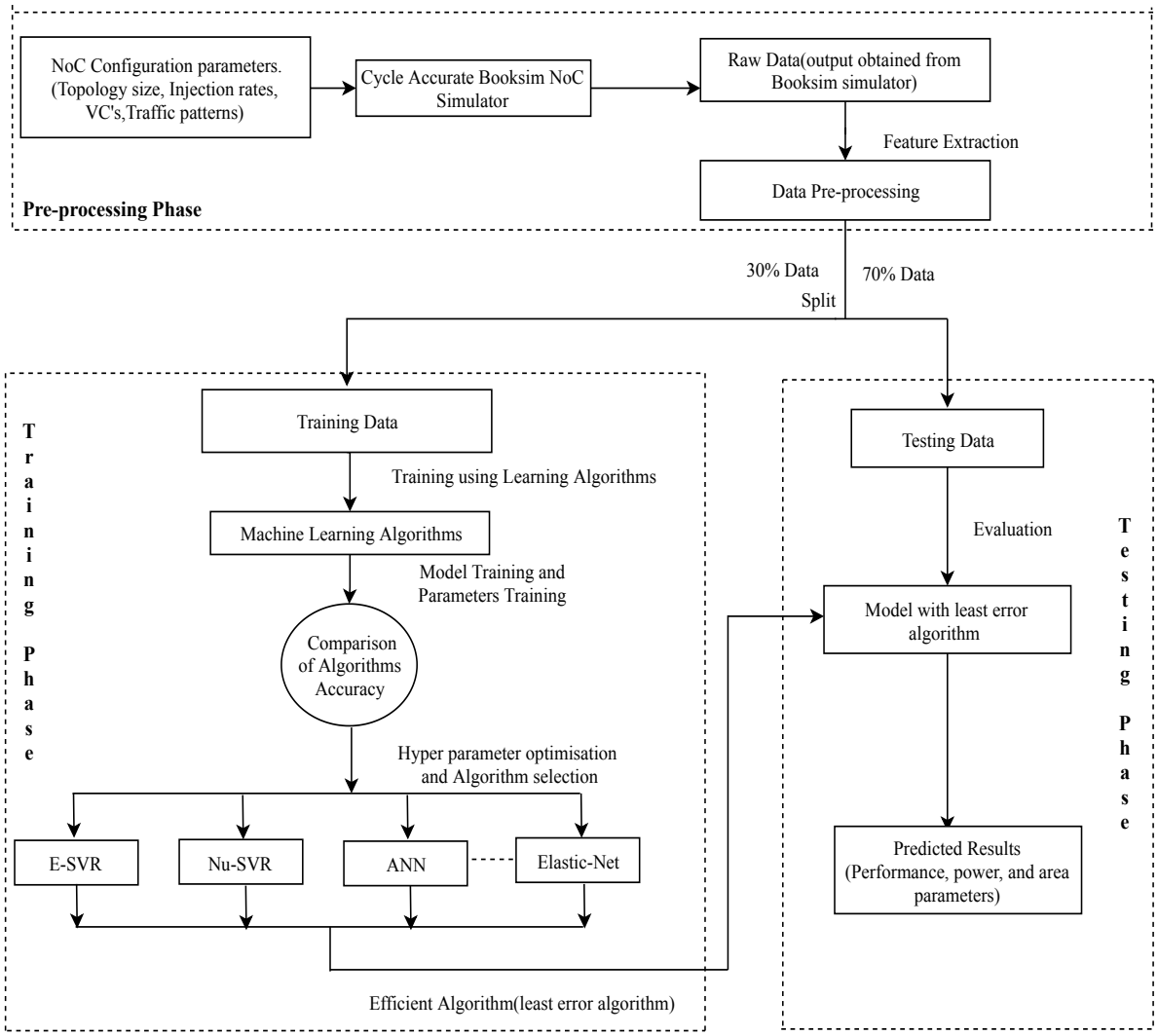


Figure 3.1: Overview of the Learning Based Framework

tor for different architectures considering various configurations as shown in Table 3.1. Among all the parameters topology sizes, injection rates, virtual channels (VCs), buffer depth, & traffic patterns have been considered as input parameters for experiments. Performance parameters: average network latency, average packet latency, average flit latency, average hop count, the power consumption of router, the total power of NoC, router area, and total area are considered as output parameters as shown in Table 3.2.

A Data for Direct topologies

Raw results from Booksim simulator are partitioned as test and train data for LBF-NoC. For uniform and tornado traffic patterns 32% of data was considered as training

Table 3.1: Configuration parameters for creating datasets

Input Parameters	Description	Values
Network		
topology	Layout and connectivity of the nodes in the network is NoC topology.	Mesh, Torus & Cmesh (Direct) Fat-Tree & Flatfly (Indirect)
k	Topology radix	2 to 45 for Mesh, Torus, & Flatfly 2 to 30 for Cmesh, Fat-Tree
n	Network dimension	2
c	Concentration (No. of PEs per router)	4
Router		
num_vcs	Total number of Virtual Channel per port	2, 3, 4, 5
vc_buf_size	Buffer size per Virtual Channel	8, 10
routing_function	The name of the routing function	Deterministic XY routing
Simulation parameters		
injection rates	It is the rate at which packets are inserted into the network by a node.	0.001, 0.0015, 0.002,.....0.1
traffic Patterns	Type of traffic in network	uniform, tornado, transpose, bitrev, bitcom & shuffle
sample_period	Total Number of measurements cycles	100000cycles

data over topology sizes (2×2 to 17×17). Remaining 68% of the data was used as test data and to validate LBF-NoC over topology sizes (18×18 to 45×45) for Mesh,& Torus architectures. For Cmesh topology (2×2 to 10×10) was considered as training data and (11×11 to 30×30) was considered as testing data.

For shuffle, transpose, bitrev, and bitcom dataset consisting of data associated with injection rates ranging from 0.001, 0.0015, 0.002 till 0.009 have been considered (any 12 injection rates can be used for training and the remaining 5 can be used for testing).

B Data for Indirect topologies

For Fat-Tree the data from *k=1 to 15* is considered as training and data from *k=16 to 30* is considered for testing for uniform, tornado and neighbor traffic patterns. Flatfly topology the architecture size from *k=2 to 24* is considered as training data and *k=26 to 50* is considered as testing data for uniform, tornado and neighbor traffic patterns.

Table 3.2: Different output features considered for LBF-NoC.

Output Features		
Performance Parameters	**Power Parameters**	**Area Parameters**
Average Network Latency(ANL)	Crossbar power(CBP)	Router Area(RA)
Average Packet Latency(APL)	Switch arbiter power(TP)	Total Area(TA)
Average Flit Latency(AFL)	Virtual Channel arbiter power(VCAP)	
Average Hop Count(AHC)	Buffer power(BP)	
	Router power(RP)	
	Total power(TP)	

For shuffle, transpose and bitrev dataset consisting of data associated with different injection rates among all the injection rates 50% was considered as training and remaining 50% data is considered as testing for both Fat-Tree and Flatfly architectures.

3.1.2 Training phase

In this phase, various ML regression algorithms are trained using the training dataset which is generated in the previous phase. As specified earlier five different input features were considered viz, architecture size, injection rates, virtual channels, buffer_size and traffic patterns. Table 3.2 lists the different output features considered. Different regression algorithms like Artificial Neural Network (ANN) with identity and relu activation functions, different generalized linear regression algorithms, i.e., lasso, lasso-lars, larsCV, bayesian-ridge, linear, ridge, elastic-net and Support Vector Regression (SVR) with *linear*, *Radial Basis Function (RBF)*, *polynomial* kernels are used to train the model for each parameter separately for direct and indirect topologies with various traffic patterns. All the trained algorithms are tuned until it provides the least error.

Mean Absolute Percentage Error (MAPE) is used to calculate the error rate in percentage and its complement has been considered as the accuracy. Figure 3.2 shows the range of accuracies produced by the regression algorithms. Where each algorithm was trained for all the architectures considered with various synthetic traffic patterns. Few algorithms like ANN and linear regression were able to predict the performance parameters but failed to predict all the output features considered. The accuracy

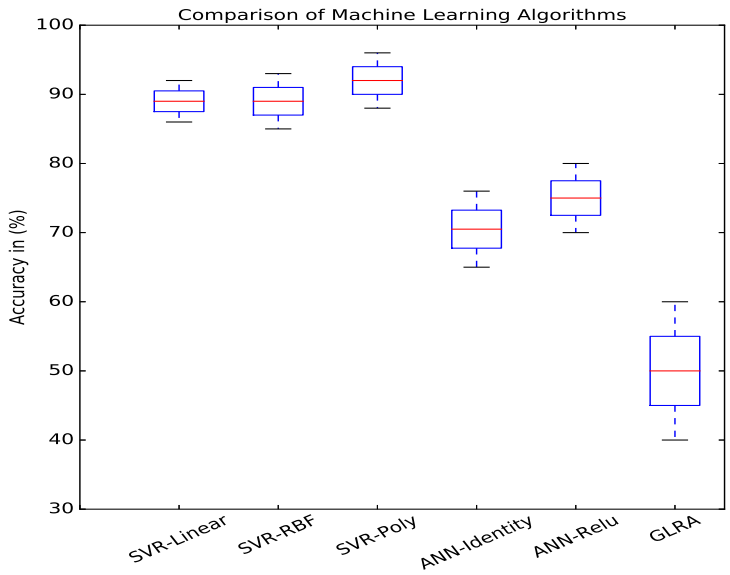


Figure 3.2: Different algorithms tested for creating Learning Based Framework. [GLRA: generalized linear regression algorithms]

of each algorithm was tuned until it provided better results. The algorithm which provided accuracy more than 85% for all the output features was considered. Among all the algorithms SVR provided accuracy more than 90% for all the parameters considering different synthetic traffic patterns. Remaining algorithms provided accuracy in the range 40% to 75%.

Hence LBF-NoC is built using SVR algorithm (Cristianini and Shawe-Taylor [2000], Smola and Schölkopf [2004]) with both variants of SVR i.e, ϵ-SVR and υ-SVR using *linear*, *RBF* and *polynomial* kernels have been used and among those the least error provided kernel was considered.

Support Vector Machine (SVM) is one of the supervised learning methods which can be used to perform classification and regression. The regression form of SVM is named as Support Vector Regression. Vapnik developed the theory of SVR in 1997. It is known as one of the essential techniques in terms of solving a regression problem (Cristianini and Shawe-Taylor [2000], Smola and Schölkopf [2004]).

The structure of SVR is similar to the SVM. In contrast, the SVR tries to fit a line or curve to the data by minimizing the cost function. The strategy of SVR is to construct a hyperplane in high-dimensional space with consideration of constraints to create a boundary for data points with upper and lower bounds as shown in Figure

3.3 (Smola and Schölkopf [2004]). The distance between the hyperplane and upper bound or lower bound is measured by ϵ. One advantage of using this function is that it can tolerate noise.

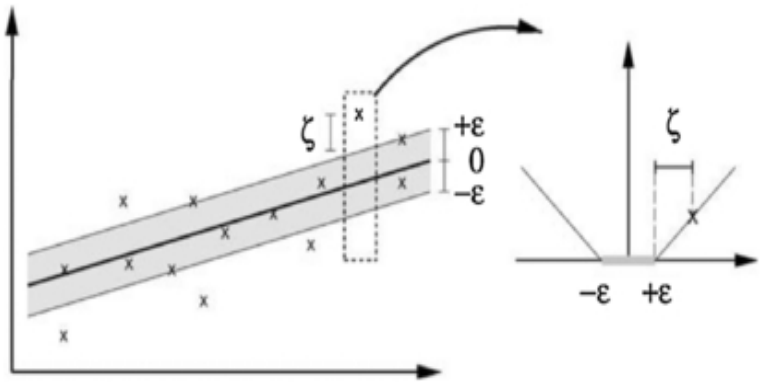


Figure 3.3: ϵ insensitive loss function for a linear SVM.

SVR approximates a linear function *g(x)* in the following form:

$$g(x) = a^T x + b \tag{3.1}$$

where a and b are the weight vector and bias term, respectively. (3.1) can be constrained as:

$$minimize \frac{1}{2}\|a\|^2 + C \times \sum_{i=1}^{l}(\xi_+ + \xi_-) \tag{3.2}$$

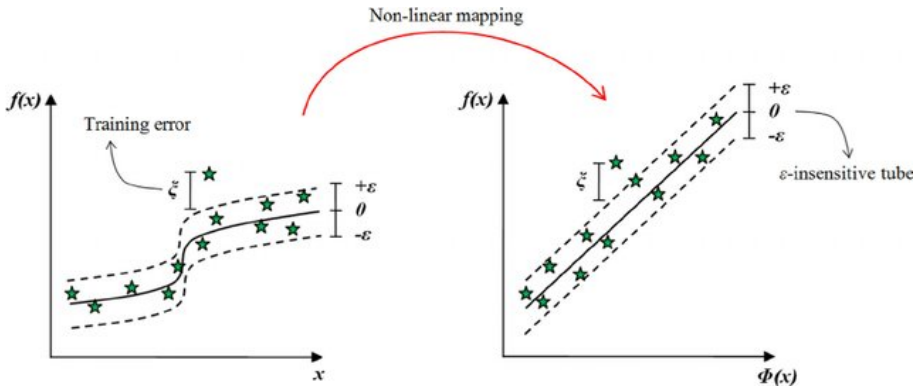


Figure 3.4: Mapping non-linear model into the feature space.

where ξ_+ and ξ_- are the two nonzero slack variables in either directions. ϵ fits the data between the boundary and constant $C \geq 0$ optimizes (3.1). The data samples

that are outside of the restricted zone within the distance of slack variables are the support vectors. The minimization function (3.1) is subject to:

$$\begin{aligned} y - (a^T x + b) &\leq \epsilon + \xi_+ \\ (a^T x + b) - y &\leq \epsilon + \xi_- \\ \xi_+, \xi_- &\geq 0 \end{aligned} \tag{3.3}$$

Standard dualization can solve the constrained optimization problem (3.1) and (3.2). Dual formulation reformulates the optimization function using the Lagrange multipliers with the help of a double set of parameters. After a set of steps explained in (Smola and Schölkopf [2004]), dual optimization problem yields the following solution:

$$f(x) = \sum_{i=1}^{n} (\alpha - \alpha^*) K(d_i, d) + b \tag{3.4}$$

where α and α^* are Lagrange multipliers; and the kernel function is represented by $K(d_i, d)$.

The SVR method is also able to solve non-linear problems. Selecting an appropriate non-linear function with the Lagrange dual formulation provides an excellent solution in non-linear models. Figure 3.4 shows the mapping of the non-linear regression model in a high-dimensional feature space (Mahdevari et al. [2014]). Value of ϵ controls the data points in ϵ-insensitive margin. Fewer support vectors are present when ϵ value is large. Complexity of the model can be adjusted using C. When C increases the complexity of model reduces and vice-verse (Burges [1998], Kuhn and Johnson [2013]).

Hence, a proper set of parameters can lead to a suitable SVR solution that can be the best model for the dataset used. Once the parameters are appropriately selected, it can be expected for a better generalization performance from the constructed SVR model.

In non-linear case a is not specified explicitly. The functions $k(d_i\ ,\ d\ )$ needs to satisfy the Mercer's condition (Smola and Schölkopf [2004]).

The kernel functions of nonlinear regression models are (Chapelle and Vapnik [2000]).

$$Linear:\ K(d_i, d) = d_i^T x$$

$$Polynomial: \ K(d_i, d) = (\phi(d_i * d) + 1)^x$$
$$Radial\ Basis\ Function: K(d_i, d) = exp(-\gamma \|(d_i - d)\|^2\ )$$

Another method of using SVR is by applying υ-SVR. ϵ-SVR and υ-SVR handle margin control and penalty parameters differently.

(Schölkopf et al. [1998]) put-forward an SVR algorithm, called $\upsilon - SVR$, adjusting the parameter epsilon automatically. $\upsilon - SV$ in $\upsilon - SVR$ controls the number of support vectors in the solution based on the samples in the dataset. In ϵ-SVR, ϵ stands for insensitive loss function, which needs to be set before the training phase. Guessing the value of ϵ in advance is the major drawback of this scheme; hence, $\upsilon - SVR$ is used to handle the situation (Schölkopf et al. [2000]).

As specified above, SVR has two variants ϵ-SVR and υ-SVR where each can use three kernels. Hence, a generalized SVR model was created which provided the information regarding the kernel and the hyper-parameter values for each output feature and the model gave the values and kernels where the error rate was less than 5% to 6%. The model was tested with various combinations of (ϵ, C, γ) and (υ, C, γ) for every output features. The hyper-parameters which provided the least error was selected. The overview of the generalized SVR model is shown in Figure 3.5. This complete process was done while collecting the data. Hence, no extra time was consumed for getting the hyper-parameter values for every output features.

Mean Square Error (MSE) was used to evaluate the performance of LBF-NoC against Booksim results. MSE is given by:

$$MSE = \frac{1}{l} \sum_{i=1}^{l} (a_i - a_i^*)^2$$

Where a_i is the actual value, a_i^* is the predicted value and 'l' is the number of data points.

3.1.3 Testing phase

The decision will be made in this phase based on the knowledge gathered during the training phase. The output of the training phase is SVR model with the least error. This model is used to predict the NoC output features and validated against the Booksim2.0 simulator.

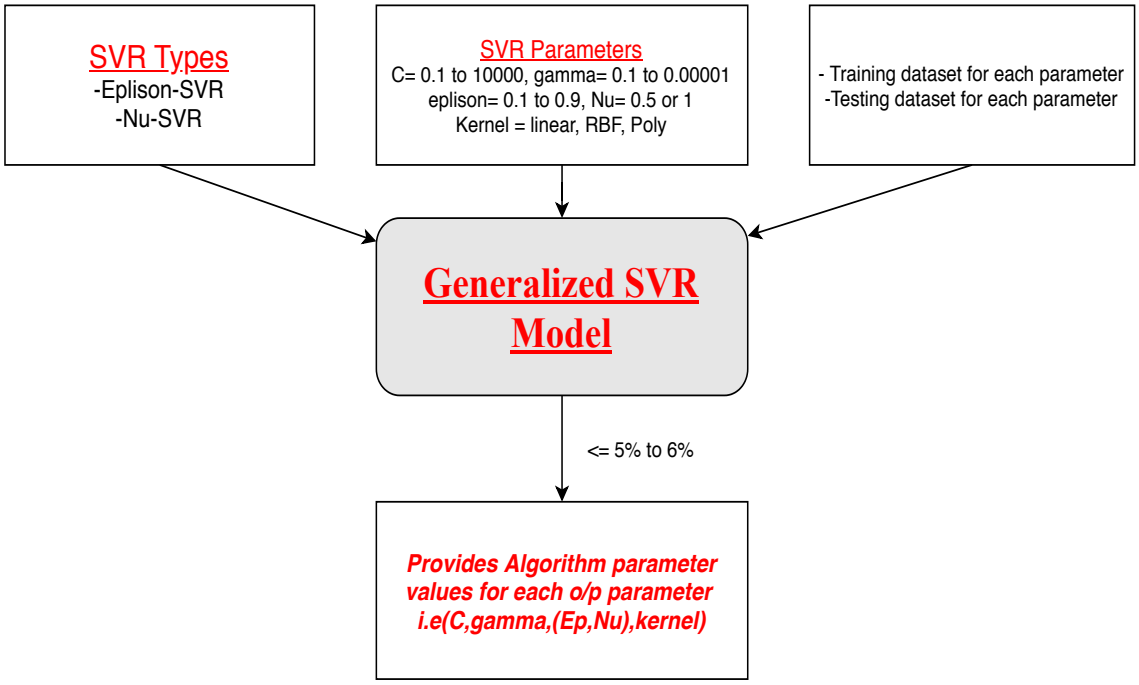


Figure 3.5: Generalized SVR Model

3.1.4 Experimental results

A Validation of Analytical Model for Performance Parameters

The Initial set of experiments on the Booksim simulator were conducted to analyze the dependency of output performance parameters v/s architecture size. Simulation results showed that there was a linear relationship between output performance parameters and the architecture size, which is also mentioned in Jiang et al. [2013]. Hence, we used curve fitting algorithms namely Least Squares Solution methods to get linear equations Arlinghaus [1994].

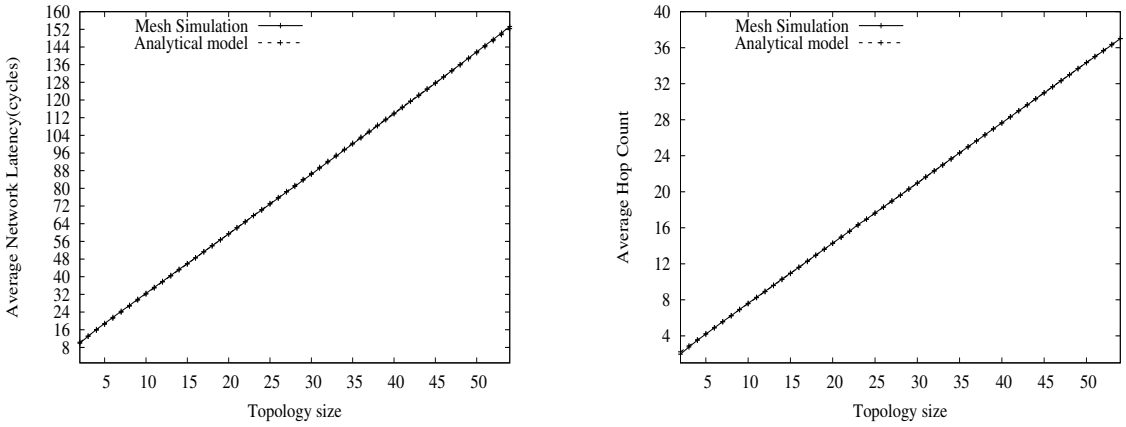
Analytical models were derived for each performance parameter viz, Average Network Latency (ANL), Average Packet Latency (APL), Average Flit Latency (AFL), Average Hop Count (AHC). Table 3.3 shows the configuration considered for the experiment and the linear curve expressions derived. The relationship between performance parameters, like ANL, APL, AFL, and AHC against topology size was studied. Experiment results show that these parameters depend linearly on the topology size (Figure 3.6).

LBF-NoC extends the simple analytical model to accept multiple input parameters (topology size, virtual size, injection rates, traffic pattern) and output multiple

Table 3.3: Configuration and results of Analytical Models

Configuration Parameters	
Topology	Mesh
Topology size	2X2 to 50X50
VC	4
Injection rate	0.002
Traffic pattern	Uniform Random
Analytical Models	
Performance parameters	Learning Curve
ANL	4.87+2.74*(topology size)
APL	4.50+2.67*(topology size)
AFC	5.35+2.71*(topology size)
AHC	0.88+0.67*(topology size)

simulation results which consist of performance, power, and area parameters.



(a) (b)

Figure 3.6: Correlation study of Performance parameters between Booksim simulator vs Analytical model for Mesh topology, where (a) Average Network Latency, (b) Average Hop count with injection rate 0.02, VC=4 respectively.

B LBF-NoC for Direct Topologies

LBF-NoC sub-models were created for each output parameter and for each traffic patterns. SVR ML algorithm was used for experiments. For performance parameters prediction linear and RBF kernels worked efficiently. For power, area parameters prediction RBF and polynomial kernels worked efficiently. All the output parameter values have been normalized using logarithmic functions.

Performance parameters comparison was done with six traffic patterns as shown in Figures 3.7, 3.8(a), 3.9, and 3.10 (a). Results have been shown for uniform and transpose traffic pattern only. The results for the tornado traffic pattern are similar to the uniform traffic results. The results for bitcom, bitrev, shuffle are similar to the transpose traffic results.

Figure 3.7(a) shows the comparison of ANL for Mesh, Torus and Cmesh topologies against Booksim. In the figure, the x-axis represents the topology size and the y-axis represents normalized network latency. LBF-NoC provided the highest accuracy of 96.7% for Mesh, 99.26% for Torus, and 97.8% for Cmesh topologies. Table 3.4 presents the MSEs of linear and RBF variants of ϵ and υ-SVRs for performance parameters. Table 3.5 presents the algorithm with accuracy which worked better in terms of accuracy for performance parameters. Figure 3.7(b), 3.8(a), 3.9 and 3.10(a) depicts the similar results for different performance parameters.

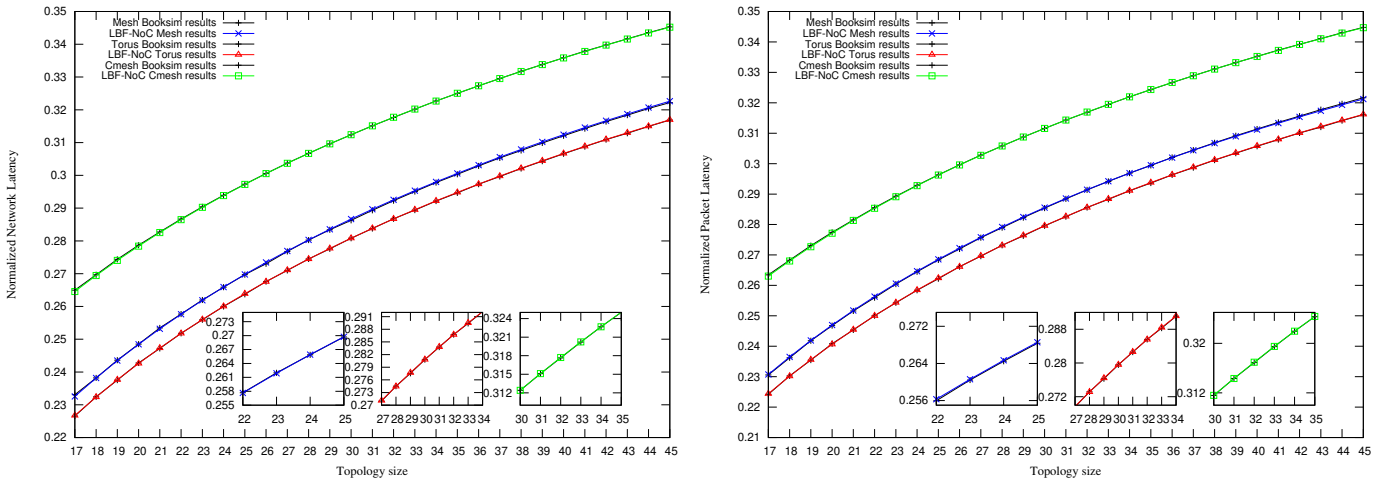


Figure 3.7: Comparison of LBF-NoC with simulation results for Mesh-based topologies with Uniform traffic pattern (a) Average Network Latency, (b) Average Packet Latency

X-axis of 3.8(b) and 3.10(b) represents three topologies namely Mesh, Torus and Cmesh. Y-axis represents average error rate respectively. These figures explain the average error rates of Mesh, Torus and Cmesh topologies against ANL, APL, and AHC.

Figure 3.11(a) 3.12(a) shows the comparison of area parameters obtained from LBF-NoC against Booksim. The results are shown for sizes 20×20, 30×30 and 40×40

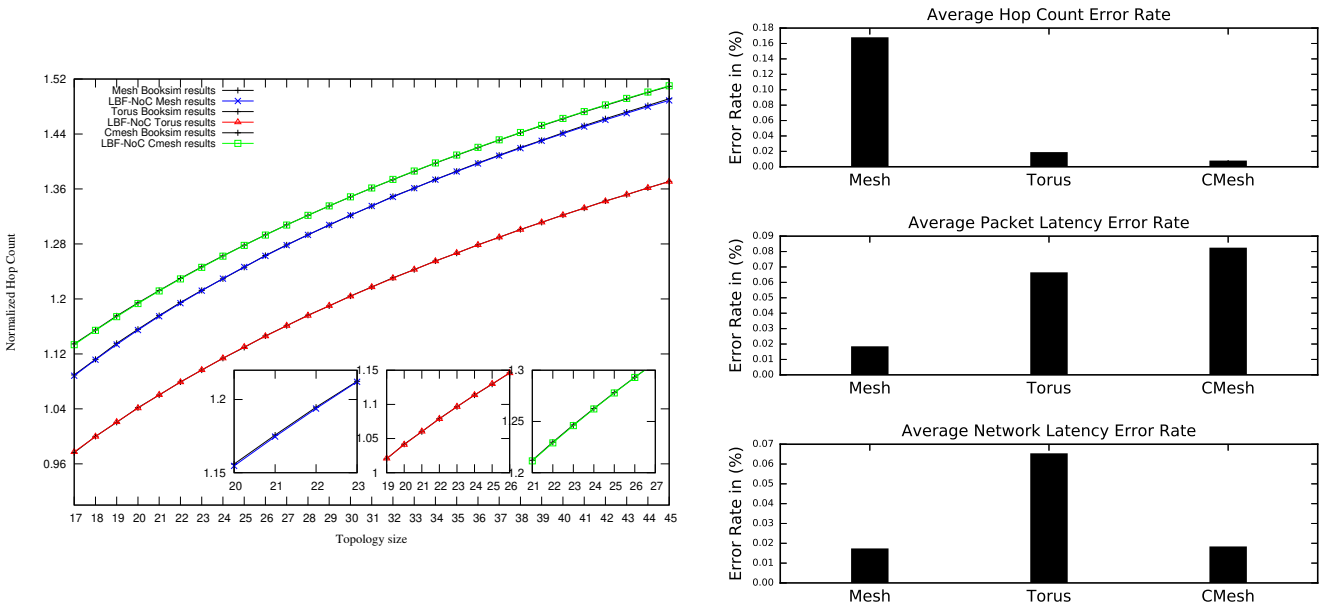


Figure 3.8: Comparison of LBF-NoC with simulation results for Mesh-based topologies with Uniform traffic pattern (a) Average hop count, (b)Average Error rates of topologies considered with injection rate 0.002, VC=4 respectively.

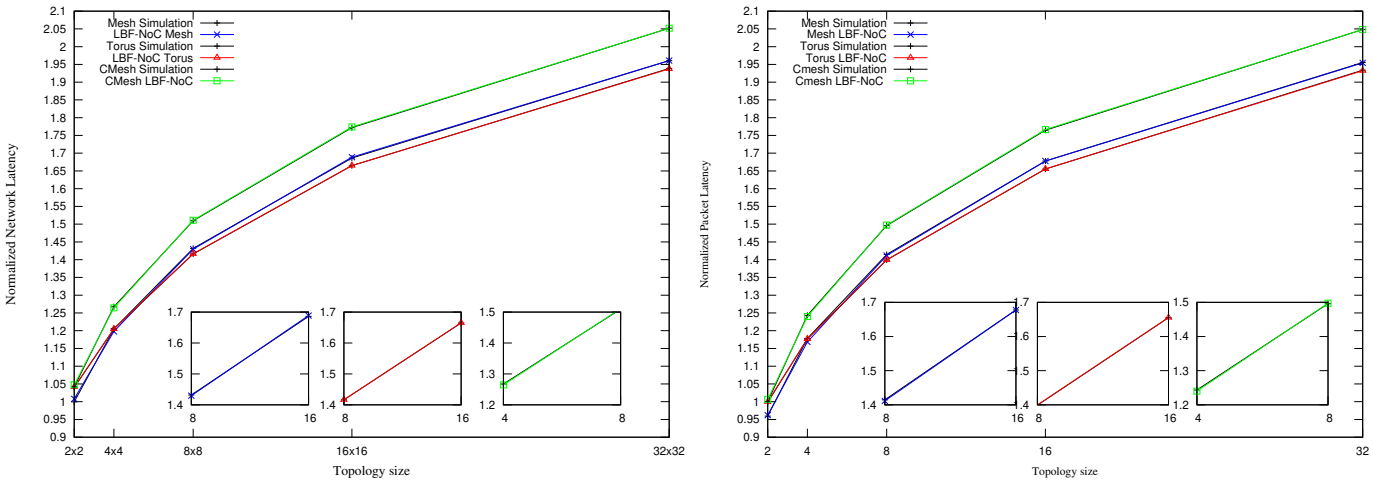


Figure 3.9: Comparison of LBF-NoC with simulation results for Mesh-based topologies with transpose traffic pattern (a) Average Network Latency, (b) Average Packet Latency

the results for remaining architecture sizes are similar. Experiments have been conducted for all the area parameters and results of Router area and total area for each topology are shown and explained.

The accuracy of the router area and total area for Mesh topology is 99.92%, 98.4% respectively. For Torus topology the accuracies are 99.13% and 99.82% respectively.

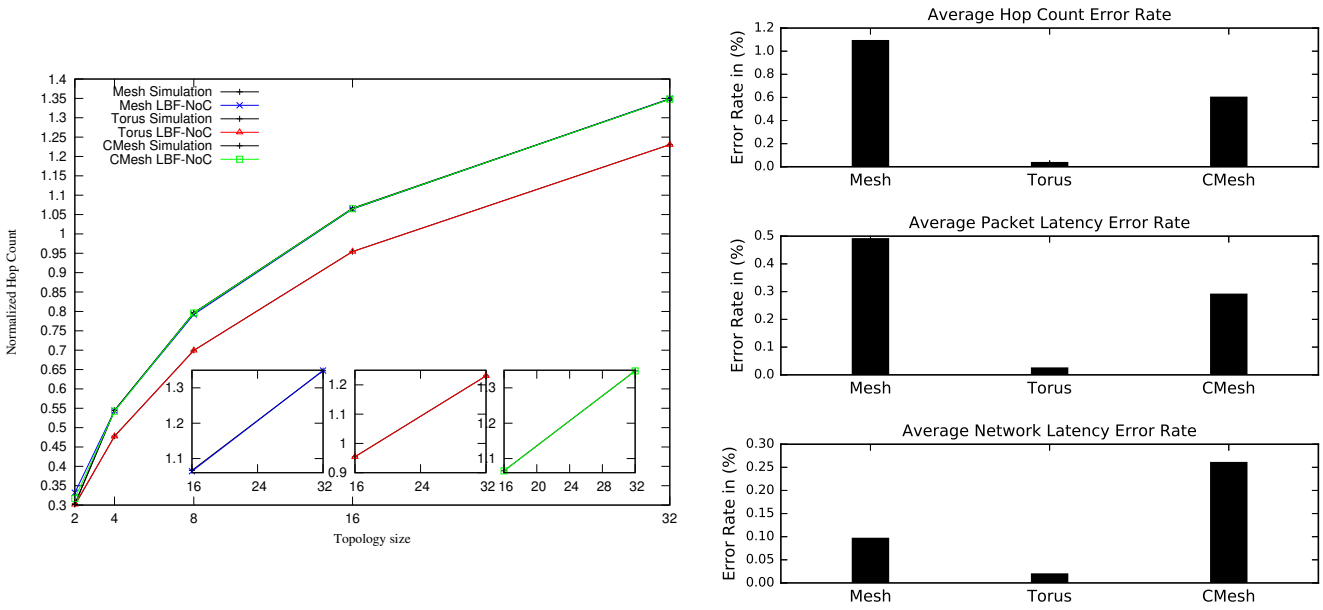
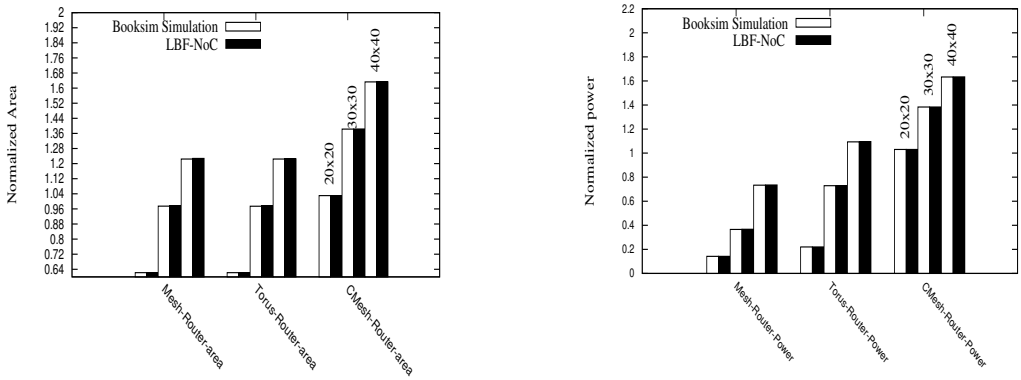


Figure 3.10: Comparison of LBF-NoC with simulation results for Mesh-based topologies with transpose traffic pattern (a) Average hop count, (b)Average error rates of topologies considered with injection rate 0.003, VC=4 respectively.

Similarly, the accuracy for Cmesh topology is 98.41% and 90.2% respectively. Figure 3.13(a) shows the error rates of LBF-NoC for router area and total area.



(a) (b)

Figure 3.11: Comparison of LBF-NoC with Booksim for Router area and Router power for 20×20, 30×30 and 40×40.

Similarly, experiments have been done for all the power parameters and results of router power and total power have been shown and explained. Figure 3.11(b), 3.12(b) shows the comparison of LBF-NoC with Booksim simulator using uniform

Table 3.4: MSE of Performance parameters for Mesh-based topologies with different traffic patterns

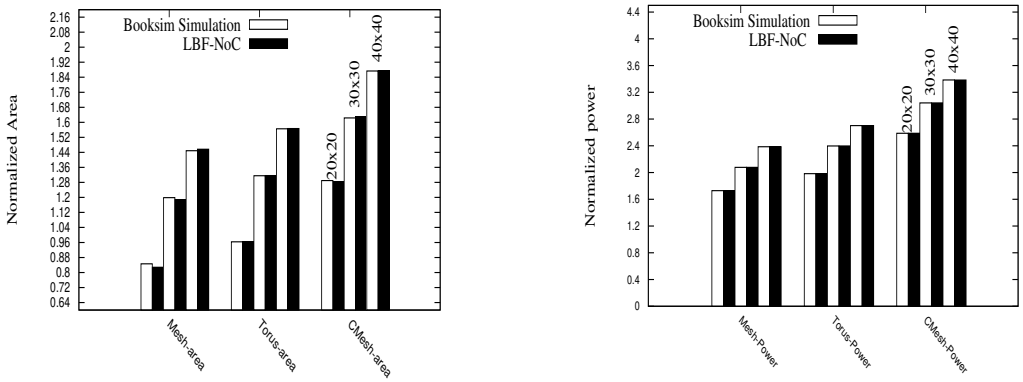
Uniform Traffic Pattern												
Algorithms used	Mesh				Torus				Cmesh			
in LBF-NoC	ANL	APL	AFL	AHC	ANL	APL	AFL	AHC	ANL	APL	AFL	AHC
$\epsilon - SVR$(Linear)	0.037	0.073	0.076	0.06	0.0074	0.007	0.007	0.013	0.022	0.022	0.023	0.013
$\epsilon - SVR$(RBF)	0.078	0.072	0.074	0.003	0.015	0.015	0.012	0.01	0.036	0.0365	0.034	0.012
$\upsilon - SVR$(Linear)	0.033	0.18	0.18	0.04	0.016	0.016	0.014	0.0002	0.025	0.0251	0.026	0.0012
$\upsilon - SVR$(RBF)	0.042	0.15	0.158	0.02	0.016	0.0163	0.016	0.001	0.030	0.036	0.03	0.015
Tornado Traffic Pattern												
$\epsilon - SVR$(Linear)	0.2	0.061	0.2	0.066	0.013	0.012	0.013	0.003	0.100	0.100	0.010	0.06
$\epsilon - SVR$(RBF)	0.05	0.09	0.05	0.021	0.013	0.017	0.018	0.010	0.031	0.030	0.032	0.01
$\upsilon - SVR$(Linear)	0.12	0.10	0.28	0.085	0.013	0.012	0.0125	0.002	0.21	0.22	0.23	0.03
$\upsilon - SVR$(RBF)	0.08	0.04	0.08	0.032	0.0077	0.0064	0.016	0.0070	0.3	0.31	0.35	0.010
Transpose Traffic Pattern												
$\epsilon - SVR$(Linear)	0.098	0.087	0.087	0.006	0.016	0.016	0.016	0.054	0.162	0.162	0.160	0.040
$\epsilon - SVR$(RBF)	0.045	0.048	0.054	0.005	0.0139	0.013	0.013	0.002	0.075	0.075	0.075	0.040
$\upsilon - SVR$(Linear)	0.099	0.086	0.085	0.006	0.030	0.030	0.0305	0.001	0.153	0.153	0.151	0.007
$\upsilon - SVR$(RBF)	0.068	0.043	0.043	0.004	0.030	0.03	0.03	0.001	0.020	0.020	0.021	0.001
Shuffle Traffic Pattern												
$\epsilon - SVR$(Linear)	0.002	0.003	0.008	0.0046	0.0076	0.0076	0.0077	0.0024	0.003	0.003	0.003	0.005
$\epsilon - SVR$(RBF)	0.002	0.003	0.001	0.004	0.004	0.004	0.0041	0.0058	0.002	0.002	0.002	0.003
$\upsilon - SVR$(Linear)	0.003	0.005	0.001	0.001	0.0043	0.0043	0.004	0.001	0.001	0.001	0.001	0.001
$\upsilon - SVR$(RBF)	0.003	0.004	0.020	0.064	0.0042	0.0042	0.0042	0.0005	0.004	0.004	0.004	0.004
Bitrev Traffic Pattern												
$\epsilon - SVR$(Linear)	0.107	0.093	0.107	0.008	0.008	0.008	0.008	0.003	0.13	0.1299	0.1296	0.0061
$\epsilon - SVR$(RBF)	0.152	0.136	0.152	0.008	0.01	0.009	0.009	0.002	0.010	0.0103	0.0103	0.0064
$\upsilon - SVR$(Linear)	0.106	0.093	0.106	0.007	0.007	0.007	0.007	0.001	0.167	0.166	0.167	0.006
$\upsilon - SVR$(RBF)	0.046	0.046	0.046	0.005	0.005	0.005	0.005	0.001	0.007	0.007	0.007	0.002
Bitcom Traffic Pattern												
$\epsilon - SVR$(Linear)	0.015	0.003	0.023	0.052	—	—	—	0.1421	0.007	0.007	0.007	0.006
$\epsilon - SVR$(RBF)	0.004	0.003	0.052	0.070	0.0086	0.0086	0.0086	0.01	0.006	0.006	0.006	0.004
$\upsilon - SVR$(Linear)	0.033	0.003	0.005	0.001	—	—	—	0.1549	0.009	0.009	0.009	0.0002
$\upsilon - SVR$(RBF)	0.038	0.014	0.003	0.005	0.0839	0.0839	0.084	0.0096	0.007	0.007	0.007	0.001

traffic pattern for the router power, total power of Mesh, Torus and Cmesh topologies. The accuracy of router power for mesh topology is 99.90%, 99.74% accuracy for Torus topology and 99.89% accuracy for Cmesh topology. Similarly, Cmesh topology accuracy is 99.70%. The error rates of LBF-NoC for router power and total power are shown in Figure 3.13(b). Table 3.6 shows the MSEs of remaining traffic patterns where it consists MSEs of both router power and total power. The accuracy of router power and total power for all the traffic patterns is more than 90%.

An LBF-NoC framework has been compared with SVR-NoC (Qian et al. [2016])

Table 3.5: Algorithm with Highest Accuracy for six traffic patterns

Traffic Patterns	NoC Topologies					
	Mesh		Torus		Cmesh	
	Algorithm	Accuracy	Algorithm	Accuracy	Algorithm	Accuracy
Uniform	$\epsilon - SVR$(RBF)	94.33	$\epsilon - SVR$(Linear)	99.14	$\epsilon - SVR$(Linear)	98
Tornado	$\epsilon - SVR$(RBF)	94.73	$\epsilon - SVR$(Linear)	98.98	$\epsilon - SVR$(RBF)	97.43
Transpose	$\upsilon - SVR$(RBF)	96.05	$\upsilon - SVR$(RBF)	97.73	$\upsilon - SVR$(RBF)	98.45
Shuffle	$\epsilon - SVR$(RBF)	99	$\epsilon - SVR$(RBF)	99.67	$\upsilon - SVR$(Linear)	99.8
Bitrev	$\upsilon - SVR$(RBF)	96.43	$\upsilon - SVR$(RBF)	98.4	$\upsilon - SVR$(RBF)	99.43
Bitcom	$\epsilon - SVR$(RBF)	96.78	$\upsilon - SVR$(RBF)	93.46	$\upsilon - SVR$(RBF)	99.45



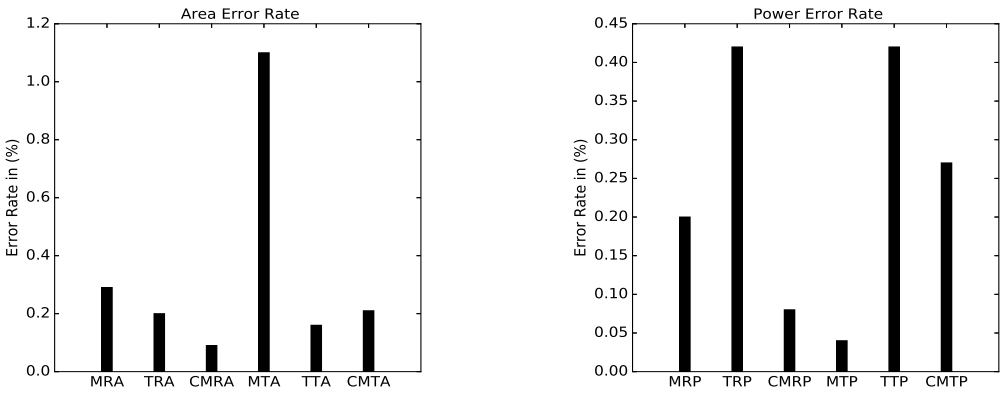
(a) (b)

Figure 3.12: Comparison of LBF-NoC with Booksim for Total area and Total power for 20×20, 30×30 and 40×40.

Table 3.6: MSE of Power parameters for Synthetic traffic pattern

MSE of Power parameters for Synthetic traffic pattern						
Traffic Patterns	Mesh		Torus		Cmesh	
	Router power	Total power	Router power	Total power	Router power	Total power
Tornado	0.018	0.024	0.091	0.11	0.08	0.14
Transpose	0.007	0.003	0.05	0.006	0.0075	0.002
Shuffle	0.008	0.012	0.01	0.0051	0.012	0.023
Bitrev	0.069	0.021	0.007	0.008	0.018	0.045
Bitcom	0.008	0.003	0.010	0.006	0.018	0.017

and Booksim2.0 (Jiang et al. [2013]). Both the works uses ML approaches. Results were compared with SVR-NoC and LBF-NoC framework. The SVR-NoC framework gave an average error rate of 12.5% for bitrev and bitcom traffic patterns whereas



(a) (b)

Figure 3.13: Error rates of LBF-NoC for router area, total area, router power and total power where MRA: Mesh Router Area TRA: Torus Router Area CMRA: Cmesh Router Area,MTA: Mesh Total Area TTA: Torus Total Area CMTA: Cmesh Total Area MRP: Mesh Router Power TRP: Torus Router Power CMRP: Cmesh Router Power MTP: Mesh Total Power TTP: Torus Total Power CMTP: Cmesh Total Power

LBF-NoC framework provided an average error rate of 1.25%. For tornado traffic pattern SVR-NoC gave 10.76% average error rate whereas LBF-NoC provided an average error rate of 0.5%. SVR-NoC framework targeted only Mesh topology for the experiments. Whereas LBF-NoC targets direct and indirect topologies to experiment and analyse the results.

Comparison of the execution time of the LBF-NoC framework and Booksim simulator execution time is made. Timing comparison made for the testing dataset provided for the LBF-NoC framework. The framework has targeted Mesh, Torus and Cmesh topologies with various topology sizes, traffic patterns, injection rates, and virtual channels. Table 3.7 shows the timing comparison of the simulator for Mesh, Torus, and Cmesh with LBF-NoC framework. Total time taken to complete the execution of all architectural sizes with different traffic patterns for Mesh topology was $8.31*10^6$ in seconds, which is 96.13 days for the simulator. Whereas LBF-NoC provided results in 321.52 seconds, i.e., 5.12 minutes to predict all parameters of NoC. For Torus topology simulator took $2.72*10^6$ seconds, which is 31.47 days. LBF-NoC took 1234.92 seconds which is 20.58 minutes. For Cmesh simulator took $1.98*10^6$ seconds, which is 22.97 in days. LBF-NoC took 470.25 seconds, 7.84 in minutes. A minimum speedup of 1000× to 1500× speedup achieved over cycle-accurate simulator.

A similar comparison was made with SVR-NoC framework where it takes 7 seconds to predict the latency values of 8×8 Mesh architecture. The LBF-NoC framework provided the same latency value in 0.3 seconds. For this experiment, LBF-NoC is 23.3× faster than SVR-NoC.

Table 3.7: Timing Comparison of Simulation results with LBF-NoC framework for different Synthetic Traffic patterns

Timing Comparison of Simulation results with LBF-NoC framework for different Synthetic traffic patterns									
Traffic Patterns	**Simulation Timings for Mesh**		**LBF-NoC Timings**	**Simulation Timings for Torus**		**LBF-NoC Timings**	**Simulation Timings for Cmesh**		**LBF-NoC Timings**
	In Seconds	In Hours	In Seconds	In Seconds	In Hours	In Seconds	In Seconds	In Hours	In Seconds
Uniform	5173049	1436.96 [59.87days]	10.22	767918.45 [8.89days]	213.31	165.15	851560	236.54 [9.86days]	96.56
Tornado	3113207	864.78 [36.03days]	245.41	1932159 [22.36days]	536.71	800.96	1088992	302.5 [12.6days]	106.32
Transpose	4875.84	1.35	5.93	7859	2.18	93.97	11342.96	3.15	64.53
Shuffle	7156.28	1.99	28.14	4716	1.31	54.08	5145.05	1.43	53.08
Bitrev	3629.91	1.01	14.13	2583.23	0.72	65.33	13001.80	3.61	69.09
Bitcom	3878.49	1.08	8.69	3710.43	1.03	55.43	14157.8	3.93	80.67

C LBF-NoC for Indirect Topologies

LBF-NoC sub-models were created for distinct traffic patterns. For all the output features both variants of ϵ-SVR and υ-SVR with polynomial and RBF kernels performed efficiently while predicting the parameters for indirect topologies.

Performance parameters comparison for Fat-Tree and Flatfly topologies have been made considering six different traffic patterns. Results of ANL for uniform and transpose traffic patterns are shown (the results for remaining traffic patterns are similar) hence, the MSE values have been shown for each traffic pattern separately in Table 3.8.

Figure 3.14 illustrates the comparison of ANL to uniform traffic pattern for Fat-Tree and Flatfly topologies against Booksim. In the figure, the x-axis represents the topology size and the y-axis represents normalized network latency. For Fat-Tree, LBF-NoC produced 96.9% accuracy and 98.7% for Flatfly topology. Figure 3.15 shows the comparison of ANL for Fat-Tree and Flatfly topologies against Booksim for bitrev traffic pattern. LBF-NoC provided 98.3% accuracy for Fat-Tree and 97.2% for Flatfly topology respectively. Table 3.8 shows the MSEs of different traffic patterns where it

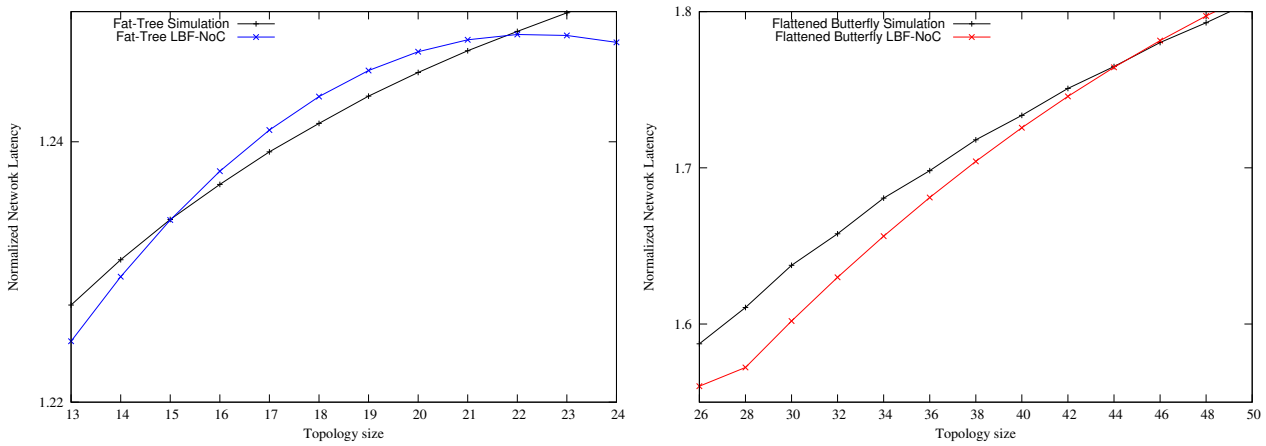


Figure 3.14: Average Network Latency comparison of LBF-NoC with simulation results for Indirect topologies with Uniform traffic pattern (a) Fat-Tree Topology, (b) Flatfly Topology

consists MSE of latencies, router power and total power.

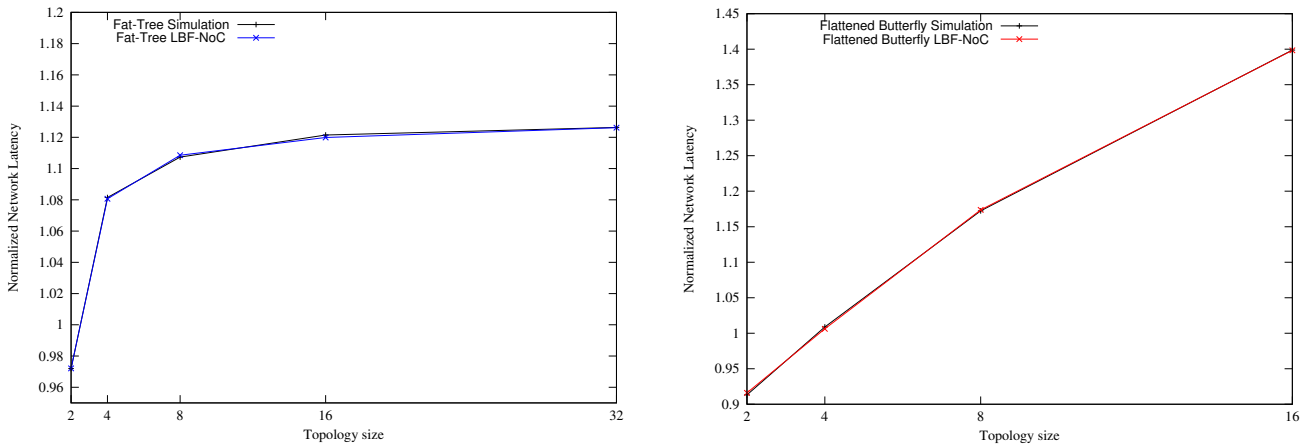
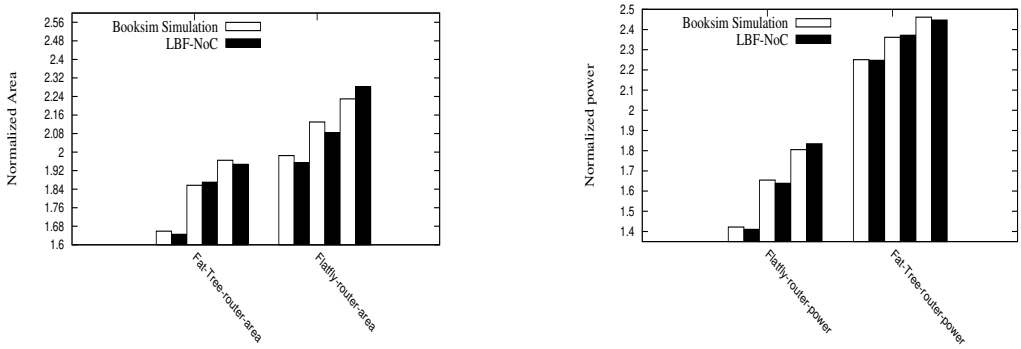


Figure 3.15: Average Network Latency comparison of LBF-NoC with simulation results for Indirect topologies with Bitrev traffic pattern (a) Fat-Tree Topology, (b) Flatfly Topology

Figure 3.16(a) & 3.17(a) shows the area parameters obtained from LBF-NoC against Booksim. The results are shown for where k=15, 20, 25 for Fat-Tree & k=30, 40, 50 for Flatfly the results for remaining architecture sizes are similar.

For Fat-Tree topology the accuracy of the router area and total area is 98.72%, 97.34% respectively. The accuracy is 96.86% and 95.67% for flatfly topology, respectively.

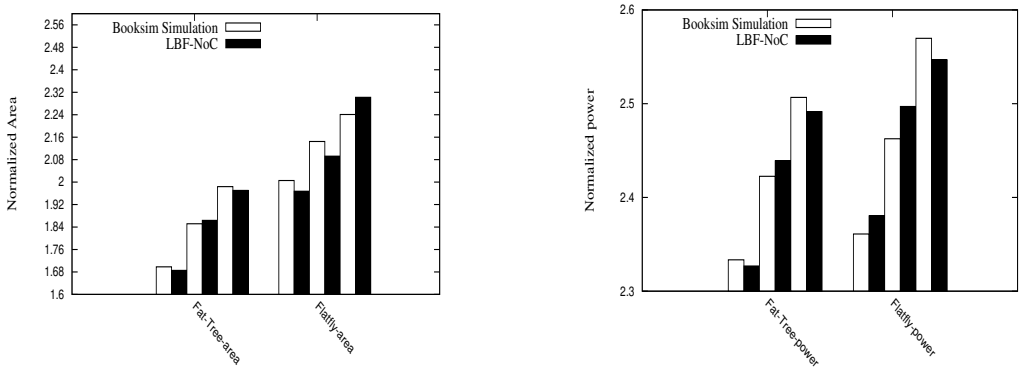
Figure 3.16(b) & 3.17(b) shows the comparison of LBF-NoC with Booksim simulator using uniform traffic pattern for the router and total power of Fat-tree and



(a) (b)

Figure 3.16: Comparison of LBF-NoC with Booksim for Router area and Router power for Fat-Tree and Flatfly topologies(where k=15, 20, 25 for Fat-Tree & k=30, 40, 50 for Flatfly).

Flatfly topologies.



(a) (b)

Figure 3.17: Comparison of LBF-NoC with Booksim for Total area and Total power for Fat-Tree and Flatfly topologies(where k=15,20,25 for Fat-Tree & k=30,40,50 for Flatfly).

The router power accuracy for Fat-Tree topology is 98.6%, the accuracy for Flatfly topology is 97.2%. The total power accuracy for Fat-tree topology is 96.3%, the accuracy for Flatfly topology is 97.4%.

Table 3.8 shows the MSEs of both router power and total power for remaining traffic patterns. For all the traffic patterns the accuracy of router power and total power is more than 94%.

Comparison of the execution time of the LBF-NoC framework and Booksim sim-

Table 3.8: MSEs of Different Performance and Power parameters for Synthetic traffic patterns

MSE of Performance and Power parameters for Synthetic traffic pattern										
Traffic Patterns	**Fattree**					**Flatfly**				
	Performance Parameters			Power parameters		Performance Parameters			Power parameters	
	ANL	APL	AFL	SP	TP	ANL	APL	AFL	SP	TP
Uniform	0.0411	0.0039	0.0044	0.0086	0.0093	0.0134	0.024	0.0121	0.0092	0.0076
Tornado	0.021	0.024	0.026	0.025	0.0058	0.0089	0.0094	0.0082	0.013	0.0028
Neighbor	0.012	0.0135	0.016	0.0061	0.0084	0.0091	0.0099	0.014	0.0073	0.0054
Bitrev	0.0037	0.0028	0.0021	0.031	0.047	0.01	0.0096	0.012	0.0081	0.0033
Transpose	0.0053	0.0061	0.0058	0.0073	0.0066	0.0012	0.0024	0.0016	0.029	0.067
Shuffle	0.0077	0.0062	0.0068	0.0027	0.0094	0.005	0.0064	0.0072	0.0011	0.032

ulator execution time is made. Timing comparison made for the testing dataset provided for the LBF-NoC framework. The framework has targeted Fat-Tree and Flatfly topologies with various topology sizes, traffic patterns, injection rates, and virtual channels. Table 3.9 shows the timing comparison of the simulator for Fat-Tree and Flatfly With LBF-NoC framework for individual traffic pattern. Total time is taken to complete the execution of all architectural sizes with different traffic patterns Fat-Tree topology was $1.66*10^8$ in seconds for the simulator. Whereas LBF-NoC provided results in 39195.2 seconds, i.e., 10.89 hours to predict all-out features for Fat-Tree topology. For Flatfly topology simulator took $1.48*10^8$ seconds. LBF-NoC took 38604.8 seconds which is 10.72 hours. A minimum speedup of 5000× to 5500× speedup for every single architecture achieved over cycle-accurate simulator.

Table 3.9: Timing Comparison of Booksim results with LBF-NoC framework for Fat-Tree and Flatfly topologies

Timing Comparison of Simulation results with LBF-NoC framework for traffic patterns						
Traffic Patterns	**Simulation Timings for Fat-Tree**		**LBF-NoC Timings**	**Simulation Timings for Flatfly**		**LBF-NoC Timings**
	In Seconds	In Hours	In Seconds	In Seconds	In Hours	In Seconds
Uniform	57553002	15989.94	11520	59858604	16627.39	12902.4
Tornado	53952561	14986.82	11598.4	62319816	17311.06	13824
Neigbhor	52924482	14701.25	10137.6	24278670	6744.08	7372.8
Transpose	114933.82	31.93	1536	1278368	355	2662.4
Shuffle	270870.72	75.24	1843.2	40349.44	11.21	1024
Bitrev	1254776.96	348.55	2560	36579.2	10.16	819.2

3.2 Enhanced Machine Learning Framework using Multiprocessing

To provide an efficient NoC design the chip designers need to simulate the simulator with various configurations considering different architecture sizes, virtual channels, buffer sizes, injection rates, traffic patterns, etc. Hence, it becomes an overhead to the chip designer as he needs to run the simulator many times to get an efficient NoC design.

To overcome this problem multiprocessing scheme was added along with machine learning model which is entitled as Multiprocessing Regression Framework (MRF-NoC). This framework generates all combinations of the input configurations given and executes them simultaneously. Along with Booksim simulator, Orion simulator was used which provides detailed power characteristics of on-chip routers in this enhanced framework. In this section, 2D & 3D Mesh NoC architectures have been considered. For 2D Mesh architecture size was considered from 2×2 to 50×50 and for 3D Mesh 2×2×2 to 30×30×2 with two layers was considered.

3.2.1 Multiprocessing Regression Framework (MRF-NoC)

MRF-NoC has been divided into two phases: Phase 1 specifies the ML model, which works the same as explained in the previous section. Phase 2 specifies the multiprocessing model.

A Machine Learning Model

In this section, the same ML model which was used in the previous section was used. The ML model is proposed to predict the design parameters of 2D and 3D Mesh NoC architectures like performance parameters (network latency, packet latency, flit latency, hop count) power consumption of various router components (buffer power, switch arbiter power, crossbar power, virtual channel arbiter power), the total power of NoC, router area and total area.

Along with MSE, three different error metrics have been used root mean square error (RMSE), mean absolute error (MAE) and r-squared to evaluate the performance of ML models which are given by the following equations:

$$MAE = \frac{\sum_{i=1}^{l} |a_i^* - a_i|}{n}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{l} (a_i^* - a_i)^2}{l}}$$

$$R^2 = 1 - \frac{\sum (a_i - a_i^*)^2}{\sum (a_i - \bar{a_i})^2}$$

where a_i is the actual value, a_i^* is the predicted value, $\bar{a_i}$ is the mean value of a_i and 'l' is the number of data points.

B Multiprocessing Model

The generated ML model considers various NoC configuration parameters, such as topology sizes, virtual channels, buffer sizes, injection rates, traffic patterns to generate different combinations of configurations. According to the configurations considered in this work, the aggregated simulations required for 2D mesh is 5628 and 3480 for 3D Mesh which is overcome by using multiprocessing. ML models are created for individual parameters and traffic patterns are simulated simultaneously using multiprocessing.

Figure 3.18 shows the overview of the MRF-NoC which is the combination of the ML model and multiprocessing model. ML models are created for individual parameters and traffic patterns are simulated simultaneously using multiprocessing.

Multiprocessing is done by utilizing the process class multiprocessing facility provided by python (McKerns et al. [2012]). Advantage of using multiprocessing is to improve the performance of the program over sequential programming. The process class uses FIFO scheduling to allocate the jobs stored in the memory. The MRF-NoC model designed creates six processes in stage 1 for each of the six traffic patterns. Each of these traffic patterns creates three other subprocesses in stage 2 and future each subprocess created in stage 2 spawn four other subprocesses in stage 3. The subprocesses created in stage 3 finally gives the results which are recorded for further analysis.

C Experimental Results and Analysis

Booksim2.0 (Jiang [(accessed 2012]) and Orion3.0 (Kahng et al. [2009]) simulators are used to accumulate the training data for 2D and 3D Mesh topologies, under the following traffic patterns: uniform, tornado, transpose, shuffle, bitrev and bitcom.

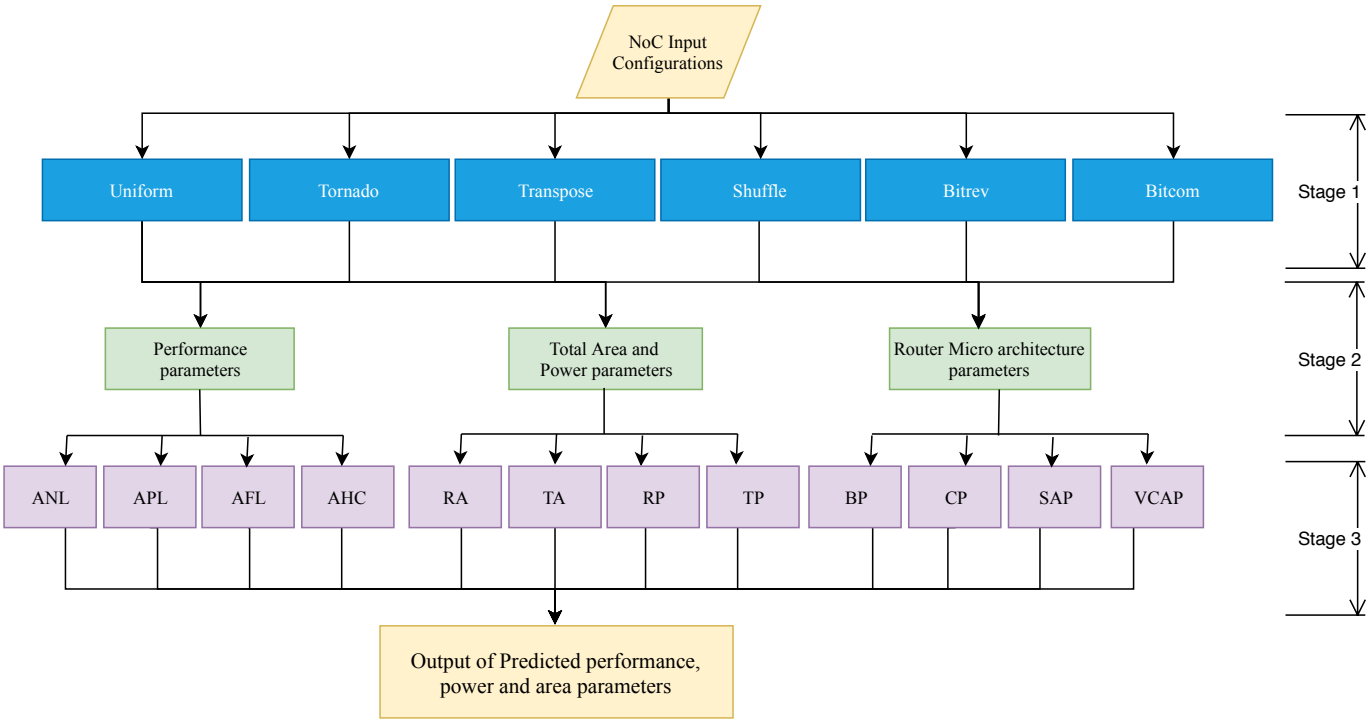


Figure 3.18: Flowchart of Multiprocessing Model.

The results for 2D Mesh architecture have been explained in the previous section 3.1.4.2 hence, the results of 3D Mesh architecture have been explained in this section.

Table 3.10: Algorithm with least error rates for various traffic patterns for 3D Mesh

Performance Parameters of 3D Mesh Topology				
Traffic Pattern	ANL	APL	AFL	AHC
Uniform	υ-SVR(linear)	υ-SVR(linear)	υ-SVR(RBF)	υ-SVR(RBF)
Tornado	ϵ-SVR(RBF)	υ-SVR(RBF)	ϵ-SVR(RBF)	υ-SVR(RBF)
Transpose	ANL	APL	AFL	
	υ-SVR(RBF)	ϵ-SVR(RBF)	υ-SVR(RBF)	
Shuffle	υ-SVR(RBF)	ϵ-SVR(RBF)	ϵ-SVR(RBF)	
Bitrev	ϵ-SVR(RBF)	υ-SVR(RBF)	ϵ-SVR(RBF)	
Bitcom	υ-SVR(RBF)	ϵ-SVR(RBF)	υ-SVR(RBF)	

Topology sizes from 2×2×2 to 10×10×2 were used for training and 11×11×2 to 30×30×2 for testing. Predictions have been made for all the parameters of NoC from 11×11×2 to 30×30×2 topology sizes for uniform and tornado traffic pattern. Transpose, shuffle, bitrev and bitcom results have been predicted using different injection

Table 3.11: Error Metrics of ML models for 3D Mesh architectures

Error Metrics for Different NoC parameters with various Synthetic traffic patterns for 3D Mesh											
Traffic Patterns	**Error Metrics**	**Performance Parameters**				**Power Parameters**					
						Router Components Power				**Total Power**	
		ANL	APL	AFL	AHC	BP	CBP	VCAP	SAP	RP	TP
Uniform	MAE	0.29	0.28	0.27	0.06	0.22	0.24	0.08	0.06	0.011	0.01
	MSE	0.098	0.0959	0.097	0.0044	0.06	0.07	0.016	0.007	0.0022	0.001
	RMSE	0.31	0.314	0.309	0.06	0.25	0.27	0.08	0.27	0.002	0.001
	R-squared	0.99	0.99	0.99	0.99	0.95	0.95	0.96	0.97	0.99	0.99
Tornado	MAE	0.21	0.23	0.22	0.08	0.37	0.33	0.15	0.38	0.04	0.01
	MSE	0.093	0.095	0.094	0.006	0.19	0.14	0.055	0.16	0.005	0.0021
	RMSE	0.26	0.28	0.265	0.08	0.43	0.37	0.23	0.4	0.06	0.0011
	R-squared	0.99	0.99	0.99	0.99	0.96	0.95	0.96	0.93	0.99	0.99
			ANL	APL	AFL	BP	CBP	VCAP	SAP	RP	TP
Transpose	MAE		0.01	0.011	0.01	0.20	0.23	0.008	0.003	0.035	0.013
	MSE		0.0001	0.0013	0.0012	0.08	0.109	0.00015	0.001	0.004	0.0021
	RMSE		0.001	0.0012	0.011	0.29	0.33	0.01	0.01	0.05	0.002
	R-squared		0.99	0.99	0.99	0.95	0.94	0.98	0.99	0.99	0.99
Shuffle	MAE		0.01	0.007	0.011	0.07	0.059	0.04	0.048	0.04	0.06
	MSE		0.0001	0.0013	0.0013	0.010	0.0061	0.102	0.01	0.006	0.008
	RMSE		0.0012	0.0011	0.0013	0.10	0.078	0.101	0.1	0.06	0.01
	R-squared		0.99	0.99	0.99	0.95	0.94	0.98	0.99	0.99	0.99
Bitcom	MAE		0.021	0.032	0.022	0.14	0.066	0.016	0.015	0.09	0.08
	MSE		0.0006	0.001	0.0006	0.024	0.0055	0.002	0.0006	0.0093	0.0075
	RMSE		0.025	0.032	0.026	0.15	0.074	0.016	0.024	0.1	0.093
	R-squared		0.99	0.99	0.99	0.90	0.91	0.92	0.95	0.98	0.95
Bitrev	MAE		0.06	0.053	0.06	0.12	0.070	0.018	0.008	0.08	0.09
	MSE		0.006	0.0039	0.0062	0.02	0.0076	0.0007	0.008	0.0076	0.0095
	RMSE		0.079	0.062	0.078	0.14	0.087	0.02	0.092	0.095	0.1
	R-squared		0.99	0.99	0.99	0.92	0.91	0.92	0.90	0.98	0.95

rates.

Table 3.10 lists the algorithms which provided the least error rates for performance parameters. Error rates for uniform, tornado are 1% to 2%, 2% to 3% and for transpose, shuffle, bitrev,bitcom are 1% to 2% respectively. These results can be seen in Table 3.11.

Power of router components for all the traffic patterns υ-SVR with *'RBF'* kernel worked efficiently with an error rate of 7% to 9%. The error rates for router power and total power is 1% to 2%. For router area and total area the error rates is 1% to 3%. υ-SVR with *'polynomial'* kernel worked efficiently for both area and power

parameters. Table 3.11 shows the different error metrics of all the NoC parameters. The complete MRF-NoC framework showed an average error rate of 7% for 3D Mesh NoC architectures.

Table 3.12: Timing comparison of simulation results with MRF-NoC framework for different Synthetic traffic patterns

Traffic Patterns	**MRF Time for 2D Mesh**	**Simulation Time for 2D Mesh**		**MRF Time for 3D Mesh**	**Simulation Time for 3D Mesh**	
	(Seconds)	(Seconds)	(Hours)	(Seconds)	(Seconds)	(Hours)
Uniform	515.64 (8.59 Minutes)	5802298	1611.75 (67.16 days)	876.7056 (14.6 minutes)	5446137	1512.82 (63.03 days)
Tornado		5543384	1539.83 (64.16 days)		25549788	7097.16 (295.72 days)
Shuffle		7156.28	1.99		355737	98.82 (4.12 days)
Transpose		4875.84	1.35		43563.2	12
Bitrev		3629.91	1.01		28397.93	7.89
Bitcom		3878.49	1.08		45329	12.61

Table 5.5 shows the execution times of the Booksim simulator and the MRF framework for Mesh architectures. Total time taken to complete the execution of all architectural sizes with different traffic patterns for 2D Mesh architectures was $1.14*10^7$ seconds for Booksim simulator and 515.64 seconds for MRF. For 3D Mesh architectures, the simulation time was $3.14*10^7$ seconds and 876.7056 seconds for Booksim and MRF respectively. Speedup of 1000× to 1500× achieved over Booksim simulator for both 2D and 3D architectures.

3.3 Summary

In this chapter, a machine learning framework named Learning-based framework (LBF-NoC) is presented which can be used to predict the performance, power and area parameters of direct (Mesh, Torus, Cmesh), & indirect (Fat-Tree, Flatfly) NoC architectures. Analysis of different regression algorithms was done. Among all the regression algorithms, the lowest error rate was observed in the SVR algorithm. Hence, the SVR algorithm has been considered in LBF-NoC. It predicts the values of NoCs similar to conventional cycle-accurate Booksim simulator. In the next section, the

machine learning framework was extended by adding multiprocessing scheme to overcome the issue of simulating NoC architecture 'n' number of times.

Chapter 4

Ensemble Learning-Based Accelerator

In this chapter, a framework named Ensemble Learning-Based Accelerator (ELBA-NoC) is presented to predict design parameters of five different architectures which consist of both 2D and 3D architectures. ELBA-NoC was designed to predict parameters in two different scenarios. In the first scenario, it will predict the worst-case latency analysis for all the architectures considered by varying topology sizes and virtual channels. In the second scenario, the framework predicts various design parameters like performance parameters: average network latency, average packet latency, average flit latency, average hop count. Power consumption of various router components (buffer power, switch arbiter power, crossbar power, virtual channel arbiter power), the total power of NoC, router area and total area.

In this chapter, five different architectures are considered consisting of 2D (Mesh, Torus, Cmesh) and 3D (Mesh, Torus) architectures. In previous chapter, the architecture size was considered till 50×50 which is 2500 nodes. Whereas in the current chapter has considered the architecture size for 2D NoCs till 75×75 which is 5625 nodes.

ELBA-NoC was created to predict the design space exploration of large-scale architectures, hence it becomes an very essential tool for the chip designers. It helps them in understanding the impact of various design parameters in the early stage thus reducing the actual development cost and simulation time.

4.1 Dataset Description

Booksim 2.0 (Jiang [(accessed 2012]) and Orion (Kahng et al. [2009]) simulators are used to generate reference data over various NoC configurations considering both 2D and 3D architectures. Booksim simulator provides performance, power and area results for each architecture and Orion simulator provides power for various components of the router.

In chapter 3, section 3.1.1 have explained how the data is generated using Booksim simulator and what are the input and output features which will be considered for the experiments.

4.2 Feature Extraction

As specified earlier experiments have been conducted in two different scenarios. Hence, datasets are also generated in two different ways.

In first scenario, data is recorded from 2×2 to 10×10 for 2D architectures and 2×2×2 to 10×10×2 for 3D architectures. The simulations are done until the architecture reaches the saturation region. Various injection rates and virtual channels are used to study saturation points for two traffic patterns(uniform and tornado traffic pattern). Among the data collected, 50% is considered for training data and the remaining 50% is considered for testing data.

In second scenario, data is recorded from 2×2 to 75×75 for 2D Mesh and Torus architectures, 2×2 to 30×30 for Cmesh architectures. 2×2×2 to 30×30×2 with two layers for 3D Mesh and Torus architectures.

For uniform and tornado traffic patterns the architecture sizes from 2×2 to 20×20 were considered as training data and remaining architectures from 21×21 to 75×75 were considered as testing data for 2D Mesh and Torus architectures, and 2×2 to 10×10 was considered as training data and 11×11 to 30×30 as testing data for Cmesh architecture. For 3D Mesh and Torus 2×2×2 to 10×10×2 architecture sizes was considered as training data and 11×11×2 to 30×30×2 was considered as testing data.

For shuffle, transpose, bitrev, and bitcom dataset consisting of data associated with injection rate is increased from 0.001 flits/cycle to 0.0098 flits/cycle in steps of 0.05 flits/cycle (any 9 injection rates can be used for training and the remaining 8

can be used for testing).

4.3 Training

ELBA-NoC was built using Random Forest (RF) algorithm. Before using the RF algorithm, ELBA-NoC was tested with other various regression algorithms like ANN with identity and relu activation functions, different generalized linear regression algorithms, i.e., lasso, lasso-lars, larsCV, bayesian-ridge, linear, ridge, ridgeCV, lars, elastic-net, elastic-netCV and SVR with *linear, RBF, polynomial* kernels. Since RF algorithm gave a better results it was considered for model development.

The algorithm which has been selected to predict the design parameters of NoC has to work for the two different scenarios considered. In the first case, as specified earlier, it was tend to predict the latency values when the network enters the saturation region. Other algorithms failed to predict the values in saturation region but the RF algorithm was able to predict values as it uses an ensemble learning approach, it can investigate nonlinear and hierarchical relationships between predictors and response. In general, while overfitting can cause inaccurate estimation of new test data and thus negatively affect the generality of the model, the RF algorithm is robust enough against overfitting. Moreover, compared to other machine learning algorithms, such as ANN and SVM, RF needs only a few tunable parameters and therefore requires little effort for offline model tuning (Breiman [2001]).

SVR was able to predict parameters for the second scenario which is parameters with higher architecture size but failed to predict in the first scenario. When compared to SVR algorithm RF was able to predict parameters in two scenarios and the training time and execution time is much faster than SVR as there are only a few tunable parameters.

4.4 Random Forest Regression

The theory behind ensemble learning techniques is based on the fact that its accuracy is higher than other machine learning algorithms because the prediction combination is more accurate than any other single model.

RF (Breiman [2001]) is an ensemble method similar to the nearest neighbour

predictor. An ML (Breiman [1996]) based tree is grown which asserts higher prediction accuracy by utilizing ensembles of trees.

The accuracy of ensemble learning techniques is higher as composed to the other ML algorithms. The reason behind it is a combination of predictions provides than a single consistent model.

Aggregation over the ensemble results in a reduction in variance, thereby increasing prediction accuracy and by randomly drawing a subset of covariates, RFs aims to reduce the association between aggregated quantities. In RF, each node is divided into a subset of randomly selected predictors at that node. An RF algorithm for regression is explained below.

1. Generate 'n' bootstrap samples from the data.
2. Grow a regression tree by random sampling 'm' of the predictors for each of the generated bootstrap samples and select the best split among those variables.
3. Average of the aggregating trees are taken for prediction of new data.

Figure 4.1 illustrates the workflow for random forests for regression. The RF regression algorithm needs the input data, the number of trees 'n' and the number of variables to use in each split 'm'. The random property derives from two factors: (a) each 'n' tree is based on a random subset of observations (b) based on a random subset of 'm' candidate variables, each split within each tree is created. Out-of-bag samples can be used to calculate a variable and unbiased error rate, eliminating the need for a test set or cross-validation. Because a large number of trees are grown, there is a limited generalization error which means there is no possibility of overfitting, which is a very useful feature for prediction (Hastie et al. [2009], James et al. [2013]).

4.5 Results and Discussion

This chapter is divided into two parts; the first part ELBA-NoC predicts the latency values of topology sizes from 6 × 6 to 10 × 10 considering various injection rates starting from 0.001, 0.0015... until it reaches saturation region for two traffic patterns i.e. uniform and tornado considering 2D & 3D architectures. The purpose of doing these experiments is to check whether the framework can predict the latency values

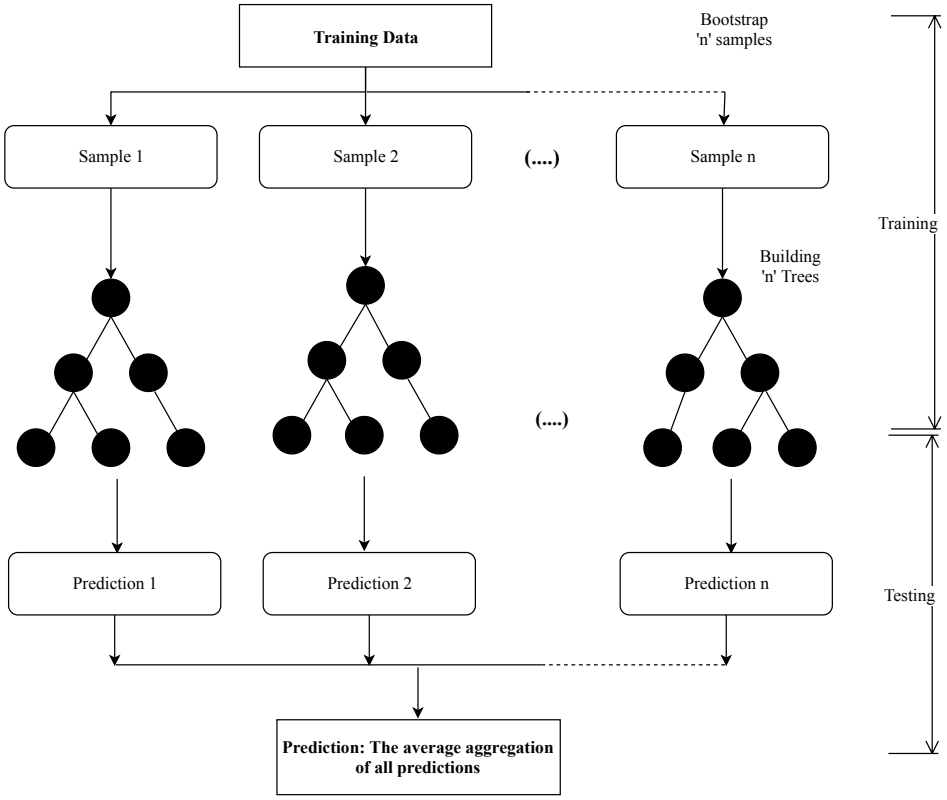


Figure 4.1: The flowchart of Random Forest for regression.

as the load on NoC increases. It is also done to check when the simulator attains saturation. Advantage of doing this is it helps the researchers and designers to restrict the amount of load.

In the second part of this work, ELBA-NoC is used to predict the performance parameters: (average network latency (ANL), average packet latency (APL), average flit latency (AFL), average hop count (AHC)), power parameters: (router, in turn, has buffer power (BP), crossbar power (CB), switch arbiter power (SAP), virtual channel arbiter power (VCAP), total router power(RP), total power(TP)) and area parameters: (router area (RA), total area (TA)) for 2D and 3D architectures considering various topology sizes, virtual channel size, buffer size, injection rates and traffic patterns.

Booksim 2.0 (Jiang [(accessed 2012]) and Orion (Kahng et al. [2009]) simulators are used to validate the ELBA-NoC results under different synthetic traffic patterns: uniform, tornado, transpose, shuffle, bitrev and bitcom (Dally and Towles [2004]).

4.5.1 Scenario-I

Injection rate method was chosen as it is one of the critical factors which affects the efficiency of the NoC communication. It will be a beneficial tool for the chip designers to put a threshold and restrict the amount of injection load on the NoC architectures.

Experimental results have been represented for 2D architectures and 3D architectures separately. Experiments were conducted from injection rate 0.001 to saturation point but, for representation in graphs, injection rates from 0.01 to saturation point were considered. Table 4.1 shows the different configurations considered for experiments.

Table 4.1: Configurations considered for Scenario-I

Booksim Network Configurations	
Topology	2D Mesh, Torus, Cmesh 3D Mesh, Torus
Topology Size	2 x 2, 3 x 3, ... 10 x 10
Traffic Pattern	Uniform, Tornado
Number of Virtual Channels(VCs)	2, 3, 4, 5
Buffers	6, 8
Injection rates	0.001, 0.0015, 0.002, 0.0025...until saturation
Sample time	100000 cycles

Figure 4.2 shows the comparison of ELBA-NoC with Booksim for Mesh architectures. In Figure 4.2 (a), (b), (c) represent results for uniform traffic pattern and Figure 4.2 (d), (e), (f) represent for tornado traffic pattern with VCs 2, 3 and 4 for topology sizes 7×7 to 10×10. As it can be observed in both traffic patterns VC2 saturates first, VC3 saturates second, then VC4 saturates later as shown in Figure 4.2. ELBA-NoC was able to predict the latency values with less than 3% error rate.

Figures 4.3, 4.4 shows the comparison for Torus and Cmesh architectures with VCs 2, 3 and 4 where (a), (b), (c) shows the results of uniform traffic pattern & (d), (e), (f) shows the results of tornado traffic pattern. ELBA-NoC was able to predict the latency values with an error rate between 2% and 4%. Both Torus and Cmesh saturate before the injection rate 0.01. Hence, for representation, injection rate from

0.006 as the initial point for plotting figures.

As specified earlier if MAE, MSE, RMSE are close to 0 and R-squared close to 1 it implies the predicted results are close to the actual results.

Table 4.2 represents the different error metrics for Mesh, Torus and Cmesh architecture. It can observed that MAE, MSE, RMSE are close to 0 and R-squared close to 1. Hence, it specifies the ELBA-NoC was able to predict the values which are very close to the actual results.

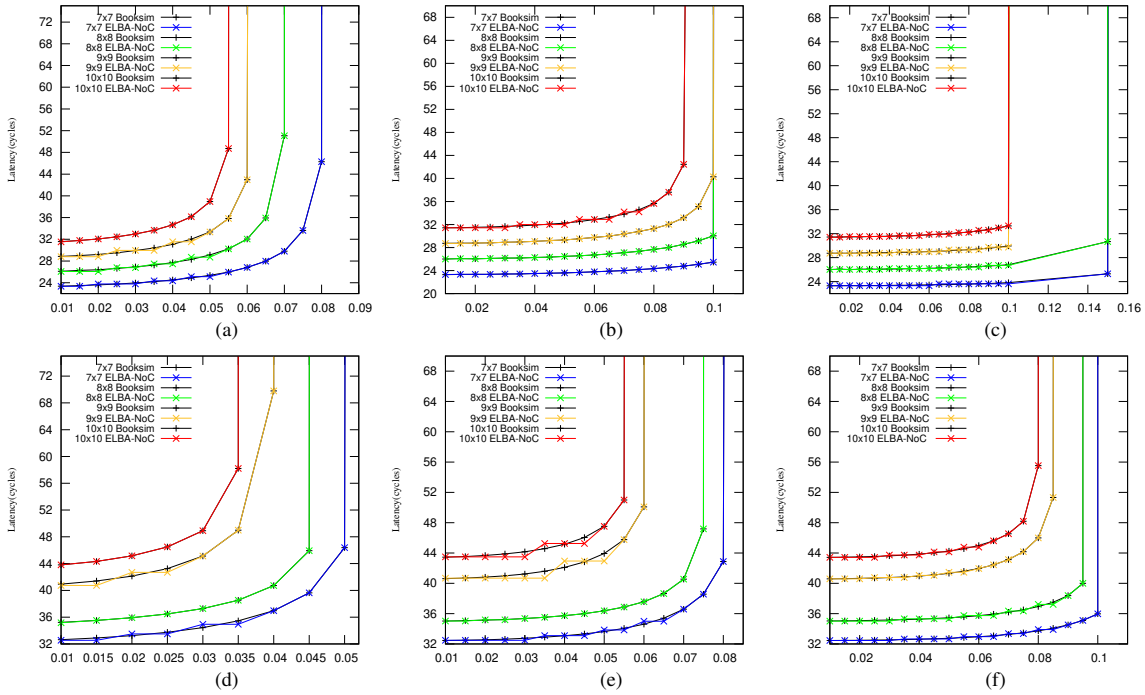


Figure 4.2: Latency comparison of ELBA-NoC with Booksim for 2D Mesh with Uniform and Tornado traffic patterns

Table 4.3 shows the execution timings of the Booksim simulator and the ELBA-NoC for 2D NoC architectures. The average simulation time to complete execution for architecture sizes from 6×6 to 10×10 with uniform and tornado traffic patterns for 2D architectures is $4.07*10^6$ seconds (47.08 days) and 3 seconds for ELBA-NoC.

Figure 4.5, and Figure 4.6 shows the comparison of 3D Mesh and Torus architectures where (a), (b), (c) shows the results of uniform traffic pattern & (d), (e), (f) shows the results of tornado traffic pattern. ELBA-NoC was able to predict the latency values of both 3D Mesh and Torus error rate between 2% to 5%. In the case of 3D architectures Torus with tornado traffic pattern saturates before 0.01 injection

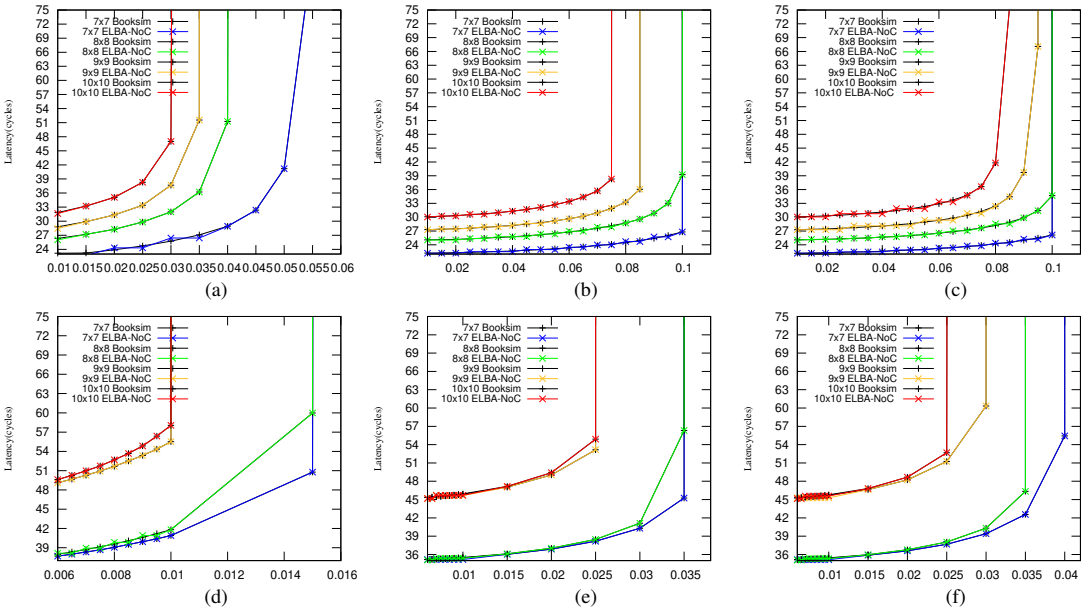


Figure 4.3: Latency comparison of ELBA-NoC with Booksim for 2D Torus with Uniform and Tornado traffic patterns

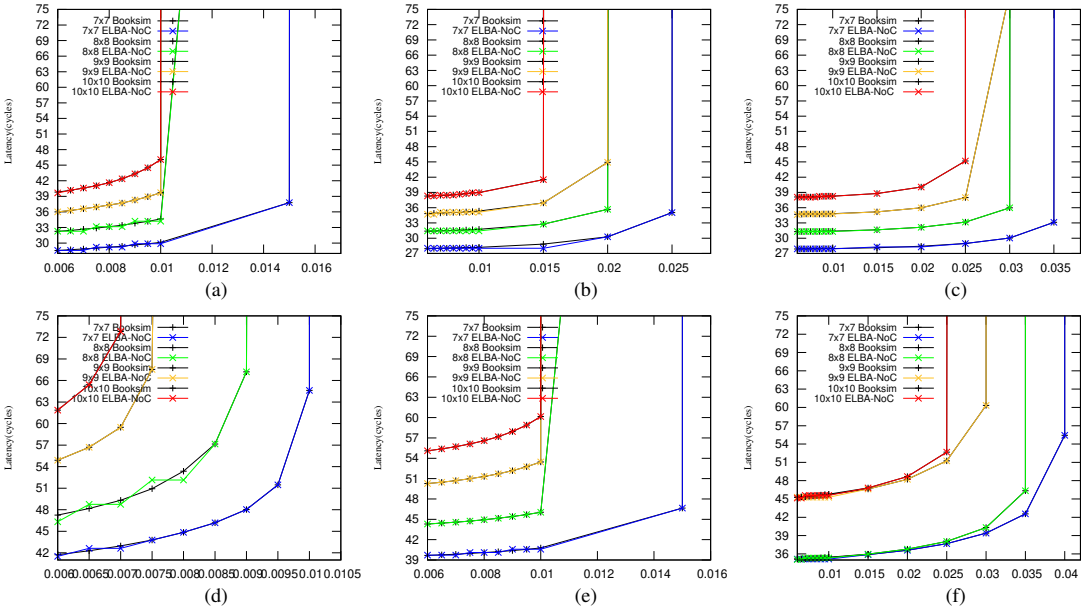


Figure 4.4: Latency comparison of ELBA-NoC with Booksim for Cmesh with Uniform and Tornado traffic patterns

rate, hence 0.005 has been considered as the initial point in representation.

Table 4.2 represents different error metrics for Mesh and Torus architectures. This implies that ELBA-NoC is able to predict latency values close to the actual values provided by Booksim simulator.

Table 4.2: Error Metrics of ELBA-NoC for Latency values with Uniform and Tornado traffic patterns for 2D & 3D architectures

Error Metrics of 2D & 3D Topologies						
Topologies / Traffic Patterns		Mesh	Torus	Cmesh	3D Mesh	3D Torus
Uniform	MAE	0.0446	0.091	0.064	0.019	0.10
	MSE	0.0084	0.02	0.06	0.0078	0.023
	RMSE	0.072	0.132	0.09	0.02	0.15
	R-Squared	0.98	0.97	0.98	0.99	0.98
Tornado	MAE	0.061	0.109	0.026	0.052	0.035
	MSE	0.019	0.0085	0.036	0.007	0.0025
	RMSE	0.126	0.114	0.103	0.08	0.05
	R-Squared	0.99	0.99	0.97	0.989	0.989

Table 4.3: Timing comparison of simulation results with ELBA-NoC for 2D Architectures

Traffic Patterns	**ELBA-NoC Time for Mesh**	**Simulation Time for Mesh**		**ELBA-NoC Time for Torus**	**Simulation Time for Torus**		**ELBA-NoC Time for Cmesh**	**Simulation Time for Cmesh**	
	(Seconds)	(Seconds)	(Hours)	(Seconds)	(Seconds)	(Hours)	(Seconds)	(Seconds)	(Hours)
Uniform	1.2	140352	38.99 (1.62 days)	1.6	1110033	308.34 (12.85 days)	2.1	700860	194.68 (8.11 days)
Tornado	2.3	445049	123.62 (5.15 days)	1.9	722309	220.6 (8.36 days)	3.3	949252	263.68 (10.99 days)

Table 4.4: Timing comparison of simulation results with ELBA-NoC for 3D Architectures

Traffic Patterns	**ELBA-NoC Time for Mesh**	**Simulation Time for 3D Mesh**		**ELBA-NoC Time for Torus**	**Simulation Time for 3D Torus**	
	(Seconds)	(Seconds)	(Hours)	(Seconds)	(Seconds)	(Hours)
Uniform	1.9	1503406	417.6 (17.4 days)	3.6	928572	257.94 (10.75 days)
Tornado	2.5	153368	426.02 (17.75 days)	4.3	2327464	646.02 (26.01 days)

Table 4.5: Error Metrics of ELBA-NoC for NoC parameters with various Synthetic traffic patterns for Mesh architectures

Error Metrics for Different NoC parameters with various Synthetic traffic patterns for 2D Mesh											
Traffic Patterns	Error Metrics	Performance Parameters				Power Parameters					
						Router Components Power				Total Power	
		ANL	APL	AFL	AHC	BP	CBP	VCAP	SAP	RP	TP
Uniform	MAE	0.12	0.21	0.12	0.048	0.022	0.025	0.0013	0.0003	0.002	0.0025
	MSE	0.020	0.065	0.023	0.005	0.0045	0.0048	0.00018	0.00087	0.0001	0.00013
	RMSE	0.15	0.25	0.15	0.07	0.067	0.069	0.004	0.009	0.002	0.003
	R-squared	0.99	0.99	0.99	0.99	0.96	0.964	0.98	0.979	0.99	0.99
Tornado	MAE	0.211	0.22	0.21	0.10	0.035	0.028	0.0014	0.00046	0.148	0.0021
	MSE	0.086	0.085	0.087	0.014	0.0087	0.0059	0.0002	0.00012	0.096	0.0023
	RMSE	0.29	0.3	0.294	0.11	0.093	0.077	0.0044	0.0011	0.31	0.0032
	R-squared	0.99	0.99	0.99	0.99	0.964	0.965	0.97	0.96	0.93	0.98
			ANL	APL	AFL	BP	CBP	VCAP	SAP	RP	TP
Transpose	MAE		0.09	0.089	0.068	0.047	0.06	0.08	0.001	0.03	0.18
	MSE		0.015	0.016	0.007	0.0032	0.018	0.013	0.009	0.002	0.11
	RMSE		0.12	0.124	0.089	0.056	0.13	0.11	0.03	0.04	0.3
	R-squared		0.99	0.99	0.998	0.94	0.946	0.95	0.935	0.96	0.94
Shuffle	MAE		0.014	0.0144	0.0141	0.054	0.059	0.02	0.038	0.025	0.05
	MSE		0.003	0.0034	0.00033	0.004	0.011	0.009	0.0017	0.008	0.007
	RMSE		0.018	0.017	0.0181	0.06	0.1	0.03	0.1	0.02	0.08
	R-squared		0.99	0.99	0.99	0.94	0.954	0.94	0.932	0.90	0.95
Bitcom	MAE		0.18	0.10	0.039	0.043	0.03	0.14	0.17	0.15	0.116
	MSE		0.03	0.014	0.003	0.04	0.03	0.028	0.04	0.06	0.04
	RMSE		0.19	0.11	0.05	0.12	0.018	0.18	0.175	0.25	0.2
	R-squared		0.99	0.99	0.99	0.95	0.97	0.99	0.99	0.89	0.972
Bitrev	MAE		0.75	0.08	0.094	0.088	0.02	0.16	0.16	0.16	0.101
	MSE		0.008	0.016	0.0167	0.023	0.005	0.039	0.03	0.06	0.03
	RMSE		0.09	0.12	0.129	0.15	0.02	0.198	0.19	0.26	0.17
	R-squared		0.99	0.99	0.99	0.94	0.97	0.99	0.98	0.941	0.97

Table 4.4 shows the execution timings of the Booksim simulator and the ELBA-NoC for 3D NoC architectures. The average simulation time to complete execution for architecture sizes from 6×6×2 to 10×10×2 with uniform and tornado traffic patterns for 3D architectures is $4.19*10^6$ seconds (56.86 days) and 4 seconds for ELBA-NoC.

4.5.2 Scenario II

This section provides a detailed study on the accuracy, errors and runtime of proposed ELBA-NoC approach. Predictions have been made for all the parameters of NoC from 21×21 to 75×75 topology sizes for uniform, tornado traffic pattern for Mesh and Torus architectures. 11×11 to 30×30 topology sizes for Cmesh architecture. Transpose,

Table 4.6: Error Metrics of ELBA-NoC for NoC parameters with various Synthetic traffic patterns for Torus architectures

Error Metrics for Different NoC parameters with various Synthetic traffic patterns for 2D Torus											
Traffic Patterns	Error Metrics	Performance Parameters				Power Parameters					
						Router Components Power				Total Power	
		ANL	APL	AFL	AHC	BP	CBP	VCAP	SAP	RP	TP
Uniform	MAE	0.09	0.096	0.1	0.009	0.02	0.015	0.0013	0.0002	0.039	0.0048
	MSE	0.01	0.032	0.041	0.002	0.0037	0.0021	0.00018	0.00037	0.0067	0.00015
	RMSE	0.15	0.25	0.15	0.05	0.061	0.046	0.004	0.00061	0.082	0.0013
	R-squared	0.98	0.987	0.98	0.99	0.97	0.97	0.98	0.964	0.955	0.95
Tornado	MAE	0.08	0.09	0.12	0.003	0.02	0.023	0.0013	0.00024	0.051	0.0026
	MSE	0.006	0.008	0.0075	0.003	0.0004	0.0039	0.00016	0.00041	0.118	0.0013
	RMSE	0.19	0.2	0.21	0.08	0.06	0.062	0.004	0.0006	0.108	0.0028
	R-squared	0.99	0.99	0.99	0.99	0.97	0.966	0.973	0.98	0.95	0.98
			ANL	APL	AFL	BP	CBP	VCAP	SAP	RP	TP
Transpose	MAE		0.06	0.08	0.07	0.047	0.06	0.08	0.001	0.03	0.18
	MSE		0.02	0.013	0.004	0.0032	0.018	0.013	0.009	0.002	0.11
	RMSE		0.09	0.096	0.088	0.056	0.13	0.11	0.03	0.04	0.3
	R-squared		0.99	0.98	0.998	0.94	0.946	0.95	0.935	0.90	0.94
Shuffle	MAE		0.02	0.016	0.011	0.054	0.059	0.02	0.038	0.025	0.05
	MSE		0.0025	0.0012	0.00038	0.004	0.011	0.009	0.0017	0.008	0.007
	RMSE		0.012	0.015	0.016	0.06	0.1	0.03	0.1	0.02	0.08
	R-squared		0.99	0.99	0.99	0.94	0.954	0.94	0.932	0.90	0.95
Bitcom	MAE		0.08	0.03	0.06	0.043	0.03	0.14	0.17	0.15	0.116
	MSE		0.008	0.009	0.0077	0.04	0.03	0.028	0.04	0.06	0.04
	RMSE		0.02	0.018	0.028	0.12	0.018	0.18	0.175	0.25	0.2
	R-squared		0.979	0.99	0.98	0.95	0.97	0.99	0.99	0.89	0.972
Bitrev	MAE		0.02	0.06	0.05	0.088	0.02	0.16	0.16	0.16	0.101
	MSE		0.005	0.007	0.0065	0.023	0.005	0.039	0.03	0.06	0.03
	RMSE		0.02	0.024	0.03	0.15	0.02	0.198	0.19	0.26	0.17
	R-squared		0.99	0.98	0.99	0.94	0.97	0.99	0.98	0.953	0.97

shuffle, bitrev and bitcom results are predicted using different injection rates.

Table 4.5 shows the error metrics for Mesh architecture with various synthetic traffic patterns and error metrics for each traffic pattern have been shown separately. Performance parameters error rate for uniform, tornado is less than 2% to 3%, for transpose, shuffle, bitrev, bitcom is 1% to 2% respectively. Error rates for power parameters are 3% to 4% for uniform and tornado traffic pattern and transpose, shuffle, bitrev, bitcom is 4% to 6% respectively.

Table 4.6 shows the error metrics for Torus architecture with various synthetic traffic patterns and error metrics for each traffic pattern have been shown separately. Performance parameters error rate for uniform, tornado is less than 1% to 2%, for

Table 4.7: Error Metrics of ML models for NoC parameters with various Synthetic traffic patterns for Cmesh architectures

Error Metrics for Different NoC parameters with various Synthetic traffic patterns for Cmesh											
Traffic Patterns	**Error Metrics**	**Performance Parameters**				**Power Parameters**					
						Router Components Power				**Total Power**	
		ANL	APL	AFL	AHC	BP	CBP	VCAP	SAP	RP	TP
Uniform	MAE	0.06	0.075	0.09	0.006	0.0069	0.054	0.0035	0.0007	0.081	0.15
	MSE	0.003	0.0045	0.006	0.0002	0.0127	0.0079	0.0033	0.0057	0.0067	0.004
	RMSE	0.08	0.09	0.1	0.08	0.112	0.089	0.0057	0.0012	0.25	0.006
	R-squared	0.99	0.98	0.97	0.99	0.97	0.967	0.975	0.96	0.98	0.98
Tornado	MAE	0.055	0.073	0.069	0.005	0.042	0.051	0.035	0.0027	0.017	0.0072
	MSE	0.0028	0.0041	0.0032	0.00049	0.0002	0.0024	0.00132	0.0007	0.09	0.0014
	RMSE	0.076	0.084	0.034	0.006	0.073	0.084	0.064	0.0057	0.12	0.0026
	R-squared	0.99	0.98	0.99	0.99	0.98	0.978	0.972	0.98	0.975	0.99
			ANL	APL	AFL	BP	CBP	VCAP	SAP	RP	TP
Transpose	MAE		0.08	0.084	0.072	0.052	0.059	0.074	0.002	0.026	0.062
	MSE		0.017	0.024	0.0178	0.009	0.0096	0.019	0.0006	0.006	0.011
	RMSE		0.098	0.101	0.086	0.081	0.088	0.093	0.004	0.035	0.077
	R-squared		0.98	0.98	0.992	0.99	0.99	0.98	0.98	0.975	0.981
Shuffle	MAE		0.026	0.017	0.018	0.053	0.057	0.023	0.032	0.027	0.056
	MSE		0.0028	0.0019	0.0004	0.0043	0.012	0.014	0.0027	0.0073	0.0092
	RMSE		0.014	0.017	0.015	0.09	0.13	0.046	0.16	0.029	0.082
	R-squared		0.99	0.99	0.99	0.96	0.956	0.97	0.965	0.96	0.95
Bitcom	MAE		0.078	0.038	0.062	0.044	0.037	0.15	0.165	0.18	0.125
	MSE		0.009	0.005	0.0075	0.0065	0.034	0.04	0.051	0.062	0.023
	RMSE		0.024	0.021	0.035	0.027	0.016	0.14	0.155	0.27	0.124
	R-squared		0.97	0.98	0.974	0.968	0.971	0.98	0.99	0.955	0.961
Bitrev	MAE		0.047	0.082	0.073	0.096	0.043	0.086	0.081	0.094	0.076
	MSE		0.007	0.0093	0.0087	0.048	0.0073	0.061	0.053	0.084	0.059
	RMSE		0.041	0.044	0.055	0.023	0.046	0.038	0.033	0.057	0.038
	R-squared		0.99	0.98	0.985	0.96	0.98	0.98	0.98	0.976	0.984

transpose, shuffle, bitrev, bitcom is 2% to 3% respectively. Error rates for power parameters are 2% to 3% for uniform and tornado traffic pattern and transpose, shuffle, bitrev, bitcom is 3% to 5% respectively.

Table 4.7 shows the error metrics for Cmesh architecture with various synthetic traffic patterns and error metrics for each traffic pattern have been shown separately. Performance parameters error rate for uniform, tornado is less than 3%, for transpose, shuffle, bitrev, bitcom is 2% to 3% respectively. Error rates for power parameters are 2% for uniform and tornado traffic pattern and transpose, shuffle, bitrev, bitcom is 4% respectively.

The error rate of router area and total area for Mesh topology are 0.08% and 1.2%

Table 4.8: Timing comparison of simulation results with ELBA-NoC for 2D NoC architectures

Traffic Patterns	ELBA-NoC Time for Mesh	Simulation Time for Mesh		ELBA-NoC Time for Torus	Simulation Time for Torus		ELBA-NoC Time for Cmesh	Simulation Time for Cmesh	
	(Seconds)	(Seconds)	(Hours)	(Seconds)	(Seconds)	(Hours)	(Seconds)	(Seconds)	(Hours)
Uniform	1792.8	26982000	7495	972	11058930	3071.92	60	6393020	1775.84
Tornado	2592	33744330	9373.42	1101.38	15626790	4340.78	84	8043888	2234.41
Shuffle	128	473280	131.47	48	100616.96	27.95	54	109761.08	30.49
Transpose	76.8	337920	93.87	62	167658.75	46.57	68	241977.85	67.22
Bitrev	132	460864.992	128.02	36	55108.48	15.31	72	277371.9	77.05
Bitcom	103	438400	121.78	68	112073.85	31.13	74	302033.15	83.9

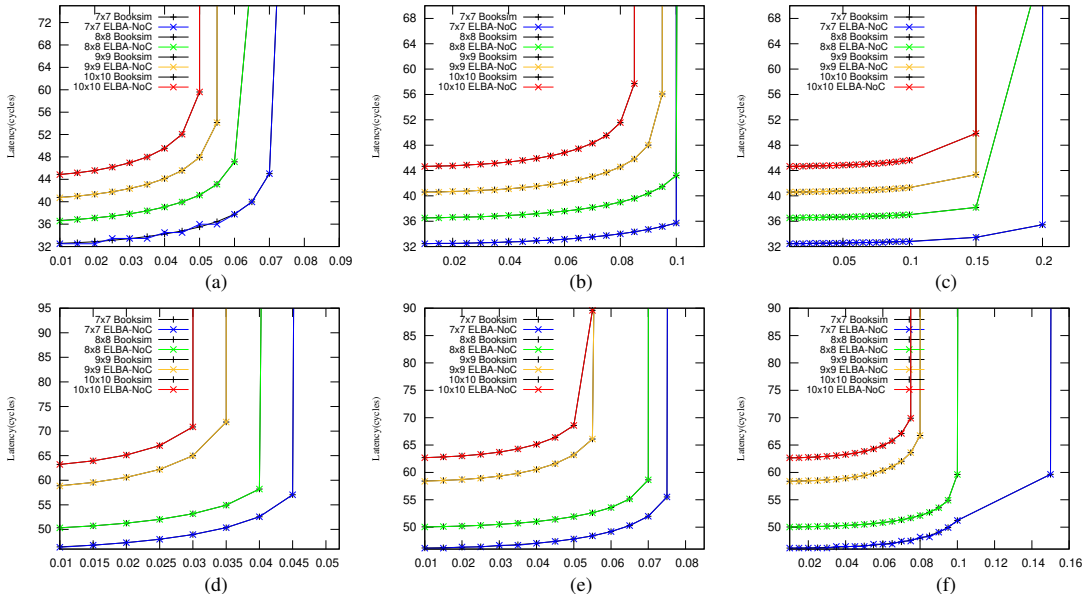


Figure 4.5: Latency comparison of ELBA-NoC with Booksim for 3D Mesh with Uniform and Tornado traffic patterns

respectively. For Torus topology, the errors are 0.7% and 0.2% respectively. Similarly, the errors for Cmesh topology is 2.1% and 3.2% respectively.

The training time of ELBA-NoC is about 1 to 2 hours for Mesh, Torus and Cmesh with six different traffic patterns. However, the training process is taken off-line and during the performance prediction, it does not incur additional overhead. Timing comparisons have been made for the testing configurations considered. The simulation time for Mesh architecture with six traffic patterns using Booksim simulator is $6.2*10^7$ seconds which is 718 days whereas ELBA-NoC was able to predict the same

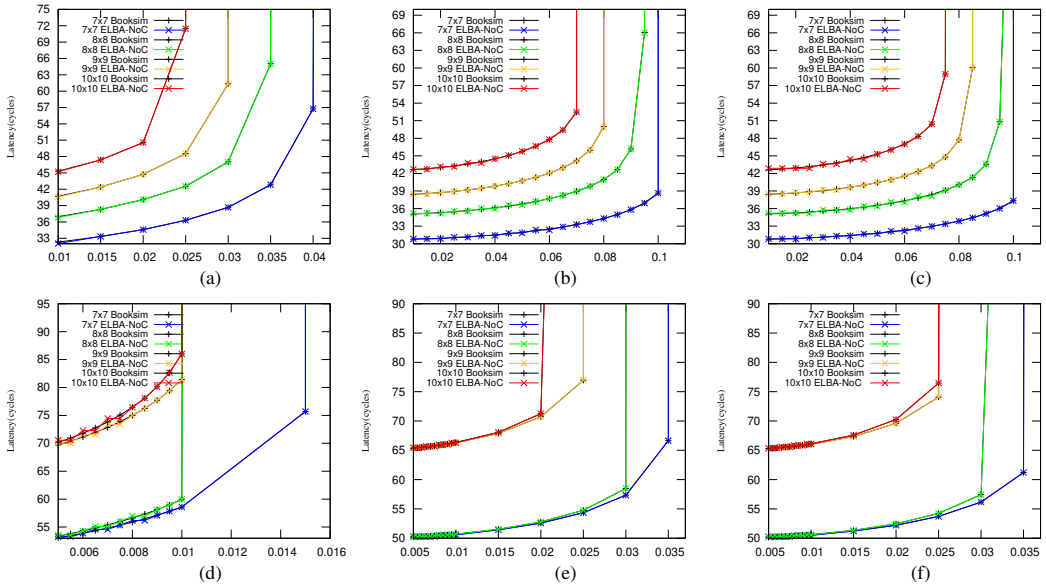


Figure 4.6: Latency comparison of ELBA-NoC with Booksim for 3D Torus with Uniform and Tornado traffic patterns

results in 4692.6 seconds which is 1.3 hours. For Torus architecture the simulation time is $2.71*10^7$ seconds which is 313.9 days and ELBA-NoC took 2287.38 seconds. For Cmesh the simulator took $1.54*10^7$ seconds which is 177.8 days and ELBA-NoC provided results in 412 seconds.

Table 4.8 shows the detailed time comparison of Mesh, Torus and Cmesh architectures for all the traffic patterns considered. Speedup of 16K× to 18K× for individual architecture was achieved over Booksim simulator.

Table 4.9 shows the error metrics for Mesh architecture with various synthetic traffic patterns and error metrics for each traffic pattern have been shown separately. Performance parameters error rate for uniform, tornado is less than 1% to 2%, for transpose, shuffle, bitrev, bitcom are 3% to 4% respectively. Error rates for power parameters are 2% to 3% for uniform and tornado traffic pattern and transpose, shuffle, bitrev, bitcom is 3% to 5% respectively.

Table 4.10 shows the error metrics for Mesh architecture with various synthetic traffic patterns and error metrics for each traffic pattern have been shown separately. Performance parameters error rate for uniform, tornado is less than 3%, for transpose, shuffle, bitrev, bitcom is 2% respectively. Error rates for power parameters are 3% for uniform and tornado traffic pattern and for transpose, shuffle, bitrev, bitcom is 4%

Table 4.9: Error Metrics of ML models for NoC parameters with various Synthetic traffic patterns for Mesh architecture

Error Metrics for Different NoC parameters with various Synthetic traffic patterns for 3D Mesh											
Traffic Patterns	Error Metrics	Performance Parameters				Power Parameters					
						Router Components Power				Total Power	
		ANL	APL	AFL	AHC	BP	CBP	VCAP	SAP	RP	TP
Uniform	MAE	0.055	0.073	0.081	0.0074	0.0061	0.059	0.0031	0.0008	0.083	0.06
	MSE	0.0028	0.0051	0.0072	0.00034	0.017	0.0072	0.0037	0.0045	0.0063	0.0038
	RMSE	0.078	0.092	0.11	0.08	0.09	0.084	0.0054	0.0011	0.09	0.05
	R-squared	0.99	0.98	0.972	0.99	0.97	0.98	0.975	0.98	0.98	0.98
Tornado	MAE	0.065	0.078	0.061	0.0045	0.047	0.053	0.038	0.0021	0.012	0.0078
	MSE	0.0023	0.0044	0.0037	0.0005	0.00027	0.0020	0.0018	0.0009	0.085	0.0013
	RMSE	0.076	0.082	0.031	0.0067	0.079	0.080	0.062	0.0055	0.17	0.0022
	R-squared	0.99	0.98	0.99	0.99	0.98	0.978	0.976	0.98	0.978	0.99
			ANL	APL	AFL	BP	CBP	VCAP	SAP	RP	TP
Transpose	MAE		0.078	0.081	0.077	0.056	0.054	0.072	0.0025	0.023	0.069
	MSE		0.013	0.027	0.0181	0.00902	0.0092	0.021	0.00074	0.0063	0.016
	RMSE		0.093	0.14	0.088	0.083	0.088	0.090	0.0044	0.039	0.072
	R-squared		0.98	0.98	0.983	0.99	0.99	0.98	0.98	0.975	0.987
Shuffle	MAE		0.022	0.011	0.013	0.052	0.058	0.026	0.034	0.027	0.053
	MSE		0.0026	0.0014	0.00042	0.0051	0.016	0.011	0.0022	0.0074	0.0096
	RMSE		0.012	0.011	0.017	0.093	0.134	0.0446	0.128	0.0275	0.0831
	R-squared		0.99	0.99	0.99	0.968	0.96	0.97	0.972	0.978	0.963
Bitcom	MAE		0.042	0.048	0.051	0.044	0.032	0.11	0.174	0.162	0.173
	MSE		0.0092	0.0057	0.0082	0.0072	0.033	0.046	0.0525	0.0694	0.0254
	RMSE		0.0232	0.0225	0.0365	0.0285	0.0159	0.19	0.184	0.282	0.161
	R-squared		0.98	0.98	0.979	0.975	0.973	0.98	0.99	0.956	0.952
Bitrev	MAE		0.052	0.055	0.059	0.092	0.041	0.083	0.087	0.096	0.071
	MSE		0.0065	0.009	0.0082	0.046	0.00701	0.063	0.057	0.084	0.0532
	RMSE		0.043	0.047	0.052	0.026	0.041	0.042	0.0365	0.0575	0.0392
	R-squared		0.99	0.98	0.98	0.968	0.98	0.98	0.98	0.972	0.98

respectively.

The training time of ELBA-NoC is about 1.5 hour for Mesh and Torus with six different traffic patterns. However, the training process is taken off-line and during the performance prediction, it does not incur additional overhead. Timing comparisons have been made for the testing configurations considered. The simulation time for Mesh architecture with six traffic patterns using Booksim simulator is $1*10^8$ seconds whereas ELBA-NoC was able to predict same results in 2460 seconds. For Torus architecture the simulation time is $9.23*10^7$ seconds whereas ELBA-NoC took 2803 seconds.

Table 4.11 shows the detailed time comparison of Mesh and Torus architectures

Table 4.10: Error Metrics of ML models for NoC parameters with various Synthetic traffic patterns for Torus architecture

Error Metrics for Different NoC parameters with various Synthetic traffic patterns for 3D Torus											
Traffic Patterns	Error Metrics	Performance Parameters				Power Parameters					
						Router Components Power				Total Power	
		ANL	APL	AFL	AHC	BP	CBP	VCAP	SAP	RP	TP
Uniform	MAE	0.042	0.055	0.058	0.0063	0.0053	0.052	0.0043	0.009	0.063	0.053
	MSE	0.0016	0.0029	0.0032	0.00047	0.0084	0.0083	0.0073	0.00843	0.022	0.0072
	RMSE	0.053	0.048	0.051	0.068	0.0492	0.0481	0.037	0.067	0.042	0.039
	R-squared	0.99	0.99	0.991	0.99	0.98	0.98	0.982	0.97	0.98	0.98
Tornado	MAE	0.0712	0.0742	0.0634	0.00381	0.047	0.053	0.0427	0.0031	0.023	0.0081
	MSE	0.0034	0.0039	0.0029	0.0011	0.037	0.043	0.0364	0.00102	0.024	0.00112
	RMSE	0.081	0.084	0.078	0.0063	0.071	0.0736	0.0711	0.0061	0.029	0.0887
	R-squared	0.99	0.99	0.99	0.99	0.99	0.983	0.971	0.99	0.986	0.984
			ANL	APL	AFL	BP	CBP	VCAP	SAP	RP	TP
Transpose	MAE		0.058	0.061	0.057	0.036	0.034	0.052	0.0015	0.003	0.049
	MSE		0.003	0.004	0.0028	0.0073	0.0072	0.0014	0.00054	0.0043	0.006
	RMSE		0.073	0.034	0.068	0.063	0.068	0.071	0.0022	0.009	0.052
	R-squared		0.99	0.989	0.99	0.99	0.99	0.98	0.99	0.973	0.981
Shuffle	MAE		0.012	0.001	0.003	0.042	0.048	0.016	0.024	0.017	0.043
	MSE		0.0015	0.0005	0.00032	0.0041	0.008	0.006	0.0012	0.0064	0.0086
	RMSE		0.006	0.001	0.013	0.083	0.093	0.075	0.094	0.064	0.0661
	R-squared		0.99	0.99	0.99	0.973	0.971	0.98	0.977	0.98	0.969
Bitcom	MAE		0.022	0.028	0.031	0.022	0.012	0.02	0.03	0.041	0.037
	MSE		0.0072	0.0031	0.0062	0.0053	0.013	0.027	0.039	0.057	0.003
	RMSE		0.014	0.017	0.018	0.016	0.011	0.014	0.0175	0.021	0.019
	R-squared		0.99	0.99	0.99	0.99	0.98	0.978	0.974	0.971	0.972
Bitrev	MAE		0.032	0.035	0.039	0.072	0.021	0.063	0.067	0.076	0.051
	MSE		0.0045	0.007	0.0062	0.025	0.0042	0.043	0.035	0.064	0.031
	RMSE		0.023	0.027	0.032	0.026	0.021	0.042	0.049	0.054	0.043
	R-squared		0.99	0.99	0.99	0.98	0.99	0.98	0.98	0.986	0.984

for all the traffic patterns considered. Speedup of 18K× to 20K× for individual architecture was achieved over Booksim simulator.

4.6 Summary

In this chapter, a framework named Ensemble Learning-Based Accelerator (ELBA-NoC) was presented for NoC parameters prediction. ELBA-NoC is built using Random Forest Ensemble regression algorithm for five different architectures including both 2D and 3D architectures (2DMesh, 2DTorus, Cmesh, 3DMesh, 3DTorus). The Framework has been used in two scenarios, first, it is used to predict average latency until the architecture reaches its saturation area for 6×6 to 10×10. Second, it is

Table 4.11: Timing comparison of simulation results with ELBA-NoC for 3D NoC architectures

Traffic Patterns	**ELBA-NoC Time for 3D Mesh**	**Simulation Time for 3D Mesh**		**ELBA-NoC Time for 3D Torus**	**Simulation Time for 3D Torus**	
	(Seconds)	(Seconds)	(Hours)	(Seconds)	(Seconds)	(Hours)
Uniform	1100	45149670	12541.57	1167	39842447.4	11067.35
Tornado	880	51871277.7	14408.96	1362	49818420	13838.48
Shuffle	102.4	603459.45	167.63	62	267533.84	74.31
Transpose	96	813253.16	225.9	58	259276.8	72.02
Bitrev	128	886701.44	246.31	66	273250.56	75.9
Bitcom	153.6	965830.4	268.29	88	1888931.975	524.7

used to predict performance, power and area parameters. To enable design space exploration of NoC, all the micro-architectural parameters like architecture sizes, injection rates, virtual channels, buffers and traffic patterns were considered. This allows system-level designers to perform design space exploration efficiently which is fast and accurate without having to know the details of the underlying architecture. The overall accuracy of 94% to 96% has been observed for both 2D and 3D NoC architectures. Framework predicts the values of NoCs similar to the conventional cycle-accurate Booksim simulator while providing a minimum 16K× speedup with 4% to 6% error rate.

Chapter 5

Floorplan-Based Learning Framework

In this chapter, two different simulators are explored one with a fixed delay between the IPs and another with the accurate dealy. Current simulators consist of a fixed delay component, and the length of the connection is determined by the physical dimension of the chip components. To determine the correct length of the link the integration of physical characteristics of the chip is required. Therefore ties of varying length and latency exist in the chip depending on the topology. Floorplan of a topology refers to the exact physical arrangement of nodes in the network. Based on the communication link in possible cases there may be many floorplans for the same topology.

In Booksim2.0 (Jiang et al. [2013]) simulator the architecture will have a fixed link delay length between the nodes. In (Halavar and Talawar [2018]) author has modified the Booksim simulator to work for floorplan based (accurate delay) architectures. Hence, both simulators one with fixed delay length and another with accurate delay are used in this thesis. Figure 5.1 shows both working of standard Booksim simulator and modified Booksim simulator.

A unified framework named Knowledgeable Network-on-Chip Accelerator (K-NoC) is presented which predicts design parameters of both simulators considered. Random Forest algorithm is used to build K-NoC and multiprocessing scheme is added along with K-NoC such that all the considered configurations can be evaluated simultaneously.

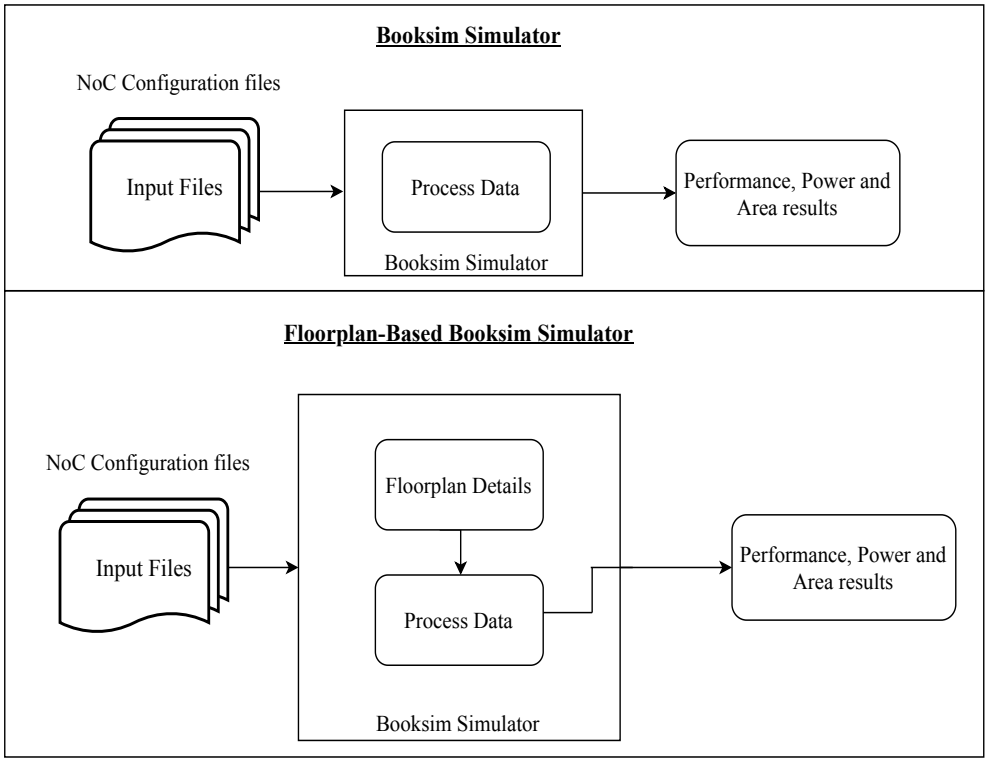


Figure 5.1: Working scenarios of Standard and Floorplan based Booksim simulators.

5.1 Study of Floorplan-Based Simulator

A detailed study was done on working of both simulators one with fixed delay and another with accurate delay. By the results it can be seen that the accurate delay simulator gives more latency when compared to the fixed delay simulator which can be observed in the Figure 5.2 where x-axis specifies the topology sizes with latency values of both simulators and y-axis specifies the average network latency in cycles. The experiment was done for uniform traffic pattern with virtual channel 4 and buffer depth 10. Latency values of two simulators specify that floor plan (accurate delay) simulator gives more latency when compared to standard (fixed delay) simulator (Halavar and Talawar [2018]).

5.2 Experimental Results

In this section, experiments were conducted for two simulators considered. Results are represented in two scenarios, the first scenario predicts the worst-case latency analysis and second scenario predicts the performance, power, and area parameters of Mesh architectures.

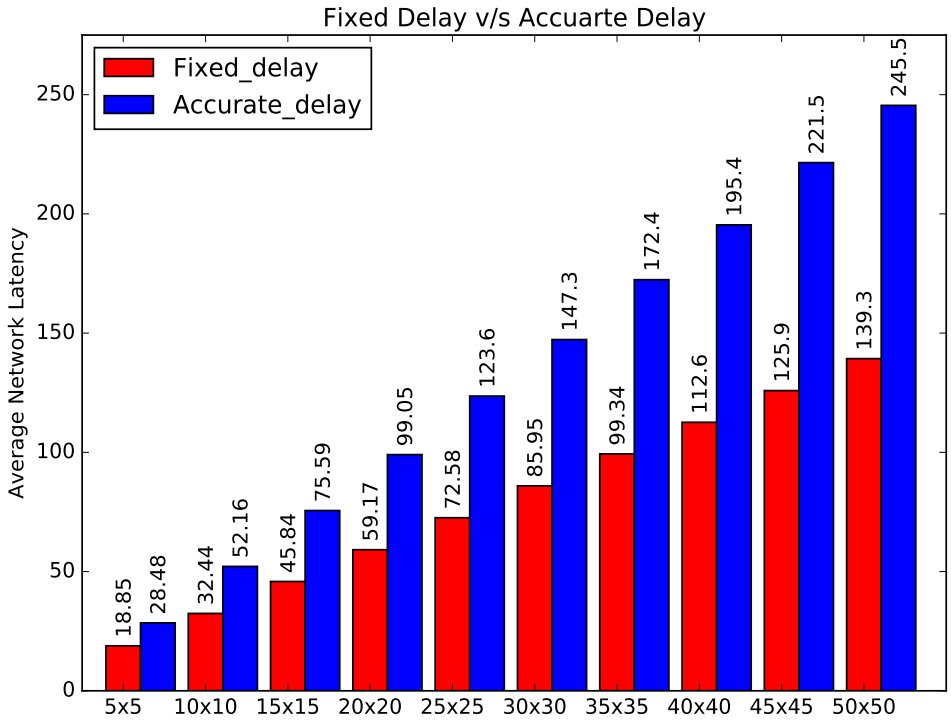


Figure 5.2: Latency values of Standard and Floorplan based Booksim simulators.

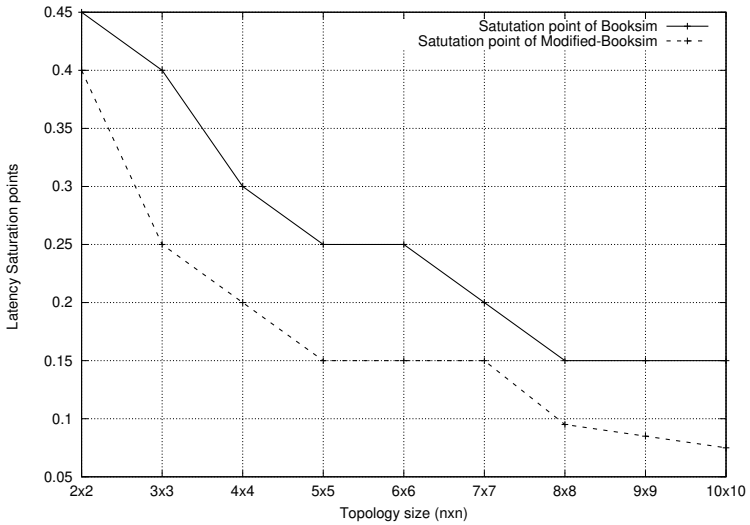


Figure 5.3: Saturation points of Booksim and Modified Booksim simulators for various architecture sizes

5.2.1 Worst Case Latency analysis

By the observed results it can be seen that modified Booksim (accurate delay) simulator saturates early when compared the actual standard Booksim (fixed delay) sim-

ulator which can be observed in Figure 5.3. As specified earlier the latency value is more for accurate delay simulator when compared to fixed delay simulator which can be seen in Figure 5.2. Hence experiments for both simulators have been done and results have been shown for 7×7, 8×8, 9×9 and 10×10 with different VCs results for other architecture sizes are similar to the results shown.

Experiments were conducted from injection rate 0.001 to saturation point but, for representation in graphs, injection rates were considered from 0.01 to saturation point.

Figure 5.4 shows the comparison of K-NoC with standard Booksim for Mesh architectures. The first three figures represent results for uniform traffic pattern and next three figures represent results for tornado traffic pattern with VCs 2, 3 and 4 for topology sizes 7×7 to 10×10. As it can be observed in both traffic patterns VC2 saturates first, VC3 saturates second, then VC4 saturates last as can be seen in Figure 5.4. K-NoC was able to predict the latency values with less than 3% error rate.

Figure 5.4 shows the comparison of K-NoC with standard Booksim for Mesh architectures. Where (a), (b), (c) shows the results of uniform traffic pattern & (d), (e), (f) shows the results of tornado traffic pattern with VCs 2, 3 and 4 for topology sizes 7×7 to 10 × 10. As it can be observed in both traffic patterns VC2 saturates first, VC3 saturates second, then VC4 saturates last as can be seen in Figure 5.4. K-NoC was able to predict the latency values with less than 3% error rate.

Figure 5.5 shows the comparison of K-NoC with modified Booksim for Mesh architectures. Where (a), (b), (c) shows the results of uniform traffic pattern & (d), (e), (f) shows the results of tornado traffic pattern with VCs 2, 3 and 4 for topology sizes 7×7 to 10×10. K-NoC was able to predict the latency values with less than 4% error rate.

K-NoC was able to predict the latency values of both simulators considered with an error rate of less than 4%. Table 5.1 represents the saturation points of various architecture sizes and different VCs for both simulators considered.

The following table will be very beneficial for the chip designers as they can limit the injection load to the architectures by referring the table in both fixed and accurate delay scenarios.

Table 5.2 represents the different error metrics for both simulators. It can be

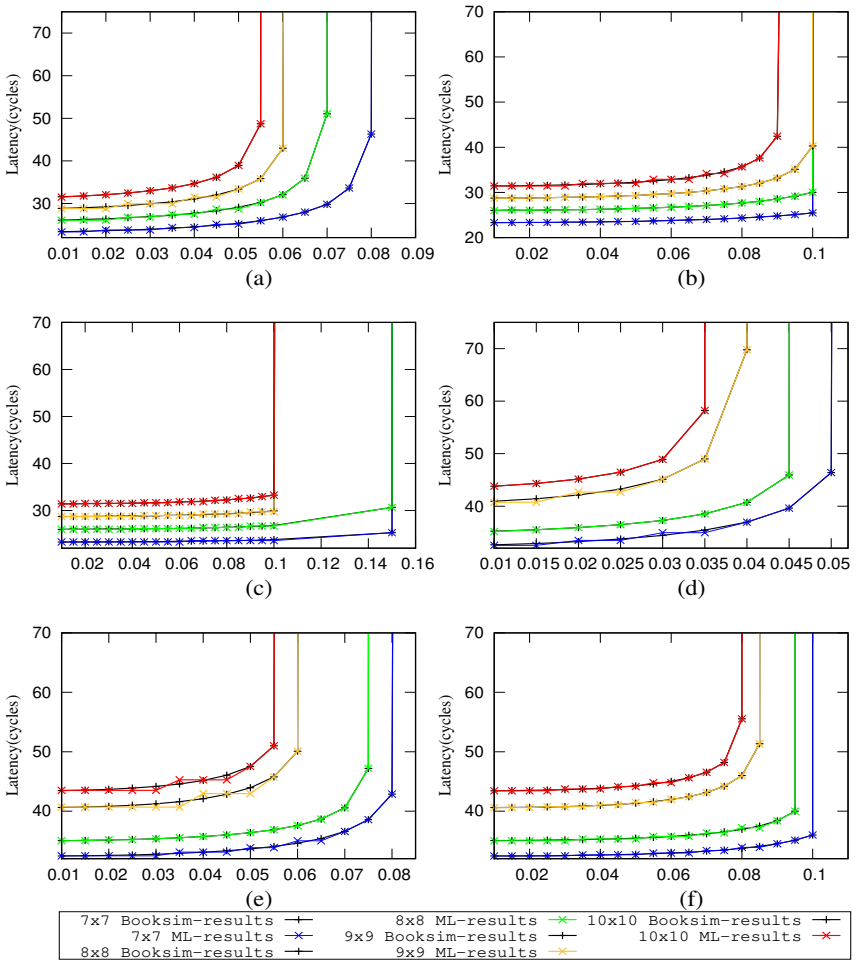


Figure 5.4: Latency comparison of K-NoC with Standard Booksim for Mesh with Uniform and Tornado traffic patterns.

Table 5.1: Saturation points of Standard and Modified Booksim Simulator for Different Architecture sizes with various Virtual Channels.

Saturation Points of Booksim Simulators for Different Virtual Channels												
Saturation points for Fixed-Delay simulator							Saturation points for Accurate-Delay simulator					
Topology	Uniform Traffic Pattern			Tornado Traffic Pattern			Uniform Traffic Pattern			Tornado Traffic Pattern		
Sizes	VC=2	VC=3	VC=4	VC=2	VC=3	VC=4	VC=2	VC=3	VC=4	VC=2	VC=3	VC=4
6x6	0.095	0.15	0.2	0.075	0.15	0.2	0.055	0.09	0.15	0.035	0.06	0.085
7x7	0.085	0.15	0.2	0.07	0.08	0.15	0.045	0.08	0.1	0.03	0.04	0.065
8x8	0.07	0.15	0.15	0.045	0.075	0.1	0.035	0.065	0.095	0.02	0.035	0.06
9x9	0.06	0.1	0.15	0.04	0.07	0.09	0.03	0.055	0.085	0.015	0.03	0.055
10x10	0.055	0.09	0.15	0.035	0.06	0.085	0.03	0.05	0.075	0.015	0.03	0.045

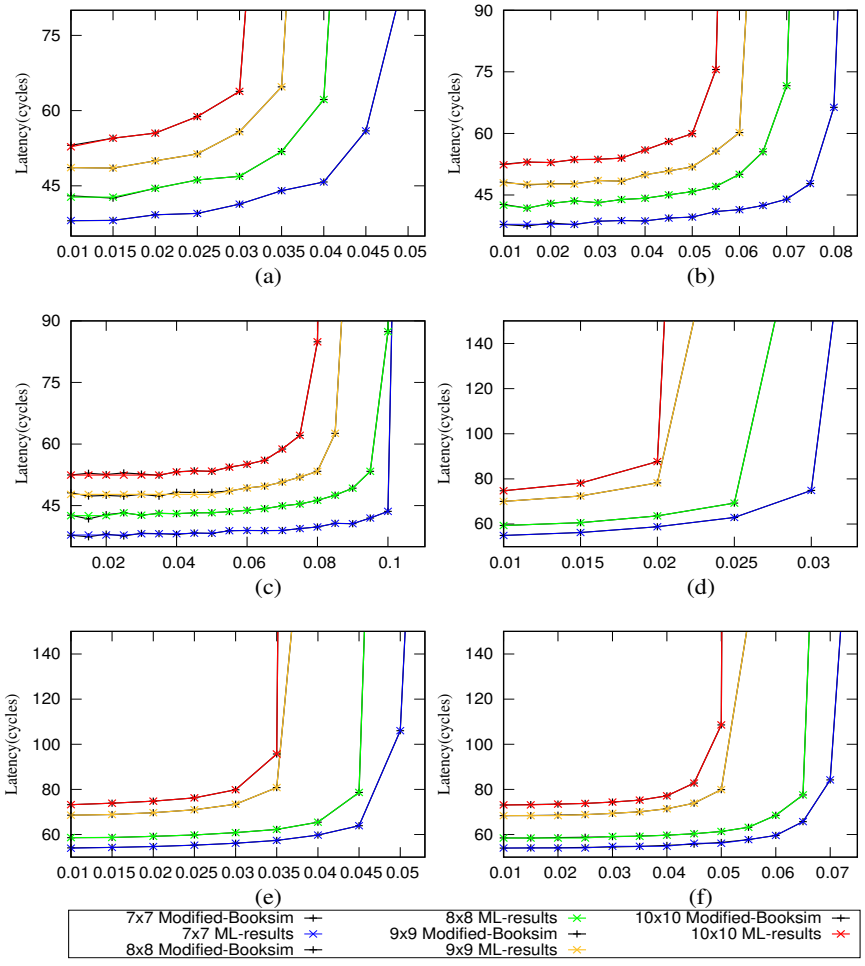


Figure 5.5: Latency comparison of K-NoC with Modified Booksim for Mesh with Uniform and Tornado traffic patterns.

observed that MAE, MSE, RMSE are close to 0 and R-squared close to 1. Hence, it specifies the K-NoC was able to predict the values which are very close to the actual results.

Table 5.2: Error Metrics of K-NoC for Latency values with Uniform and Tornado traffic patterns

Error Metrics of K-NoC for Booksim Simulators				
Metrics	Standard Booksim		Modified Booksim	
	Uniform	Tornado	Uniform	Tornado
MAE	0.0446	0.061	0.085	0.092
MSE	0.0084	0.019	0.032	0.38
RMSE	0.072	0.126	0.24	0.302
R-Squared	0.98	0.97	0.962	0.966

5.2.2 Predictions for Different Mesh NoC Architectures

In this section, results for different architecture sizes with various configurations are shown. Predictions have been made for all the parameters of NoC from 21×21 to 75×75 topology sizes for uniform and tornado traffic patterns for both standard and modified Booksim simulators.

Table 5.3 shows the different error metrics for standard Booksim simulator. All the values shown in the table are the average error rates of all the experiments conducted. Experiments have been demonstrated for all traffic patterns separately by varying input parameters. K-NoC was able to predict performance parameters with the error rate of 3% to 5%. In table NL (Network Latency), PL (Packet Latency) and FL (Flit Latency) specifies the mean of average and maximum latencies. For power parameters, the error rate was 4% to 6%. It showed an average accuracy considering all the parameters is around 95% to 97% for all the architectures.

Table 5.4 shows the different error metrics for modified Booksim simulator. All the values shown in the table are the average error rates of all the experiments conducted. Experiments have been demonstrated for all traffic patterns separately by varying input parameters. K-NoC was able to predict performance parameters with the error

Table 5.3: Error Metrics of K-NoC for NoC parameters of Standard Booksim Simulator

Error Metrics for Different NoC parameters with various Synthetic Traffic Patterns							
Traffic Patterns	**Error Metrics**	**Performance Parameters**				**Power Parameters**	
		NL	**PL**	**FL**	**AHC**	**RP**	**TP**
Uniform	MAE	0.12	0.21	0.12	0.048	0.010	0.049
	MSE	0.020	0.065	0.023	0.005	0.0001	0.004
	RMSE	0.15	0.25	0.15	0.07	0.013	0.07
	R-squared	0.99	0.99	0.99	0.99	0.95	0.978
Tornado	MAE	0.211	0.22	0.21	0.10	0.098	0.12
	MSE	0.086	0.085	0.087	0.014	0.016	0.02
	RMSE	0.29	0.3	0.294	0.11	0.12	0.024
	R-squared	0.99	0.99	0.99	0.99	0.95	0.974
Transpose		**NL**	**PL**	**FL**	**RP**	**TP**	
	MAE	0.09	0.089	0.068	0.03	0.18	
	MSE	0.015	0.016	0.007	0.002	0.11	
	RMSE	0.12	0.124	0.089	0.04	0.3	
	R-squared	0.99	0.99	0.998	0.90	0.94	
Shuffle	MAE	0.014	0.0144	0.0141	0.025	0.05	
	MSE	0.003	0.0034	0.00033	0.008	0.007	
	RMSE	0.018	0.017	0.0181	0.02	0.08	
	R-squared	0.99	0.99	0.99	0.90	0.95	
Bitcom	MAE	0.18	0.10	0.039	0.15	0.116	
	MSE	0.03	0.014	0.003	0.06	0.04	
	RMSE	0.19	0.11	0.05	0.25	0.2	
	R-squared	0.99	0.99	0.99	0.89	0.972	
Bitrev	MAE	0.75	0.08	0.094	0.16	0.101	
	MSE	0.008	0.016	0.0167	0.06	0.03	
	RMSE	0.09	0.12	0.129	0.26	0.17	
	R-squared	0.99	0.99	0.99	0.93	0.97	

rate of 4% to 6% and for power parameters, the error rate was 5% to 7%. It showed an average accuracy considering all the parameters is around 94% to 96% for NoC architectures. The error rates of router area and total area for Mesh topology are 1% to 2% respectively.

According to the configurations considered in this work, the individual simulator needs to simulated for around 10860 times to get the results of various NoC architec-

Table 5.4: Error Metrics of K-NoC for NoC parameters of Modified Booksim Simulator

Error Metrics for Different NoC parameters with various Synthetic Traffic Patterns							
Traffic Patterns	**Error Metrics**	**Performance Parameters**				**Power Parameters**	
		NL	PL	FL	AHC	RP	TP
Uniform	MAE	0.19	0.26	0.21	0.052	0.018	0.062
	MSE	0.026	0.074	0.031	0.006	0.0093	0.0086
	RMSE	0.21	0.3	0.26	0.082	0.014	0.081
	R-squared	0.98	0.98	0.98	0.99	0.95	0.963
Tornado	MAE	0.29	0.27	0.26	0.2	0.12	0.16
	MSE	0.091	0.093	0.0912	0.022	0.021	0.03
	RMSE	0.34	0.38	0.4	0.16	0.18	0.036
	R-squared	0.98	0.98	0.98	0.98	0.94	0.96
Transpose		NL	PL	FL	RP	TP	
	MAE	0.11	0.093	0.076	0.043	0.23	
	MSE	0.027	0.023	0.0091	0.0046	0.26	
	RMSE	0.23	0.27	0.093	0.08	0.41	
	R-squared	0.97	0.98	0.98	0.94	0.93	
Shuffle	MAE	0.027	0.019	0.026	0.034	0.061	
	MSE	0.008	0.0071	0.0023	0.0092	0.0082	
	RMSE	0.028	0.031	0.025	0.033	0.086	
	R-squared	0.98	0.98	0.98	0.93	0.96	
Bitcom	MAE	0.22	0.21	0.044	0.23	0.28	
	MSE	0.051	0.023	0.0041	0.071	0.061	
	RMSE	0.22	0.18	0.081	0.29	0.31	
	R-squared	0.98	0.98	0.98	0.92	0.96	
Bitrev	MAE	0.62	0.091	0.11	0.24	0.20	
	MSE	0.012	0.027	0.022	0.073	0.042	
	RMSE	0.093	0.18	0.19	0.33	0.29	
	R-squared	0.98	0.98	0.98	0.93	0.95	

tures. The advantage of using multiprocessing scheme in ML framework is that all 21720 simulation results can be obtained in a single run with good accuracy and with a minimum speedup of 12K×.

Timing comparisons have been made for the testing configurations considered. Table 5.5 shows the execution times of the Booksim simulators and the K-NoC for Mesh architectures. Total time taken to complete the execution of all architectural

sizes with different traffic patterns for standard Booksim simulator was $6.1*10^7$ seconds which is 706.63 days whereas K-NoC was able to predict same results in 4824.6 seconds which is 1.34 hours. Whereas modified Booksim simulator took to simulate all the architectures is $3.8*10^7$ seconds which is 440.1 days and K-NoC predicted the same results in 4530 seconds which is 1.26 hours. A minimum speedup of 12K× to 15K× speedup achieved over cycle-accurate simulator for both simulators considered.

Table 5.5: Timing Comparison of Simulation results with K-NoC for different Synthetic Traffic patterns

Traffic Patterns	**K-NoC Timings for Standard Booksim**	**Standard Booksim Simulation Timings**		**K-NoC Timings for Modified Booksim**	**Modified Booksim Simulation Timings**	
	(Seconds)	(Seconds)	(Hours)	(Seconds)	(Seconds)	(Hours)
Uniform	4824.6 (1.34 hours)	26955540	7487.65 (311.99 days)	4530 (1.26 hours)	19194120	5331.7 (222.15 days)
Tornado		33720300	9366.75 (390.28 days)		18818415	5227.34 (217.81 days)
Shuffle		167136	46.43		2932	0.81
Transpose		198411.36	55.11		2680.32	0.74
Bitrev		129648	36.01		3241	0.9
Bitcom		147456	40.96		3091	0.86

5.3 Summary

A unified framework named Knowledgeable Network-on-Chip Accelerator (K-NoC) has been proposed to predict the performance, power, and area parameters of Mesh topology for the fixed delay and accurate delay between IPs under various synthetic traffic patterns. Experiments were conducted to predict results in two different situations, first was predicting worst-case latency analysis and second was to predict the all output parameters like network latency (average, maximum), packet latency (average, maximum), flit latency (average, maximum), average hop count, switch area, total area, switch power and total power for Mesh topology. The overall accuracy of K-NoC for standard Booksim results is 95% to 97% has been observed and for modified Booksim results K-NoC predicted with the accuracy of 94% to 96%. K-NoC predicted the values of NoCs similar to the conventional cycle-accurate Booksim simulators while providing minimum 12K× speedup with an error rate less than 8%.

Chapter 6

Summary and Conclusions

NoCs became an integral part of modern many-core processors; NoCs research and development play a key role in the design of future large-scale architectures with hundreds to thousands. However, the lack of fast modelling methodologies that can provide a high degree of accuracy is a major obstacle to research and development of large-scale NoCs. This dissertation proposed frameworks to resolve the problem of simulation speed, thus allowing for fast and accurate simulation of NoCs with up to thousands of nodes.

This thesis summaries the research work is done so far. Firstly, a detailed study of Booksim NoC simulator was done as it is one of the most widely used simulator in NoC community. Simulation time increases as the resources and size of the architecture increases. To overcome this problem, a highly parameterized machine learning framework named Learning-Based Framework (LBF-NoC) was proposed which can be used to predict the performance, power and area parameters of direct (Mesh, Torus, Cmesh), indirect(Fat-Tree, Flatfly) NoC architectures. Analysis of different regression algorithms namely artificial neural networks with identity and relu activation functions, different generalized linear regression algorithms, thatis lasso, lasso-lars, larsCV, bayesian-ridge, linear, ridge, elastic-net and support vector regression (SVR) with linear, radial basis function, polynomial kernels have been made while building the framework. Among all the regression algorithms, the lowest error rate was observed in the SVR algorithm. Hence, the SVR algorithm has been considered in LBF-NoC. It predicts the values of NoCs similar to conventional cycle-accurate Booksim simulator while providing a 5000× speedup with 5% to 6% error.

To get an efficient NoC design, chip designers need to simulate the simulator or

the proposed framework with various configurations. To overcome this problem multiprocessing scheme was added along with machine learning framework such that all combinations of considered NoC configurations can be executed in a simultaneously.

Another framework was proposed named Ensemble Learning-Based Accelerator (ELBA-NoC) which was used to predict parameters in two different scenarios. In the first scenario, it is used to predict the worst-case latency analysis, as injection rate is one of the critical factors which affects the efficiency of the NoC communication. In the second scenario, it predicts the complete design space exploration of five different architectures considering both 2D and 3D NoCs. ELBA-NoC was tested with above-mentioned regression algorithms and failed to predict the latency values when the network enters the saturation region. Random Forest algorithm was able to predict the latency values as the network reaches the saturation region and the same RF algorithm was used to predict the NoC parameters as it worked efficiently than SVR. The overall accuracy of 94% to 96% has been observed for both 2D and 3D NoC architectures. ELBA-NoC predicts the values of NoCs similar to the conventional cycle-accurate Booksim simulator while providing a minimum 16K× speedup with 4% to 6% error rate.

A modified Booksim simulator was adopted as it had an accurate delay between the nodes and the standard Booksim has a fixed delay between the nodes. A unified framework was proposed which predicts the complete design space exploration of two simulators. The overall accuracy of the proposed framework for standard Booksim results is 95% to 97% and for modified Booksim results it is 94% to 96%. A minimum 12K× speedup with error rate less than 6% was achieved.

The future work focus on creating a generalized framework which can be used to predict all 2D and 3D topologies by considering other NoC simulators. Evaluating the machine learning framework with real time benchmarks.

Bibliography

Sriram Vangal. *Performance and Energy Efficient Network-on-Chip Architectures.* PhD thesis, Institutionen för systemteknik, 2007.

Boris Grot, Joel Hestness, Stephen W Keckler, and Onur Mutlu. Express cube topologies for on-chip interconnects. In *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, pages 163–174. IEEE, 2009.

Partha Pratim Pande, Cristian Grecu, Michael Jones, Andre Ivanov, and Resve Saleh. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE transactions on Computers*, 54(8):1025–1040, 2005.

Robert R Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6): 52–59, 1997.

John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach.* Elsevier, 2011.

William J Dally and Brian Towles. Route packets, not wires: on-chip inteconnection networks. In *Proceedings of the 38th annual Design Automation Conference*, pages 684–689, 2001. ACM, 2001.

Luca Benini and Giovanni De Micheli. Networks on chips: A new soc paradigm. *computer*, 35(1):70–78, 2002.

Kangmin Lee, Se-Joong Lee, and Hoi-Jun Yoo. Low-power network-on-chip for high-performance soc design. *IEEE transactions on very large scale integration (vlsi) systems*, 14(2):148–160, 2006.

Sriram R Vangal, Jason Howard, Gregory Ruhl, Saurabh Dighe, Howard Wilson, James Tschanz, David Finan, Arvind Singh, Tiju Jacob, Shailendra Jain, et al. An

80-tile sub-100-w teraflops processor in 65-nm cmos. *IEEE Journal of solid-state circuits*, 43(1):29–41, 2008.

Jason Howard, Saurabh Dighe, Yatin Hoskote, Sriram Vangal, David Finan, Gregory Ruhl, David Jenkins, Howard Wilson, Nitin Borkar, Gerhard Schrom, et al. A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. In *2010 IEEE International Solid-State Circuits Conference-(ISSCC)*, pages 108–109. IEEE, 2010.

Shane Bell, Bruce Edwards, John Amann, Rich Conlin, Kevin Joyce, Vince Leung, John MacKay, Mike Reif, Liewei Bao, John Brown, et al. Tile64-processor: A 64-core soc with mesh interconnect. In *2008 IEEE International Solid-State Circuits Conference-Digest of Technical Papers*, pages 88–598. IEEE, 2008.

Shekhar Borkar. Thousand core chips: a technology perspective. In *Proceedings of the 44th annual design automation conference*, pages 746–749, 2007.

Shekhar Borkar. Future of interconnect fabric: a contrarian view. In *Proceedings of the 12th ACM/IEEE international workshop on System level interconnect prediction*, pages 1–2, 2010.

Yatin Hoskote, Sriram Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar. A 5-ghz mesh interconnect for a teraflops processor. *IEEE Micro*, 27(5):51–61, 2007.

Michael Bedford Taylor, Jason Kim, Jason Miller, David Wentzlaff, Fae Ghodrat, Ben Greenwald, Henry Hoffman, Paul Johnson, Jae-Wook Lee, Walter Lee, et al. The raw microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE micro*, 22(2):25–35, 2002.

Li-Shiuan Peh and William J Dally. A delay model for router microarchitectures. *IEEE Micro*, 21(1):26–34, 2001.

Umit Y Ogras, Paul Bogdan, and Radu Marculescu. An analytical approach for network-on-chip performance analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(12):2001–2013, 2010.

Bheemappa Halavar, Ujjwal Pasupulety, and Basavaraj Talawar. Extending booksim2. 0 and hotspot6. 0 for power, performance and thermal evaluation of 3d noc architectures. *Simulation Modelling Practice and Theory*, 96:101929, 2019.

Ahmed Ben Achballah and Slim Ben Saoud. A survey of network-on-chip tools. *arXiv preprint arXiv:1312.2976*, 2013.

Nan Jiang, James Balfour, Daniel U Becker, Brian Towles, William J Dally, George Michelogiannakis, and John Kim. A detailed and flexible cycle-accurate network-on-chip simulator. In *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, pages 86–96. IEEE, 2013.

Vincenzo Catania, Andrea Mineo, Salvatore Monteleone, Maurizio Palesi, and Davide Patti. Noxim: An open, extensible and cycle-accurate network on chip simulator. In *26th IEEE International Conference on Application-specific Systems, Architectures and Processors, ASAP 2015, Toronto, ON, Canada, July 27-29, 2015*, pages 162–163, 2015, 2015. doi: 10.1109/ASAP.2015.7245728.

Anh T Tran and Bevan Baas. Noctweak: a highly parameterizable simulator for early exploration of performance and energy of networks on-chip. *VLSI Computation Lab, ECE Department, University of California, Davis, Tech. Rep. ECE-VCL-2012-2*, 2012.

Pablo Abad, Pablo Prieto, Lucia G Menezo, Valentin Puente, José-Ángel Gregorio, et al. Topaz: An open-source interconnection network simulator for chip multiprocessors and supercomputers. In *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, pages 99–106. IEEE, 2012.

Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 simulator. *ACM SIGARCH computer architecture news*, 39(2):1–7, 2011.

Andrew B Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration.

In *2009 Design, Automation & Test in Europe Conference & Exhibition*, pages 423–428. IEEE, 2009.

Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *2009 IEEE international symposium on performance analysis of systems and software*, pages 33–42. IEEE, 2009a.

Anil Kumar and Basavaraj Talawar. Machine learning based framework to predict performance evaluation of on-chip networks. In *2018 Eleventh International Conference on Contemporary Computing (IC3)*, pages 1–6. IEEE, 2018.

Hari Angepat, Derek Chiou, Eric S Chung, and James C Hoe. Fpga-accelerated simulation of computer systems. *Synthesis Lectures on Computer Architecture*, 9 (2):1–80, 2014.

Danyao Wang, Charles Lo, Jasmina Vasiljevic, Natalie Enright Jerger, and J Gregory Steffan. Dart: A programmable architecture for noc simulation on fpgas. *IEEE Transactions on Computers*, 63(3):664–678, 2012.

Qi Guo, Tianshi Chen, Yunji Chen, and Franz Franchetti. Accelerating architectural simulation via statistical techniques: A survey. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(3):433–446, 2015.

Thiem Van Chu, Shimpei Sato, and Kenji Kise. Fast and cycle-accurate emulation of large-scale networks-on-chip using a single fpga. *ACM Trans. Reconfigurable Technol. Syst.*, 10(4), December 2017. ISSN 1936-7406. doi: 10.1145/3151758. URL https://doi.org/10.1145/3151758.

Thiem Van Chu. *Ultra-Fast and Accurate Simulation for Large-Scale Many-Core Processors*. PhD thesis, Tokyo Institute of Technology, 2015.

Thiem Van Chu, Shimpei Sato, and Kenji Kise. Enabling fast and accurate emulation of large-scale network on chip architectures on a single fpga. In *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 60–63. IEEE, 2015.

Daniel Sanchez and Christos Kozyrakis. Zsim: Fast and accurate microarchitectural simulation of thousand-core systems. *ACM SIGARCH Computer architecture news*, 41(3):475–486, 2013.

George Kurian, Jason E Miller, James Psota, Jonathan Eastep, Jifeng Liu, Jurgen Michel, Lionel C Kimerling, and Anant Agarwal. Atac: A 1000-core cache-coherent processor with on-chip optical network. In *2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, pages 477–488. IEEE, 2010.

Axel Jantsch, Hannu Tenhunen, et al. *Networks on chip*, volume 396. Springer, 2003.

Jose Duato, Sudhakar Yalamanchili, and Lionel Ni. Interconnection networks: An engineering approach, m. kaufmann pub. *Inc., USA*, 2002.

Mahmoud Moadeli, Partha Maji, and Wim Vanderbauwhede. Quarc: A high-efficiency network on-chip architecture. In *2009 International Conference on Advanced Information Networking and Applications*, pages 98–105. IEEE, 2009.

Axel Jantsch, Robert Lauter, and Arseni Vitkowski. Power analysis of link level and end-to-end data protection in networks on chip. In *2005 IEEE International Symposium on Circuits and Systems*, pages 1770–1773. IEEE, 2005.

Érika Cota, Alexandre de Morais Amory, and Marcelo Soares Lubaszewski. *Reliability, Availability and Serviceability of Networks-on-chip*. Springer Science & Business Media, 2011.

Jose Duato, Sudhakar Yalamanchili, and Lionel Ni. *Interconnection networks*. Morgan Kaufmann, 2003.

Sriram Vangal, Jason Howard, Gregory Ruhl, Saurabh Dighe, Howard Wilson, James Tschanz, David Finan, Priya Iyer, Arvind Singh, Tiju Jacob, et al. An 80-tile 1.28 tflops network-on-chip in 65nm cmos. In *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pages 98–589. IEEE, 2007.

James Balfour and William J Dally. Design tradeoffs for tiled cmp on-chip networks. In

ACM International conference on supercomputing 25th anniversary volume, pages 390–401, 2006.

Jesus Camacho, José Flich, Antoni Roca, and José Duato. Pc-mesh: A dynamic parallel concentrated mesh. In *2011 International Conference on Parallel Processing*, pages 642–651. IEEE, 2011.

Abdul Quaiyum Ansari, Mohammad Rashid Ansari, and Mohammad Ayoub Khan. Performance evaluation of various parameters of network-on-chip (noc) for different topologies. In *2015 annual IEEE India conference (INDICON)*, pages 1–4. IEEE, 2015.

John Kim, William J Dally, and Dennis Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *Proceedings of the 34th annual international symposium on Computer architecture*, pages 126–137, 2007.

Natalie Enright Jerger and Li-Shiuan Peh. On-chip networks, synthesis lectures on computer architecture. *Morgan & cLaypool publishers*, 2009.

Tobias Bjerregaard and Shankar Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys (CSUR)*, 38(1):1–es, 2006.

Ankur Agarwal, Cyril Iskander, and Ravi Shankar. Survey of network on chip (noc) architectures & contributions. *Journal of engineering, Computing and Architecture*, 3(1):21–27, 2009b.

Wang Zhang, Ligang Hou, Jinhui Wang, Shuqin Geng, and Wuchen Wu. Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip. In *2009 WRI Global Congress on Intelligent Systems*, volume 3, pages 329–333. IEEE, 2009.

Brett Stanley Feero and Partha Pratim Pande. Networks-on-chip in a three-dimensional environment: A performance evaluation. *IEEE Transactions on computers*, 58(1):32–45, 2008.

Yuan Xie, Gabriel H Loh, Bryan Black, and Kerry Bernstein. Design space exploration

for 3d architectures. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2(2):65–103, 2006.

Anna W Topol, DC La Tulipe, Leathen Shi, David J Frank, Kerry Bernstein, Steven E Steen, Arvind Kumar, Gilbert U Singco, Albert M Young, Kathryn W Guarini, et al. Three-dimensional integrated circuits. *IBM Journal of Research and Development*, 50(4.5):491–506, 2006.

Christopher M Bishop. Pattern recognition and machine learning (information science and statistics), 2007.

Fahimeh Farahnakian, Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, and Juha Plosila. Bi-lcq: A low-weight clustering-based q-learning approach for nocs. *Microprocessors and Microsystems*, 38(1):64–75, 2014.

Kwangok Jeong, Andrew B Kahng, Bill Lin, and Kambiz Samadi. Accurate machine-learning-based on-chip router modeling. *IEEE Embedded Systems Letters*, 2(3): 62–66, 2010.

Andrew B Kahng, Bill Lin, and Kambiz Samadi. Improved on-chip router analytical power and area modeling. In *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 241–246. IEEE, 2010.

Masoumeh Ebrahimi, Masoud Daneshtalab, Fahimeh Farahnakian, Juha Plosila, Pasi Liljeberg, Maurizio Palesi, and Hannu Tenhunen. Haraq: Congestion-aware learning model for highly adaptive routing algorithm in on-chip networks. In *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, pages 19–26. IEEE, 2012.

Dominic DiTomaso, Avinash Karanth Kodi, Ahmed Louri, and Razvan Bunescu. Resilient and power-efficient multi-function channel buffers in network-on-chip architectures. *IEEE Transactions on Computers*, 64(12):3555–3568, 2015.

Yuho Jin, Eun Jung Kim, and Timothy Mark Pinkston. Communication-aware globally-coordinated on-chip networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(2):242–254, 2011.

Adesh Kumar, Paawan Sharma, Mukul Kumar Gupta, and Roushan Kumar. Machine learning based resource utilization and pre-estimation for network on chip (noc) communication. *Wireless Personal Communications*, 102(3):2211–2231, 2018.

Fahimeh Farahnakian, Masoumeh Ebrahimi, Masoud Daneshtalab, Juha Plosila, and Pasi Liljeberg. Adaptive reinforcement learning method for networks-on-chip. In *2012 International Conference on Embedded Computer Systems (SAMOS)*, pages 236–243. IEEE, 2012.

Luca P Carloni, Andrew B Kahng, Swamy V Muddu, Alessandro Pinto, Kambiz Samadi, and Puneet Sharma. Accurate predictive interconnect modeling for system-level design. *IEEE transactions on very large scale integration (VLSI) systems*, 18 (4):679–684, 2009.

Alessandro Pinto, LP Carloni, and AL Sangiovanni-Vincentelli. A methodology and an open software infrastructure for constraint-driven synthesis of on-chip communications. Technical report, Technical Report, 2007.

Chaochao Feng, Zhonghai Lu, Axel Jantsch, Jinwen Li, and Minxuan Zhang. A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip. In *Proceedings of the Third International Workshop on Network on Chip Architectures*, pages 11–16, 2010.

Dominic DiTomaso, Ashif Sikder, Avinash Kodi, and Ahmed Louri. Machine learning enabled power-aware network-on-chip design. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1354–1359. IEEE, 2017.

Zhiliang Qian, Da-Cheng Juan, Paul Bogdan, Chi-Ying Tsui, Diana Marculescu, and Radu Marculescu. Svr-noc: A performance analysis tool for network-on-chips using learning-based support vector regression model. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 354–357. IEEE, 2013.

Zhi-Liang Qian, Da-Cheng Juan, Paul Bogdan, Chi-Ying Tsui, Diana Marculescu, and Radu Marculescu. A support vector regression (svr)-based latency model for network-on-chip (noc) architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(3):471–484, 2015.

Sourav Das, Janardhan Rao Doppa, Partha Pratim Pande, and Krishnendu Chakrabarty. Design-space exploration and optimization of an energy-efficient and reliable 3-d small-world network-on-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(5):719–732, 2016.

Justin Boyan and Andrew W Moore. Learning evaluation functions to improve optimization by local search. *Journal of Machine Learning Research*, 1(Nov):77–112, 2000.

Sung Joo Park, Bumhee Bae, Joungho Kim, and Madhavan Swaminathan. Application of machine learning for optimization of 3-d integrated circuits and systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(6):1856–1865, 2017.

Jieming Yin, Yasuko Eckert, Shuai Che, Mark Oskin, and G Loh. Toward more efficient noc arbitration: A deep reinforcement learning approach. In *Proceedings of the 1st International Workshop on AI-assisted Design for Architecture (AIDArc)*, 2018.

Mahnaz Rafie, Ahmad Khademzadeh, and Midia Reshadi. Performance improvement of application-specific network on chip using machine learning algorithms. *International Journal of High Performance Systems Architecture*, 5(2):71–83, 2014.

Jih-Sheng Shen, Pao-Ann Hsiung, and Chun-Hsian Huang. Learning-based adaptation to applications and environments in a reconfigurable network-on-chip for reducing crosstalk and dynamic power consumption. *Computers & Electrical Engineering*, 39(2):453–464, 2013.

Suleyman Tosun. Cluster-based application mapping method for network-on-chip. *Advances in Engineering Software*, 42(10):868–874, 2011.

Mohadaseh Zaman Dageleh and Mohammad Ali Jabraeil Jamali. V-castnet3d: A novel clustering-based mapping in 3-d network on chip. *Nano communication networks*, 18:51–61, 2018.

A Aravindhan, S Salini, and G Lakshminarayanan. Cluster based application mapping strategy for 2d noc. *Procedia Technology*, 25:505–512, 2016.

Fahimeh Farahnakian, Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, and Juha Plosila. Q-learning based congestion-aware routing algorithm for on-chip network. In *2011 IEEE 2nd International Conference on Networked Embedded Systems for Enterprise Applications*, pages 1–7. IEEE, 2011.

Da-Cheng Juan and Diana Marculescu. Power-aware performance increase via core/uncore reinforcement control for chip-multiprocessors. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, pages 97–102, 2012.

Nicolas Genko, David Atienza, Giovanni De Micheli, Jose Manuel Mendias, Roman Hermida, and Francky Catthoor. A complete network-on-chip emulation framework. In *Design, Automation and Test in Europe*, pages 246–251. IEEE, 2005.

Pascal T Wolkotte, Philip KF Holzenspies, and Gerard JM Smit. Fast, accurate and detailed noc simulations. In *First International Symposium on Networks-on-Chip (NOCS'07)*, pages 323–332. IEEE, 2007.

Yana E Krasteva, Francisco Criado, Eduardo de la Torre, and Teresa Riesgo. A fast emulation-based noc prototyping framework. In *2008 International Conference on Reconfigurable Computing and FPGAs*, pages 211–216. IEEE, 2008.

Swapnil Lotlikar, Vinayak Pai, and Paul V Gratz. Acenocs: A configurable hw/sw platform for fpga accelerated noc emulation. In *2011 24th Internatioal Conference on VLSI Design*, pages 147–152. IEEE, 2011.

Michael K Papamichael. Fast scalable fpga-based network-on-chip simulation models. In *Ninth ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMPCODE2011)*, pages 77–82. IEEE, 2011.

Michael K Papamichael, James C Hoe, and Onur Mutlu. Fist: A fast, lightweight, fpga-friendly packet latency estimator for noc modeling in full-system simulations. In *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip*, pages 137–144, 2011.

Tobias Drewes, Jan Moritz Joseph, and Thilo Pionteck. An fpga-based prototyping framework for networks-on-chip. In *2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–7. IEEE, 2017.

Hadi Mardani Kamali and Shahin Hessabi. Adapnoc: A fast and flexible fpga-based noc simulator. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–8. IEEE, 2016.

Hadi Mardani Kamali, Kimia Zamiri Azar, and Shaahin Hessabi. Ducnoc: A high-throughput fpga-based noc simulator using dual-clock lightweight router micro-architecture. *IEEE Transactions on Computers*, 67(2):208–221, 2017.

William James Dally and Brian Patrick Towles. *Principles and practices of interconnection networks*. Elsevier, 2004.

Nan Jiang. *Booksim Simulator*, (accessed 2012). http://nocs.stanford.edu/booksim.html.

Nello Cristianini and John Shawe-Taylor. An introduction to support vector machines, 2000.

Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, August 2004. ISSN 0960-3174. doi: 10.1023/B:STCO.0000035301.49549.88. URL http://dx.doi.org/10.1023/B:STCO.0000035301.49549.88.

Satar Mahdevari, Kourosh Shahriar, Saffet Yagiz, and Mohsen Akbarpour Shirazi. A support vector regression model for predicting tunnel boring machine penetration rates. *International Journal of Rock Mechanics and Mining Sciences*, 72:214–229, 2014.

Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

Max Kuhn and Kjell Johnson. *Applied predictive modeling*, volume 26. Springer, 2013.

Olivier Chapelle and Vladimir Vapnik. Model selection for support vector machines. In *Advances in neural information processing systems*, pages 230–236, 2000, 2000.

B Schölkopf, P Bartlett, A Smola, and R Williamson. Support vector regression with automatic accuracy control. In *ICANN 98*, pages 111–116. Springer, 1998.

Bernhard Schölkopf, Alex J Smola, Robert C Williamson, and Peter L Bartlett. New support vector algorithms. *Neural computation*, 12(5):1207–1245, 2000.

Sandra Arlinghaus. *Practical handbook of curve fitting*. CRC press, 1994.

Zhi-Liang Qian, Da-Cheng Juan, Paul Bogdan, Chi-Ying Tsui, Diana Marculescu, and Radu Marculescu. A support vector regression (svr)-based latency model for network-on-chip (noc) architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(3):471–484, 2016.

Michael M McKerns, Leif Strand, Tim Sullivan, Alta Fang, and Michael AG Aivazis. Building a framework for predictive science. *arXiv preprint arXiv:1202.1056*, 2012.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

Bheemappa Halavar and Basavaraj Talawar. Accurate performance analysis of 3d mesh network on chip architectures. In *2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 1–6. IEEE, 2018.

List of Publications

Journal Publications

1. Anil Kumar and Basavaraj Talawar, LBF-NoC: Learning-Based Framework to Predict Performance, Power and Area for Network-on-Chip Architectures, Journal of Circuits, Systems, and Computers, World Scientific. **[Revision Submitted]**

2. Anil Kumar and Basavaraj Talawar, *Estimating Design Decision Parameters of 2D & 3D Network-on-Chip Architectures using Multiprocessing Regression Framework*, Arabian Journal for Science and Engineering **[Submitted]**

3. Anil Kumar and Basavaraj Talawar, *ELBA-NoC: Ensemble Learning-Based Accelerator for 2D & 3D Network-on-Chip Architectures*, International Journal of Computational Science and Engineering, 2020 Vol.23 No.4, pp.319-335 DOI: 10.1504/IJCSE.2020.113176

4. Anil Kumar and Basavaraj Talawar, *Knowledgeable Network-on-Chip Accelerator for Fast and Accurate simulations using Supervised Learning Algorithms and Multiprocessing*, International Journal of Intelligent Engineering Informatics **[Revision Submitted].**

Conference Publications

1. Anil Kumar and Basavaraj Talawar, *Machine Learning Based Framework to Predict Performance Evaluation of On-Chip Networks*, 2018 Eleventh International Conference on Contemporary Computing (IC3), Aug. 2018, Noida, India.

2. Anil Kumar and Basavaraj Talawar, *UPM-NoC: Learning Based Framework to Predict Performance Parameters of Mesh Architecture in On-Chip Networks*, Intelligent Computing Techniques for Smart Energy Systems (ICTSES 18), Dec-2018, India.

3. Anil Kumar and Basavaraj Talawar, *Floorplan Based Performance Estimation of Network-on-Chips using Regression Techniques*, 2019 IEEE 5th International Conference for Convergence in Technology (I2CT), Mar-2019, India.

4. Anil Kumar and Basavaraj Talawar, *Accurate Router Level Estimation of Network-on-Chip Architectures using Learning Algorithms*, 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT) Nov-2019, India.

5. Anil Kumar and Basavaraj Talawar, *A Support Vector Regression-Based Approach to Predict the Performance of 2D & 3D On-Chip Communication Architectures*, 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT) Nov-2019, India.

6.1 Brief Bio-Data

Personal Details

Name - Anil Kumar

Date of Birth - 21 May 1989

Work Address

Anil Kumar

Research Scholar, Department of CSE,

NITK Surathkal, Mangalore,

Karnataka, 575 025.

Email: anilkumar.nitk1@gmail.com

Permanent Address

Anil Kumar

#1-12-75, Daddy Colony

Anand Sadan, Raichur

Karnataka 584101.

Phone No: +91 (886)7732 2505

Qualification

- M.Tech in Computer Science and Engineering from VTU University, Belgavi, Karnataka, India (2012-2014)
- B.E in Information Science and Engineering from VTU University, Belgavi, Karnataka, India (2009-2012)