

Final Report

|  |
| --- |
| School of Computer Science  Faculty of Engineering AND PHYSICAL SCIENCES |

Discrete-event Simulation of Routerless Network-on-Chip

Muhammad Ihtisyam Bin Shahrulnizam

Submitted in accordance with the requirements for the degree of  
Meng Computer Science

**2024/25**

**COMP3931 Individual Project**

The candidate confirms that the following have been submitted*:*

|  |  |  |
| --- | --- | --- |
| **Items** | **Format** | **Recipient(s) and Date** |
| *Final Report* | *PDF file* | *Uploaded to Minerva (DD/MM/YY)* |
| *Scanned participant consent forms* | *PDF file / file archive* | *Uploaded to Minerva (DD/MM/YY)* |
| *Link to online code repository* | *URL* | *Sent to supervisor and assessor (DD/MM/YY)* |
| *User manuals* | *PDF* | *Sent to client and supervisor (DD/MM/YY)* |

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)

© <Year of Submission> The University of Leeds and <full name of candidate>

# Summary

*A concise overview of the problem statement regarding simulation routerless NoC simulation, methodology and key findings.*

# Acknowledgements

*Acknowledgements to supervisor and others who provided guidance*

Table of Contents

[Summary iii](#_Toc193849142)

[Acknowledgements iv](#_Toc193849143)

[Table of Contents (example of how to format) v](#_Toc193849144)

[Chapter 1 Introduction and Background Research 1](#_Toc193849145)

[.1 Introduction 1](#_Toc193849146)

[.2 Background Research and Literature Review 1](#_Toc193849147)

[.2.1 Evolution of Network-on-Chip 1](#_Toc193849148)

[.2.2 Routerless NoC Design Principles 1](#_Toc193849149)

[.2.3 Discrete-event Simulation 1](#_Toc193849150)

[.2.4 Performance Metrics and Evaluation Methods 1](#_Toc193849151)

[Chapter 2 Methodology 2](#_Toc193849152)

[2.1 Simulation Framework Design 2](#_Toc193849153)

[2.2 Routerless NoC Implementation 2](#_Toc193849154)

[2.3 Traffic Generation 2](#_Toc193849155)

[2.4 Validation Strategy 2](#_Toc193849156)

[Chapter 3 Implementation and Validation 3](#_Toc193849157)

[3.1 Implementation Overview 3](#_Toc193849158)

[3.2 Routerless NoC Implementation 3](#_Toc193849159)

[3.3 Performance Monitoring and Statistics Collection 3](#_Toc193849160)

[3.4 Testing and Validation 3](#_Toc193849161)

[Chapter 4 Results, Evaluation and Discussion 4](#_Toc193849162)

[4.1 Experimental Setup 4](#_Toc193849163)

[4.2 Performance Results 4](#_Toc193849164)

[4.3 Comparative Analysis 4](#_Toc193849165)

[4.4 Discussion 4](#_Toc193849166)

[4.5 Evaluation 4](#_Toc193849167)

[4.6 Conclusions 4](#_Toc193849168)

[4.7 Future Work 5](#_Toc193849169)

[List of References 6](#_Toc193849170)

[Appendix A Self-appraisal 7](#_Toc193849171)

[A.1 Critical self-evaluation 7](#_Toc193849172)

[A.2 Personal reﬂection and lessons learned 7](#_Toc193849173)

[A.3 Legal, social, ethical and professional issues 7](#_Toc193849174)

[A.3.1 Legal issues 7](#_Toc193849175)

[A.3.2 Social issues 7](#_Toc193849176)

[A.3.3 Ethical issues 7](#_Toc193849177)

[A.3.4 Professional issues 7](#_Toc193849178)

[Appendix B External Materials 8](#_Toc193849179)

# Chapter 1 Introduction and Background Research

## 1.1 Introduction

As the chip technology experiences rapid growth of the integrated circuit number on the chips, the application of high performance on-chip communication has become a necessity. Bus-based systems were not recommended to be used for the application of dozens or hundreds of cores which is the reason Network-on-Chip (NoC) architectures have been developed. They do this, for instance, by presenting not only the most efficient data integration at the processor level but also a parallel data transfer electrical path that is provided without increasing the power consumption or thermal state of the device.

Additionally, traditional routers are located below the programmable switches that interconnect the cores. Each switch can be controlled allowing the routers to select the best routing paths for the packets of data that are directed to each node. The routing flexibility and built-in redundancy are said to be the advantages of this architecture. Nonetheless, the additional complexity introduced to the design with router logic is seen in the area and power overhead it generates. Router logic could be responsible for more than 30-40% of the total power consumed in some designs.

The current paper is concerned with the use of an inventive input-output free Network-on-Chip (NoC) architecture. Input free NoCs get rid of an unnecessary complex router and instead simple loop-based routing structure replaces them, on this way, their hardware overhead is significantly lowered while the performance of the network is still acceptable. The design achieves data transportation through the cycles or "loops" of the packets are traversing, which were already set in advance and they are going to be moved through. Thus, the thing is that NoC without dedicated routers have the potential not only for better power, silicon area, and design management but also at least throughputs and latency performance metrics.

In this research, the implementation methodology is a discrete-event simulation framework that serves the purpose of creating models and evaluating routerless NoC performance through different network configurations, traffic patterns and injection rates. The simulation environment permits the analysis of packets as they traverse the network and measures essential performance metrics like packet latency, throughput and link utilisation. The research findings serve to add on to the collection of studies on efficient on-chip interconnect design and therefore they form the basis for the future hardware implementations of communication infrastructures in multi core complex systems.

## Background Research

A rapid transformation in computer systems, from multi core chips to many core architectures, was a direct outcome of the parallel computing revolution. Having gone from the birth of the need for high speed interconnection between processing elements to the invention of routerless approaches, it is by these steps that the process is explained and justified.

### 1.2.1 Evolution of Network-on-Chip

The shared bus structures were initially viewed as the solution to the issue of on-chip communication. In this case, all the processing elements use a single communication channel which they share and take alternative turns to send data. While this is easy to implement, however, the design incurs serious scalability problems like the case of the "Mason" bottleneck [Mason 1975] since it creates bandwidth bottlenecks and contention problems as the number of elements increases.

The other choice was to use the point-to-point interconnections by which the processing elements were connected through different paths. This way, the processors achieve a beneficial high bandwidth and low latency directly connected components outcome but it hits reality by the withdrawing number of cores because of the connection’s quadratic growth.

A novelty in the case of Network-on Chip designs would come from large network technologies in spite of the fact that they should reflect their connection to that large network. Accordingly, they build on-chip interconnections to a network level with routers and links which is scalable and can accommodate dozens or hundreds of cores. The NoC majority is either grid or torus. In many cases, a node is a router that controls incoming traffic using dynamic routing. Thus the system is flexible and redundant. The downside of it is that the router logic is complicated and may be very high power consumer. It is even possible for it to consume up to 30-40% in particular cases.

### 1.2.2 Routerless NoC Concept

The routerless NoC logic is against the regular one in the design path. To be specific, throughtout the network nodes there are no specific routers. Instead, the data follows the predetermined cycles or loops which are equivalent of joining nodes that are two way. Therefore, there are various paths between the origin and the destination. In a no-router architecture, the packets are injected into the assigned loops that they will eventually traverse through it and so packets do not hop by routing, they follow the loop assigned to them until they reachthe destination where they exits the network. The existence of this effect results in the simplification of the amount of hardware that is needed at each node. Instead of the complex routing logic where there are only injection and ejection mechanisms which are simple.

The main features of routerless NoCs’ architectures are :

* Loops : Predefined cyclic paths connecting multiple nodes
* Injection mechanisms : Hardware that inserts packets into appropriate loops
* Ejection mechanisms : Hardware that extracts packets when they reach their destination
* Extended Bufferd (EXB) : Storage elements that may be used to handle contetion when multiple packets compete for the same loop resources

On a side note, this approach is beautifully simple as it reduces the hardware requirement and power consumption and it mught run at a higher frequency as the critical path is simpler. But it faces with some challenges like loop selection, avoiding deadlocks and managing the network congestion.

## Related Work and Literature Review

As designers seek alternatives to the traditional router based model, research routerless and loop based NoC architectures has become the mainstream definitely in the past few years. This part shows the main contributions to them and the former attemps to solve similar problems.

### Conventional Router Based Network-on-Chip

NoC studies before were mostly applying router microarchitecture optimisation interventions, availing efficient routing algorithms and looking at several netwrok topologies. The major part of the studies concerning NoC design principles was initiated by Dally and Towles who were also the very first to introduce the ideas, wormhole routing and virtual channels that most systems today consider standard. Preceding research advanced these ideas by incorporating adaptive routed, quality service techniques and singular application optimisations.

These techniques have considerably improved the designs. However, still very often they operate strictly within the framework of the traditional router based platforms where the nodes implement the buffering for data, route computation and arbitration for nodes all exist in each node. Even if these routers are specialised, they still have the area, power and latency costs directly related to the are aof the network.

### Ring Based and Loop Based

The ring based interconnects are one of the simpler NoC structures. Discussion on Ring Interconnect, a multiprocessor architecture which Intel built was used in several generations. It proved that ring based communication is commercially viable. Nonetheless, the implementations are mostly similar since they also use the approach of the router-to-node connection and are not feasible due to the router scalability bottleneck caused by the addition of cores.

More recently, the researchers came up with the idea of routerless architectures. Zhao et al. were the first ones to present routerless concept as the routerless NoC that has shown up to 26% savings in power and 16% in area compared to conventional mesh NoCs. Their research exclusively pointed out that compared to routers, the most significant remains involved in the layout but not in the setting of a dedicated network performing at similar levels.

Based on the foundation, the following studies have dive deep into several loop contruction alogrithms that target the improvement of different metrics. For instance, RLrec emphasises minimum loop count with a sure connectivity while other methods are designed for average hop count minimisatio. These developments have served as the major theoretical base for learning routerless NoC functionality and they also help in taking implementation decisions.

### Performance Analysis and Evaluation

The NoC evaluation technique has transformed from early stage when it consisted only measuring the bandwidth and latency to the huge different of metrics. Energy efficiency which is accounted in terms of energy per bit transferred will cause problems if chip power restrictions effect design. The efficiency and scalability which related more to network size have also become a primary concerns.

The simulation frameworks have presented an important part of the NoC evaluation and the ones like Booksim, Noxim and Simpy were developed to allow thorough performance testing on chips with varied packet traffic and workload patterns. These frameworks allow researchers to examine many different kind of design configurations without costing much in terms of hardware.

The available application of traffic worms interest has increased on the traffic patterns along with the costs associated with synthetic benchmarking only. This is because the real worlds application performance often linked to specific communication patterns resulting from applications run on systems. The trend points out the importance of using real scenarios in evaluating NoC devices that reflect how they will actually be used.

## Problem Statement and Research Objectives

### 1.4.1 Problem statement

Routerless NoC present many benefits in terms of hardware simplicity and efficiency. However several major discussions are still on debate in the aread of their performance, scalability limits and design optimisation. The main challenges are :

* To understand the impact of loop construction algorithms on network performance indicators such as latency, throughput and energy efficiency
* To find suitable approaches to resolve the contention when several packets try to use the same network resources
* To investigate the scalability constraints as the network size increases and the traffic patterns become more complex
* To introduce efficient mechanisms of injection and ejection that will overhead the network as less as possible

A thorough simulation framework is a must to indentify optimal configurations and the design space to prevent low level component design decisions.

### 1.4.2 Research Objectives

The goal of this project is to give solutions to each problems as follows

* Recognition of the influence of different loop construction algorithms on network performance across various traffic patterns and network sizes and rank and apply the best one
* Evaluation of the control system efficiency of Extended Buffer (EXB) mechanisms in contention management and performance improvements
* Examination of routerless NoCs scalability traits and revealing possible bottlenecks as the network scale increases
* Performance comparison of time slicing NoCs with traditional routers using same metrics and measures

### 1.4.3 Research Methodology

To achieve these goals, this research make use discrete-event simulation techniques as simulation based methodology. The Simpy library is the core of the simulation that implements the major components of routerless NoCs such as :

* Network topology and loop contruction
* Packet generation and injection
* Flit level movement through the network
* Contention handling through Extended Buffers (EXB-
* Packet ejection and completion

The simulation permits precise observation of each packet and flit as they go through the network making it possible to perform a very detail examination of contention spots, buffer utilisation and timing characteristics. The measures taken for the performance assessment encompass an average packet latency, throughput, buffer utilisation and link activity.

The assessment covers both synthetic traffic patterns and traces from real applications where the availability allows. A set of different network configurations is analysed concerning the network size, loop construction algorithms, buffer capacities and injection rates to show the design trade offs thoroughly.

* Ring-based interconnection
* Path-based routing
* Traffic flow management
* Power and area advantages

### Discrete-event Simulation

* Principles of discrete-event simulation
* Application to network systems
* Existing simulators for NoC
* Limitations of current simulators

### Performance Metrics and Evaluation Methods

* Latency
* Throughput
* Power Consumption
* Scalability measures
* Quality of Services

# Chapter 2 Methodology

## 2.1 Research Approach

This project followed a step by step plan to devise a framework for discrete event of simulation routerless Network-on-Chip architectures. The methodology employed a phased development model starting with comprehensive literature review to understand basic concepts. This followed by design and implementation of the framework, validation testing, and performance evaluation in various configurations and traffic patterns.

The research’s primary focus is on measuring key performance metrics such as latency, throughput, and scalability of the design. Simulation based methodology was chosen due to its flexibility and cost efficient when testing configurations without a need of physical hardware implementation. This approach proved to be a good method to continue which made the process a whole lot more faster and easier for me to try different configurations before the hardware was deployed.

## 2.2 Simulation Framework Design

### 2.2.1 Discrete Event Simulation Model

The focus of this study is discrete event simulation (DES) model, which was created with the help of the Python library called Simpy. DES is the main simulation paradigm due to its capability to effectively depicting systems in which state changes occur at discrete points in time. Within this model, events like packet injection, packet ejection, and flit transmission are scheduled and processed in an ordered time.

The simulation framework employs an event-driven model where each part of the Network-on-Chip (NoC) interacts through events that take place at certain simulation timestamps. The advancement of simulation time occurs in several jumps, each jump is equivalent to one clock cycle in the NoC hardware.

Routerless Network-on-Chip architectures present simulation challenges due to the loop based topology and distinct flow control mechanisms. To model these systems, I developed a discrete event simulation framework focused on capturing packet movement through ring based interconnections without traditional routers. This framework provides a platform to analyse performance characteristics and validating design choices in routerless NoC implementations.

After evaluating several implementations options, I chose discrete event simulation as the foundation for the framework. This approach models the system as discrete events occurring at specifics points in time making it suitable for network simulations where components interact through message passing. The Simpy library was

### 2.2.2 Architectural Components

The simulation was made to imitate the structure of Routerless NoC:

* Nodes: Represent processing elements in NoC that are able to generate, receive, and forward network traffic. Every node has Extended Buffer (EXB) to store the flits temporarily when the loop is busy.
* Loops: Unidirectional ring paths that are used to connect multiple nodes. They are the main routing paths in this architecture.
* Links: Directional connections between adjacent nodes on a loop which are capable of carrying one flit at a time.
* Flits: Network traffic is modelled as packets that are fragmented into smaller pieces known as flits (flow control units). Each packet contains head, body and tail.
* Buffers: Temporary storage components at each node for holding flits, including loop buffers and extended buffers (EXB).

These components are based on the idea of realistic implementation of hardware specifications in a routerless NoC while providing variable of network parameters.

* System requirements and specifications
* Architecture considerations
* Implementation approach

## 2.3 Network Topology Generation

### 2.3.1 Small Network Configuration (2x2)

For a 2x2 mesh network, two predefined loops were implemented:

* A clockwise loop connecting nodes 0, 1, 3, 2, 0
* A counter-clockwise loop connecting nodes 0, 2, 3, 1, 0

This bidirectional approach make sure that each pair of nodes has multiple possible ways to go through which makes congestion decrease and improves fault tolerance.

### 2.3.2 Medium Network Configuration (4x4)

In 4x4 network, a more complicated configuration was applied with multiple layers of loops:

* Inner loops connecting the central nodes ( 5, 6, 9, 10)
* Outer perimeter loops connecting boundary nodes
* Multiple cross-mesh loops dividing the network into regions
* Specialised loops for specific traffic patterns

This multi-layer scheme allows communication throughout the network while keeping the principle of routerless design the loops were designed carefully to assure the sufficient path diversity and the minimum average hop counts is between node pairs

### 2.3.3 Scalable Network Generation

For larger network (beyond 4x4), a systematic loop generation algorithm was implemented:

* Perimeter loops connecting all nodes on the boundary of the mesh
* Horizontal loops traversing each row of the mesh
* Vertical loops traversing each column of the mesh

This method is actually one that concerns the maintenance of full connectivity through looping gates. The design is also scalable for any network size provided that hardware resources are present to support it.

* Node
* Link modelling
* Ring interconnect implementation

## 2.4 Routing Mechanism

### 2.4.1 Path Based Routing Algorithm

The routing system implemented is the path based one instead of the routers table based routing mechanism that is found in the router based NoCs. Each node in turn picks the optimum loop by the path that is shortest to the target. The routing decision’s directions are :

* Determine the loops that are common to both the source and the destination nodes
* Coounter the length of the path (in hops) for each candidate loop
* Choose the one that is minimal hop-count

This method is advantageous in a sense it minimises latency first by choosing the most direct way possible. The routing is decided only at the source node. After a packet is introduced in a certain loop, it will continue to follow that loop until it reaches the destination or completes a full loop of the circuit.

### 2.4.2 Extended Buffer Management

A consideration taken in the design of a routerless No Cis management of the Extended Buffes (EXB). EXB acts as a temporal holding spot when a node has to connect a packet to a loop that is occupied at that moment. The EXB management strategy proceeds according to the rules of the following lines :

* When a loop is occupied, the incoming flits will be kept in the EXB
* When the loop gets free, the flits will come out of the EXB in the FIFO order
* An EXB is attached to a certain loop where it gets rid of this and keeps the packets in the proper path.
* Finally, when the EXB is depleted, it disengages from the loop again which makes it accessible for any other packets

This practice makes sure that the buffer resources are worked efficiently while at the same time makes sure that the packets are ordered and deadlock does not occur.

* Synthetic traffic patterns
* Realistic application
* Traffic injection rate control
* Packet generation

## 2.5 Packet Flow Control

### 2.5.1 Packet Segmentation

WIthin the simulation there is a flit based flow control where each packet is sent in the form of multiple flits :

* Head flit : Routing information is together with the mark of a start of a packet
* Body flit : These are the capacity bearing payloads of the packets
* Tail flit : The packet ends with that

This segmentation provides accurate control over the network resources and allows the pipelining of packets throughout the network

### 2.5.2 Injection and Ejection Mechanisms

Packet injection is handled under a meticulously designed protocol that draws the network traffic :

* In the first instance the source node would come up with a loop that is best for the packet misplaced
* In case the loops is available, flits will be injected directly. However if it is busy then the stored flits are put in the EXB memory and the injection will take place when the loop is freed. Between any pair of flit injections a time interval is set up which is not less than theoretical hardware constraints are supposed to be

The second activity is packet ejection which follows :

* Once a flit’s destination node is reached, the node performs attempt to eject it
* On the other hand if available, ejections resources will allow the flit to exit the network
* Conversely if the resources is busy, the flit will be circulated on their loop
* If number of circulation attemps exceeds threshold, then a reserved link for ejection will be secured

The mechanisms of the counters operations provide smooth packet operations. At the same time deadlocks are prevented.

## 2.6 Traffic Generation

### 2.6.1 Synthetic Traffic Patterns

The framework is flexible to accept different traffic patterns for the measure of network performance in several scenarios :

* Directed Traffic : Packets will be generated by certain source-destination pairs to control the tests
* Random Traffic : A source-destination paor will be decided randomly with the condition that the nodes for the source and destination must differ
* Poisson Traffic : The packet injection process will be based on the use of Poisson process of an initial rate which is overridden by user defined rate

The use of these traffic patterns allows network to be moderated within a variety of contexts, ranging from predictable communication patterns to veggie workloads.

### 2.6.2 Injection Rate Control

Traffic injection was taken care of through the use of an exponential distribution model that simulates arrival times between packets. This approach helped in precise control of the network load while modeling the actual environment traffic accurately. The injection rate varies to promote evaluation of network performance under varying load conditions.

## 2.7 Performance Metrics Collection

### 2.7.1 Latency Measurement

Packet latency is calculated based on the time difference between the head flit insertion and ejection of a packet. This gives an accurate view of the total time the packet has spent in the network, including queueing delays, transmission time, and ejection delays. Furthermore the simulation framework collects these latency values for statistical analysis.

* Individually measured packet latency values are recorded for packets that are delivered without errors
* The minimum, average and maximum of the above mentioned latencies are reported
* The distribution of the latency is investigated to deduce the performance variability

### 2.7.2 Throughput Calculation

Throughput is defined as the count of packets that are properly delivered per elapsed time (number of cycles) which is the primary measure of evaluating the network performance. Throughput is implied as a measure of the effective network capacity thus it is an indicator for the overall performance. Two types of throughput measurement are included in the design :

* Absolute throughput : This figure is simply the sum of all the packets delivered in the entire simulation period
* Normalised throughput : Packet delivery per operational cycle, a measure of performance that is standardised across various implementations

### 2.7.3 Network Utilisation Monitoring

The simulation framework monitors buffer levels and link usage to get insight into network efficiency and possible bottlenecks. THese metrics will show congestion points and give a base for potential optimisations for the network design.

## 2.8 Experimental Design

### 2.8.1 Factorial Experiments

Routerless NoC design is applied to all the experimental tests, which is routed through a factorial experiment

* Network size ranges (2x2, 4x4)
* Packet injection rates
* Packet sizes
* Traffic patterns

These designs enable exploration of parameter space and work out interactions between different parameters

### 2.8.2 Comparative Analysis

The experimental methodology is built on comparative analysis of :

* Different routerless NoC configurations
* Traffic patterns which differ among tests and congestion states
* Performance metrics across different network sizes

The randomisation of traffic routes through the traffic patterns accomplised is important for the comparison to be performed

## 2.9 Validation Methodology

### 2.9.1 Functional Validation

The simulation framework conducted elaborate checking throughout to ensure the right modeling of the routerless NoC parts were built :

* Component level validation was done by ascertaining the autonomous working of individual nodes links and buffers
* Integration testing involved verification of connections between the components and overall system operation
* Packet tracing was done meticulously which made it possible to check for the right movement of packets through the network and flow control mechanisms

### 2.9.2 Performance Validation

Performance validation tests were done to compare the simulated results to theoretical models and benchmarks that have been published on the matter of routerless NoC :

* Latency was measured against theoretical lower bounds
* Throughput was checked against the theoretical network capacity
* Network scaling laws were referenced for verification

These validation steps substantiate that the simulator framework is capable of providing accurate and reliable performance readings for routerless NoC structure

* In case the loop is available, flits will be injected directly
* Comparisons with existing simulators (if applicable)
* Test cases design

# Chapter 3 Implementation and Validation

## 3.1 Implementation Overview

Discrete-event simulation of routerless Network-on-Chip (NoC) architecture is designed and implemented according to the design in this chapter. This implementation is based on the principles introduced in Chapter 2, Methodology, which is built with Python’s Simpy library to be a comprehensive and flexible simulation framework. The proposed method is the same with principal architecture routerless NoCs. The traditional routers were replaced by loops and simplified interfaces that were predefined. The key parts of the implementation are:

* Packet and flit based communication model
* Loops based interconnection topology
* Node interfaces with extension buffers (EXB)
* Interconnection mechanisms for injection, ejection and forwading of packets
* Methods to avoid network abnormalities such as deadlock, livelock and starvation

The appendix code for the complete implementation is given in Appendix B, yet this chapter provides only the design decisions, architectural components and validation methodologies.

* Software architecture
* Core components description
* Class relationships
* Data structure and algorithm

## 3.2 Core Components

### 3.2.1 Simpy Environment

Discrete-event simulation is done in Simpy, a Python framework for process oriented discrete-event simulation. Simpy offers the vital functions that are available for concurrent process modelling including:

* Event scheduling and management
* Process synchronisation
* Resource allocation and management
* Time management

The environment of Simpy is the simulation’s central manager. It runs processes in the right order and moves simulation time forward. With the support of this environment, it becomes proper for the concurrent operation of the routerless NoC to be modelled. For example:

* Simultaneous packet transmissions across multiple loops
* Contention for resources like extension buffers and ejection links
* Propagation delays between nodes

### 3.2.2 Packet and Flit Classes

The communication model works at flit level where it breaks packets into smaller units called flits to manage the flow of data more efficiently. The two classes that are the basis of the implementation are Packet and Flit.

A packet makes up an identification that contains a source, destination, size, and creation timestamp. The packet segmentation method (flitation()) divides each of the packets into three types flits:

* Head: The first flit contains the routing information
* Body: The intermediate flits contain data
* Tail: The last flit marks packet end

Every flit retains its individual state data like the current node, injection and ejection times and a livelock prevention “circling count”.

### 3.2.3 Loop and Link Classes

The most fundamental component of the routerless NoC is the loop based interconnection network. The implementation includes Loop and Link classes. The Loop class declares the circular pathway through more than two nodes. Besides, it has methods to:

* Specify the next node in the loop series
* Compute the path length for every source to destination

The Link class in the loop is the physical connection between adjacent nodes that also consists of a signle flit buffer that is implemented as Simpy Store with capacity of one

### 3.2.4 Node Interface

Node interface is the most complex component which is responsible for managing the injection, forwarding and ejection of packets. The Node class implements:

* Loop Buffers: Single flit buffers for each passing loop
* Extension Buffer (EXB): A multi flit buffer that can be attached to any loop to handle multi flit packets
* Ejection Links: Limited resources for packet ejection
* Routing Table: Maps destinations to best loops
* Loop Management: Tracking which loops have EXBs attached

Node interface is also keeping packet injection, ejection and latency stats for performance analysis.

* Node architecture implementation
* Network topology generation
* Routing algorithm implementation
* Flow control mechanisms

## 3.3 Network Topology Generation

The simulation framework provides diverse networking condigurations. For example, 2x2, 4x4 and larger grids. The loops are produced by the methods structured in the examined papers and especially the composed layered approach of (Alazemi et al.) is exploited.

For the two networks (2x2 and 4x4), some routes are stored in advance to guarantee complete connectivity and perfect performance. In the case of a larger network, RLrec (Routerless Recursive) algorithm is implemented which:

* Builds the loops for the outer edges of the grid
* Adds horizontal row loops for connecting nodes in the same row
* Adds vertical column loops for connecting nodes in the same column
* Metrics collection
* Data analysis techniques
* Statistical analysis

## 3.4 Key Processes

### 3.4.1 Packet Injection

Packet injection process is the one that concerns the insertion of the new packets into the network. When a packet is to be injected, the node first finds the right loop by looking into the routing table. If there are several loops that can reach the destination, the one that has a shorter path is picked.

Moreover, for packets which occupy more than one flit in total, the node separately checks if it has an Extension Buffer (EXB) and if the one is used that is not full. In case the EXB is full, injection step is postponed and it is retried later.

The following steps entailed the injection process:

* Picks the most efficient loop to the destination
* Applies an Extension Buffer (EXB) for multi flit packets
* Injects the flits sequentially into the loop
* Manages the full buffer condition when necessary by trying again later

### 3.4.2 Flit Forwarding

When the flits are injected into the network, they are routed via the network. In each node, the incoming flits are processed according to their destination:

* When the destination is equal to the current node, it tries to eject the packet
* When the ejection fails or when flits have another destination, they continue to the next node in the loop

The forwarding process:

* Establishes whether the flit has reached its destination or not
* If it is at the destination it will try to eject it or else it will forward it
* The next step is to calculate the next node in the loop in case of forwarding
* The it applies some delay as one cycle to model the transmission time
* Finally it uses the network transmit function to loop the flit to the next node.

### 3.4.3 Packet Ejection

The ejection process is about the packets which have reached their destinations and should be removed. This is considered as one of the strategic aspects of the routerless NoC realisation since it must manage the contention for resoureces due to several packets arriving together.

To uphold ejection efficiency versus hardware simplicity, the implementation limits the number of ejection links to each node. When the number of packets furthermore exceeds the ejection nodes, some are temporarily redirected into the network for looping and attempt the ejection later on. The ejection process is as following:

* Checks ejection links availability
* Uses a reserved node for the packet that achieved circling count threshold
* Records statistics
* Adds up the circling count when the ejection fails
* Preserves one ejection link to avoid livelock issues if a packet is circling for too long

## 3.5 EXB Management

Extension Buffer (EXB) is a mechanism which allows efficient handling of transferring multi flit packets while using few buffer resources. In contrast to the previous design where one packet sized buffer was required for each node in the loop, EXB approach used buffer pools that can be pooled together and dynamically assigned to any loop needed. The procedures of EXB management:

* Appending an EXB to the loop while injecting a multi flit packet
* Storing flits in an EXB if the loop is busy
* Draining flits from EXB when the loop has been freed
* Removing EXB from the loop when it becomes empty

This arrangement is more efficient in terms of buffer occupancy compared to the alternative methods while continuing to maintain the performance of the network

## 3.6 Abnormality Countermeasure

TBC

## 3.7 Traffic Generation and Simulation

The implementation is constructed with traffic generation framework that provides the ability to analyse and assess performance under different scenarios. It supports:

* Different injection rates to model different network loads
* Poisson arrival process with exponentially distributed inter arrival times
* Variable sized packets
* Different traffic patterns
* Custom traffic scenarios for specific evaluations

The injection rate can be amended via a parameter that sets the average amount of flits per node per cycle. The increase in the number of flits signifies greater network load, enabling the evaluation of performance under stress conditions

## 3.8 Performance Metrics

Implementation carries out the collection of various performance parameters in order to examine the routerless NoC architecture. Specific stats regarding the number of packets that are injected, ejected and average latency are gathered at every node. This data are then summed up at the global network level to give better information on overall performance.

These metrics () are useful since they allow detailed routerless NoC assessment in terms of various confidurations and traffic textures

## 3.9 Implementation Validation

TBC

* Unit testing approach
* Integration testing methodology
* Validation against theoretical model
* Benchmarks and comparison

# Chapter 4 Results, Evaluation and Discussion

## 4.1 Experimental Setup

* Simulation parameters
* Workload characteristics
* Test cases description

## 4.2 Performance Results

* Latency analysis under different traffic patterns
* Throughput analysis
* Scalability evaluation with increasing network size
* Power efficiency estimation

## 4.3 Comparative Analysis

* Benchmarking against existing literature
* Analysis of advantages and limitations

## 4.4 Discussion

* Interpretation of results
* Performance bottlenecks identification
* Implications for routerless NoC design
* Unexpected findings and anomalies (if applicable)

## 4.5 Evaluation

* Analysis of simulation accuracy
* Computational efficiency
* Framework limitations

## 4.6 Conclusions

* Summary of achievement
* Discussion of how objectives were met
* Key insights from the project

## 4.7 Future Work

* Potential improvements to the simulation framework
* Additional features for future implementation
* Potential applications of the simulator

# List of References

*List of academic papers, books, websites, and other sources used*

# Appendix A Self-appraisal

## A.1 Critical self-evaluation

* Analysis of achievement against objectives
* Strengths and weaknesses of the approach
* Technical challenges encountered and solutions

## A.2 Personal reﬂection and lessons learned

* Knowledge and skills gained
* Project management reflections
* Changes that would be made

## A.3 Legal, social, ethical and professional issues

### A.3.1 Legal issues

* Intellectual property aspect

### A.3.2 Social issues

* Potential impact on computing infrastructure
* Energy efficiency distributions

### A.3.3 Ethical issues

* Research integrity
* Open source vs proprietary considerations

### A.3.4 Professional issues

* Adherence to software engineering best practices
* Documentation
* Maintainability

# Appendix B External Materials

* List of libraries used
* External code sources
* Development tools and environments