CONGESTION-AWARE ADAPTIVE ROUTING ALGORITHM DESIGN FOR 3D NETWORK-ON-CHIP

3D YONGA-ÜSTÜ-AĞLAR İÇİN TIKANIKLIĞA DUYARLI UYARLANABİLEN YÖNLENDİRME ALGORİTMA TASARIMI

NURETTİN BÖLÜCÜ

PROF. DR. SÜLEYMAN TOSUN

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Master of Science

in Computer Engineering.

2021

ABSTRACT

CONGESTION-AWARE ADAPTIVE ROUTING ALGORITHM DESIGN FOR 3D NETWORK-ON-CHIP

Nurettin BÖLÜCÜ

Master of Science, Computer Engineering Department
Supervisor: Prof. Dr. Süleyman TOSUN
June 2021, 63 pages

New communication methods have become inevitable due to the continuous increase in the number of components on the integrated circuits. Designers, who prioritize performance, scalability, power consumption need better solutions. Network-on-Chip (NoC) architecture emerged as a solution to these requirements with its parallelism and scalability. As the technology advances in integrated circuit systems, cores can process more data and require better communication. As the number of cores increase, NoC desings need to have shorter average path than classical communication methods. 3D-NoC design has been proposed to meet these demands due to its architecture, speed, average path, and power consumption. However, the routing problems for 3D is more complicated than 2D version. Since deterministic algorithms encounter congestion problems, adaptive algorithms are able to distribute the traffic load to give better results. Motivated by the effectivenes of learning algorithms, in this thesis, we present a Q-Learning based routing algorithm for 3D-NoCs to solve this type of problems. In our algorithm, each router node maintains a Q-Table and updates it by receiving the traffic information from neighboring routers. We select the output port by using the

state of the router and corresponding Q-Values. We have tested our proposed method with the deterministic XYZ and 3D elevator-first West-First algorithm under different traffic models and compared the results. We have observed an 8% average performance improvement compared to the other routing algorithms.

Keywords: Network-on-Chip(NoC), Q-Learning, 3D-NoC, Routing, Adaptive Routing, Congestion Awareness

ÖZET

3D YONGA-ÜSTÜ-AĞLAR İÇİN TIKANIKLIĞA DUYARLI UYARLANABİLEN YÖNLENDİRME ALGORİTMA TASARIMI

Nurettin BÖLÜCÜ

Yüksek Lisans, Bilgisayar Mühendisliği
Tez Danışmanı: Prof. Dr. Süleyman TOSUN
Haziran 2021, 63 sayfa

Entegre devrelerdeki komponent sayılarının artışına bağlı olarak yeni haberleşme yöntemlerinin geliştirilmesi kaçınılmaz olmuştur. Performans, ölçeklenebilirlik, güç tüketimi gibi metriklerin öncelikli olduğu tasarımcılar daha iyi çözümlere ihtiyaç duymaktadır. Yonga-üstü-Ağlar(YüA) ölçeklenebilirlik ve paralel çalışabilme özellikleri ile bu ihtiyaçlara çözüm olabilmektedir. Entegre devre teknolojilerindeki ilerleme ile birlikte işlemciler daha fazla işlem gücüne sahipler ve daha iyi haberleşme sistemlerine ihtiyaç duymaktalar. Entegre sistemlerde işlemci sayılarının artması ile beraber YüA sistemleri de daha iyi ve ortalama mesafenin daha kısa olduğu tasarımlara ihtiyaç duymaya başladılar. 3 boyutlu (3B) YüA mimarileri olanak sağladıkları hızları, ortalama mesafe ve güç tüketimleri ile bu ihtiyaçları karşılamak adına geliştirilmişlerdir. Fakat 3B sistemlerin daha karmaşık tasarımları ile birlikte, yönlendirme algoritmaları problem olmaya başlamıştır. Saptanabilir tasarımların sıkça trafik yoğunlaşması sorunlarına karşılık, uyarlanabilir yönlendirme algoritmaları daha iyi performans sergilemektedir. Bu motivaston ile birlikte 3B Yonga-üstü-Ağlar için Q-Öğrenme tabanlı yönlendirme algoritma sunmaktayız. Sunduğumuz algoritma içerisinde her anahtar içerisine bir Q-Tablo

tutmakta ve komşu anahtarlardan aldığı traffik bilgisine göre bu tabloyu güncellemektedir. Çıkış kanalı anahtarın o anki durumuna ve Q-Tablodaki değerlere göre belirlenmektedir. Sunduğumuz metod ile saptanabilir yönlendirme algoritması ve Z düzlemi öncelikli batı-öncelikli yönlendirme kıyaslanmış ve %8 oranında iyileştirme kaydedilmiştir.

Anahtar Kelimeler: Yogca-üstü-Ağ, YüA, Uyarlanabilir Yönlendirme Algoritması, Tıkanıklık Farkındalığı

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor, Professor Dr. Süleyman Tosun, for supporting and guide this thesis with his endless patience, valuable advice, encouragement, and immense knowledge. At every stage of this thesis, he supported me to push forward and work harder.

I would like to thank TUBİTAK, for giving me valuable support for this thesis under the project 117E130.

I would like to thank my big family for their constant support throughout my master's education and my life in general. They have motivated and supported me to start university again to pursue my goal to be an engineer. I would like to thank my mother and father for their sacrifices to create the best environment for our education. I would like to thank Necva Bölücü, especially for her technical support and guidance throughout writing this thesis.

I would like to thank my colleagues, dear friends Atif Ofluoğlu, Metin Dündar Özkan, Uğur Usuğ for their support, patience, and technical guidance when I needed the most.

Last but not least, I would like to thank my girlfriend, Büşra Nakipoğlu. I am both proud and lucky to have her in my life. I thank her for making me a better and happier person every day and motivate me to follow my dreams.

CONTENTS

ABSTRACT i
ÖZET iii
ACKNOWLEDGEMENTS v
CONTENTS vi
FIGURES ix
TABLES x
SYMBOLS AND ABBREVIATIONS xi
1. INTRODUCTION 1
1.1. Overview 1
1.2. Contributions 2
1.3. Outline of the Thesis 3
2. BACKGROUND 5
2.1. System on Chip 5
2.1.1. Point-to-Point and Bus 5
2.2. Network on Chip 6
2.2.1. Components of Network on Chip 6
2.2.2. Features of Network on Chip 9
2.3. Q-Routing 14
3. RELATED WORK 16
3.1. Related Work on 3D-NoC Designs 16
3.2. Related Work on Adaptive Routing 18
3.2.1. Partially Adaptive Routing 18
3.2.2. Fully Adaptive Routing 20
4. CONGESTION-AWARE ADAPTIVE ROUTING ALGORITHM 23
4.1. Introduction 23
4.2. Deadlock Avoidance 24
4.3. Routing Algorithm and Q-Table 27
4.3.1. Q-Table Format 27

4.3.2. Feedback Message 28
4.3.3. H Value Transfer 31
4.3.4. Elevator First 32
4.4. An Example 32
5. EXPERIMENTAL RESULTS 35
5.1. Simulation Environment 35
5.2. Performace Metrics 35
5.3. Results Analysis and Comparison 36
5.3.1. Uniform Traffic Test 36
5.3.2. Hotspot Traffic Test 37
6. CONCLUSION 40
6.1. Conclusion 40
6.2. Future Research Directions 40
REFERENCES 41
THESIS ORIGINALITY REPORT 47
CV 48

FIGURES

Figure 1.1. (a) Bus-based (b) P2P (c) NoC 1

Figure 2.1. Moore‘s Law 6

Figure 2.2. Typical NoC architecture 7

Figure 2.3. Five-port router architecture 8

Figure 2.4. (a) Mesh (b) Torus (c) Ring (d) Star topologies 11

Figure 2.5. Deadlock Example 13

Figure 2.6. Q-Learning algorithm process 14

Figure 2.7. Q-Table for (a) Node-based and (b) Direction-based 15

Figure 3.1. Turn models for (a) Mad-y and (b) LEAR (c) CARM 21

Figure 3.2. Q-Table formats for (a) Q-Routing and (b) C-Routing 21

Figure 4.1. 4x4x4 NoC Example. 23

Figure 4.2. Router representation for the mad-y method in 2D-NoC. 24

Figure 4.3. Eight possible cycles with 90-degree turns. 25

Figure 4.4. Allowed turns for (a) 90-degree, (b) 0-degree, and (c) 180-degree. 25

Figure 4.5. Channel numbering for (a) mad-y method and (b) HARA method. 26

Figure 4.8. Feedback packet. 28

Figure 4.6. Encoding the latency information 29

Figure 4.7. Encoding the direction information 29

Figure 4.9. 2D Network 30

Figure 4.10. Learning rate comparison under uniform test model. 31

Figure 4.11. Learning rate comparison under hotspot test model. 31

Figure 4.12. Output Channel Selection Algorithm 33

Figure 4.13. Example for path selection. 34

Figure 4.14. Example for updating Q-Table. 34

Figure 5.1. Latency under uniform traffic test. 37

Figure 5.2. Throughput under uniform traffic test. 38

Figure 5.3. Latency under hotspot traffic test. 38
Figure 5.4. Throughput under hotspot traffic test. 39

TABLES

Table 3.1. Virtual Channel Assignments for DyXYZ..... 17
Table 3.2. Output port options for HamFA 18

Table 4.1. Potential output channels for each input channel and destination. 27
Table 4.2. An example Q-Table. 30

SYMBOLS AND ABBREVIATIONS

Abbreviations

2D	Two Dimensional
3D	Three Dimensional
AMT	Average Message Time
BU	Business Unit
CARM	Congestion Aware Routing method
CATRA	Congestion Aware Trapezoid based Routing Algorithm
D	Down layer
DBAR	Destination Based Adaptive Routing
Dir	Direction
DSP	Digital Signal Processor
DyAD	Dynamic-Adaptive-Deterministic
DyXY	Dyncamin XY
DyXYZ	Dyncamin XYZ
E	East
EDXY	Enhanced Dynamic XY
FAR	Fully Adaptive Routing
FDWR	Force Directed Wormhole Routing
FL-RuNS	First-Last Runtime and Resilient 3D Networks-on-Chip Scheme
FIFO	First In First Out
HamFA	Hamiltonian-based Fault-tolerant routing Algorithm
HARAQ	Highly Adaptive Routing Algorithm using Q-Learning
HLAFT	Hybrid-Look-Ahead-Fault-Tolerant
IP	Intellectual Property
InCh	Input Channel
L	Local
LAFT	Look-Ahead-Fault-Tolerant

Abbreviations

LA-XYZ	Look-Ahead XYZ
LEAR	Load based Energy Aware Routing
MPSoC	Multiple Processor SoC
MU	Memory Unit
NI	Network Interface
N	North
N1	North - Virtual Channel 1
N2	North - Virtual Channel 2
NE	Northeast
NW	Northwest
NoC	Network-on-Chip
OE	Odd-Even
OutCh	Output Channel
P2P	Point-to-Point
PE	Processing Element
RCA	Regional Congestion-Aware
S	South
S1	South - Virtual Channel 1
S2	South - Virtual Channel 2
SE	Southeast
SW	Southwest
SoC	System-on-Chip
TSV	Through-Silicon-Via
U	Upper layer
VC	Virtual Channel
VCT	Virtual Cut Through
W	West

1. INTRODUCTION

1.1. Overview

Number of transistors has been doubling every two years in the same chip area according to the Moore‘s law [1]. This made it possible to have a significant number of transistors and also processing units in a single chip. This phenomenon is known as System-on-Chip(SoC) design, and every year SoC systems grow exponentially. Traditional SoCs are using bus-based communications and point-to-point (P2P) links for inter-module communications. These designs are getting inefficient every year, with the number of cores is increasing significantly. Therefore the classical bus-based and P2P communications becoming unpractical. Additionally, processing elements are getting more advance and complex over time, and they now require more efficient, faster communication with less power consumption. Therefore Network-on-Chips (NoC) design has emerged as a new approach to address higher communication demands [2].

Figure 1.1. shows the bus-based, P2P, and NoC architectures. P2P communication architecture stands out as the fastest. However, it gets inefficient due to its high power and area consumption as the number of cores increases. Shared bus architecture uses single or multiple busses to allow communication between the IPs in different time slots. Bus-based communications require good scheduling to support many connected components. However bus-based communications are not scalable to the number of cores.

Figure 1.1. (a) Bus-based (b) P2P (c) NoC

NoC technology is a new approach to modern SoC problems due to its efficiency, scalability, and reusability characteristics. NoCs mainly consist of processing elements, routers, network interfaces, and links. Processing elements can be a DSP, memory unit, processor, etc. It allows connecting processors, cores, and other business units with switching routers to propagate packets using classic network technologies. NoC interconnection technology is often called as "a front-end solution to a back-end problem".

In recent years, 3D-NoC designs have emerged as the number of business units continues to grow. 3D-NoC has significant advantages regarding the length of the path and number of interconnection compared to 2D-NoC designs. The average number of hops increases exponentially with respect to an increase in the number of nodes in the 2D networks. However, an increase in the number of hops in 3D networks is more linear [3].

3D-NoC designs connect 2D-NoC layers on top of each other using Through-Silicon-Via (TSV). Through-Silicon-Via is a type of connection for three-dimensional integrated circuits and is accepted as the most popular for vertical connection with planes [4]. TSVs enable connecting ICs vertically with the ability to pass through the silicon. TSVs are fundamental for 3D-NoCs with their advantage of sort connection. Shorter connections also use less power, thus they reduce the power consumption as well as the latency.

In 3D architectures, the number of business units assigned to different layers increases the transistor density. With reducing the length of the connection using TSV, 3D-NoC designs also aim to reduce the number of clock cycles it takes to transfer a packet. More global interconnection with shorter connection length allows 3D-NoCs to have higher connectivity, higher performance while having less latency, and consuming less power compared to its 2D counterparts [5].

1.2. Contributions

In this thesis, we present a Congestion-aware adaptive routing algorithm design for 3D-NoCs. Our approach is fundamentally based on designing an adaptive routing algorithm using the traffic congestion information and selecting the ideal path depending on this information. In our method, each router keeps a Q-table to store traffic and expected latency information to send a message to each direction and output channel combination. Additionally, decisions to send the message to the upper or lower layer are made based on the traffic

information around the current router and the neighboring routers. H value is used to determine the congestion information for inter-layer packet transfer, and the H value is calculated by taking the average of Q-Table values of minimum paths. Although Q-Learning-based routing algorithms have been used for 2D networks, we have proposed a congestion-aware routing for interlayer communication while using Q-Learning-based routing for intra-layer communication.

To avoid deadlock, we have adapted the mad-y model [6]. The mad-y uses two virtual channels on the Y dimension to be able to restrict some possible turns to avoid deadlock. The mad-y method is a minimal routing algorithm; it prohibits 180-degree turns and only allows turns in ascending order. However, a prior study [7] proposed a new routing method based on mad-y method to allow 180-degree turns. This method modifies the numbering mechanism of mad-y and allows the 180-degree turns that do not cause deadlock. To be able to implement adaptiveness, we have adapted our routing algorithm to adapt this numbering mechanism to avoid deadlock.

The proposed model has been developed using the Noxim Simulation framework [8] to test and compare against several routing algorithms under uniform and hotspot traffic models. Noxim is a NoC simulator and developed in SystemC. SystemC is a C++ programming language extension and gives an opportunity for hardware to be modeled in software. We have observed that our method outperformed them in terms of latency and throughput.

1.3. Outline of the Thesis

The thesis is structured as follows:

In **Chapter 2**, the essential background knowledge to ease the understanding of this thesis for any reader is given. The background knowledge is given under two sections: Network-on-Chips and Q-Learning background.

Previous studies on NoCs will be given in **Chapter 3**. In that section, models will be discussed under two main subjects; adaptive routing algorithms and 3D-NoCs. We also discuss the partially and fully adaptive routing algorithms for better understandings.

In **Chapter 4**, our proposed Congestion-Aware Adaptive Routing Algorithm Design for 3D Network-on-Chip will be explained in detail. First, we present turn models that avoid deadlock and livelock. Then we will explain the adaptive routing algorithm and information sharing structure for adaptiveness.

In **Chapter 5**, the experimental results of the proposed model will be presented along with a discussion of the results. Results are compared with other approaches in the literature.

Finally, conclusion nad future directions will be given in **Chapter 6**.

2. BACKGROUND

2.1. System on Chip

System-on-Chips (SoCs) are integrated circuits that integrate multiple components such as processors, memories, I/O (Input/Output) ports, and more. Following *Moore's law*, the number of transistors on a single chip doubles every two years [1]. Figure 2.1. shows the number of transistors on Intel's processors. Multiple cores can run multiple processors on an SoC with the technology, and these designs are called Multiple Processor SoC(MPSoC). These advancements of having these many cores on chips bring other issues such as communication, bandwidth, and latency problems. These issues increase the importance of the connection and communication networks of the chips. Before NoC designs, Point-to-Point and Bus-based connections were the main approaches for SoC connections. Number of processors on a single chip increased while the processing power of each processor improved, communication has played an essential role in power consumption. Connections use 50% of the dynamic power, and this share is increasing [9].

2.1.1. Point-to-Point and Bus

Figure 1.1. shows the P2P and Bus-based communications. As seen from the figure, P2P architecture connects every Processing Element if they need to communicate. This approach is applicable if a core or Processing Element is requested to communicate with a few Processing Elements without using the shared bandwidth. Bus-based communications are one of the most common ways of communication in an SoC design. Bus-based communication connects multiple components via a single bus or multiple busses. These busses are acting as the system's spine, and all components use these busses for communication. Bus-based communications have simple architecture, this grants advantages such as simplicity, scalability, cost. However same simple architecture can cause problems since all components use a single bus. If the network traffic increase, every component will be affected, and this may cause starvation for some components. Also, if the bus fails, the entire system would crash. There are several bus-based communication such as AMBA [10], Wishbone [11], Avalon [12], PowerPC bus [13], etc.

Figure 2.1. Moore's Law [1]

2.2. Network on Chip

NoC technology is a new approach to modern SoC problems due to its efficiency, scalability, and reusability characteristics. NoC interconnection technology is often called as "a front-end solution to a back-end problem". In this section, we have explained NoC architectures in general with their characteristics and their components. This allows shrinking the chip size, integrating more components, and increasing the SoCs' processing power.

2.2.1. Components of Network on Chip

As seen in Figure 2.2., a typical NoC consist of processing elements(PE), routers(R), network interfaces(NI), and links.

2.2.1.1. Processing Elements

[1] Source : Wikipedia (wikipedia.org/wiki/Tranistor_count)

Figure 2.2. Typical NoC architecture

Processing elements are the main elements of these network designs. Processing elements are mainly computing cores with varying purposes such as processor cores, memory modules, DSPs, etc. NoC systems aim to connect these processing elements efficiently with each other regarding speed and power consumption.

2.2.1.2. Network Interfaces

Network interfaces are the interfaces between processing elements and the routers that provide, organizes, and handles packet-based communication. Network interfaces are responsible for communication protocols, assembling and disassembling packets, organizing buffers, and helping the routers.

2.2.1.3. Routers

Routers manage how packet transfers are executed in the network based on the implemented routing algorithm, packet selection, and connections. Thus routers perform the flow-control,

routing, switching functionalities to transfer the packets. Figure 2.3. shows an example five-port router architecture.

Figure 2.3. Five-port router architecture

2.2.1.4. Links

Links are the elements that connect the routers physically utilized for data transfers. Characteristics of these links are decided regarding the network designs, such as bandwidth and length.

2.2.1.5. Virtual Channel

Input buffers can be divided into multiple virtual channels, which use a single physical link. Using various virtual channels allows transferring multiple packets without blocking each other while a packet is in an idle state. Virtual channels allow improving latency performance and increase the throughput [14]. Virtual channels can be utilized for deadlock avoidance and introduced to avoid deadlock for cyclic networks [15] and torus networks [16].

2.2.2. Features of Network on Chip

Although the architecture and the components are mostly the same for the NoCs, other significant defining features include topology, routing algorithm, and flow control.

2.2.2.1. Topology

Topology determines the scheme and the connection between the routers as well as the physical layout. Topology directly affects scalability, performance, latency, and power consumption. Many topology designs have been proposed to address different routing problems. Mesh, Torus, Ring, Butterfly, Star, etc., can be given as an example. Figure 2.4. shows the most commonly used four topologies, which are Mesh, Torus, Ring, and Star. Mesh and Torus topologies are used as the base architecture for some of the most common network topologies. 60% of the 2D-NoC topologies use either mesh-based or torus-based topology [17]. In mesh networks, every router is connected to the four neighboring routers. In torus networks, routers are connected to neighboring routers as in the mesh. Unlike mesh networks, torus uses wrap-around channels to connect the edge routers. Torus networks decrease the average number of hops for packet transfer, but due to long wrap-around channels, transmission delay is significantly higher on these channels.

- **Mesh:** In Mesh topologies, each device is connected to neighboring routers, consist of n rows and m columns. The address of the routers is usually defined as coordinates (x,y). Mesh topologies have the tolerance for having failed links. Also, mesh does not have to centralize authorities, and adding new devices would not disturb the network.
- **Torus:** Torus topology also consists of n rows and m columns. Each device is connected to neighboring routers, as well as the wrap-around connections to the opposite edge. Torus topology has proposed to reduce the latency for mesh topologies while keeping the simplicity.
- **Ring:** Ring topologies can be classified as Bidirectional and Unidirectional based on their output channels. In-Ring topologies, every router is connected to two other routers and creates a closed circle. Ring topologies are chip to implement but has high latencies.

- **Star:** In Star topologies, every device is connected to a central device or hub. Star topologies have the disadvantage of having longer cable compare to bus-based topologies. However, only one device would be affected by the failure in star topologies if a link fails. If a device wants to send a message to another device, the message needs to be transferred over the central hub.

For ideal topology selection, several factors need to be considered, such as flow control, routing, traffic, number of cores, channel load, etc. Topologies are selected based on the performance and costs with evaluating these factors of the system.

2.2.2.2. Routing Algorithm

The routing algorithm is responsible to determine the path to send the packet to its destination. The path selection is closely dependent on the topology of the network. The purpose of routing algorithms is to find the best possible path for the whole network to minimize the average latency. We can classify routing algorithms with several factors, such as adaptivity and decision locality.

Routing algorithms can be classified with respect to the location of the routing decision is made as *source routing* and *distributed routing*.

- **Source:** In source routing, path selection is made before the message is transferred. Thus, the message will be transferred over the pre-defined path. This type of routing algorithm is mostly used with minimal routing algorithms.
- **Distributed:** In distributed routing, every router selects the path according to the information in the header flit. Path selection can be minimal, non-minimal, or adaptive.

Routing algorithms can be classified as *minimal routing* and *non-minimal routing* according to the minimality. Minimal routing algorithms always send the message in the direction of the destination. Minimal routing algorithms can be adaptive or deterministic. However, Non-minimal routing algorithms mostly use fully adaptive routing algorithms for their path selection strategy.

Figure 2.4. (a) Mesh (b) Torus (c) Ring (d) Star topologies

- **Minimal:** In a minimal routing algorithm, the message is always sent closer to the destination. The biggest advantage of minimal routing is that livelock does not occur in minimal routing algorithms. Compare to non-minimal routing, and minimal routing algorithms are more scalable and less complex to implement.

- **Non-Minimal:** In non-minimal routing, message can be sent to every direction. With non-minimal routing, congested nodes and fault links can be avoided. Non-minimal routing algorithms need to handle livelock scenarios.

Routing algorithms can be categorized as *deterministic routing* and *adaptive routing* algorithms depending on their degree of adaptiveness.

- **Deterministic:** The path is fixed for all source-destination combinations. Does not rely on the state of the network.
- **Adaptive:** The path is selected dynamically depending on the faulty links, congestion, nonhomogenous traffic, etc.

2.2.2.3. Flow control

Flow Control is the main feature to efficiently use the network's resources, such as buffers and channels. Flow control refers to control of the message flow between routers and resource allocations [18]. A good flow control design must avoid deadlock and distribute the resources evenly to avoid long waits for packets. Most common flow control approaches are: *store-and-forward*, *wormhole*, and *virtual cut through*

- **Store-and-Forward:** For each packet, every flit is stored in the buffers, and the router waits until each flit is received to forward it to the next router. This approach causes higher per-packet latency and also requires larger buffers to be able to store bigger messages.
- **Virtual Cut Through(VCT):** VCT requires to store every flit of a packet before sending it to next router. Unlike Store-and-Forward, it allows forwarding the stored flits to the next router when the next router is available to store the whole packet. This approach also requires larger buffers but has reduced per-packet latency compare to the store-and-forward approach [19, 20].
- **Wormhole:** Wormhole approach is the one of the most common flow control. The message packet is divided into three types of flits (header, body, tail), while header flits generally store the communication information. Flits are sent to the next router if there is a free buffer slot. Thus, the Wormhole approach has lower latency than other approaches with better buffer management and less buffer space requirement. Thus, with wormhole switching, the buffer size can be reduced, which also reduces the power consumption and the buffering area cost [21].

2.2.2.4. Deadlock, Livelock and Starvation

Deadlock, livelock, and starvation are all defined as a state in a packet that cannot reach its final destination due to a lack of resources.An ideal routing algorithm is expected to be deadlock and livelock free.

Deadlock:
Deadlock occurs when two or more packets hold a resource while waiting for another resource in cyclic order. Figure 2.5. shows an example where P_1, P_2, P_3, and P_4 is sent to R_3, R_4, R_1, and R_2 respectively. Every packets are requesting an output channel that is hold by another packet.

Figure 2.5. Deadlock Example

Livelock:
Livelock can occur in non-minimal adaptive routing algorithms. Livelock occurs when a packet does not blocked but travels in a circular path and does not reach its destination. Minimal routing algorithms can avoid it; however, for adaptive routing algorithms, livelock detection or avoidance algorithms need to be implemented.

Starvation:
Starvation occurs when a message waits for a resource that is always granted for other packets.

2.3. Q-Routing

Q-Routing is Q-Learning-based adaptive routing. It uses Q-Learning for optimal path selection for the network, and each node decides which direction it should be sent the packet. Q-Routing offers more flexible and quick adaptiveness compare to regular adaptive routing algorithms. In regular adaptive routing algorithms, latency or congestion information is sent back to the sender after reaching the final destination. In the Q-Routing approach, each node holds local or global information depending on the design, which allows the network to be updated more quickly.

Q-Learning is an algorithm of model-free reinforcement learning in the machine learning proposed by Christopher Watkins [22]. Q-Learning is worked by learning with Q-Functions to model the environment and take action accordingly. Q-Learning approach has Q-Tables that hold state-action pairs Q(s, a) to utilize its actions. Based on the algorithm, Q-Learning updates its Q-Values at every step. Figure 2.6. shows the Q-Learning algorithm process.

Q-Table can be explained as simply a lookup table where we store the calculated expected reward for action at each state.

Figure 2.6. Q-Learning algorithm process

Q-Function uses the Bellman equation for calculating the Q-Values iteratively. Using equation (1) Q-Values are updated. α is given as the learning rate to update the Q-Value with the

feedback message *r*. With respect to the action (a), state-action Q-Value is updated with α and *r*.

$$\overbrace{Q(s,a)}^{\text{new Q-Value}} = (1-\alpha) * \overbrace{Q`(s,a)}^{\text{old Q-Value}} + \overbrace{\alpha}^{\text{learning rate}} * \overbrace{r}^{reward} \tag{1}$$

Q-Routing stores a Q-Table in each node that stores congestion information. Congestion information can be stored for each destination or each direction. Figure 2.7. shows example Q-Tables for 4x4 2D-NoC. Storing expected latency information for each node can be more reliable to find the optimal path. However, as seen from Figure 2.7. storing expected latency information for each node increases the size of the Q-Table.

Figure 2.7. Q-Table for (a) Node-based and (b) Direction-based

3. RELATED WORK

Multi-core designs became inevitable due to the increase in demand for processing power over time. Communication between the multi-core designs plays an essential role in performance.

In this chapter, we present some of the related work focusing on 3D-NoC designs and routing algorithms that use reinforcement learning for better communication performance.

3.1. Related Work on 3D-NoC Designs

3D networks are rising for their multiple benefits such as lower power consumption, smaller area usage, shorter average path between source and destination pairs. Many routing algorithms have been proposed for 3D architectures.

Some of these routing algorithms can be categorized as deterministic routing algorithms, such as XYZ, while many of them are adaptive routing algorithms such as 3D-FAR [23] and DyXYZ [24].

DyXYZ[24] is a fully adaptive routing algorithm implemented proposed for 3D networks and proven deadlock-free. DyXYZ routers use 4, 4, 2 virtual channels on X, Y, and Z dimensions and divide the network into eight subnetworks. DyXYZ guarantees deadlock freedom by using different virtual channels to connect subnetworks as seen in Table 3.1. 3D-FAR [23] routing algorithms reduce the virtual channels to 2, 2, and 4 on X, Y, Z dimensions, respectively.

3D-FAR algorithm use four subnetworks as $((X^+)(Y^+)(Z^*)$, $(X^-)(Y^+)(Z^*)$, $(X^+)(Y^-)(Z^*)$, $(X^-)(Y^-)(Z^*))$ where "$+$", "$-$" represents the channels on positive and negative direction and "$*$" mean "dont-care".

Feero et al. [25] has worked on 3D-NoC architectures to evaluate the benefits of 3D designs with respect to Latency, Throughput, Energy, and Area. In their evaluation, Feero et al. showed that the 3D designs increase the number of links by 29%, allowing more flits in the network and decreasing hops by 40%, which drops average packet energy proportionally.

Subnetwork	Virtual Channels
ENU	X0,Y0,Z0
ESU	X2,Y0,Z1
END	X1,Y1,Z0
WNU	X0,Y2,Z2
ESD	X3,Y1,Z1
WND	X1,Y3,Z2
WSD	X3,Y3,Z3
WSU	X2,Y2,Z3

Table 3.1. Virtual Channel Assignments for DyXYZ

LA-XYZ routing algorithm has been proposed Ahmed et al. LA-XYZ aim to minimize the latency by pre-computing the next port [26]. Look-Ahead routing improves the efficiency of pipelining to reduce the latency. LA-XYZ routing algorithm is implemented with 3D-OASIS-NoC router architecture [27].

Akbari et al. proposed AFRA deadlock-free routing algorithm for 3D networks [28]. The proposed algorithm mainly focused on faulty links on vertical links. If there are no faults detected on the path, the algorithm uses the ZXY path. The algorithm gives X-direction priority and uses the XZXY path in case of any fault on the vertical link. AFRA does not tolerate faults on horizontal links, and AFRA is not congestion aware. Ebrahimi et al. have improved AFRA algorithm in their proposed algorithm HamFA [29]. They have added a feature to tolerate faulty links on horizontal links as well. In addition, faulty information is distributed to all vertical links of the same row.

HamFA [29] uses Hamiltonian-based path selection to simplify complexity of the fault-tolerant algorithm. Table 3.2. show the potential output ports with respect to destination port.

Charif et al. proposed a routing solution for 3D-NoC designs using TSV in [30]. The proposed model partially connects the layer to form 3D-NoC. To avoid deadlok without increasing the complexity, the First-Last method includes two virtual channels for the east and north direction and one virtual channel for other directions. The channel designs give the ability to support any topology, such as asymmetric topologies. With FL-RuNS [31], an improved version of the First-Last method has been proposed. FL-RuNS supports fault tolerance to

Destination Positions		Y=Even	Y=Odd
N, D, ND, SD, WN, EN, END, WND	Z=Odd	D,N,E,D	D,N,W,D
N, U, NU, SU, WN, EN, ENU, WNU	Z=Even	U,N,E,U	U,N,W,U
U, EU, WU, ENU, WNU	Z=Odd	U,W,S,U	U,E,S,U
D, ED, WD, END, WND	Z=Even	D,W,S,D	D,E,S,D
ED, WD, ESD, WSD	Z=Odd	D,E,N,D	D,W,N,D
EU, WU, ESU, WSU	Z=Even	U,E,N,U	U,W,N,U
S, NU, SU, ES, WS, ESU, WSU	Z=Odd	U,S,W,U	U,S,E,U
S, ND, SD, ES, WS, ESD, WSD	Z=Even	D,S,W,D	D,S,E,D
W	Z=Odd	W,S,U,N	W,N,D,S
W	Z=Even	W,S,D,N	W,N,U,S
E	Z=Even	E,N	E,S
E	Z=Odd	E,N	E,S

Table 3.2. Output port options for HamFA [29]

overcome the failures of vertical TSV links with keeping the performance of the First-Last model.

HLAFT [32] uses local information as well as the look-ahead information to improve Look-Ahead-Fault-Tolerant [33] routing algorithm. LAFT ensures that calculated routes do not cross congested areas. LAFT uses Random-Access-Buffer(RAB) to ensure deadlock-freedom. RAB detects the flits that cause the deadlock and drops the flit to process the next flits to encounter the deadlock scenarios.

3.2. Related Work on Adaptive Routing

In recent years, with traffic congestion becoming a major issue after the processing elements become more powerful and the number of cores increased in an integrated circuit, adaptive routing algorithms draw more attention. There have been many researches and improvements on adaptive routing algorithms to avoid congestion. Adaptive routing algorithms can be divided into two categorize as partially and fully adaptive routing algorithms.

3.2.1. Partially Adaptive Routing

Partially adaptive routing algorithms aim to improve power and energy consumption performance while using less area. These algorithms generally solve the problem of deadlock but do not present fully adaptive routing to avoid congestion.

Turn restriction based routing algorithms can be examples of these algorithms such as Odd-Even(OE) routing [34], Negative-First routing algorithm [35], Planar-Adaptive routing algorithm [36]. To explain more detailly, the Negative-First routing algorithm arranges directions to give west and south directions(negative directions) priority. If the destination is on the northwest or southeast of the source, the algorithm sends the packet non-adaptively and uses the shortest path otherwise.

DyAD [37] routing algorithms benefit from the advantages provided by both deterministic and partially adaptive routing algorithms. DyAD switches between adaptive and deterministic routing depending on the traffic load of the network. Each router makes its decision on using either deterministic or adaptive routing. When the network becomes more congested, the algorithm avoids using congested links.

Gratz et al. has proposed Regional Congestion-Aware(RCA) [38] routing algorithms to propagate local congestion information with all nodes in the same row and column. At each hop, congestion information is propagated in the network. With the RCA routing algorithm, every node in the network has a sense of traffic load information on the network. The problem with RCA is in the path selection phase routing algorithm to evaluate all nodes' congestion information in the same direction. Assess the traffic congestion of nodes that is not in the path can cause non-optimal path selection. DBAR [39] routing algorithm has been proposed by Ma Sheng et al. to solve this problem. DBAR uses destination location as a metric for path selection to avoid using the congestion information beyond nodes. DBAR routing algorithm uses both local and regional traffic information and offers a more effective routing algorithm.

NoP [40] routing algorithm uses the congestion information from the nodes on the minimal path to the destination and within two hops. NoP algorithm does not have enough information to be able to select optimal paths for all scenarios.

CATRA [41] routing algorithm is another regional algorithm. Unlike other regional algorithms, which uses congestion information of neighboring routers on the same row or column, CATRA uses the congestion information in the trapezoid positions. CATRA method aims to efficiently distribute the traffic load by using both local congestion information and global traffic load information.

CATRA, NoP, and DBAR are all based on the mad-y [6]. Mad-y is minimal adaptive routing algorithm uses two virtual channel on Y dimension. The mad-y method restricts some of

the possible turns to avoid deadlock in the system. Mad-y uses a numbering mechanism and only allows turns in ascending order to prove deadlock freedom.

3.2.2. Fully Adaptive Routing

Fully adaptive routing algorithms decide the route between destination and current node at run time depending on the routing information collected from the neighboring routers or the routing table. In these routing algorithms, routers share network and traffic information with each other. Storing and updating these routing tables takes power and energy. Thus power consumption is higher compare to different routing algorithms.

An adaptive routing algorithm DyXY [42] has also proposed to use local congestion information to propagate messages in either X or Y direction depending on the allocation of the input buffers. Analyzing the usage rate of the input rates gives local information of the neighboring routers in that direction. To improve the DyXY routing algorithm, the Enhanced DyXY routing algorithm EDXY [43] has been proposed. In EDXY, congestion information is shared to its row and column nodes with additional wires.

Routing algorithms such as HARAQ [7], CARM [44], FDWR [45] use global information of the network to utilize fully adaptive routing algorithm with using Q-Tables to perform routing.

Congestion Adaptive Routing Method (CARM) [44] is a congestion aware non-minimal routing algorithm. CARM is proposed based on the mad-y [6] while increasing the efficiency of the virtual channels. CARM expands and updates the restrictions of the mad-y method as seen in Figure 3.1.

HARAQ [7] is another fully adaptive routing algorithm that uses Q-Learning. HARAQ uses Q-Tables for congestion awareness. HARAQ uses two virtual channels on the Y dimension and uses the structure to prohibit some turns to avoid deadlock. Although prohibited turns are determined using the mad-y method, they also analyzed all possible turns to detect any possible cycles and allow some of the 180-degree turns by taking the numbering into considerations.

Puthal et al. use another approach to use Q-Learning in their proposed C-Routing algorithm [46]. In C-Routing network is divided into clusters to reduce the Q-Table size and save space.

Figure 3.1. Turn models for (a) Mad-y and (b) LEAR (c) CARM

C-Routing uses both deterministic and adaptive routing algorithms. Deterministic routing is used for the north direction to avoid deadlock and livelock situations. The clustering-based approach divides the Q-Table into two sections: cluster part and switch part as seen from Figure 3.2. Cluster part have a size of $c \times m \times k$ where c, m and k are number of clusters, number of output channels and size of each entry respectively. $(l+c) \times m \times k$ is the total size of Q-Table where l represents the number of switches.

Figure 3.2. Q-Table formats for (a) Q-Routing and (b) C-Routing

Ebrahimi et al. proposed LEAR based on the mad-y method. LEAR utilize two virtual channel on Y dimension. LEAR uses the same turns as mad-y and imports 180-degree turns from virtual channel 1 (vc1) to virtual channel 2 (vc2) as these turns do not create cycles. The turns that LEAR allows can be seen from Figure 3.1.

4. CONGESTION-AWARE ADAPTIVE ROUTING ALGORITHM

Improving the performance of the NoC routing in 3D mesh topologies is the main objective of this thesis. Previously Q-Learning-based routing algorithms are used in 2D topologies using a deterministic approach for the third dimension. This thesis aims to improve the latency performance of 3D-NoCs by adapting a partially adaptive routing algorithm in the Z dimension. This chapter introduces the congestion-aware highly adaptive routing algorithm for 3D-NoCs.

This chapter introduces the proposed model, deadlock avoidance approach, and the used routing algorithm. In section 4.1. proposed model and its operation, in general, is introduced. The approach for deadlock avoidance is explained in section 4.2. Section 4.3. explains the routing algorithm and usage of Q-Learning and Q-Table.

4.1. Introduction

In this work, Congestion-Aware Adaptive Routing Algorithm Design for 3D Network-on-Chip is proposed that uses reinforcement learning to distribute the traffic load, avoid congestions to find the optimal paths for the health of the whole network. The proposed method is developed based on 3D Mesh topology as seen in Figure 4.1. with using Wormhole packet switching for packet transfer.

Figure 4.1. 4x4x4 NoC Example.

4.2. Deadlock Avoidance

In 2D-NoC, each router has five output ports (north (N), south (S), east (E), west (W), local (L)). Each incoming packet to a router has three possible turns to reach the destination. If an adaptive routing is selected, unbalanced traffic may cause deadlock problems. To avoid deadlock, we use one channel on the x dimension while there are two virtual channels (VC) on the y dimension based on the mad-y method [6]. Figure 4.2. illustrates the router used in the mad-y network (North-VC1 (N1), North-VC2 (N2), South-VC1(S1), South-VC2(S2), East(E), West(W), and Local(L)). In 3D-NoC, the routers have also up (U) and down (D) ports to send packets to different layers in the z dimension.

Figure 4.2. Router representation for the mad-y method in 2D-NoC.

There are sixteen possible 90-degree turns with six channels. Deadlock can occur in 0-degree, 90-degree, and 180-degree turns. The mad-y method prohibits some possible turns to avoid deadlock in 2D-NoC. These turns may cause eight cycles, as shown in Figure 4.3. If we eliminate at least one turn from each cycle as shown in Figure 4.4.a, we can prevent cycles in 90-degree turns resulting in at most twelve possible turns. When it comes to 0-degree turns, two types are possible; turns that change the virtual channel and the turns that do not. All combinations of the former 0-degree turns are allowed in mad-y routing. For the latter, it only turns that transit from VC1 to VC2 is permitted, as shown in Figure 4.4.b.

Figure 4.3. Eight possible cycles with 90-degree turns.

Figure 4.4. Allowed turns for (a) 90-degree, (b) 0-degree, and (c) 180-degree.

The mad-y method is a minimal routing algorithm; it prohibits 180-degree turns and only allows turns in ascending order. However, authors of [7] proposed a new routing method allowing 180-degree turns. This method modifies the numbering mechanism of mad-y and allows the 180-degree turns that do not cause deadlock. The numbering mechanism can be seen in Figure 4.5. Figure 4.4.(c) shows the allowed 180-degree turns.

With allowing turns with ascending orders, we ensure that no cyclic dependency can occur between any channels. Thus, proving that this method is deadlock-free. Also, after a message is transmitted in the east direction, the message can no longer be routed to the west direction. Therefore, the message can move in the east direction after reaching the leftmost column for

the worst-case scenario. This proves that the message can reach its destination after several hops, and it is livelock-free.

Figure 4.5. Channel numbering for (a) mad-y method and (b) HARA method.

Prohibiting the turns solves the potential deadlock problem. However, it does not guarantee that there will be an ideal path to the destination. To avoid such situations, a lookup table introduced after analyzing all combinations of input channels and destinations. Table 4.1. shows all potential output channels for all varieties.

Deadlock may also occur in intra-layer communication in 3D-NoCs. To avoid this, we prohibit the turns to the previous layer. In other words, if a packet is sent from layer A to B, it can never go back to layer A.

Although prohibiting some turns helps avoid deadlock, it may also increase the latencies. Additionally, uneven traffic loads on the network may further increase the latencies. In section 4.3. we present our Q-Learning-based routing algorithm that tries to find the best possible path by learning the previous network loads.

To avoid such situations, we analyzed all combinations of output channels and destinations. We used a lookup table to guarantee that the ideal path will not be prohibited on the sent router. Table 4.1. shows potential output ports for each message configuration. Figure 4.12. shows the output channel selection algorithm for packet transfers.

Table 4.1. Potential output channels for each input channel and destination.

Input Chn. \ Destination	N	S	E	W	NE	NW	SE	SW
L	N1, N2, S1, W	N1, S1, S2, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W
N1	N1, N2, S1, W	N1, S1, S2, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W
N2	N2	S2	N2, S2, E	-	N2, S2, E	-	N2, S2, E	-
S1	N1, N2, W	N1, S2, W	N1, N2, S2, E, W	N1, W	N1, N2, S2, E, W	N1, W	N1, N2, S2, E, W	N1, W
S2	N2	-	N2, E	-	N2, E	-	N2, E	-
E	N1, N2, S1, W	N1, S1, S2, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W	N1, N2, S1, S2, E, W	N1, S1, W
W	N2	S2	N2, S2, E	-	N2, S2, E	-	N2, S2, E	-

4.3. Routing Algorithm and Q-Table

Q-Learning is in the family of reinforcement learning. This machine-learning algorithm learns the environment from the previous actions and makes the next move based on this learning. The steps in our routing problem are decided based on the traffic rates of the destination region in an attempt to minimize the latency of the traveling packet. Routing algorithms that use Q-Learning are called Q-Routing [47]. The main goal of these routing algorithms is to reduce latency. Q-Learning-based routing algorithms use Q-Tables in each switch to select the ideal output channel with respect to Q-Values. Q-Tables store Q-Values which are the expected latency for a message to be delivered to the destination.

4.3.1. Q-Table Format

We keep a Q-Table that store expected latency information for all output channel to select the output channel with respect to this congestion information. Each cell of the Q-Table contains a value for the direction of the destination and possible output channels. The higher the value in a cell, the more congested the region of the output channel. The highest potential value in a cell can be fifteen. The Q-Table values represent the latency for sending a packet from the selected port to the corresponding direction as seen in example Q-Table 4.2.. Every router has a Q-Table to choose the shortest path.

We have used Region-based routing (R-Routing) to reduce the Q-Table size compare to Q-Routing and C-Routing. Q-Table size is fixed for each router regardless of the network size. The Q-Table size is $8x6$ for eight different destinations (N, S, E, W, NE, NW, SE, SW) and six different output channels (N1, N2, S1, S2, E, W). Q-Tables size of Q-Routing is nxm where n corresponds to the number of nodes in the network, and m corresponds to the number of output channels.

Initially, the Q-Table cells are set to zero for the minimal path output channels and set to eight for others to force minimal path routing for congestion-free networks. As we can see from Table 4.2. some cells have no entry since some turns are not allowed in the mad-y method. The possible directions are selected by the channel selection algorithm given in Figure 4.12. The tables are initialized to encourage the minimal path for non-congested networks. To use minimal paths nor low traffic scenarios, non-minimal paths are not allowed to have a value less than eight.

4.3.2. Feedback Message

Q-Routing model uses run time information to learn network information based on the feedback messages and information from the neighboring routers. After processing the tail flit of each incoming message, every router generates a feedback message to send back to the last router with the information of its expected latency for sending the message to the destination and the time spent for the previous hop. Feedback messages are created for the messages from North, South, East, and West with the additional link in 2D. Feedback messages contain local latency (L), global latency (gL), direction (Dir), and virtual channel (VC). The latency(L) value uses 3-bit information to indicate the waiting time between reaching the input buffer and successfully transmitted to the next router or local processing unit. Latency value is encoded as shown in Figure 4.6. Global latency (gL) uses 4-bit information indicating its expected latency to send the message to its destination. Q-Table has a maximum value of 15. Thus Global latency used 4-bit data. Virtual channel (VC) information shows which virtual channel has been used to receive the message. VC uses 1-bit information ("1" indicates virtual channel 2). Direction(Dir) shows where the router should send the message for the first router and encoded as shown in Figure 4.7. The reason for storing the output channel and direction information to store minimum information for each message in the router.

Figure 4.8. Feedback packet.

Figure 4.6. Encoding the latency information

```
Delay : Waiting time between reaching the input buffer
            and successfully transmitted to next router
AMT   : Average Message Time
L     : Latency
-----------------------------------------------------------------------
if (Delay < 1xAMT)
    L = 0;
else if (Delay < 2xAMT)
    L = 1;
else if (Delay < 3xAMT)
    L = 2;
else if (Delay < 5xAMT)
    L = 3;
else if (Delay < 9xAMT)
    L = 4;
else if (Delay < 15xAMT)
    L = 5;
else if (Delay < 27xAMT)
    L = 6;
else
    L = 7;
```

Figure 4.7. Encoding the direction information

```
Enumeration {
    NORTH     : 0,
    SOUTH     : 1,
    EAST      : 2,
    WEST      : 3,
    NORTHEAST : 4,
    NORTHWEST : 5,
    SOUTHEAST : 6,
    SOUTHWEST : 7
}
```

For example, when router 0 is trying to send a message to router 5 for the given network in Figure 4.9. Let's assume router 4 gets the message first from the S2 input channel and selected the East channel to send the message. Let's assume the time spent from router 0 to router 4 is 5xAMT, and the Q-Table value for sending a message to router 5 with the East output channel is 12. Router 4 will create a feedback packet as follows :

- **Lateny :** The Latency information encoded as shown in Figure 4.6. (3).
- **Global Lateny :** Expected latency information to send a message from router 4 to router 5 (12).

Table 4.2. An example Q-Table.

Direction \ Output Chn.	N1	N2	S1	S2	E	W
N	3	4	8	8	8	8
S	8	8	6	5	8	8
E	8	8	8	8	2	8
W	8	8	8	8	8	3
NE	7	7	8	8	6	8
NW	4	4	8	8	8	5
SE	8	8	4	4	5	8
SW	8	8	5	5	8	7

- **Direction :** Router 5 is on northeast of router 0 (4):
- **VC :** The virtual channel which router 0 used to transfer the message (1):

Figure 4.9. 2D Network

When the message arrives at the router X from router Y, it updates its Q-Table using the following equation. We have compared different learning rate values to select the optimum value. Figure 4.10. and Figure 4.11. shows the learning rate comparisons under different traffic models. In this equation, 0.25 is the learning ratio. The learning ratio has been selected as 0.25 to give a higher value to long-term cumulative value rather than short-term spikes and short trends in the network. This approach has advantages for stable networks in that each processing element primarily working with the same intensity over time. However, it also has disadvantages for volatile networks and has a slow response for flexible systems.

$$Q_X[Dir][OutC] = 0.25 * (gL_Y + L_Y) + 0.75 * Q_X[Dir][OutC] \tag{2}$$

Figure 4.10. Learning rate comparison under uniform test model.

Figure 4.11. Learning rate comparison under hotspot test model.

4.3.3. H Value Transfer

Each router stores an H value. H value is the average value of minimal path values on Q-Table. H values reflect the congestion information for each router and its neighboring nodes. H values and transferred to the neighboring router if the message received from Up or Down. We decide if the message will move on the same layer or be sent to the next layer

based on the traffic loads using H value. If the traffic rate of the router on the destination layer direction (i.e., $H[Up]$ and $H[Down]$ in Figure 4.12.) is smaller than the current router (i.e., H[Local]), the packet is sent to the next layer. Otherwise, it is sent to the neighbor router in the same layer, which is selected based on our Q-Learning-based algorithm. We call this layer selection procedure *elevator-first* since we give priority to send packets to intra-layer routers first if possible. H value uses 6-bit value have more resolution. So if the average value is 9.6, the H value will be $9.6 * 4 = 38$.

4.3.4. Elevator First

In a 3D-NoC routing, based on the destination node's location and the node with the message, both can be in a different layer. We have proposed an Elevator-first routing algorithm to send the messages using the traffic load information for interlayer messages.

Figure 4.12. shows the proposed channel selection algorithm for adaptive packet transfer. If the router is the destination router, the message is sent to the local channel. If the (x,y) coordinates are the same as the current node while the destination is on different layer, the message uses the vertical links as output channel with respect to the location of the destination. If the destination is on a different layer and the (x,y) coordinates of the current node is different, the output channel is selected based on the traffic rates.

4.4. An Example

In Figure 4.14. we give an example of how output channels are selected and Q-tables are updated for the network shown in Figure 4.13.

Figure 4.13. shows an example of a 3D-NoC. The darker nodes represent higher traffic loads. In this figure, node 0 in layer 0 generate a message and select destination node as node 43 in layer 2. As node 0 receive the message from the local processor, node 0 can select all possible output channels (N1, N2, E, U). Since the U port has a higher traffic rate (i.e., H value), the North sends the message via virtual channel 1(vc1). Router 4 examines the H values of both itself and the upper layer to decide between intra-layer and inter-layer transfer. Upon examination, router 4 select the output channel with the lowest latency from the potential output channels(N1, N2, S1, S2, E). After processing the message to the output channel,

Figure 4.12. Output Channel Selection Algorithm

```
Dir : Destination Direction
InCh : Input Channel
OutCh : Potential Output Channel List
Dest_x,Dest_y,Dest_z : x,y,z coordinates of destination node
Curr_x,Curr_y,Curr_z : x,y,z coordinates of current node
-------------------------------------------------------------------
if (Dir==Local) {
    OutCh = Local;
} else if ((Dest_x==Curr_x) && (Dest_y==Curr_y)) {
    if (Dest_z > Curr_z)
        OutCh = U
    else
        OutCh = D
} else if ((Dest_z > Curr_z) && (H[Up] < H[Local])) {
        OutCh = U
} else if ((Dest_z < Curr_z) && (H[Down] < H[Local])) {
        OutCh = D
} else {
    if (InCh==(L or N1 or S1 or E))
        OutCh += W + N1;
    if (InCh==(L or N1 or E))
        OutCh += S1;
    if (Dir==(E or NE or SE))
        OutCh += E;
    if (Dir==(N or E or NE or SE))
        OutCh += N2;
    if ((InCh != S2) && ( Dir==(S or E or NE or SE)))
        OutCh += S2;
}
```

router 4 creates a feedback message with local latency and expected latency information. Router 0 uses this information to update its Q-Table for packet transfer to NE direction for N2 output channel with the Equation (2). Router 0 will send the message to the neighboring less congested routers in the same layer until the upper router is less congested, which is router 5. When the message is processed by router 5 since the upper neighbor is now less congested than horizontal neighbors, router 5, will send the message to the upper layer from channel U. This will continue until the message arrives at its final destination.

Figure 4.13. Example for path selection.

Figure 4.14. Example for updating Q-Table.

5. EXPERIMENTAL RESULTS

To evaluate and compare the proposed method, many simulations have been conducted over a virtual simulation environment for both the proposed method and the HARAQ method. To have an understanding of the experiment results, the simulation environment and the performance metrics are described in section 5.1. The results are given in section 5.3. with discussions of the outputs.

5.1. Simulation Environment

The proposed model has been developed using the Noxim Simulation framework [8]. Noxim is a cycle-accurate NoC simulator developed in SystemC. SystemC is a C++ programming language extension and gives an opportunity for hardware to be modeled in software. Noxim is a flexible simulator organized in 2D mesh topology with features like employing and generating different routing algorithms, having simple source code. Based on the routing algorithm and PE module, Noxim can simulate a network and analyze the performance by calculating the performance metrics such as latency, average latency, throughput, total packet transfer, expected packet transfer, etc.

In this work, Noxim has been modified to support 3D topology and select several virtual channels for each direction. And a new router and new routing algorithm have been designed to evaluate the proposed method.

An additional channel is implemented for sending feedback messages without interfering with the routing channels. This would reduce the impact on bandwidth usage and create a more scalable and configurable approach for future works and more improvements. However, we can see that this approach would also increase the complexity of the hardware design and power consumption while reducing the software complexity. Feedback channels use two buffer locations on each input.

5.2. Performace Metrics

- **Latency (L)**
  Latency is the time spent until the packet is delivered to its destination. The latency

can be calculated by difference between the time that packet generation and the time it arrives to the destination PE.

$$L = T_a - T_g \tag{3}$$

- **Throughput (T)**
  Throughput is the rate of successfuly delivered message over time. Expected throughput can be calculated as in Equation (4). PIR is packet injection rate, N_f is number of flits per packet, N_{pe} is number of nodes in the network.

$$T = PIR * N_f * N_{pe} \tag{4}$$

The packet size is assigned to be ten flits long for the test environment, and the bandwidth is one flit per cycle. Additionally, every input channel has a buffer with a size of 8 flits. Each test is conducted to increase the number of injected packets per node per cycle ($flit/node/cycle$). The network is warmed up for 10.000 cycles to generate the Q-Table values ideally, and the tests are performed over 100.000 cycles to obtain the average performance outputs. To test the proposed routing algorithm and developed environment, both uniform and hotspot tests are conducted to monitor the performance for both best-case scenarios and stressed and congested networks.

5.3. Results Analysis and Comparison

To compare and evaluate the proposed method, 3D deterministic routing algorithm and elevator-first West-First [48] routing algorithm implemented. To understand the benefits of the third dimension for the adaptive routing algorithm, 2D HARAQ is also implemented with the same processing elements. West-First routing algorithm has been developed for 3D networks prioritizing inter-layer message transfer if the destination is not on the same layer.

5.3.1. Uniform Traffic Test

Uniform traffic tests have equal packet injection rates for every router, and each node has the same probability of being a destination. This method equally distributes the traffic load to the network and allows to analyze the best-case scenario. Figure 5.1. and Figure 5.2. shows the

latency and throughput performances under uniform traffic test. All algorithms have similar performance under low traffic as expected. As the injection rate increases, we can observe that 3D-NoC starts to outperform 2D-NoC.

The throughput results in Figure 5.2. show the same characteristic for the three methods. Figure 5.1. and Figure 5.2. show that our method achieves approximately 8% of improvement against 3D deterministic routing and 4% against the West-First routing algorithm for latency performance. The throughput performance of the proposed algorithm is 1.5% better than the 3D deterministic and 1% better than the West-First routing algorithm.

Figure 5.1. Latency under uniform traffic test.

5.3.2. Hotspot Traffic Test

In hotspot traffic, multiple processors are assigned to have higher injection rate. We used a coefficient value k to multiply the injection rate for the specified processors. The nodes in the coordinates (2,3,2) and (2,3,3) for 3D-NoC and (4,3) and (4,4) for 2D-NoC have higher injection rate. We have conducted our tests when $k = 5$. Figure 5.3. and Figure 5.4. shows the latency and throughput results for all algorithms. As seen from the figures, we have observed similar performance under low traffic loads. On relatively lower injection rates, we have started observing performance gain for non-deterministic routing algorithms. Similarly,

Figure 5.2. Throughput under uniform traffic test.

our method outperforms other methods in terms of latency and throughput values, with approximately 8% improvement obtained compared to 3D deterministic and approximately 3% improvement obtained compared to the West-First routing algorithm.

Figure 5.3. Latency under hotspot traffic test.

Figure 5.4. Throughput under hotspot traffic test.

6. CONCLUSION

6.1. Conclusion

In this work, we have proposed a traffic-aware highly adaptive routing algorithm for 3D-NoCs inspired by the Q-Learning algorithm for its path selection algorithm. The proposed method uses two virtual channels on the Y dimension for a better cost and performance ratio. We have proposed a congestion-aware communication for inter-layer communication while already used a Q-Learning-based routing algorithm for 3D networks. This approach also aims to avoid deadlock and livelock probabilities. To avoid congested areas, the Q-Learning system has been used to estimate the latency from each output to each 2D direction. Traffic-Aware routing has been used for path selection for interlayer packets.

Compared to similar works, the proposed congestion-aware adaptive routing algorithm shows latency and throughput improvements for 3D networks. Using two virtual channels for only the Y dimension and using Q-Learning as a reinforcement learning approach allows a simple and modern design for improving the current NoC routing algorithms.

6.2. Future Research Directions

The proposed method is open to improvement with a fully adaptive routing algorithm on the Z dimension that allows 180-degree turns. Increasing the size of Q-Table to $26x8$ for 26 different destinations (N, S, E, W, U, D, NE, NW, SE, SW, UN, US, UW, UE, DN, DS, DW, DE, UNE, UNW, USE, USW, DNE, DNW, DSE, DSW) and six different output channels (N1, N2, S1, S2, E, W, U, D) can allow fully adaptive routing algorithm while increasing the cost for Q-Table size. Another improvement can be accomplished by increasing the Q-Table size for the 2D table to increase the resolution for the destination.

We are using an additional link not to share the bandwidth of the data link in our proposed method. This approach can cause hardware complexity for the network. Optimized communication can be developed to use the same link for data and feedback transfer.

REFERENCES

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

[2] 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**.

[3] Vasilis F Pavlidis and Eby G Friedman. 3-d topologies for networks-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(10):1081–1090, **2007**.

[4] Vasilis F Pavlidis, Ioannis Savidis, and Eby G Friedman. *Three-dimensional integrated circuit design*. Newnes, **2017**.

[5] Konstantinos Tatas, Kostas Siozios, Dimitrios Soudris, and Axel Jantsch. *Designing 2D and 3D network-on-chip architectures*. IKEEBOOK-2017-046. Springer, **2014**.

[6] Christopher J Glass and Lionel M Ni. Maximally fully adaptive routing in 2d meshes. In *In International Conference on Parallel Processing, volume I*. Citeseer, **1992**.

[7] 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**.

[8] Vincenzo Catania, Andrea Mineo, Salvatore Monteleone, Maurizio Palesi, and Davide Patti. Noxim: An open, extensible and cycle-accurate network on chip simulator. In *2015 IEEE 26th international conference on application-specific systems, architectures and processors (ASAP)*, pages 162–163. IEEE, **2015**.

[9] Nir Magen, Avinoam Kolodny, Uri Weiser, and Nachum Shamir. Interconnect-power dissipation in a microprocessor. In *Proceedings of the 2004 international workshop on System level interconnect prediction*, pages 7–13. **2004**.

[10] David Flynn. Amba: enabling reusable on-chip designs. *IEEE micro*, 17(4):20–27, **1997**.

[11] W Peterson et al. Wishbone soc architecture specification. *Revision B*, 2, **2002**.

[12] Avalon bus specification. `https://www.intel.cn/content/dam/altera-www/global/zh_CN/pdfs/literature/manual/mnl_avalon_bus.pdf`. Accessed: 2021-05-27.

[13] Keith Diefendorff, Rich Oehler, and Ron Hochsprung. Evolution of the powerpc architecture. *IEEE Micro*, 14(2):34–49, **1994**.

[14] William J Dally et al. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed systems*, 3(2):194–205, **1992**.

[15] William J Dally and Charles L Seitz. Deadlock-free message routing in multiprocessor interconnection networks. **1988**.

[16] William J Dally and Charles L Seitz. The torus routing chip. *Distributed computing*, 1(4):187–196, **1986**.

[17] Erno Salminen, Ari Kulmala, and Timo D Hamalainen. On network-on-chip comparison. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pages 503–510. IEEE, **2007**.

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

[19] Ville Rantala, Teijo Lehtonen, Juha Plosila, et al. *Network on chip routing algorithms*. Citeseer, **2006**.

[20] Fernando Moraes, Ney Calazans, Aline Mello, Leandro Möller, and Luciano Ost. Hermes: an infrastructure for low area overhead packet-switching networks on chip. *Integration*, 38(1):69–93, **2004**.

[21] Edwin Rijpkema, Kees Goossens, Andrei Rădulescu, John Dielissen, Jef van Meerbergen, Paul Wielage, and Erwin Waterlander. Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip. *IEE Proceedings-Computers and Digital Techniques*, 150(5):294–302, **2003**.

[22] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, **1992**.

[23] Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, and Hannu Tenhunen. Fault-tolerant method with distributed monitoring and management technique for 3d stacked meshes. In *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)*, pages 93–98. IEEE, **2013**.

[24] Masoumeh Ebrahimi, Xin Chang, Masoud Daneshtalab, Juha Plosila, Pasi Liljeberg, and Hannu Tenhunen. Dyxyz: Fully adaptive routing algorithm for 3d nocs. In *2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, pages 499–503. IEEE, **2013**.

[25] Brett Feero and Partha Pratim Pande. Performance evaluation for three-dimensional networks-on-chip. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI'07)*, pages 305–310. IEEE, **2007**.

[26] Akram Ben Ahmed and Abderazek Ben Abdallah. La-xyz: low latency, high throughput look-ahead routing algorithm for 3d network-on-chip (3d-noc) architecture. In *2012 IEEE 6th International Symposium on Embedded Multicore SoCs*, pages 167–174. IEEE, **2012**.

[27] Akram Ben Ahmed, Abderazek Ben Abdallah, and Kenichi Kuroda. Architecture and design of efficient 3d network-on-chip (3d noc) for custom multicore soc. In *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, pages 67–73. IEEE, **2010**.

[28] Sara Akbari, Ali Shafiee, Mahmoud Fathy, and Reza Berangi. Afra: A low cost high performance reliable routing for 3d mesh nocs. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 332–337. IEEE, **2012**.

[29] Masoumeh Ebrahimi, Masoud Daneshtalab, and Juha Plosila. Fault-tolerant routing algorithm for 3d noc using hamiltonian path strategy. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1601–1604. IEEE, **2013**.

[30] Amir Charif, Alexandre Coelho, Masoumeh Ebrahimi, Nader Bagherzadeh, and Nacer-Eddine Zergainoh. First-last: A cost-effective adaptive routing solution for tsv-based three-dimensional networks-on-chip. *IEEE Transactions on Computers*, 67(10):1430–1444, **2018**.

[31] Alexandre Coelho, Amir Charif, Nacer-Eddine Zergainoh, and Raoul Velazco. Fl-runs: A high-performance and runtime reconfigurable fault-tolerant routing scheme for partially connected three-dimensional networks on chip. *IEEE Transactions on Nanotechnology*, 18:806–818, **2019**.

[32] Akram Ben Ahmed and Abderazek Ben Abdallah. Graceful deadlock-free fault-tolerant routing algorithm for 3d network-on-chip architectures. *Journal of Parallel and Distributed Computing*, 74(4):2229–2240, **2014**.

[33] Akram Ben Ahmed and Abderazek Ben Abdallah. Architecture and design of high-throughput, low-latency, and fault-tolerant routing algorithm for 3d-network-on-chip (3d-noc). *The Journal of Supercomputing*, 66(3):1507–1532, **2013**.

[34] Ge-Ming Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on parallel and distributed systems*, 11(7):729–738, **2000**.

[35] Christopher J Glass and Lionel M Ni. Fault-tolerant wormhole routing in meshes. In *FTCS-23 The Twenty-Third International Symposium on Fault-Tolerant Computing*, pages 240–249. IEEE, **1993**.

[36] Andrew A Chien and Jae H Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. *ACM SIGARCH Computer Architecture News*, 20(2):268–277, **1992**.

[37] Jingcao Hu and Radu Marculescu. Dyad: smart routing for networks-on-chip. In *Proceedings of the 41st annual Design Automation Conference*, pages 260–263. **2004**.

[38] Paul Gratz, Boris Grot, and Stephen W Keckler. Regional congestion awareness for load balance in networks-on-chip. In *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pages 203–214. IEEE, **2008**.

[39] Sheng Ma, Natalie Enright Jerger, and Zhiying Wang. Dbar: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip. In *Proceedings of the 38th annual international symposium on Computer architecture*, pages 413–424. **2011**.

[40] Giuseppe Ascia, Vincenzo Catania, Maurizio Palesi, and Davide Patti. Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip. *IEEE Transactions on Computers*, 57(6):809–820, **2008**.

[41] Masoumeh Ebrahimi, Masoud Daneshtalab, Pasi Liljeberg, Juha Plosila, and Hannu Tenhunen. Catra-congestion aware trapezoid-based routing algorithm for on-chip networks. In *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 320–325. IEEE, **2012**.

[42] Ming Li, Qing-An Zeng, and Wen-Ben Jone. Dyxy: a proximity congestion-aware deadlock-free dynamic routing method for network on chip. In *Proceedings of the 43rd annual Design Automation Conference*, pages 849–852. **2006**.

[43] Pejman Lotfi-Kamran, Amir-Mohammad Rahmani, Masoud Daneshtalab, Ali Afzali-Kusha, and Zainalabedin Navabi. Edxy–a low cost congestion-aware routing algorithm for network-on-chips. *Journal of Systems Architecture*, 56(7):256–264, **2010**.

[44] Manoj Kumar, Vijay Laxmi, Manoj Singh Gaur, Seok-Bum Ko, and Mark Zwolinski. Carm: congestion adaptive routing method for on chip networks. In *2014 27th International Conference on VLSI Design and 2014 13th International Conference on Embedded Systems*, pages 240–245. IEEE, **2014**.

[45] Timo Schonwald, Jochen Zimmermann, Oliver Bringmann, and Wolfgang Rosenstiel. Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures. In *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pages 527–534. IEEE, **2007**.

[46] Manas Kumar Puthal, Virendra Singh, Manoj Singh Gaur, and Vijay Laxmi. C-routing: An adaptive hierarchical noc routing methodology. In *2011 IEEE/IFIP 19th International Conference on VLSI and System-on-chip*, pages 392–397. IEEE, **2011**.

[47] Justin A Boyan and Michael L Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in neural information processing systems*, pages 671–678. Citeseer, **1994**.

[48] Christopher J Glass and Lionel M Ni. The turn model for adaptive routing. *ACM SIGARCH Computer Architecture News*, 20(2):278–287, **1992**.