ON ENHANCING THE PERFORMANCE OF BUFFERLESS NETWORK-ON-CHIP

By

Mohamed Assem Abd ElMohsen Ibrahim

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Computer Engineering

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2016

ON ENHANCING THE PERFORMANCE OF BUFFERLESS NETWORK-ON-CHIP

By
Mohamed Assem Abd ElMohsen Ibrahim

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Computer Engineering

Under the Supervision of

Dr. Hatem M. El-Boghdadi

Professor
Computer Engineering Department
Faculty of Engineering, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2016

ON ENHANCING THE PERFORMANCE OF BUFFERLESS NETWORK-ON-CHIP

By
Mohamed Assem Abd ElMohsen Ibrahim

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Computer Engineering

Approved by the
Examining Committee

_____
Dr. Hatem M. El-Boghdadi, Thesis Main Advisor
- Professor at the Faculty of Engineering, Cairo University

_____
Dr. Amr G. Wassal, Internal Examiner
- Associate Professor at the Faculty of Engineering, Cairo University

_____
Prof. Dr. Mohammad Z. Abdel Majeed, External Examiner
- Professor at the Faculty of Engineering, Al-Azhar University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2016

Engineer's Name:	Mohamed Assem Abd ElMohsen Ibrahim
Date of Birth:	07/07/1988
Nationality:	Egyptian
E-mail:	mohamedassem@eng.cu.edu.eg
Phone:	+2 01007846437
Address:	407N, Pyramids Gardens Giza, Egypt

Registration Date:	01/10/2010
Awarding Date:	…./…./2016
Degree:	Master of Science
Department:	Computer Engineering
Supervisors:	Dr. Hatem Mahmoud El-Boghdadi
Examiners:	Prof. Dr. Mohammad Zaki Abdel-Majeed (External examiner) - Professor at the Faculty of Engineering, Al-Azhar University Dr. Amr Jalal El-Deen Wassal (Internal examiner) Dr. Hatem Mahmoud El-Boghdadi (Thesis main advisor)

Title of Thesis:
On Enhancing the Performance of Bufferless Network-on-Chip

Key Words:
Bufferless Network-on-Chip; Selection Function; Maximum Flexibility; Ranking Policies; Congestion Management;

Summary:

With the arrival of chip multiprocessor systems, Network-on-Chip (NoC) has started to form the backbone of communication within a microprocessor chip. However, unfortunately, the performance of NoC is bounded by the limited power and area budgets. Bufferless NoC has emerged as a solution to reduce power and area. Bufferless NoC eliminates the buffers used for routing and/or flow control and handle contention using packet dropping or packet deflection. In this thesis, we focus on enhancing the performance (latency and deflection count) of deflection-based bufferless NoC running latency-sensitive applications.

First, we present an analytical study for the traffic in bufferless NoC under the Maximum Flexibility (MaxFlex) selection function with different step sizes. We also provide an experimental study under MaxFlex. Simulation results show that with large values of step size, the latency could be reduced by 97% over using Straight Line selection function. The proposed analysis explains the outperforming experimental results.

Then we propose different flit ranking policies that focus on decreasing the deflection count of the flits. Simulation results show that the proposed ranking policies can reduce the latency by up to 58% compared to Oldest First policy.

Finally, we consider relaxing the effect of congestion in bufferless NoC under high injection rate. We propose two approaches for congestion prevention. The first considers running applications on NoC with extra nodes. The second considers dividing a certain load into a sequence of lighter loads. Simulation results show that the proposed approaches enhance the latency by up to 61% in addition to operating at higher injections rates.

Acknowledgments

بسم الله الرحمن الرحيم

"سُبْحَانَكَ لَا عِلْمَ لَنَا إِلَّا مَا عَلَّمْتَنَا ۖ إِنَّكَ أَنتَ الْعَلِيمُ الْحَكِيمُ﴿٣٢﴾" البقرة.
“Glory to You (O Lord), we have no knowledge except what you have taught us. Indeed, it is You who is the knowing, the wise (32)” Al-baqarah

" وَمَا تَوْفِيقِي إِلَّا بِاللَّـهِ ۚ عَلَيْهِ تَوَكَّلْتُ وَإِلَيْهِ أُنِيبُ ﴿٨٨﴾" هود.
“And my success is not but through Allah. Upon him I have relied and to Him I return (88)” Hood.

I would like to express my sincere gratitude to my advisor, Dr. Hatem El-Boghdadi, for his huge support, patience and immense knowledge. His guidance helped me in all the time of working on this thesis. I could not have imagined having a better advisor and mentor for my masters study.

Also, I take this opportunity to express gratitude to all of the Computer Engineering department members for their help and support.

I would like to thank my family for their encouragement, support, and attention without which I would never have made it to the end.

Last but not the least; I would like to thank my wife, Yousra, for being always there for me and for her support and kindness. My uttermost gratitude goes to Allah that I met her in such critical point in my life.

Table of Contents

ACKNOWLEDGMENTS ..... **I**

TABLE OF CONTENTS ..... **II**

LIST OF TABLES ..... **V**

LIST OF FIGURES ..... **VI**

ABSTRACT ..... **X**

CHAPTER 1 : INTRODUCTION **1**

1.1. BASIC BACKGROUND 2
1.1.1. Buffered NoCs 2
1.1.2. Bufferless NoCs 3
1.1.3. Selection Functions 3
1.1.4. Maximum Flexibility Selection Function 3
1.1.5. Flit Ranking Policies 4
1.1.6. Congestion Management 4
1.2. RELATED WORK 5
1.3. SCOPE OF THE THESIS 6
1.3.1. Increasing and Varying Step Size Under MaxFlex 7
1.3.2. Evaluating Flit Ranking Policies 7
1.3.3. Preventing the Congestion 8
1.4. CONTRIBUTION OF THE THESIS 8
1.5. ORGANIZATION OF THE THESIS 9

CHAPTER 2 : BACKGROUND **10**

2.1. INTERCONNECTION NETWORK 10
2.2. NETWORK-ON-CHIP (NOC) 11
2.3. BUFFERLESS NETWORK-ON-CHIP 12
2.4. SELECTION FUNCTIONS 13
2.5. FLIT RANKING POLICIES 14
2.6. CONGESTION MANAGEMENT 15

CHAPTER 3 : MODIFIED MAXIMUM FLEXIBILITY SELECTION FUNCTION **16**

3.1. PROPOSED APPROACH 16
3.2. ANALYSIS OF MMAXFLEX SELECTION FUNCTION 17
3.2.1. Type 1 Packets 19
3.2.2. Type 2 Packets 20
3.2.3. Type 3 Packets 20
3.2.4. Type 4 Packets 21
3.2.5. Type 5 Packets 21
3.2.6. Type 6 Packets 25
3.2.6.1. Type 6 (a) 25

3.2.6.2. Type 6 (b) 27
3.2.6.3. Type 6 (c) 29
3.2.6.4. Type 6 (d) 29
3.2.7. Type 7 Packets 29
3.2.7.1. Type 7 (a) 30
3.2.7.2. Type 7 (b) 30
3.2.7.3. Type 7 (c) 31
3.2.7.4. Type 7 (d) 31
3.2.8. Type 8 Packets 31
3.2.8.1. Type 8 (a) 35
3.2.8.2. Type 8 (b) 36
3.2.9. Type 9 Packets 36
3.2.9.1. Type 9 (a, c) 36
3.2.9.2. Type 9 (b, d) 37
3.2.10. Type 10 Packets 38
3.2.10.1. Type 10 (a, c) 38
3.2.10.2. Type 10 (b, d) 39
3.2.11. Type 11 Packets 40
3.2.11.1. Type 11 (a) 40
3.2.11.2. Type 11 (b) 42
3.2.11.3. Type 11 (c) 43
3.2.11.4. Type 11 (d) 43
3.2.12. Type 12 Packets 44
3.2.12.1. Type 12 (a) 44
3.2.12.2. Type 12 (b) 45
3.2.12.3. Type 12 (c) 46
3.2.12.4. Type 12 (d) 46
3.2.13. Summary of Packets Count Calculations 48
3.3. Proof of Packet Types Completeness 49
3.4. Packets Distribution Analysis Results 51
3.5. Experimental Setup 51
3.5.1. Experimental Methodology 52
3.5.2. Interconnection Network Model 52
3.5.3. Evaluation Metrics 52
3.6. Simulation Results 53
3.7. Estimation of the Value of the Step Size 55
3.8. Concluding Remarks 55

CHAPTER 4 : VARIABLE STEP SIZE MAXIMUM FLEXIBILITY SELECTION FUNCTION 56

4.1. Motivation 56
4.2. Proposed Variable Step Size Approaches 56
4.2.1. Using the Manhattan distance between NoC nodes (NMDVS) 57
4.2.2. Using the Manhattan distance between NoC regions (RMDVS) 58
4.2.3. Using In-Region and Out-Region routing (IORVS) 58
4.2.4. Using the Manhattan distance between NoC nodes for Out-Region routing (ORMDVS) 59
4.3. Simulation Results 59
4.4. Concluding Remarks 73

CHAPTER 5 : NEW FLIT RANKING POLICIES FOR DEFLECTION-BASED BUFFERLESS NOCS 75

5.1. MOTIVATION 75
5.1.1. Oldest First Ranking Policy (OF) 75
5.1.2. Most Deflection First Ranking Policy (MDF) 76
5.2. PROPOSED FLIT RANKING POLICIES 76
5.2.1. Deflection Age Ratio Ranking Policy (DAR) 76
5.2.2. Deflection Distance Ratio Ranking Policy (DDR) 77
5.2.3. Last Dimension Ranking Policy (LD) 77
5.3. SIMULATION RESULTS 77
5.4. CONCLUDING REMARKS 79

CHAPTER 6 : TIME-SENSITIVE CONGESTION MANAGEMENT MECHANISMS 80

6.1. MOTIVATION 80
6.2. PROPOSED APPROACHES 81
6.2.1. Using Larger NoCs (LNoC) 81
6.2.2. Using Sequential Injection (SI) 81
6.3. SIMULATION RESULTS 82
6.4. CONCLUDING REMARKS 85

CHAPTER 7 : DISCUSSION AND CONCLUSION 86

7.1. FUTURE WORK 86

REFERENCES 88

APPENDIX A: 2D MESH TERMINOLOGIES 92

PUBLICATIONS 94

List of Tables

Table 1: Up traffic passing through switch *C* 23
Table 2: Type 5 *Count* calculation for an increasing diagonal switches under up traffic using different *SS* values 24
Table 3: Down traffic passing through switch *C* 24
Table 4: Type 5 *Count* calculation for an increasing diagonal switches under down traffic using different *SS* values 24
Table 5: Up traffic passing through switch C_{Solid} 27
Table 6: Type 6(a) *Count* calculation for the solid diagonal switches under up traffic using different *SS* values 27
Table 7: Type 6(a) *Count* calculation for the dotted diagonal switches under up traffic using different *SS* values 27
Table 8: Down traffic communication passing through switch D_{Dotted} 28
Table 9: Type 6(b) *Count* calculation for the dotted diagonal switches under down traffic using different *SS* values 28
Table 10: Type 6(b) *Count* calculation for the solid diagonal switches under down traffic using different *SS* values 28
Table 11: Down traffic passing through switch *A* 33
Table 12: Summary for the data collected in Table 11 33
Table 13: Down traffic passing through switch *B* 33
Table 14: Down traffic passing through switch *C* 34
Table 15: Down traffic passing through switch *D* 34
Table 16: Up traffic passing through switch Z_{Solid} 42
Table 17: Up traffic passing through switch Y_{Solid} 45
Table 18: Formulas for different traffic types 47
Table 19: Common variables used in Table 19 49
Table 20: A and B values for up and down traffic 49
Table 21: Multiplier value for Type 8, Type 9 and Type 10 49
Table 22: Values for Type 9 up traffic communication 49
Table 23: First category cases 50
Table 24: Second category cases 50
Table 25: Step size to mesh dimension percentage 55

List of Figures

Figure 1: Generic switch in a 2D mesh 2
Figure 2: Example of interconnection network 10
Figure 3: Generic switch in a 2D mesh 11
Figure 4: The operation of MaxFlex selection function using step size of one 13
Figure 5: The operation of MaxFlex selection function using step size of one 17
Figure 6: Increasing and decreasing diagonals in a 2D mesh 18
Figure 7: Up and down traffic in 2D mesh 18
Figure 8: Location of $W(i,j)$ in 2D mesh row 20
Figure 9: Type 3 example for a row in 5x5 and 6x6 meshes 21
Figure 10: Type 3 *Count* calculation for a row switch in a 5x5 mesh 21
Figure 11: Location of $W(i,j)$ in 2D mesh diagonal 22
Figure 12: Type 5 example for an increasing diagonal 23
Figure 13: Type 6 example for an increasing diagonal under both up and down traffics 25
Figure 14: Location of $W(i,j)$ in 2D mesh diagonal 26
Figure 15: Type 7 example for an increasing diagonal under both up and down traffics 30
Figure 16: Type 8 example for an increasing diagonal in a 12x12 mesh 32
Figure 17: Procedure for counting the packets passing through a switch for Type 8(a) 35
Figure 18: Procedure for counting the packets passing through a switch for Type 8(b) 36
Figure 19: Procedure for counting the packets passing through a switch for Type 9(a, c) 37
Figure 20: Procedure for counting the packets passing through a switch for Type 9(b, d) 38
Figure 21: Procedure for counting the packets passing through a switch for Type 10(a, c) 39
Figure 22: Procedure for counting the packets passing through a switch for Type 10(b, d) 40
Figure 23: Location of $W(i,j)$ in 2D mesh 41
Figure 24: Type 11 example for an increasing diagonal under both up and down traffics 42
Figure 25: Type 12 example for an increasing diagonal under both up and down traffics 45
Figure 26: Number of packet passing through sample border and core switches over different fixed step size values 51
Figure 27: Average packet latency for different fixed step size values 53
Figure 28: Average deflection count for different fixed step size values 53
Figure 29: Average packet latency for different fixed step sizes at flit injection rate = 0.22 flit/cycle/node 53
Figure 30: Average packet latency for fixed step size of 8 compared with different selection functions 54
Figure 31: Average deflection count for fixed step size of 8 compared with different selection functions 54
Figure 32: 4x4 mesh divided into four 2x2 regions 57
Figure 33: Average packet latency for NMDVS using different % values 60
Figure 34: Average deflection count for NMDVS using different % values 60

Figure 35: Average packet latency for RMDVS compared with RMDVS`.....61
Figure 36: Average deflection count for RMDVS compared with RMDVS`.....61
Figure 37: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 1$ using 2x2 region size.....62
Figure 38: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 1$ using 5x5 region size.....62
Figure 39: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 1$ using 2x2 region size.....62
Figure 40: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 1$ using 5x5 region size.....62
Figure 41: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 2$ using 2x2 region size.....63
Figure 42: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 2$ using 5x5 region size.....63
Figure 43: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 2$ using 2x2 region size.....63
Figure 44: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 2$ using 5x5 region size.....63
Figure 45: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 3$ using 2x2 region size.....64
Figure 46: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 3$ using 5x5 region size.....64
Figure 47: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 3$ using 2x2 region size.....64
Figure 48: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 3$ using 5x5 region size.....64
Figure 49: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 4$ using 2x2 region size.....65
Figure 50: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 4$ using 5x5 region size.....65
Figure 51: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 4$ using 2x2 region size.....65
Figure 52: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 4$ using 5x5 region size.....65
Figure 53: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 5$ using 2x2 region size.....66
Figure 54: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 5$ using 5x5 region size.....66
Figure 55: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 5$ using 2x2 region size.....66
Figure 56: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 5$ using 5x5 region size.....66
Figure 57: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 6$ using 2x2 region size.....67
Figure 58: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 6$ using 5x5 region size.....67
Figure 59: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 6$ using 2x2 region size.....67

Figure 60: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 6$ using 5x5 region size 67
Figure 61: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 7$ using 2x2 region size 68
Figure 62: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 7$ using 5x5 region size 68
Figure 63: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 7$ using 2x2 region size 68
Figure 64: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 7$ using 5x5 region size 68
Figure 65: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 8$ using 2x2 region size 69
Figure 66: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 8$ using 5x5 region size 69
Figure 67: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 8$ using 2x2 region size 69
Figure 68: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 8$ using 5x5 region size 69
Figure 69: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 9$ using 2x2 region size 70
Figure 70: Average packet latency for different $SSOutRegion$ values under in $SSInRegion = 9$ using 5x5 region size 70
Figure 71: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 9$ using 2x2 region size 70
Figure 72: Average deflection count for different $SSOutRegion$ values under in $SSInRegion = 9$ using 5x5 region size 70
Figure 73: Average packet latency using different $SSInRegion$ values and 60% under 2x2 region size 72
Figure 74: Average deflection count using different $SSInRegion$ values and 60% under 2x2 region size 72
Figure 75: Average packet latency using different $SSInRegion$ values and 60% under 5x5 region size 72
Figure 76: Average deflection count using different $SSInRegion$ values and 60% under 5x5 region size 72
Figure 77: Average packet latency for different variable step size formulas 73
Figure 78: Average deflection count for different variable step size formulas 73
Figure 79: Average packet latency for different ranking policies 78
Figure 80: Average deflection count for different ranking policies 78
Figure 81: Average packet latency for LD enhancement over other ranking policies 78
Figure 82: Average deflection count for LD enhancement over other ranking policies 78
Figure 83: Using 4x4 mesh instead of 3x3 mesh 81
Figure 84: Example of two phase sequential injection 82
Figure 85: Average packet latency for fifteen nodes in different mesh sizes 83
Figure 86: Average deflection count for fifteen nodes in different mesh sizes 83
Figure 87: Average packet latency for different number of extra nodes in different locations in 10x10 mesh 84
Figure 88: Average deflection count for different number of extra nodes in different locations in 10x10 mesh 84

Figure 89: Average packet latency for two phase SI using different number of nodes in different locations in 10x10 mesh 85
Figure 90: Average deflection count for two phase SI using different number of nodes in different locations in 10x10 mesh 85
Figure 91: Main increasing and decreasing diagonals in 5x5 mesh.....92

Abstract

Network-on-Chip (NoC) is commonly used to connect different computing components. With the arrival of chip multiprocessor systems, NoC has started to form the backbone of communication between cores and memory within a microprocessor chip. Although NoC has started to form the backbone of communication between cores, the performance of such interconnection network is bounded by the limited power and area budgets. Bufferless NoC has emerged as a solution to reduce power and area. Bufferless NoC eliminates the buffers used for routing or flow control and handle contention using packet dropping or packet deflection.

We focus on enhancing the performance (in particular, packet latency and deflection count) of deflection-based bufferless NoC running latency-sensitive applications. We divide the work to focus on three aspects of NoC. First, we focus on selecting an output port for the outgoing packet. After that, we shift our focus to ranking the flits in order to select which one to serve first. Finally, we investigate relaxing the effect of congestion under high injection rate.

In the first part, we study the effect of Maximum Flexibility selection function (MaxFlex) on 2D bufferless meshes when a fixed or a variable step size is used. The selection function selects an output channel from a set of channels supplied by the routing function. MaxFlex is a well-known selection function that tries to maximize the number of routing choices as a packet approaches its destination. We investigate the distribution of packets through the NoC via increasing and/or varying the used step size as improving the distribution leads to better utilization and thus better performance. Simulation results show that using a larger step size can enhance the performance by up to 95% compared to using Straight Line selection function. Also, the results show that using variable step size enhances the performance compared to fixed step size by up to 29 %.

Concerning the second part, we devise and evaluate different flit ranking policies. A flit ranking policy chooses which flit should be served first, thus it determines which flit can select an output port first. In this work, we propose novel ranking policies that take the deflection behavior of the bufferless NoC into account. Via the experimental study, we compare these policies to the Oldest First (OF) ranking policy. Simulation results show that the performance of the proposed policies excels over fixed step size MaxFlex with OF as ranking policy by up to 58%.

Finally, we focus on congestion prevention for bufferless NoC running latency-sensitive applications. NoC congestion is one of the main roadblocks that prevent the bufferless NoC to operate under high injection rates. Thus, by relaxing the congestion, bufferless NoCs can approach the performance of buffered NoCs but without the extra cost of using buffers (power and area). To address this problem, we propose prevention mechanisms that target the deflection count of the flits. The proposed approaches aim to give more space for the flits to roam leading to fewer deflections which directly affects the overall packet latency. Via simulation, we show that the proposed approaches enhance the packet latency by 61% compared to fixed step size MaxFlex.

Chapter 1 : Introduction

In the last few years, there is an industry wide switch to many-core and multi-core systems. In such systems, the performance of the communication system is very critical to the performance of the whole system.

Network-on-Chip (NoC) has emerged as a solution for the limitations in the traditional communications approaches (e.g. buses) especially after the tremendous increase in the number of the communicating modules within a single silicon chip [1,2]. NoC is a group of switches connecting homogeneous or heterogeneous nodes in a multiple point-to-point fashion [3,4]. NoC switches forward the data to/from the nodes/switches over links equipped with input and output buffers.

Buffered NoCs became the de facto approach for communication between cores within chip as they are more scalable, reliable, and predictable. Buffered NoCs were shown to consume significant power and chip area. For instance, in the Intel Teraflops chip and the MIT RAW chip, NoC fabric consumes around 30% and 36% power respectively [5,6]. Focusing on a single NoC switch, a considerable fraction of power and area is used by the internal buffers of the switch. In [7,8], the buffers within a single switch consume around 37% power and 80% area. In addition to being heavy power and area consumers, buffered NoCs are more complex to design as they require extra handlers for packets placement and buffer overflow.

Bufferless NoC has emerged as a solution to decrease power and area requirements [9,10,11,12]. Bufferless NoC eliminates the buffers used within switches; which has a direct impact on power and area. In contrast to the traditional buffered NoC; when two packets compete for the same output port, the allocator either drops or deflects (misroute) the losing packet instead of buffering it. Dropped packet should be retransmitted again. On the other hand, deflected packet follows a non-productive port. Due to the hazards accompanying the dropping mechanism such as handling positive (ACK)/negative (NACK) acknowledgement (NACK buffers [9], NACK network [11]), storing the packet within the source node (extra storage), and retransmission (increase the total network load), in this thesis, we adapt the deflection approach.

Even though bufferless NoCs have their advantages regarding area and power consumption, they have their own problems. Eliminating buffers helps in decreasing the chip area and limiting the consumed power, but at the same time, the flits have no place to reside in case of port contention which leads to dropping or deflecting the flits. This dropping/deflecting mechanism results in increasing the NoC traffic volume which in turn consumes link bandwidth.

Both mechanisms under low to medium rates lightly affect the performance (packet latency and deflection count) leading to a performance approaching buffered NoCs. On the other hand, under high injection rates, the number of packets increases leading to more contention, as a result, using bufferless NoCs leads to reducing the total available bandwidth (as a result of increasing the traffic volume due to retransmitting the flits or deflecting the flits away from their destination) which eventually leads to a performance worse than buffered NoCs. Thus, bufferless NoC is shown generally to function efficiently under moderate loads and smaller NoC sizes [10].

In this thesis, we study several aspects of bufferless NoC to serve latency-sensitive applications. In other words, we aim to operate latency-sensitive applications on

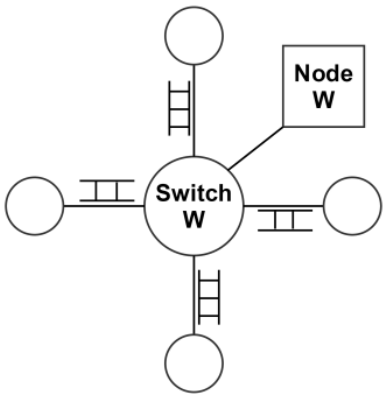

Figure 1: Generic switch in a 2D mesh

bufferless NoCs under high injection rates without inducing extra power or chip area usage. This work follows three tracks; enhancing performance through output selection functions, enhancing performance through flit ranking policies, and finally, enhancing performance through congestion prevention.

1.1. Basic Background

In this section, we formally introduce some notations that shall assist in describing the scope and contribution of this work. Specially: (1) buffered NoCs, (2) bufferless NoCs, (3) selection functions, (4) maximum flexibility, (5) flit ranking policies, and (6) congestion management. We now discuss these topics briefly.

1.1.1. Buffered NoCs

A 2D buffered NoC is a two dimensional array of nodes. Each node is connected to the network using a switch. The switches are connected in a multiple point-to-point fashion. Switches forward the data to/from the nodes and/or switches over links. Each link is equipped with input and output buffers. The data is delivered as packets where each packet is divided into several flow control units called flits. Topology defines the networks logical layout (connections). A sample switch in a 2D mesh is shown in Figure 1.

Buffered NoCs are used widely as a communication fabric. To handle the contention that may occur between two flits arriving simultaneously at an output port, buffered NoCs use the input buffer to store the incoming flits. By doing this, the switch can store the flits that lost the arbitration and forward the winning flits.

Buffered NoCs have the drawback of consuming significant power and area. For example, the NoC fabric in the Intel Teraflops chip and the MIT RAW chip consumes 30% and 36% respectively of the required power [5,6]. Also the network occupies large chip area (for example, 80% area [7,8]) due to buffer usage. Beside consuming power and area, buffered NoCs are complex to design due to the need to implement different scenarios for handling the buffers logic. One way to reduce the required power and chip area is to eliminate the buffers within the network; i.e. bufferless NoC.

1.1.2. Bufferless NoCs

Bufferless NoCs have been proposed to reduce the power and area consumption and to simplify the design process. This is done by removing the input and output buffers.

Bufferless NoCs handle the output port contention by either dropping the losing packet or by deflecting it. Bufferless NoCs that use the dropping mechanism chooses to drop the packet that lost the contention competition. By dropping the packet, bufferless NoCs have to retransmit this packet which leads to an increase in the network traffic and/or the hardware cost and design complexity.

The deflecting bufferless NoCs choose not to drop any contending packet. Instead, bufferless NoCs forward all the incoming packets to output ports even if it means to forward the packets through longer paths (non-productive ports). The deflecting buffered NoCs are preferred due to their simpler design, and less power and area cost.

However, using bufferless NoCs can cause degradation in the performance. A recent study [10] showed that the power and area gains exceed the degradation in the network performance when NoC load is low to medium, which matches many of the real-life applications.

1.1.3. Selection Functions

To route a packet successfully from a source node to a destination node, it is required to have a routing function and a selection function. The routing function calculates the path to follow between a source–destination pair and offers a set of output ports to get closer to the destination. The selection function selects an output port from the supplied set of ports.

Routing could be classified as deterministic or adaptive based on the selection function. Routing is deterministic if the selection function delivers the same port for each source-destination combination each time. On the other hand, routing is adaptive if the selection function delivers a port based on the network state, thus the selection function may deliver different port each time it is used [3].

Many selection functions exist for 2D meshes such as Straight Line (similar to dimension order routing DO) which favors *X* (or *Y*) dimension than *Y* (or *X*) till no more steps left in *X* (or *Y*) and then alternate to the other dimension i.e. *Y* (or *X*). Another selection function is Random Productive Port which selects one of the flit's productive ports randomly. One of the well-known selection functions is Maximum Flexibility.

1.1.4. Maximum Flexibility Selection Function

Maximum Flexibility (MaxFlex) is a selection function that is similar to the z^2 routing proposed in [13]. It selects the output port on the dimension with more hops to the destination (i.e. longest distance to the destination). By doing this, MaxFlex maximizes the number of productive ports provided by the routing function as the flit approaches its destination. In other words, MaxFlex prevents the flit from being stuck in one dimension leading to one productive port only.

MaxFlex tries to move the packets on a diagonal between the source and the destination. Packet initially follows the dimension with higher hop count. When it reaches a switch where the difference in the *X*-dimension is equal to the difference in

the *Y*-dimension, it follows the diagonal. The path of the diagonal is dependent on the step size used. Step size of *SS* means that a packet moves *SS* steps in *X*-dimension and then *SS* steps in *Y*-dimension.

MaxFlex causes the traffic to be concentrated in the central part of the network bisection as it tries to move on the diagonals. This leads to uneven switches utilization which degrades the performance [14].

1.1.5. Flit Ranking Policies

Under normal operation, a NoC switch can receive several flits at the same cycle from the neighboring switches and/or from the node connected to it. For example, if a switch, in a 2D mesh topology, is connected to four switches, it can receive up to maximum five flits at the same cycle. Each flit has its own destination and wants to pass through its productive port i.e. wants to get closer to its destination. How the switch determines the order by which it will serve the incoming flits is determined by the ranking policy. In other words, it determines which flit can select an output port first.

Different ranking policies lead to different service order for the flits travelling through the NoC. To be more specific, different policies lead to different arrival order for the NoC flits which leads to a different set of flits reaching their destination before the others. The delivery of a certain set can result in a better performance but this is not the only factor. Flit ranking schemes have a direct effect on livelock property in the NoC [3,4]. Some schemes can result in packets travelling indefinitely the NoC and never reaching their destinations.

In bufferless NoC, due to the buffers elimination, the ranking policies have a greater effect on the overall performance as the flit that fails to get its productive port will be forced to take a detour.

Different criteria can be used as ranking policies. For example; Oldest First (OF) policy ensures there is a total age order among flits and prioritizes older flits, Closest First (CF) policy prioritizes the flits with smaller distance to their destinations before flits whose remaining distance is larger, Most Deflections First (MDF) policy gives higher priority to the flits with more deflections, and Round Robin (RR) policy ranks the flits from different input ports in a round robin fashion.

1.1.6. Congestion Management

Under high injection rates, the traffic volume in the NoC increases causing more strain on the NoC links and buffers. When the NoC reaches a point where the buffers and the links are occupied and can’t handle the traffic load, then the NoC is said to be congested. Under congestion, the NoC can’t function properly and can’t retain its normal performance. Specifically, the flits simply continue roaming in the NoC without reaching their destinations which increases the traffic volume and prevents the injection of new flits (i.e. nodes starvation).

In bufferless NoC, the congestion can arise and develop more quickly and severely as the links are the only available buffering resources. Bufferless NoCs have been shown to function efficiently under moderate loads and smaller NoC sizes [10]. But under high injection loads and due to the lack of buffers, bufferless NoCs fail to operate and scale efficiently causing a collapse in the overall performance. This prevents

bufferless NoCs from competing with buffered NoCs performance especially under high injection rates.

To tackle the congestion in a NoC, one of the approaches is to detect the congestion and then control its effect to retain the normal NoC behavior. Another approach is to provide the needed resources and take various measures to prevent the congestion from forming in the first place.

1.2. Related Work

In this section we summarize the previous work done related to bufferless NoCs and to our work specifically. We list the different algorithms, techniques and ideas related to bufferless routing algorithms, output port selection functions, flit ranking policies, and congestion management. We survey these topics state-of-art briefly.

Concerning bufferless routing, several previous works examined the use of both dropping and deflecting routing approaches in bufferless. [9,15,11] proposed dropping based routing algorithms where the packets with low priority are dropped once a port contention occurs. These previous studies suffered extra performance loss given the fact that they require a separate network for the ACK/NACK packets delivery, and they induce extra traffic load due packet retransmission. In order to reduce the packet dropping, [16] proposed a selective packet-dropping routing. In [10], a set of deflecting routing algorithms for bufferless routing (BLESS) was proposed. This study used real applications and synthetic workloads to evaluate the network energy consumption, performance, latency, and area requirements of bufferless routing. Their algorithms resulted in around 40% energy reduction with a small degradation in the performance under light traffic. Also, the algorithms save around 60% of area requirements. However, in [12], BLESS was shown to be complex for hardware implementation due to its output allocator. The work done in MaS [17] solved some of the drawbacks in [12] by using packet-sized buffer at each switch which is used to hold the packet with higher priority in case of contention thus decreasing the receiver side buffering requirements caused by the out-of-order delivery of BLESS (caused by either the truncation or by considering each flit as a head flit) by 80%. Also, MaS achieves better average packet latency and average power consumption compared to BLESS by 10% and 9% respectively. Also, in [18], a simplified bufferless router (CHIPPER) was presented, in which a permutation network was designed to solve the output allocation problem in BLESS. However, its deflection rate is high at the medium-to-heavy traffic load. Several works [19,20,21] have been proposed to reduce the packet deflections by adding a few buffers. In [20], a hybrid bufferless router (MinBD) was presented, in which a bufferless router is combined with a small side-buffer. In addition, a buffer controller was designed for identifying the packets which would be deflected and are needed to be temporarily stored in the side-buffer. While in [19], a hybrid bufferless router with an adaptive flow control (AFC) was presented, in which the routing scheme switches between the buffered and bufferless routing according to the network load. However, using buffers in [19,20,21] weakens the primary advantage of the bufferless NoC in cost and energy. The authors in [22] approached the problem by trying to decrease the deflection count as a cause for the performance degradation in bufferless NoCs. They constructed three deflection models to analyze the deflection causes, and proposed a deflection routing based on turn model to reduce the deflections during packet transmissions. The experimental results for [22] showed a reduction in the deflection rate by 41% compared to other bufferless networks.

As for the selection functions, in [13], the authors proposed zigzag (z^2) selection function (the inspiration for MaxFlex) as an optimal selection function for mesh topology. However, in [14,23,24], the authors analyzed different selection functions for mesh topology and found that z^2 is not the best for this topology. [25] presented a topology-independent selection function. None of the previous studies evaluated the MaxFlex on bufferless NoCs or evaluated the effect of changing the value for the used step size. Other studies focused their attention on other topologies such as fat-trees. [26] was the first to propose and evaluate different selection functions for fat-trees. The study showed that a selection function dependent on current switch address and destination address (SADP) has slightly better performance in case of uniform traffic as it balances the load on the links. The authors in [27] proposed and analyzed a selection function dependent on the stage and the source node (SAOP) that outperformed other selections functions in hot-spot traffic. In [28], the authors proposed a cost-efficient congestion management mechanism for fat-trees that detects the current traffic pattern and switch to a certain selection function that is proved to give better performance under the detected traffic pattern. The work done in [29] proposed Cool Centers Priority (CCP) selection function for buffered 2D meshes to eliminate hot-spots, and to guarantee load balancing.

Concerning flit ranking policies, in [30], the authors showed that ranking-based policies using global or history-related criterion are beneficial in a deflection-based NoC. [10] evaluated several flit ranking polices (OF, CF, MDF, and RR) under BLESS and selected OF as their primary ranking policy as it is guaranteed to avoid livelock. However, the authors in [22] chose MDF as their main ranking policy as they aimed to decrease the overall deflection count.

Finally, for the congestion management, [31,32,33] were proposed to adjust the network load. These previous studies controlled the injection rate of each node, and restricted the injection of latency-insensitive processing node if the network load becomes heavy. However, these studies lacked the detailed understanding of the workloads, which made the system design more difficult. In [34], the authors proposed a distributed congestion control mechanism (Cbufferless) for bufferless NoC. This study detected network congestion by monitoring deflection information of the flits and used dynamic node throttling for the node(s) causing network congestion.

1.3. Scope of the Thesis

This thesis has three main directions regarding bufferless NoCs. These three directions aim for a performance similar to buffered NoCs under high injection rates. The first direction targets the selection functions specifically MaxFlex. It reduces the packet latency and the average deflection count via increasing and varying the used step size. The second investigates the flit ranking policies. It enhances the performance via using policies that exploit the properties of the bufferless NoCs specifically the deflection behavior. Finally, the third direction aims to relax the NoC congestion. It achieves that by giving more space for the flits to roam and/or organizing the applications' injection behavior. In each direction, we propose new approaches that enhance the performance while trying to keep the area and the power intact.

1.3.1. Increasing and Varying Step Size Under MaxFlex

The first direction of this thesis focuses on enhancing the traffic distribution under MaxFlex in order to decrease port contention in 2D bufferless meshes. Typical MaxFlex (Step Size = 1) tends to focus the traffic on the NoC diagonal (central part) which leads to increase in the port contention and as a result increases the deflection count and the packet latency. In Chapter 3, we present a study, both analytical and experimental, on the effect of increasing the step size under MaxFlex on the traffic distribution and eventually on the overall performance. We also propose the ideal step to use under different mesh sizes.

For the analytical part; we identify 12 types of traffic that constitute collectively the MaxFlex traffic in the network. The analysis shows that increasing the step size leads to a better load distribution over the NoC switches. In other words, the central part of the network bisection becomes more relaxed.

Then, we simulate a 10x10 mesh under uniform traffic and use step size values ranging from 1 to 9 to check the effect the NoC performance. The results show that increasing the step size leads to better packet latency and smaller deflection count thus enhancing the NoC performance. To be exact, using a fixed step size of 8 enhances the packet latency and the deflection count by 95% and 38% respectively compared to using Straight Line selection function. Also, for different mesh sizes, simulation results show that a step size of 60-80% of the mesh dimension leads to better performance.

In Chapter 4, we address the idea of using different step size for each packet. By using variable step size, we tend to further enhance the traffic distribution aiming to utilize more links and hence decrease the contention and the deflections. We propose different formulas for determining the variable step size value for each packet. Each of the proposed formulas is devised to be a function in the distance between the source and destination. The formulas fall into one or more of the following categories; formulas that consider the distance between the source and destination as nodes, formulas that consider assigning the NoC nodes to virtual regions and then consider the distance between the source and destination regions, and finally, formulas that also use the regions concept but differ between in-region and out-region routing.

Simulating these formulas under 10x10 mesh conforms that varying the step size (using a valid formula) leads to better distribution for the flits among the NoC links thus better NoC performance. Specifically, the results show that using a variable step size can enhance the results over using a fixed step size of 8 by up to 29%. The results come in line with the analytical results of increasing the fixed step size.

1.3.2. Evaluating Flit Ranking Policies

In this direction, we exploit the deflecting bufferless NoCs properties to provide better performance. Based on the results from the fixed/variable step size study and from recent bufferless NoC study [22], in Chapter 5, we study the effect of the flit ranking policies on 2D bufferless meshes' performance, and propose various policies tailored to decrease the flits' deflections in the NoC. In other words, the proposed approaches favor the flit with more deflections as extra detouring for this flit leads to extra delay thus increasing the overall packet latency.

We investigate the usage of the flit's deflection count along with its age and the distance between its source and destination. Also, we develop an enhancement over the proposed policies. The enhancement favors the flit with steps in one direction only as any deflection shall result in at least two hops to correct its path.

We simulate a 10x10 mesh using the enhanced deflection-based approaches. The experiments show that proposed policies lead to better performance. Specifically, using the proposed enhancement along with the proposed policies decreases the packet latency by 58% compared with using fixed step size MaxFlex with Oldest First ranking policy.

1.3.3. Preventing the Congestion

The final direction aims to prevent the bufferless NoC congestion. In Chapter 6, we study the congestion in bufferless NoCs, and propose two mechanisms for preventing the congestion development. Both of the proposed mechanisms prevent the congestion by providing more space for the flits to move by decreasing and/or dividing the traffic volume.

We investigate how to relieve the traffic volume thus preventing the congestion from developing in the first place. To be able to do that, we provide more links bandwidth to the flits so that they have more freedom in their movement towards their destinations. We propose two mechanisms to perform this freedom.

First, we propose running the applications on a NoC larger than what is required. For example, instead of running the application mix on a 3x5 mesh, we propose running the same application mix on a 4x4 mesh. The idea behind this mechanism is to take advantage of the extra links provided as a result of using the larger NoC thus providing extra space for the flits to move with less competition with the other flits.

Second, we propose dividing the application mix into smaller sets, and then run the smaller sets sequentially on the whole NoC. The smaller application mix in combination with the sequential operation leads to injecting less data into the NoC in each smaller run which directly affects the deflection count and the packet latency in a positive way.

We simulate both mechanisms on a 10x10 mesh and measure the enhancements in the performance. Using the proposed prevention mechanisms enhances the packet latency and the deflection count by 61% and 68% respectively compared with using fixed step size MaxFlex.

1.4. Contribution of the Thesis

With the increasing demand on mobile processing, two main engineering factors come into sight: chip area and power. With the interconnection as an important element of modern processors and a main contributor to chip area and power; the decision of optimizing the interconnection area and power is one of the top-list goals in design.

In this thesis, we aim to make the bufferless NoCs work in a fashion similar to buffered NoCs under high injection rates while keeping the area and power gains. We optimize bufferless NoCs through adopting multiple approaches: enhance using selection functions, enhance using ranking policies, and enhance using congestion prevention. The proposed approaches aim to decrease the overall packet latency and average deflection count. Additionally, the approaches aim to push the injection rate boundary for the bufferless NoCs making it feasible in a wider range of practical applications instead of using the heavy area and power consumer - the buffered NoCs.

First, we propose using larger step sizes under MaxFlex selection function (instead of using a step size of one). We thoroughly analyze the MaxFlex under uniform traffic and identified 12 types constituting the traffic. Through using larger step sizes, the

traffic concentration becomes more distributed among the center and border switches leading to less contention among the flits. Using larger step sizes, we are able to use bufferless NoCs under higher injection rates while keeping the average packet latency and average deflection count feasible. We also propose novel approaches for using variable step size for each flit instead of using fixed step size for all the flits. These approaches utilize the NoC links better leading to even better performance. These enhancements are achieved without using any extra buffers thus we keep the chip area small. Also, we distribute the traffic leading to using more links but the frequency of each link decreases which keeps the power usage in its normal figures.

Ranking policies aims to put an order for serving the flits. Knowing that the deflection count for the flits plays a great role in the overall performance, we propose several policies that aim to decrease the overall deflections resulting in better performance. By devising polices based on the flits' deflection count, we aim to favor the flits that suffered more deflections while not causing extra new deflections for other flits. Also, as in the selection functions, these ranking schemes don't use any extra buffers leading to good area and power performances.

Finally, the main roadblock for bufferless NoCs is quickly becoming congested. We propose novel mechanisms for preventing the congestion by mitigating the initial cause for the congestion i.e. the traffic volume. As the traffic volume increases, and in addition to the absence of buffers, the flits have to compete with each other more frequently leading to more deflections and thus detouring. This unneeded detours make the flits travel in the NoC without reaching their destinations. The proposed mechanisms prevent the congestion by decreasing the traffic volume via using larger NoC or via organizing the applications work load. The proposed mechanisms fit the latency-sensitive applications that can be divided and allocated to different parts of the NoC. By organizing the applications allocation and operation, we achieve low packet latency and deflection count while keeping feasible power and area figures.

1.5. Organization of the thesis

This thesis is organized as follows. Chapter 2 explains the preliminaries of the concepts adopted in this thesis. In Chapter 3, we analyze and simulate the usage of fixed step size greater than one under MaxFlex. Chapter 4 presents a study for the use of variable step size under MaxFlex. In Chapter 5, we propose several flits ranking policies and show their performance. Chapter 6 addresses the congestion problem in bufferless NoCs by proposing and evaluating two congestion prevention mechanisms. Finally, in Chapter 7 we summarize our findings and make some concluding remarks concerning the current and future work.

Chapter 2 : Background

In this chapter, we present some preliminaries and concepts that are used in this thesis. We start with interconnection networks in Section 2.1. In Section 2.2 and Section 2.3, we discuss Network-on-Chip and bufferless Network-on-Chip respectively. Section 2.4 explains the idea behind selection functions. Then, Section 2.5 discusses flit ranking policies and their use. Finally, congestion management is discussed in Section 2.6.

2.1. Interconnection Network

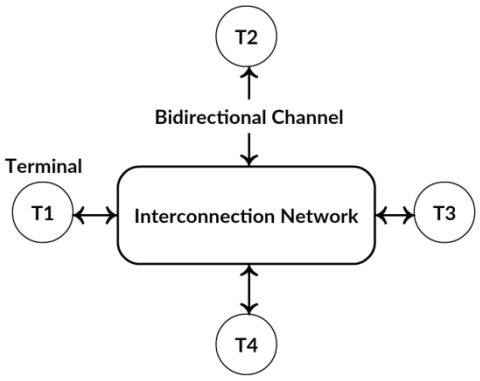

Figure 2: Example of interconnection network

An interconnection network is a programmable system that transports data between terminals. The interconnection network system is composed of buffers, channels, switches, and controls that function together to deliver data. Figure 2 shows an example for an interconnection network with four terminals ($T1 \rightarrow T4$) connected to it. To communicate with terminal T_i, T_j sends a data message into the network and the network delivers the message to T_i, where $1 \leq i, j \leq 4$ and $i \neq j$.

The network is considered programmable as it makes different connections at different points in time. For example, the interconnection network in Figure 2 can send a message from T2 to T3 in one cycle and then send a message from T2 to T1 in the next cycle using the same resources.

Many systems with different scale fall under the above definition. For example, on-chip networks can deliver data between memory, registers, and arithmetic modules within a processor. On the other hand, system-level networks connect processors, memories, input/output (I/O) ports. Finally, local-area and wide-area networks connect different systems together within an enterprise or across large geographical distance.

Interconnection networks can be found in almost all digital systems. Specifically, in computer systems, they connect processors to memories and I/O devices to I/O controllers. While, in communication switches and network routers, they connect input ports to output ports. Also, they connect sensors and actuators to processors in control systems.

Around the late 1980s, most of the mentioned systems used the bus architecture as their interconnection network. However, recently all high-performance interconnections

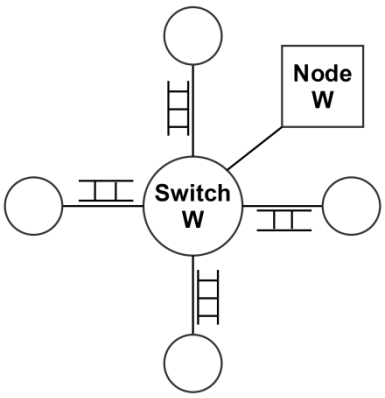

Figure 3: Generic switch in a 2D mesh

use point-to-point interconnection networks rather than buses. This change is due to the inability of buses to keep up with the enhanced processor performance and the bandwidth demand. On the other hand, point-to-point interconnection networks operate faster than buses.

Interconnection networks are important because they are a limiting factor in the performance of many systems. The interconnection network connecting processor and memory largely determines two main performances metrics in a computer system, namely, the memory latency and memory bandwidth. In communication switches, the performance of the interconnection network determines the data rate and the number of ports of the switch.

2.2. Network-on-Chip (NoC)

In a chip multiprocessor (CMP) architecture, the NoC generally connects the processor nodes and their private caches with the shared cache modules and memory controllers. In a typical NoC, each node has a high-speed buffered switch that connects the node to its neighbors by links. The width of a link varies. Nodes send and receive packets; typical packets are small request and control messages, such as cache block read requests, and larger data packets containing cache block data. Packets are partitioned into flits which are the atomic unit of traffic. Flits have size equal to the width of a link. Typically, links have a latency of only one or two cycles, and are pipelined, so that they can accept a new flit every cycle.

NoC topology defines the networks logical layout (connections). Various NoC topologies exist, but the most used topology is the 2D mesh [3,4], which is implemented in several commercial products [35,36] and research prototype many-core processors [7,37,38]. In mesh topology, each switch has maximum of 5 input and 5 output channels/ports; one from each neighboring switch and one from the node connected to it. A sample switch in a 2D mesh is shown in Figure 3.

Since the switch plays a crucial role in the NoC, its design needs to be simple to simplify the overall NoC design. As a result, current implementations tend to use simple routing algorithms. The most common routing algorithm is Dimension Order routing (DO) which route the flit first along the *X* direction until the destination's *Y* coordinate is reached; then route to the destination in *Y* direction.

NoC has a set of characteristics that differentiate it from the traditional networks. We summarize these characteristics in the following points:

1) *Topology*: The topology is statically known, and usually very regular. A change in topology impacts various aspects, such as routing and traffic-load.
2) *Latency*: Links and switches have latency much lower than traditional networks.
3) *Routing*: Routing logic is designed to minimize the complexity and the latency as the NoC switch stages must take no more than a few cycles.
4) *Coordination*: Global coordination is possible and often less expensive than distributed adaptive mechanisms, due to a relatively small known topology, and low latency.
5) *Links*: Links are expensive in terms of both hardware complexity and on-chip area.
6) *Traffic Patterns*: Cache miss behavior of the running applications drive traffic patterns in a NoC.
7) *Power*: The existence of a constrained power budget differentiates NoCs from traditional networks.

2.3. Bufferless Network-on-Chip

Recent work has shown that it is possible to eliminate buffers from the NoC switches. In such bufferless NoCs, application performance degrades minimally for low-to-moderate network intensity workloads, while some work shows that power consumption decreases by 20-40%, router area on die is reduced by 75%, and implementation complexity also decreases [10]. While other evaluations have shown that optimizations to traditional buffered router designs can make buffers more area- and energy-efficient [12], bufferless design techniques such as those in [18,20,17,22] address inefficiencies in bufferless design. In a bufferless NoC, the general system architecture does not differ from traditional buffered NoCs. However, the lack of buffers requires different injection and routing algorithms in the network.

As in a buffered NoC, a bufferless NoC injects and routes flits synchronously across all nodes/switches. The node is able to inject each flit of the packet into the network as long as one of its output links is free. Injection requires a free output link as there is no buffer to hold the packet in the switch. If no output link is free, the flit remains queued inside the node. A flit is routed to a neighbor based on the routing algorithm, and the arbitration policy.

Flits are arbitrated to output ports based on the required direction and the ranking policy used. If flits contend for the same output port, their ranks are compared, and the one with higher rank (priority) obtains the port. The other contending flit(s) are either dropped or by deflected.

Bufferless NoCs that uses the dropping mechanism chooses to drop the packet that lost the contention competition. By dropping the packet the NoC has to retransmit this packet which leads to an increase in the network traffic and/or the hardware cost and design complexity.

The deflecting bufferless NoCs choose not to drop any contending packet. Instead, it forwards all the incoming packets to output ports even if it means to forward the packets through longer paths (non-productive ports). The deflecting buffered NoCs are preferred due to its simpler design and less power and area cost. An example for a deflecting bufferless NoC is BLESS [10].

Previous work [10] has shown significant reductions in chip power and area from eliminating buffers in the NoC, however, that work has focused primarily on low-to-

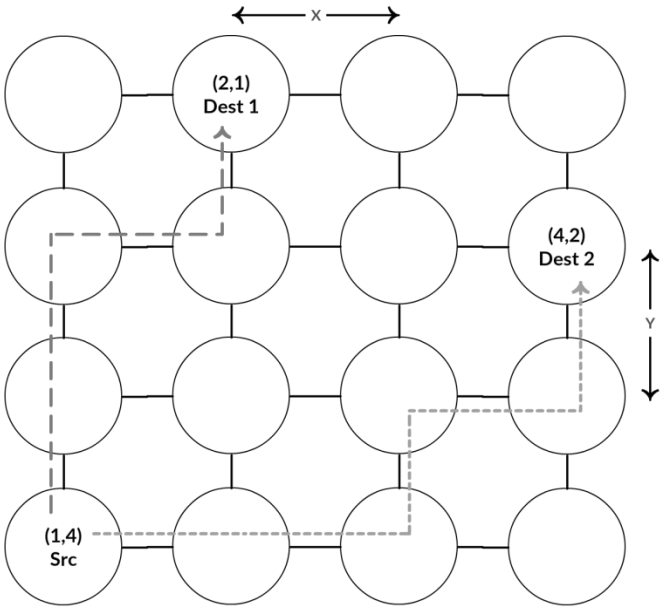

Figure 4: The operation of MaxFlex selection function using step size of one

medium network load. Higher levels of network load remain a challenge, and improving performance in these cases increases the applicability of bufferless NoCs. Furthermore, as the size of the CMP increases, the efficiency gains from bufferless NoCs become more important.

2.4. Selection Functions

A routing algorithm is divided into two functions: routing function and selection function. The routing function provides a set of productive output ports based on the current node and the destination node. The selection function selects from this set based on the status of the output ports at the current node. This selection is performed in such a way that a free channel (if any) is supplied. The routing function determines whether the routing algorithm is deadlock-free or not. However, the selection function only affects performance.

There are two ways to perform the selection: the selection function can ignore the network state, for example, the selection can be random; or the selection can take into account the status of output ports and channels at the current node. Obviously, the selection second approach is better as it works based on some sort of feedback from the NoC.

When several output ports are available, some policy is required to select one of them. Policies can have various goals, for example, balancing the use of resources, reserving some bandwidth for high-priority packets, or even delaying the use of resources to be used for deadlock avoidance. However, under any policy, the selection function should give preference to ports belonging to minimal paths i.e. productive ports. Otherwise the selection function may produce livelock.

Various selection functions exist for n dimensional meshes with the goal of maximizing performance. Three of the well-known selection functions are Minimum Congestion (MinCon), Maximum Flexibility (MaxFlex), and Straight Line (SL).

In MinCon, a virtual channel is selected in the direction with the most available virtual channels. This selection function works with buffered NoCs and tries to balance the use of virtual channels in different physical channels. The idea behind this selection function is since the packet transmission is pipelined, then flit transmission rate is limited by the slowest stage in the pipeline. Balancing the use of virtual channels balances the bandwidth allocated to different virtual channels.

In MaxFlex, a channel (physical or virtual) is selected in the dimension with the greatest distance to travel to the destination. This selection function tries to maximize the number of routing options as a packet approaches its destination. This selection function can perform under both buffered and bufferless NoCs. Specifically, MaxFlex first moves the flit till the number of hops left in the *X*-dimension is equal to the number of hops left in the *Y*-dimension. After that, MaxFlex moves the flit one step on the *X*-dimension and then one step on the *Y*-dimension i.e. MaxFlex tends to move the flit on a diagonal. Figure 4 shows the operation of MaxFlex selection function.

In meshes, MaxFlex selection function concentrates the traffic in the central part of the network bisection producing uneven channel utilization which degrades the NoC performance. This downside has more effect in buffered NoCs than in deflection-based bufferless NoCs due to the lack of buffers and the deflecting behavior in the latter case. The absence of buffers forces the flits to be deflected, in contrast to moving into one of the available buffers (in case of buffered NoCs). This deflection behavior moves small portion of the traffic away from the central NoC switches, thus decreasing the concentration.

Finally, in SL, a channel (physical or virtual) is selected in the dimension closest to the destination. So, the packet travels in the same dimension whenever possible. This selection function tries to route packets in dimension order unless the requested port in the corresponding dimension is not available. This selection function can perform under both buffered and bufferless NoCs. In meshes, this selection function achieves a good distribution of traffic across the network bisection as it tends to move the traffic more towards the NoC borders.

2.5. Flit Ranking Policies

As mentioned above, routing algorithms compute the productive port(s) to move the flit from the current switch to the destination switch via a routing function, and then select the output port for the flit via a selection function. If multiple flits simultaneously request the same output port, some sort of arbitration must be provided between them.

Different arbitration approaches can be used to allocate the required channel bandwidth including random, round robin (RR), or ranking policies. For random selection, any flit is selected randomly without considering the NoC status. For RR selection, output ports are arranged in a circular list. When a port transfers a flit, the next port in the list is selected for the next flit transmission. Finally, ranking policies

uses various criteria to determine which flits should be served first. Ranking policies require some information to be carried in each flit to be used as thee ranking criterion.

Different criteria can be used as ranking policies. For example; Oldest First (OF) policy ensures there is a total age order among flits and prioritizes older flits, Closest First (CF) policy prioritizes the flits with smaller distance to their destinations before flits whose remaining distance is larger, and Most Deflections First (MDF) policy gives higher priority to the flits with more deflections.

Different arbitration (and ranking policies) leads to different service order for the flits travelling through the NoC. To be more specific, different arbitration leads to different arrival order for the NoC flits which leads to a different set of flits reaching their destination before the others. The delivery of a certain set can result in a better performance but this is not the only factor. The selected arbitration has a direct effect on livelock property in the NoC [3,4]. Specifically, an arbitration approach can result in flits travelling indefinitely the NoC and never reaching their destinations.

In bufferless NoCs, due to the buffers elimination, the used arbitration approach has a greater effect on the overall performance compared to buffered NoCs as the flit that fails to get its productive port will be forced to take a detour. However, in buffered NoCs, if the requested port is busy, the flit remains in the input buffer and shall be routed again after the port is freed and if it successfully arbitrates for the port.

2.6. Congestion Management

Under high injection rates, the traffic volume in the NoC increases causing more strain on the NoC links and buffers. When the NoC reaches a point where the buffers and the links are occupied and can't handle the traffic load, then the NoC is said to be congested. Under congestion, the NoC can't function properly and can't retain its normal performance. Specifically, the flits travel in the NoC without reaching their destinations which increases the traffic volume and prevents the injection of new flits.

In bufferless NoC, the congestion can arise and develop quickly and severely as the links are the only buffering resources. Bufferless NoCs have been shown to function efficiently under moderate loads and smaller NoC sizes [10]. But under high injection loads, and due to the lack of buffers, bufferless NoCs fail to operate and scale efficiently causing a collapse in the overall performance. This prevents bufferless NoCs from competing with buffered NoCs performance especially under high injection rates.

To tackle the congestion in a NoC, one of the approaches is to detect the congestion and then control its effect to retain the normal NoC behavior. Another approach is to provide the needed resources and take various measures to prevent the congestion from forming in the first place.

The detection and control approaches apply heuristics and monitor the NoC performance to detect the congestion once it arises. If congestion is detected, these approaches apply a control mechanism to relieve the congested areas. The problem with these approaches is that if the heuristics used to monitor the performance or the actions taken to relieve the congestion are biased or excessive, the overall performance of the system is affected.

On the other hand, the prevention approaches uses extra resources to decrease the probability of developing the congestion. The idea is to use the extra resources to provide other options for the flits in case of contention under high traffic volume. For example, a buffered NoC can use extra buffers to host the flits in case of increased traffic volume.

Chapter 3 : Modified Maximum Flexibility Selection Function

As stated before, routing is composed of routing function and selection function. Maximum Flexibility (MaxFlex) selection function [13] was introduced with the advantage of maximizing the number of productive ports provided by the routing function as the flit approaches its destination. However, MaxFlex selection function uses a step size of one.

In this chapter, we investigate the effect of using a step size larger than one under MaxFlex selection function. First, we propose the modified MaxFlex selection function (MMaxFlex) and show its operation. Then, we provide a thorough analytical study for MMaxFlex. In our analysis, we begin by analyzing the traffic in 2D meshes under MMaxFlex for any step size. Then, we prove that any packet passing through a node can be classified into one of twelve traffic types. Finally, we derive the count of packets for each type passing through a switch. We also provide simulation results and explain how it conforms to the analysis.

The chapter is organized as follows; Section 3.1 provides the motivation behind the MMaxFlex, and how it works. Then, we analyze the effect of the step size under MMaxFlex on the packets distribution in Section 3.2. In Section 3.3, we prove that any packet under MMaxFlex passing through a switch can be classified into one of twelve traffic types. We provide the effect of using MMaxFlex on the distribution of packets within the NoC in Section 3.4. Sections 3.5 and 3.6 present the experimental setup and the simulation results respectively. In Section 3.7, we estimate the value of the step size based on the dimensions of the NoC. Finally, Section 3.8 makes some concluding remarks.

3.1. Proposed Approach

MaxFlex selection function tends to alternate the flit's movement on both dimensions as a way to make more productive ports available for the flit. Specifically, MaxFlex first moves the flit till the number of hops left in the *X*-dimension is equal to the number of hops left in the *Y*-dimension. After that, MaxFlex moves the flit one step on the *X*-dimension and then one step on the *Y*-dimension i.e. MaxFlex tends to move the flit on a diagonal. Figure 5 shows the operation of MaxFlex selection function.

The main problem with MaxFlex is that it tends to concentrate the traffic on the central part of the NoC leading to more contention between the flits to get the required output ports, thus leading to more deflections in case of bufferless NoC. Eventually, the flit takes more cycles to reach the destination i.e. higher average packet latency.

In this chapter, we propose a modified version of MaxFlex (MMaxFlex) to keep the freedom provided by the MaxFlex selection function while relaxing the contention on the central NoC switches. To achieve our goal, we incorporate the idea of the Straight Line (similar to DO) selection function property to the MaxFlex selection function. Straight Line selection function tends to focus the movement of flits on the NoC's border switches while MaxFlex tends to focus the movement of flits on the NoC's central switches (i.e. the switches in the middle of the NoC).

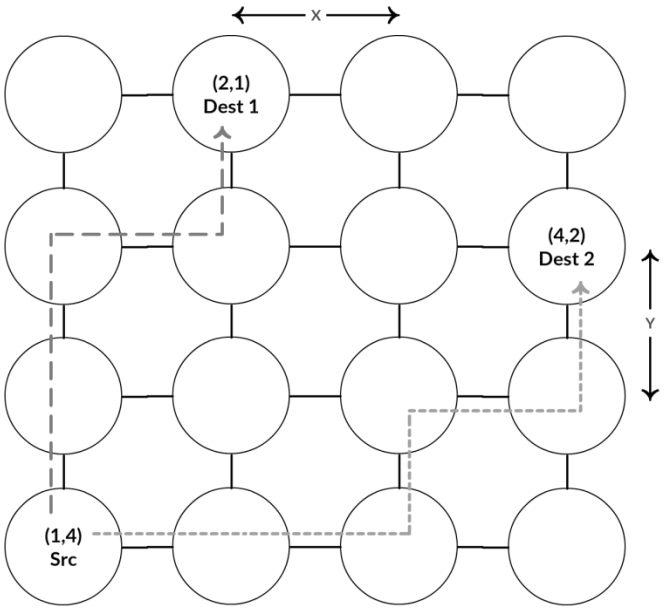

Figure 5: The operation of MaxFlex selection function using step size of one

As a result, we propose using MaxFlex with a step size greater than one. This moves the traffic further towards the borders and decreases the concentration on the NoC's central switches. This approach leads to less contention and subsequently less deflections and smaller packet latency. In the next section, we analyze MMaxFlex in bufferless 2D meshes for any step size.

3.2. Analysis of MMaxFlex Selection Function

In this section, we study the effect of the step size on the distribution of packets through bufferless $n \times n$ two-dimensional mesh network. In doing that, for a certain step size, we count the number of packets passing through each switch (all ports included). To simplify the analysis, we divide the traffic going through a switch into 12 different types. Finally, we derive the number of passing packets belonging to each type.

In the following analysis, we assume that:

1) Each node sends only one packet to each other node (i.e. each node sends n^2-1 packets)
2) Packet length is one Flit
3) No deflections (i.e. path of each packet is only affected by the value of step size and not by misrouting due deflection). This assumption is set to ease the analysis.

Before going into the analysis details, we present some definitions and terminologies that are used throughout the analysis. First, we differentiate between increasing and decreasing diagonals in a 2D mesh. Figure 6 shows both of the diagonal types. In the *decreasing diagonal*, both the *X* and *Y* indices increases for each node along the diagonal. In contrast, the *X* index increases while the *Y* index decreases for each node along the *increasing diagonal*. A typical 2D mesh switch belongs to an

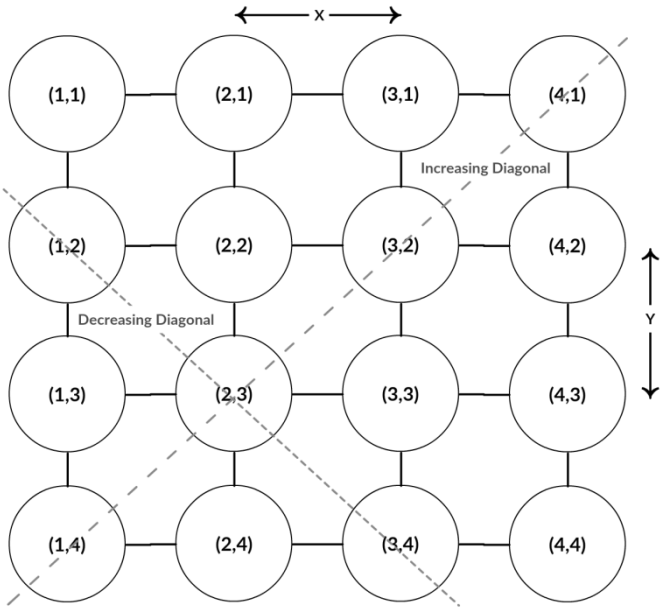

Figure 6: Increasing and decreasing diagonals in a 2D mesh

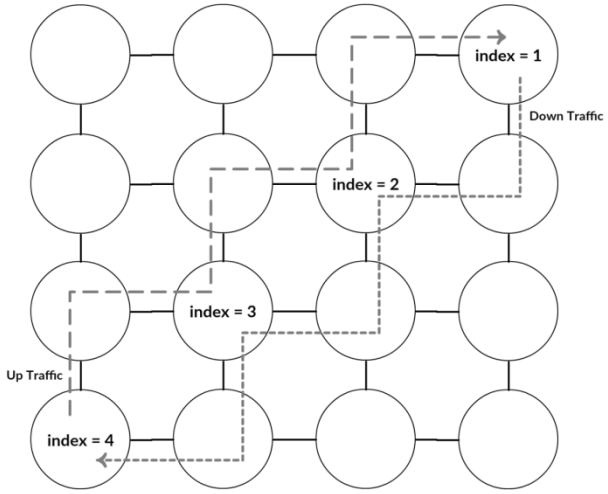

Figure 7: Up and down traffic in 2D mesh

increasing diagonal as well as a decreasing diagonal but not necessarily of same size (Check Appendix A for more details).

Second, concerning the traffic moving on a diagonal, we divide such traffic to up traffic and down traffic. *Up traffic* is the traffic from nodes with higher *index* to nodes with lower *index*, where *index* is the position of the node in a NoC row or column or diagonal (depends on the traffic type under study). On the other hand, *down traffic* is the traffic from nodes with lower *index* to nodes with higher *index*. Figure 7 shows the difference between up and down traffic.

In the following sub-sections, we study each traffic type separately. All the traffic types fall under one of two categories:

1) Type resulted from a communication behavior not related to MaxFlex exclusively
2) Type resulted from a communication behavior related to MaxFlex

The first category is concerned with the traffic types that can be a result of other selection functions, not only MaxFlex. For example, this category can exist if Straight Line selection function is used. However, the second category exists only due to the unique behavior of MaxFlex operation.

The twelve traffic types are summarized as follows. Type 1 and Type 2 are concerned with the traffic resulted from the ejection and injection respectively. Type 3 is a result of row nodes communicating with each other. Similarly, Type 4 is a result of column nodes communicating with each other. All the previous types (1, 2, 3, and 4) belong to the first category.

As for the second category, Type 5 is for the communication between the nodes belonging to the same diagonal. Type 6 and Type 7 are similar to Type 5; however, both of them are concerned with row or column nodes communicating with diagonal nodes. Specifically, Type 6 focuses on the communication between row nodes and diagonal nodes (movement on both row and diagonal switches), while Type 7 focuses on the communication between column nodes and diagonal nodes (movement on both column and diagonal switches).

Communication behavior described in Type 8, Type 9, and Type 10 is a result of the effect of the communication that occurs in Type 5, Type 6, and Type 7 respectively. For example, in Type 5, as the diagonal nodes do not have direct links between them, the packet has to move through other switch (not belonging to the same diagonal under study) to reach the next diagonal node. This kind of movement leads to affecting switches other than the diagonal under study switches. Types 8, 9, and 10 are concerned with such effect.

Also, as Type 6 and Type 7 involve row and column movement respectively beside the diagonal movement; other non-diagonal (row and column) switches are affected by such communication behavior. Type 11 and Type 12 are concerned with the effect caused on row and column switches by Type 6 and Type 7 respectively.

Concerning the analysis, we note the following; since the communication behavior on increasing diagonal is similar to the one done on decreasing diagonal, we focus our analysis on increasing diagonals only. Additionally, as Type 6 and Type 7 are similar, we analyze Type 6 in details. Following the same analysis, the equations related to Type 7 are straight forward. This relation resembles the relation between Type 8, Type 9, and Type 10, and between Type 11 and Type 12. Finally, for all the types except Type 8, Type 9, and Type 10, we derive equations for counting the number of packets passing through a switch as a result of the type under study. For Type 8, Type 9, and Type 10, we count the number of passing packets using pseudo code not equation for its easier analysis and explanation.

For the analysis, we use the following terminology: For an $n \times n$ mesh, let $W(i,j)$ be a switch located at *index*, where $1 \leq i, j \leq n$, and *index* is the position of the switch in a NoC row or column or diagonal based on the traffic type under study. Let P be a packet going from source node $S(X_{\text{Src}}, Y_{\text{Src}})$ to destination $D(X_{\text{Dst}}, Y_{\text{Dst}})$. Let $\Delta X = |X_{Src} - X_{Dst}|$ and $\Delta Y = |Y_{Src} - Y_{Dst}|$. Now, we present the traffic 12 types going through $W(i,j)$. In each type, we describe the communication behavior, and the number of packets (denoted as *Count*) passing through switch $W(i,j)$ as a result of the type under study.

3.2.1. Type 1 Packets

Description: Packets destined to node $W(i,j)$.
Count: $n^2 - 1$

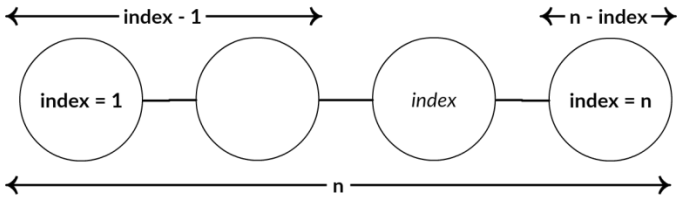

Figure 8: Location of *W*(*i,j*) in 2D mesh row

Proof: Since there are n^2-1 nodes sending packets to node *W*(*i,j*) as their destination (check the assumptions), then the number of packets passing through switch *W*(*i,j*) is equal to the number of packets ejected to node *W*(*i,j*).

■

3.2.2. Type 2 Packets

Description: Packets injected by node *W*(*i,j*).
Count: $n^2 - 1$
Proof: Since node *W*(*i,j*) is sending one packet to each of the remaining n^2-1 nodes (check the assumptions), then the number of packets passing through switch *W*(*i,j*) is equal to the number of packets injected by node *W*(*i,j*).

■

3.2.3. Type 3 Packets

Description: Packets passing through *W*(*i,j*) injected by node (*i,k*) and destined to node (*i,m*) where $1 \leq k, m \leq n$ and $j \neq k \neq m$ (i.e. same row communication).
Count: $2(index - 1)(n - index)$
Where *index* is the position of *W*(*i,j*) in the row under study, $1 \leq index \leq n$.
Proof: In this type, as shown in Figure 8, switch *W*(*i,j*) belongs to a 2D mesh row at position *index*. We have (*index* - 1) nodes before node *W*(*i,j*) and (*n* - *index*) nodes after node *W*(*i,j*).
For the same row communication, the number of packet passing through switch *W*(*i,j*) is a result of the following:

1) The nodes before *W*(*i,j*) send packets to the nodes after *W*(*i,j*) i.e. (*index* - 1) * (*n* - *index*) packets.
2) The nodes after *W*(*i,j*) send packets to the nodes before *W*(*i,j*) i.e. (*n* - *index*) * (*index* - 1) packets.

Hence, the overall count is 2 * (*index* - 1) * (*n* - *index*) packets.

■

Examples for illustrating traffic type three are shown in both Figure 9 and Figure 10. Figure 9 shows the value of *index* and *Count* for each row switch in 5x5 and 6x6 meshes. Also, it shows the number of packet passing through each switch as a result of this traffic type. On the other hand, Figure 10 shows an example on how to calculate the number of passing packets for a row switch in a 5x5 mesh. In Figure 10, the number of packets passing through switch *B*($index$ = 2, n = 5) equals 6 due to node *A* sending packets to nodes (*C, D, E*) i.e. 3 in addition to (*C, D, E*) sending packets to *A* i.e. 3.

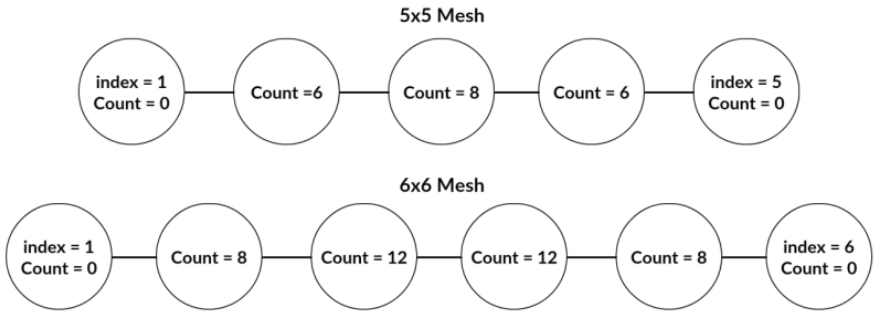

Figure 9: Type 3 example for a row in 5x5 and 6x6 meshes

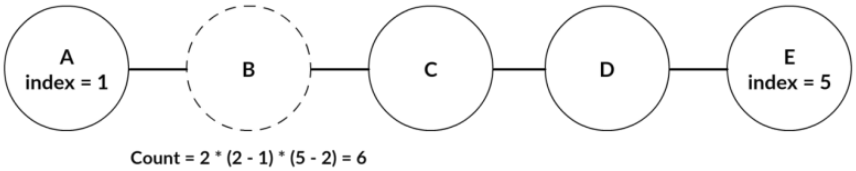

Figure 10: Type 3 *Count* calculation for a row switch in a 5x5 mesh

3.2.4. Type 4 Packets

Description: Packets passing through $W(i,j)$ injected by node (k,j) and destined to node (m,j) where $1 \leq k, m \leq n$ and $i \neq k \neq m$ (i.e. same column communication). [1]
Count: $2(index-1)(n-index)$
Where *index* is the position of $W(i,j)$ in the column under study, $1 \leq index \leq n$.
Proof: The proof for this type is similar to the proof of Type 3.

3.2.5. Type 5 Packets

Description: Packets passing through $W(i,j)$ as a result of communication between nodes on the same diagonal as node $W(i,j)$.

Count: $Count_{Up} + Count_{Down} = (index-1)\left\lfloor \frac{n'-index}{SS} \right\rfloor + (n'-index)\left\lfloor \frac{index-1}{SS} \right\rfloor$

Where n' is the number of nodes in the diagonal under study, $1 \leq n' \leq n$; *index* is the position of $W(i,j)$ in the diagonal under study, $1 \leq index \leq n'$; and SS is the value of the step size.

This equation counts the number of packets passing through a given increasing diagonal switch $W(i,j)$ under up traffic and down traffic. [2]

[1] Type 1, 2, 3, and 4 are not related to MaxFlex selection function only as these types will occur in almost all routing algorithms under the same assumptions. In other words, no step size is involved in the
[2] Any of the following analysis (and proof) can be applied to both increasing and decreasing diagonals.

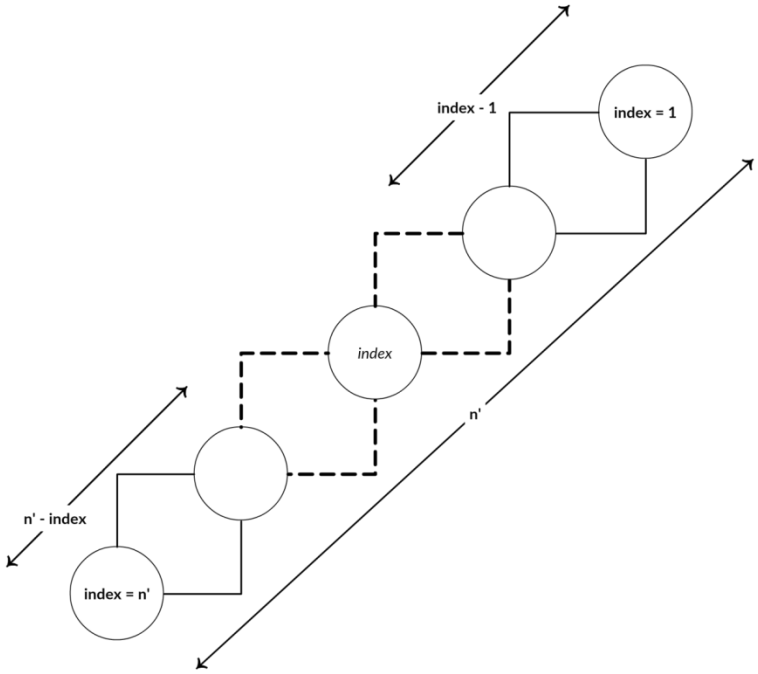

Figure 11: Location of *W*(*i*,*j*) in 2D mesh diagonal

Proof: We consider increasing diagonal for the proof; however, the same proof applies to decreasing diagonal. In this type, as shown in Figure 11, switch *W*(*i*,*j*) belongs to a 2D mesh diagonal with n' nodes. Switch W(I,j) is at position *index*. We have (*index* - 1) nodes before node *W*(*i*,*j*) and (n' - *index*) nodes after node *W*(*i*,*j*).

For the same diagonal communication, the number of packets passing through switch *W*(*i*,*j*) is a result of the following:

1) Under up traffic, some of the nodes after node *W*(*i*,*j*) send packets to the nodes before *W*(*i*,*j*) based on the step size *SS* used. Specifically, $\left\lfloor \frac{n' - index}{SS} \right\rfloor$ nodes after *W*(*i*,*j*) send packets to the nodes before *W*(*i*,*j*). Hence, the number of packets is (*index* - 1) * $\left\lfloor \frac{n' - index}{SS} \right\rfloor$ packets.
2) Under down traffic, some of the nodes before node *W*(*i*,*j*) send packets to the nodes after *W*(*i*,*j*) based on the step size *SS* used. Specifically, $\left\lfloor \frac{index - 1}{SS} \right\rfloor$ nodes before W(I,j) send packets to the nodes after *W*(*i*,*j*). Hence, the number of packets is (n' - *index*) * $\left\lfloor \frac{index - 1}{SS} \right\rfloor$ packets.

Thus, the overall count is (*index* - 1) * $\left\lfloor \frac{n' - index}{SS} \right\rfloor$ + (n' - *index*) * $\left\lfloor \frac{index - 1}{SS} \right\rfloor$ packets.

■

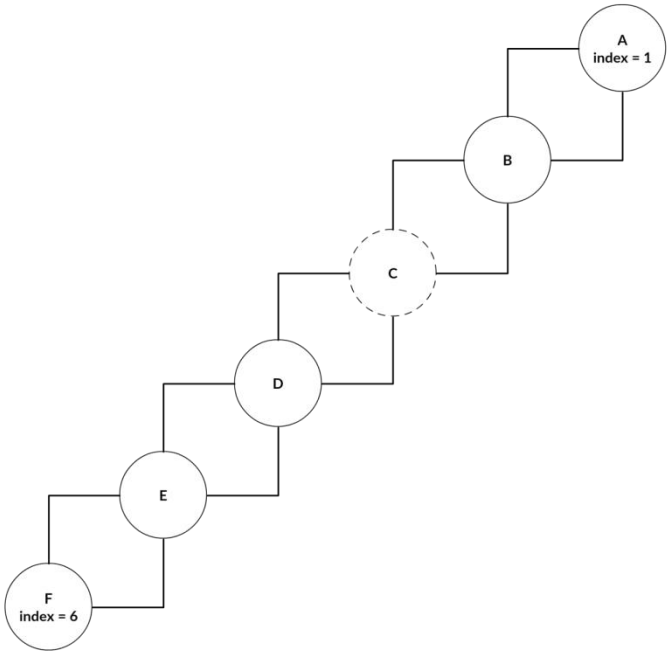

Figure 12: Type 5 example for an increasing diagonal

Table 1: Up traffic passing through switch *C*

Source → Destination	Pass *W*(*i*,*j*) or Not
$D \rightarrow B$	×
$D \rightarrow A$	×
$E \rightarrow B$	Pass
$E \rightarrow A$	Pass
$F \rightarrow B$	×
$F \rightarrow A$	×

In order to illustrate the *Count* calculations, in Figure 12, we show the up traffic passing through switch $C(i = 3, n' = 6)$ i.e. traffic from (*D*, *E*, *F*) to (*A*, *B*) using step size of two. Table 1 lists all the communication from (*D*, *E*, *F*) to (*A*, *B*) and whether the packets to (*A*, *B*) will pass switch *C* or not.

Table 1 states that only two packets pass the red switch under up traffic i.e. *Count* = 2. Also, applying the up traffic *Count* equation, the number of packets passing through the switch *C* is two i.e. *Count* = 2 which matches the value deduced from Table 1.

Table 2 shows the *Count* values for the diagonal switches using different step sizes under up traffic.

In a similar manner, in Figure 12, the down traffic passing through switch $C(i = 3, n' = 6)$ i.e. traffic from (*A*, *B*) to (*D*, *E*, *F*) using step size of two. Table 3 lists all the communication from (*A*, *B*) to (*D*, *E*, *F*) and whether the packets to (*D*, *E*, *F*) will pass switch *C* or not.

Table 2: Type 5 *Count* calculation for an increasing diagonal switches under up traffic using different *SS* values

index	*SS* = 1	*SS* = 2	*SS* = 3
1	0	0	0
2	4	2	1
3	6	2	2
4	6	3	0
5	4	0	0
6	0	0	0

Table 3: Down traffic passing through switch *C*

Source → Destination	**Pass *W*(*i*,*j*) or Not**
$A \rightarrow D$	Pass
$A \rightarrow E$	Pass
$A \rightarrow F$	Pass
$B \rightarrow D$	×
$B \rightarrow E$	×
$B \rightarrow F$	×

Table 4: Type 5 *Count* calculation for an increasing diagonal switches under down traffic using different *SS* values

index	*SS* = 1	*SS* = 2	*SS* = 3
1	0	0	0
2	4	0	0
3	6	3	0
4	6	2	2
5	4	2	1
6	0	0	0

Table 3 states that three packets pass the red switch under down traffic i.e. *Count* = 3. Also, applying the down traffic *Count* equation, the number of packets passing through switch *C* is three i.e. *Count* = 3 which matches the value deduced from Table 3.

Table 4 shows the *Count* values for the diagonal switches using different step sizes under down traffic.

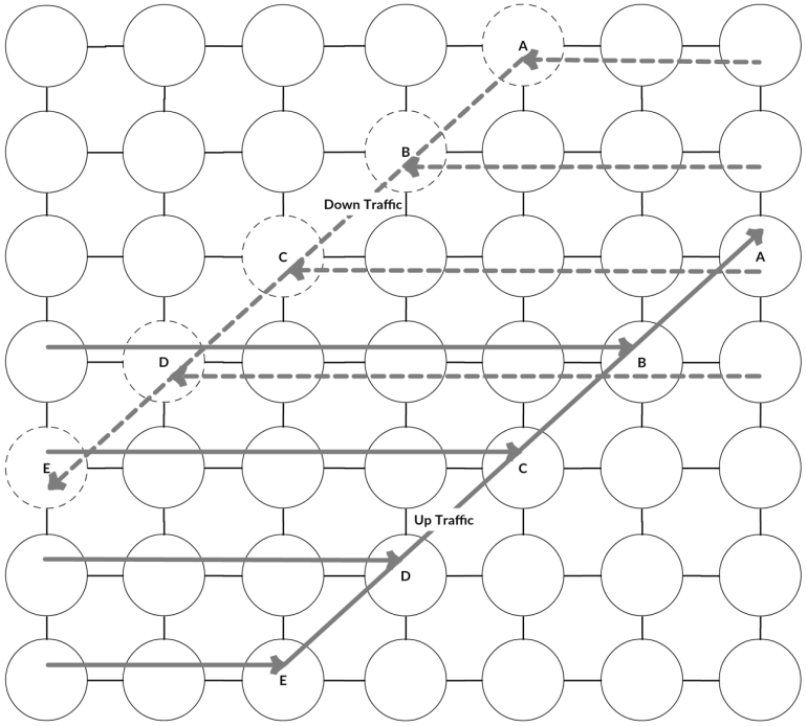

Figure 13: Type 6 example for an increasing diagonal under both up and down traffics

3.2.6. Type 6 Packets

Description: Packets passing through $W(i,j)$ as a result of communication destined to nodes on the same diagonal as node $W(i,j)$ from nodes with $\Delta X > \Delta Y$ (i.e. leads to moving on a row first).[3]

Figure 13 shows an example for the up and down traffic on increasing diagonal under traffic Type 6. We divide the discussion of Type 6 into four separate sub-types to ease the calculation for each of them. The sub-types are listed in the following sub-sections.

3.2.6.1. Type 6 (a)

Description: Packets passing through switch $W(i,j)$ as a result of row nodes communicating with nodes with lower *index* on increasing diagonal i.e. up traffic.

Count: $(index-1)\sum_{x=index}^{n'} Q\times(1-\left\lceil\frac{(x-index)\bmod SS}{SS}\right\rceil)$

If diagonal is below main diagonal i.e. $n+1-j<i$

$$Q = n - x$$

Else

$$Q = n' - x$$

[3] Types 6 and 7 are based on the behavior of any two nodes communicating using MaxFlex (except for same row and column communication). In other words, any two communicating nodes will have to move on a diagonal.

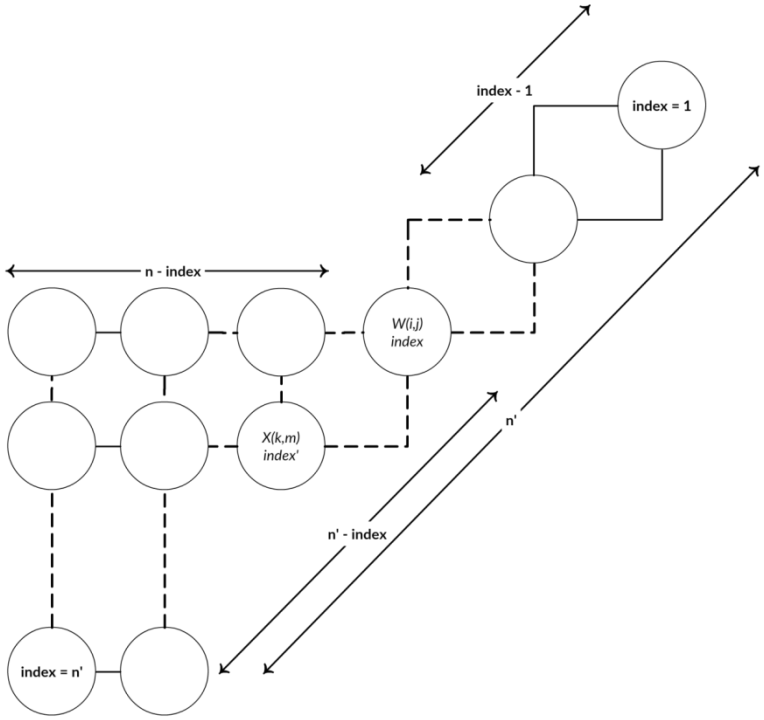

Figure 14: Location of *W*(*i*,*j*) in 2D mesh diagonal

Where n' is the number of nodes in the diagonal under study, $1 \leq n' \leq n$; *index* is the position of *W*(*i*,*j*) in the diagonal under study, $1 \leq index \leq n'$; and SS is the value of the step size.

Proof: We consider increasing diagonal for the proof; however, the same proof applies to decreasing diagonal. In this type, we follow the same steps as in Type 5. However, in Type 6, we focus on the communication between the row nodes and the diagonal nodes.

In this type, as shown in Figure 14, switch *W*(*i*,*j*) belongs to a 2D mesh diagonal with n' nodes at position *index*. We have (*index* - 1) diagonal nodes before node *W*(*i*,*j*) and (n' - *index*) diagonal nodes after *W*(*i*,*j*). Also, there are (*n* – *index*) on the same row as *W*(*i*,*j*) before node *W*(*i*,*j*).

For this type, under up traffic, each of the row nodes belonging to the (n' - *index*) diagonal nodes after *W*(*i*,*j*) send packets to the (*index* - 1) diagonal nodes before node *W*(*i*,*j*). For each of the diagonal nodes *X*(*k*,*m*) at position $index'$ after *W*(*i*,*j*), if *X*(*k*,*m*) communication with the (*index* - 1) nodes before *W*(*i*,*j*) passes through *W*(*i*,*j*), then the row nodes belonging to the same row as the given node shall pass *W*(*i*,*j*) as well i.e. $(n - index)$ nodes. Hence, the overall count for each of *X*(*k*,*m*) in case it passes through *W*(*i*,*j*) is (*index* - 1) * $(n - index')$ packets.

■

In order to illustrate the *Count* calculations, in Figure 13, we show the up traffic passing through switch $C_{Solid}(i = 3, n = 7, n' = 6)$ i.e. traffic from row nodes before (C_{Solid}, D_{Solid}, E_{Solid}) to (A_{Solid}, B_{Solid}) using step size of two. Table 5 lists all the communication from (C_{Solid}, D_{Solid}, E_{Solid}) rows to (A_{Solid}, B_{Solid}) and whether the packets to (A_{Solid}, B_{Solid}) will pass switch C_{Solid} or not.

Table 5: Up traffic passing through switch C_{Solid}

Source → Destination	Pass $W(i,j)$ or Not
$C_{Solid-Row} \rightarrow A_{Solid}$	Pass
$C_{Solid-Row} \rightarrow B_{Solid}$	Pass
$D_{Solid-Row} \rightarrow A_{Solid}$	×
$D_{Solid-Row} \rightarrow B_{Solid}$	×
$E_{Solid-Row} \rightarrow A_{Solid}$	Pass
$E_{Solid-Row} \rightarrow B_{Solid}$	Pass

Table 6: Type 6(a) *Count* calculation for the solid diagonal switches under up traffic using different *SS* values

index	*SS* = 2	*SS* = 3
1	0	0
2	(5 + 3)	(5 + 2)
3	(4 + 2) + (4 +2)	(4) + (4)
4	(3) + (3) + (3)	(3) + (3) + (3)
5	(2) + (2) + (2) + (2)	(2) + (2) + (2) + (2)

Table 7: Type 6(a) *Count* calculation for the dotted diagonal switches under up traffic using different *SS* values

index	*SS* = 2	*SS* = 3
1	0	0
2	(3 + 1)	(3)
3	(2) + (2)	(2) + (2)
4	(1) + (1) + (1)	(1) + (1) + (1)
5	0	0

Table 5 states that twelve packets pass switch C_{Solid} under up traffic (*Count* = 4 + 4 + 2 + 2 = 12). Also, applying the *Count* equation, the number of packets passing through switch C_{Solid} is twelve (*Count* = (3 – 1) * ((7 – 3) * 1 + (7 – 4) * 0 + (7 – 5) * 1) = 2 * (4 + 0 + 2) = 12) which matches to the value deduced from Table 5.

We show the *Count* values for the solid and dotted diagonal switches under up traffic using different step sizes in Table 6 and Table 7 respectively.

3.2.6.2. Type 6 (b)

Description: Packets passing through switch $W(i,j)$ as a result of row nodes communicating with nodes with higher index on increasing diagonal i.e. down traffic.

Count: $(n'-index)\sum_{x=1}^{index} Q \times \left(1-\left\lceil \frac{(index-x) \bmod SS}{SS} \right\rceil\right)$

Table 8: Down traffic communication passing through switch D_{Dotted}

Source → Destination	Pass $W(i,j)$ or Not
$A_{Dotted-Row} \rightarrow E_{Dotted}$	Pass
$B_{Dotted-Row} \rightarrow E_{Dotted}$	×
$C_{Dotted-Row} \rightarrow E_{Dotted}$	×
$D_{Dotted-Row} \rightarrow E_{Dotted}$	Pass

Table 9: Type 6(b) *Count* calculation for the dotted diagonal switches under down traffic using different *SS* values

index	*SS* = 2	*SS* = 3
1	(2) + (2) + (2) + (2)	(2) + (2) + (2) + (2)
2	(3) + (3) + (3)	(3) + (3) + (3)
3	(2 + 4) + (2 + 4)	(4) + (4)
4	(3 + 5)	(2 + 5)
5	0	0

Table 10: Type 6(b) *Count* calculation for the solid diagonal switches under down traffic using different *SS* values

index	*SS* = 2	*SS* = 3
1	0	0
2	(1) + (1) + (1)	(1) + (1) + (1)
3	(2) + (2)	(2) + (2)
4	(1 + 3)	(3)
5	0	0

If diagonal is above main diagonal i.e. $n+1-j > i$

$$Q = x - 1 + n - n'$$

Else

$$Q = x - 1$$

Where n' is the number of nodes in the diagonal under study, $1 \leq n' \leq n$; *index* is the position of $W(i,j)$ in the diagonal under study, $1 \leq index \leq n'$; and *SS* is the value of the step size.

Proof: The proof for this type is similar to the proof of Type 6(a).

In order to illustrate the *Count* calculations, in Figure 13, we show the down traffic passing through switch $D_{Dotted}(i = 4, n = 7, n' = 5)$ i.e. traffic from row nodes before (A_{Dotted}, B_{Dotted}, C_{Dotted}, D_{Dotted}) to E_{Dotted} using step size of three. Table 8 lists all the communication from (A_{Dotted}, B_{Dotted}, C_{Dotted}, D_{Dotted}) rows to E_{Dotted} and whether the packets to E_{Dotted} will pass switch D_{Dotted} or not.

Table 8 states that seven packets pass switch D_{Dotted} under down traffic (*Count* = 2 + 5 = 7). Also, applying the *Count* equation, the number of packets passing through switch D_{Dotted} is twelve (*Count* = (5 – 4) * (2 * 1 + 3 * 0 + 4 * 0 + 5 * 1) = 1 * (2 + 5) = 7) which matches to the value deduced from Table 8.

We show the *Count* values for the dotted and solid diagonal switches under up traffic using different step sizes in Table 9 and Table 10 respectively.

3.2.6.3. Type 6 (c)

Description: Packets passing through switch *W*(*i*,*j*) as a result of row nodes communicating with nodes with lower index on decreasing diagonal i.e. up traffic.[4]

Count: $(index-1)\sum_{x=index}^{n'} Q\times(1-\left\lceil\frac{(x-index)\bmod SS}{SS}\right\rceil)$

If diagonal is above main diagonal i.e. $j>i$

$$Q=n-x$$

Else

$$Q=n'-x$$

Where n' is the number of nodes in the diagonal under study, $1\le n'\le n$; *index* is the position of *W*(*i*,*j*) in the diagonal under study, $1\le index\le n'$; and SS is the value of the step size.

Proof: The proof for this type is similar to the proof of Type 6(a).

3.2.6.4. Type 6 (d)

Description: Packets passing through switch *W*(*i*,*j*) as a result of row nodes communicating with nodes with higher index on decreasing diagonal i.e. down traffic.

Count: $(n'-index)\sum_{x=1}^{index} Q\times(1-\left\lceil\frac{(index-x)\bmod SS}{SS}\right\rceil)$

If diagonal is below main diagonal i.e. $j<i$

$$Q=x-1+n-n'$$

Else

$$Q=x-1$$

Where n' is the number of nodes in the diagonal under study, $1\le n'\le n$; *index* is the position of *W*(*i*,*j*) in the diagonal under study, $1\le index\le n'$; and SS is the value of the step size.

Proof: The proof for this type is similar to the proof of Type 6(a).

3.2.7. Type 7 Packets

Description: Packets passing through *W*(*i*,*j*) as a result of communication destined to nodes on the same diagonal as node *W*(*i*,*j*) from nodes with $\Delta X<\Delta Y$ (i.e. leads to moving on a column first).[5]

[4] Types 6(c) and 6(d) are same as 6(a) and 6(b) but for decreasing diagonals.

[5] Type 7 is similar to Type 6 but for column nodes instead of row nodes.

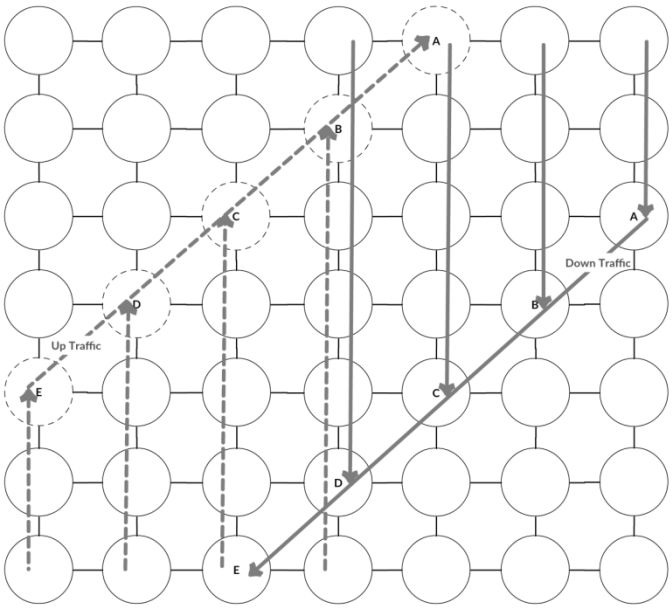

Figure 15: Type 7 example for an increasing diagonal under both up and down traffics

Figure 15 shows an example for the up and down traffic on increasing diagonal under traffic Type 7. We divide the discussion of Type 7 into four separate sub-types to ease the calculation for each of them. The sub-types are listed in the following sub-sections.

3.2.7.1. Type 7 (a)

Description: Packets passing through switch $W(i,j)$ as a result of column nodes communicating with nodes with lower index on increasing diagonal i.e. up traffic.

Count: $(index-1)\sum_{x=index}^{n'} Q\times(1-\left\lceil\frac{(x-index)\bmod SS}{SS}\right\rceil)$

If diagonal is above main diagonal i.e. $n+1-j>i$

$$Q=n-x$$

Else

$$Q=n'-x$$

Where n' is the number of nodes in the diagonal under study, $1\le n'\le n$; *index* is the position of $W(i,j)$ in the diagonal under study, $1\le index\le n'$; and SS is the value of the step size.

Proof: The proof for this type is similar to the proof of Type 6(a).

3.2.7.2. Type 7 (b)

Description: Packets passing through switch $W(i,j)$ as a result of column nodes communicating with nodes with higher index on increasing diagonal i.e. down traffic.

Count: $(n'-index)\sum_{x=1}^{index} Q\times(1-\left\lceil\frac{(index-x)\bmod SS}{SS}\right\rceil)$

If diagonal is below main diagonal i.e. $n+1-j<i$

$$Q = x-1+n-n'$$

Else

$$Q = x-1$$

Where n' is the number of nodes in the diagonal under study, $1 \le n' \le n$; *index* is the position of $W(i,j)$ in the diagonal under study, $1 \le index \le n'$; and SS is the value of the step size.

Proof: The proof for this type is similar to the proof of Type 6(a).

3.2.7.3. Type 7 (c)

Description: Packets passing through switch $W(i,j)$ as a result of column nodes communicating with nodes with lower index on decreasing diagonal i.e. up traffic.

Count: $(index-1)\sum_{x=index}^{n'} Q \times (1-\left\lceil \frac{(x-index) \bmod SS}{SS} \right\rceil)$

If is diagonal below main diagonal i.e. $j<i$

$$Q = n-x$$

Else

$$Q = n'-x$$

Where n' is the number of nodes in the diagonal under study, $1 \le n' \le n$; *index* is the position of $W(i,j)$ in the diagonal under study, $1 \le index \le n'$; and SS is the value of the step size.

Proof: The proof for this type is similar to the proof of Type 6(a).

3.2.7.4. Type 7 (d)

Description: Packets passing through switch $W(i,j)$ as a result of column nodes communicating with nodes with higher index on decreasing diagonal i.e. down traffic.

Count: $(n'-index)\sum_{x=1}^{index} Q \times (1-\left\lceil \frac{(index-x) \bmod SS}{SS} \right\rceil)$

If diagonal is above main diagonal i.e. $j>i$

$$Q = x-1+n-n'$$

Else

$$Q = x-1$$

Where n' is the number of nodes in the diagonal under study, $1 \le n' \le n$; *index* is the position of $W(i,j)$ in the diagonal under study, $1 \le index \le n'$; and SS is the value of the step size.

Proof: The proof for this type is similar to the proof of Type 6(a).

3.2.8. Type 8 Packets

Description: Packets passing through $W(i,j)$ as a result of communication between nodes on a diagonal other than node $W(i,j)$ diagonal.[6,7]

[6] Types 8, 9, 10, 11 & 12 are concerned by the effect of adjacent nodes (row, column, diagonal) communication on other nodes.

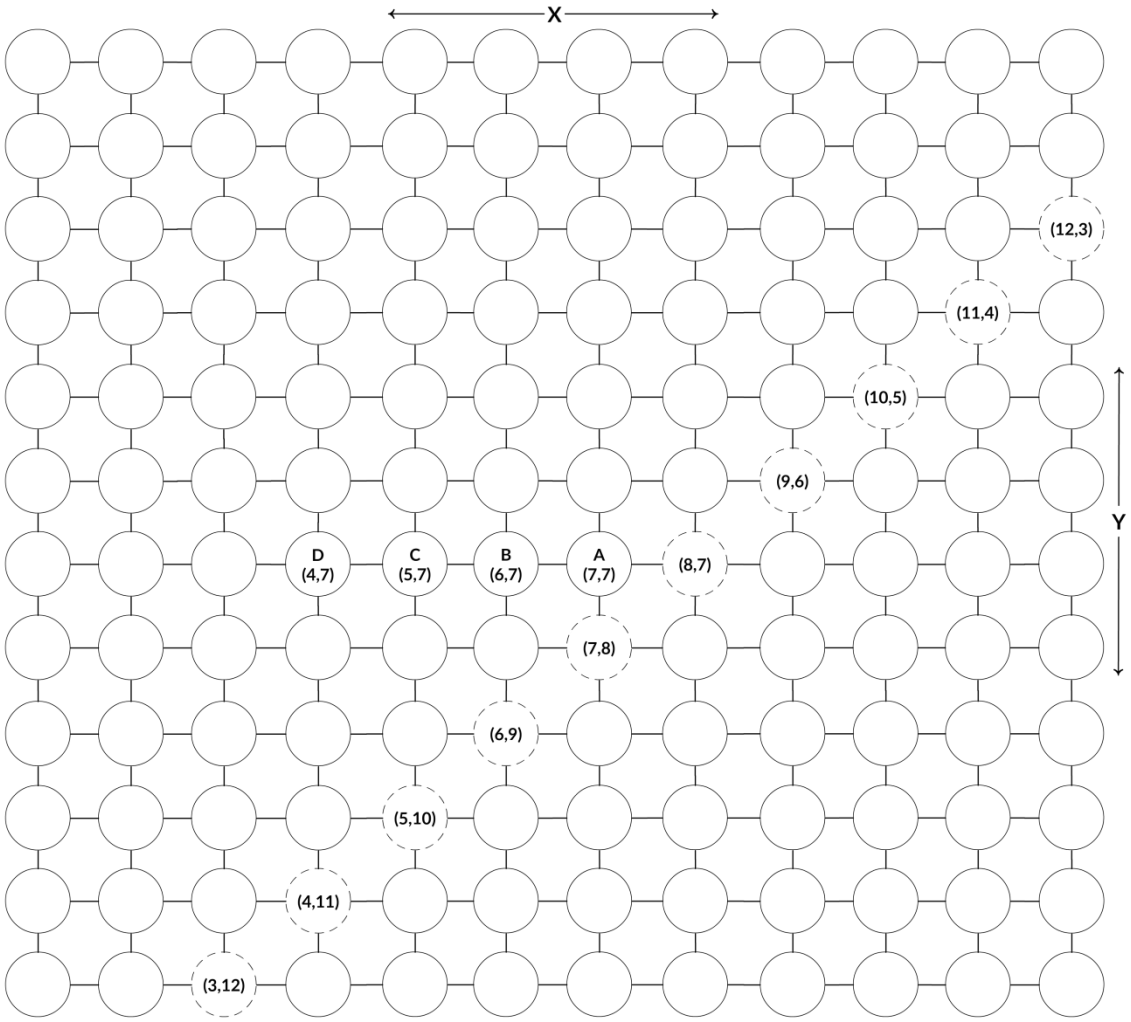

Figure 16: Type 8 example for an increasing diagonal in a 12x12 mesh

In this type, we start with an illustrating example instead of the order followed in the previous sections of the analysis. From the example, we deduce the general relations for this type.

In order to understand the effect and the behavior of some communicating nodes on other switches and depending on the MaxFlex selection function default behavior (i.e. move on X then Y), we check ΔX between the switch under study and the node originating the traffic.

In Figure 16, Consider switch $A(index_{Diagonal} = 4, index_{Row} = 1, n = 12, n' = 10)$ and the switches on the left of it, where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $index_{Diagonal}$ is the position of $W(i,j)$ row in the diagonal originating traffic, $1 \leq index_{Diagonal} \leq n'$; $index_{Row}$ is the distance (number of steps in X-dimension) from switch $W(i,j)$ to the diagonal originating traffic; and SS is the value of the step size. From Figure 16, we list the $(\Delta X, \Delta Y)$ for the nodes originating the traffic that affect the switch under study (switch A + switches on the left) under down traffic in tables 11, 13, 14, and 15. Each of these tables is divided into

[7] The next analysis is the same for increasing and decreasing diagonals. Also, it is the same for up and down traffic in the number of reached nodes formula.

Table 11: Down traffic passing through switch *A*

$index_{Row}$ = 1 (Switch *A*)											
SS = 1			*SS* = 2			*SS* = 3			*SS* = 4		
All	(1,0)	Right	All	(1,0)	Right	All	(1,0)	Right	All	(1,0)	Right
All	(2,1)	Right	All	(2,1)	Up	One	(2,1)	Up	One	(2,1)	Up
All	(3,2)	Right	All	(3,2)	Right	All	(3,2)	Up	One	(3,2)	Up
All	(4,3)	Right	All	(4,3)	Up	All	(4,3)	Right	All	(4,3)	Up
All	(5,4)	Right	All	(5,4)	Right	One	(5,4)	Up	All	(5,4)	Right

Table 12: Summary for the data collected in Table 11

$index_{Row}$ = 1	
ΔX [8]mod *SS*	Number of Nodes Reached
0	All
1	All
2	One
3	One
> 3	One

Table 13: Down traffic passing through switch *B*

$index_{Row}$ = 2 (Switch *B*)								
SS = 2			*SS* = 3			*SS* = 4		
All	(2,0)	Right	All	(2,0)	Right	All	(2,0)	Right
×	(3,1)	×	All	(3,1)	Up	One	(3,1)	Up
All	(4,2)	Right	×	(4,2)	×	All	(4,2)	Up
×	(5,3)	×	All	(5,3)	Right	X	(5,3)	×
All	(6,4)	Right	All	(6,4)	Up	All	(6,4)	Right

a group of columns for each step size value. Each group lists (ΔX , ΔY) values for all the originating nodes, how many diagonal nodes reached, and the port used to reach these nodes.

From Table 11, we summarize the collected data based on ΔX mod *SS*. The summary lists all the values for ΔX mod *SS* and whether any diagonal nodes are reached or not. In case of reaching diagonal nodes, Table 12 lists the number of the reached nodes.

From Table 12, we notice that if the ΔX mod *SS* is zero or one, then all of the intended diagonal nodes can be reached. However, if ΔX mod *SS* is greater than one, only one diagonal node can be reached. The following relations summarize out findings.

[8] We monitored ΔX value due to MaxFlex selection function default behavior i.e. move on X then Y.

Table 14: Down traffic passing through switch *C*

$index_{Row}$ = 3 (Switch *C*)					
SS = 3			*SS* = 4		
All	(3,0)	Right	All	(3,0)	Right
×	(4,1)	×	All	(4,1)	Up
×	(5,2)	×	×	(5,2)	×
All	(6,3)	Right	×	(6,3)	×
×	(7,4)	×	All	(7,4)	Right

Table 15: Down traffic passing through switch *D*

$index_{Row}$ = 4 (Switch *D*)		
SS = 4		
All	(4,0)	Right
×	(5,1)	×
×	(6,2)	×
×	(7,3)	×
All	(8,4)	Right

ΔX mod $SS \in (0,1)$ then All[9]
ΔX mod $SS > 1$ then One[10]

From Table 13 and in a similar manner to what was done in Table 11, we notice that if the ΔX mod *SS* is zero or two, then all of the intended diagonal nodes can be reached. Also, if ΔX mod *SS* equals one, then none of the diagonal nodes can be reached. Finally, if ΔX mod *SS* is greater than two, only one diagonal node can be reached. The following relations summarize out findings.

ΔX mod $SS \in (0,2)$ then All[11]
ΔX mod $SS = 1$ then × (Zero)
ΔX mod $SS > 2$ then One

Similarly, in Table 14 and Table 15, we summarize our findings concerning ΔX mod *SS* and the number of diagonal nodes reached in the relations following each table. From Table 14, we notice the following relations.

ΔX mod $SS \in (0,3)$ then All[12]
ΔX mod $SS \in (1,2)$ then × (Zero)
ΔX mod $SS > 3$ then One

From Table 15, we notice the following relations.

[9] Reach all nodes below or on same row as the switch under study (in this case 5 nodes)
[10] Reach the node on the same column only
[11] Reach all nodes below or on the same row as the switch under study (in this case 4 nodes)
[12] Reach all nodes below or on the same row as the switch under study (in this case 3 nodes)

```
Procedure Count
Inputs: i, j, SS, X_src, X_dst
Outputs: Count
Count = 0
// Nodes on diagonal
for i = 1 to n'
        // Nodes affected by nodes on diagonal from i to n'
        for j = 1 to SS
                // Nodes originating the traffic
                for index = i to n'
                        if (ΔX mod SS ) ∈ (0, j)
                                Count += (i – j)
                        else if (ΔX mod SS > j)
                                Count += 1
```

Figure 17: Procedure for counting the packets passing through a switch for Type 8(a)

$$\Delta X \bmod SS \in (0,4) \text{ then All}^{13}$$
$$\Delta X \bmod SS \in (1,2,3) \text{ then } \times \text{ (Zero)}$$
$$\Delta X \bmod SS > 4 \text{ then One}$$

In order to generalize for $index_{Row}$, we consider all the relations deduced for each $index_{Row}$ value discussed in the previous tables. From these tables and the deduced relations, in Table 16, we calculate the number of reached diagonal nodes under up and down traffic. Also, we calculate the X and Y index of the switch under study i.e. the switch we calculate the number of passing packets for.

Additionally, we generalize the relations for $index_{Row}$ under up and down traffic.

$$\Delta X \bmod SS \in (0, index_{Row}) \text{ then All}$$
$$\Delta X \bmod SS \in (1,2 \ldots index_{Row}\text{-}1) \text{ then } \times \text{ (Zero)}$$
$$\Delta X \bmod SS > index_{Row} \text{ then One}$$

For Type 8, we represent the formulas in form of pseudo code not an equation for easier analysis and explanation. We present pseudo code for up traffic and other for down traffic in the following sub-sections.

3.2.8.1. Type 8 (a)

Description: Packets passing through $W(i,j)$ as a result of up traffic communication between nodes on a diagonal other than node $W(i,j)$ diagonal. This sub-type studies the effect of Type 5 under up traffic on the switches adjacent to a given diagonal.

Count: To calculate the number of packets passing through a switch under up traffic, we use the pseudo code in Figure 17.

[13] Reach all nodes below or on the same row as the switch under study (in this case 2 nodes)

```
Procedure Count
Inputs: i, j, SS, X_src, X_dst
Outputs: Count
Count = 0
// Nodes on diagonal
for i = 1 to n'
        // Nodes affected by nodes on diagonal from 1 to i
        for j = 1 to SS
                // Nodes originating the traffic
                for index = 1 to i
                        if (ΔX mod SS ) ∈ (0, j)
                                Count += (n' – i – j + 1)
                        else if (ΔX mod SS > j)
                                Count += 1
```

Figure 18: Procedure for counting the packets passing through a switch for Type 8(b)

3.2.8.2. Type 8 (b)

Description: Packets passing through $W(i,j)$ as a result of down traffic communication between nodes on a diagonal other than node $W(i,j)$ diagonal. This sub-type studies the effect of Type 5 under down traffic on the switches adjacent to a given diagonal.
Count: To calculate the number of packets passing through a switch under down traffic, we use the pseudo code in Figure 18.

3.2.9. Type 9 Packets

Description: Packets passing through $W(i,j)$ as a result of communication destined to nodes on a diagonal other than node $W(i,j)$ diagonal from nodes with $\Delta X > \Delta Y$.[14]

Similar to what was done in Type 8, for Type 9; we represent the formulas in form of pseudo code for up traffic and down traffic in the following sub-sections.

3.2.9.1. Type 9 (a, c)

Description: Packets passing through $W(i,j)$ as a result of up traffic communication destined to nodes on a diagonal other than node $W(i,j)$ diagonal from nodes with $\Delta X > \Delta Y$. This sub-type studies the effect of sub-type 6(a) and sub-type 6(b) on the switches adjacent to a given diagonal.
Count: To calculate the number of packets passing through a switch under up traffic, we use the pseudo code in Figure 19.

[14] Type 9 and Type 10 analysis is the same as Type 8.

```
Procedure Count
Inputs: i, j, SS, X_src, X_dst
Outputs: Count
-----------------------------------------------------
Count = 0
// Nodes on diagonal
for i = 1 to n'
        // Nodes affected by nodes on diagonal from i to n'
        for j = 1 to SS
                // Nodes originating the traffic after and including index node
                for index = i to n'
                        if (ΔX mod SS ) ∈ (0, j)
                                Count += (i – j) * Multiplier
                        else if (ΔX mod SS > j)
                                Count += Multiplier
```

Figure 19: Procedure for counting the packets passing through a switch for Type 9(a, c)

Where for ***increasing diagonal***
If diagonal is below main diagonal i.e. $n+1-Node_y < Node_x$

$$Multiplier = n - index$$

Else

$$Multiplier = n' - index$$

And for ***decreasing diagonal***
If diagonal is above main diagonal i.e. $Node_y > Node_x$

$$Multiplier = n - index$$

Else

$$Multiplier = n' - index$$

Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $Node_x$ is the X index of the node belonging to both the diagonal originating the traffic and $W(i,j)$ row; and $Node_y$ is the Y index of the node belonging to both the diagonal originating the traffic and $W(i,j)$ row.
Proof: The proof for this type is similar to the proof of Type 8.

3.2.9.2. Type 9 (b, d)

Description: Packets passing through $W(i,j)$ as a result of down traffic communication destined to nodes on a diagonal other than node $W(i,j)$ diagonal from nodes with $\Delta X > \Delta Y$. This sub-type studies the effect of sub-type 6(b) and sub-type 6(d) on the switches adjacent to a given diagonal.
Count: To calculate the number of packets passing through a switch under down traffic, we use the pseudo code in Figure 20.

```
Procedure Count
Inputs: i, j, SS, X_src, X_dst
Outputs: Count
-----------------------------------------------------------
Count = 0
// Nodes on diagonal
for i = 1 to n'
        // Nodes affected by nodes on diagonal from 1 to i
        for j = 1 to SS
                // Nodes originating the traffic before and including index node
                for index = 1 to i
                        if (ΔX mod SS ) ∈ (0, j)
                                Count += (n' – i – j + 1) * Multiplier
                        else if (ΔX mod SS > j)
                                Count += Multiplier
```

Figure 20: Procedure for counting the packets passing through a switch for Type 9(b, d)

Where for ***increasing diagonal***
If diagonal is above main diagonal i.e. $n+1-Node_y > Node_x$

$$Multiplier = index-1+n-n'$$

Else

$$Multiplier = index-1$$

And for ***decreasing diagonal***
If diagonal is below main diagonal i.e. $Node_y < Node_x$

$$Multiplier = index-1+n-n'$$

Else

$$Multiplier = index-1$$

Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $Node_x$ is the X index of the node belonging to both the diagonal originating the traffic and $W(i,j)$ row; and $Node_y$ is the Y index of the node belonging to both the diagonal originating the traffic and $W(i,j)$ row.
Proof: The proof for this type is similar to the proof of Type 8.

3.2.10. Type 10 Packets

Description: Packets passing through $W(i,j)$ as a result of communication destined to nodes on a diagonal other than node $W(i,j)$ diagonal with from nodes with $\Delta X < \Delta Y$.

Similar to what was done in Type 8 and Type 9, for Type 10; we represent the formulas in form of pseudo code for up traffic and down traffic in the following sub-sections.

3.2.10.1. Type 10 (a, c)

Description: Packets passing through $W(i,j)$ as a result of up traffic communication destined to nodes on a diagonal other than node $W(i,j)$ diagonal with from nodes with $\Delta X < \Delta Y$. This sub-type studies the effect of sub-type 7(a) and sub-type 7(c) on the switches adjacent to a given diagonal.

```
Procedure Count
Inputs: i, j, SS, X_src, X_dst
Outputs: Count
-----------------------------------------------------------
Count = 0
// Nodes on diagonal
for i = 1 to n'
        // Nodes affected by nodes on diagonal from i to n'
        for j = 1 to SS
                // Nodes originating the traffic after and including index node
                for index = i to n'
                        if (ΔX mod SS ) ∈ (0, j)
                                Count += (i – j) * Multiplier
                        else if (ΔX mod SS > j)
                                Count += Multiplier
```

Figure 21: Procedure for counting the packets passing through a switch for Type 10(a, c)

Count: To calculate the number of packets passing through a switch under up traffic, we use the pseudo code in Figure 21.

Where for ***increasing diagonal***
If diagonal is above main diagonal i.e. $n+1-Node_y > Node_x$

$$Multiplier = n - index$$

Else

$$Multiplier = n' - index$$

And for ***decreasing diagonal***
If diagonal is below main diagonal i.e. $Node_y < Node_x$

$$Multiplier = n - index$$

Else

$$Multiplier = n' - index$$

Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $Node_x$ is the X index of the node belonging to both the diagonal originating the traffic and $W(i,j)$ row; and $Node_y$ is the Y index of the node belonging to both the diagonal originating the traffic and $W(i,j)$ row.
Proof: The proof for this type is similar to the proof of Type 8.

3.2.10.2. Type 10 (b, d)

Description: Packets passing through $W(i,j)$ as a result of down traffic communication destined to nodes on a diagonal other than node $W(i,j)$ diagonal with from nodes with $\Delta X < \Delta Y$. This sub-type studies the effect of sub-type 7(b) and sub-type 7(d) on the switches adjacent to a given diagonal.
Count: To calculate the number of packets passing through a switch under down traffic, we use the pseudo code in Figure 22.

```
Procedure Count
Inputs: i, j, SS, X_src, X_dst
Outputs: Count
-----------------------------------------------------------
Count = 0
// Nodes on diagonal
for i = 1 to n'
        // Nodes affected by nodes on diagonal from 1 to i
        for j = 1 to SS
                // Nodes originating the traffic before and including index node
                for index = 1 to i
                        if (ΔX mod SS ) ∈ (0, j)
                                Count += (n' – i – j + 1) * Multiplier
                        else if (ΔX mod SS > j)
                                Count += Multiplier
```

Figure 22: Procedure for counting the packets passing through a switch for Type 10(b, d)

Where for ***increasing diagonal***
If diagonal is below main diagonal i.e. $n+1-Node_y < Node_x$

$$Multiplier = index-1+n-n'$$

Else

$$Multiplier = index-1$$

And for ***decreasing diagonal***
If diagonal is above main diagonal i.e. $Node_y > Node_x$

$$Multiplier = index-1+n-n'$$

Else

$$Multiplier = index-1$$

Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $Node_x$ is the X index of the node belonging to both the diagonal originating the traffic and $W(i,j)$ row; and $Node_y$ is the Y index of the node belonging to both the diagonal originating the traffic and $W(i,j)$ row.
Proof: The proof for this type is similar to the proof of Type 8.

3.2.11. Type 11 Packets

Description: Packets passing through $W(i,j)$ as a result of communication between node (i,k) from same row as node $W(i,j)$ and nodes on node (i,m) diagonal where $1 \leq k, m \leq n$ and $j \neq k \neq m$.

We divide the discussion of Type 11 into four separate sub-types to ease the calculation for each of them. The sub-types are listed in the following sub-sections.

3.2.11.1. Type 11 (a)

Description: Packets passing through switch $W(i,j)$ as a result of same row nodes communicating with nodes with lower index on increasing diagonal i.e. up traffic.
Count: $(index_{Row}-1)(index_{Diagonal}-1)$
Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $index_{Diagonal}$ is the position of switch $W(i,j)$ row in the destination diagonal,

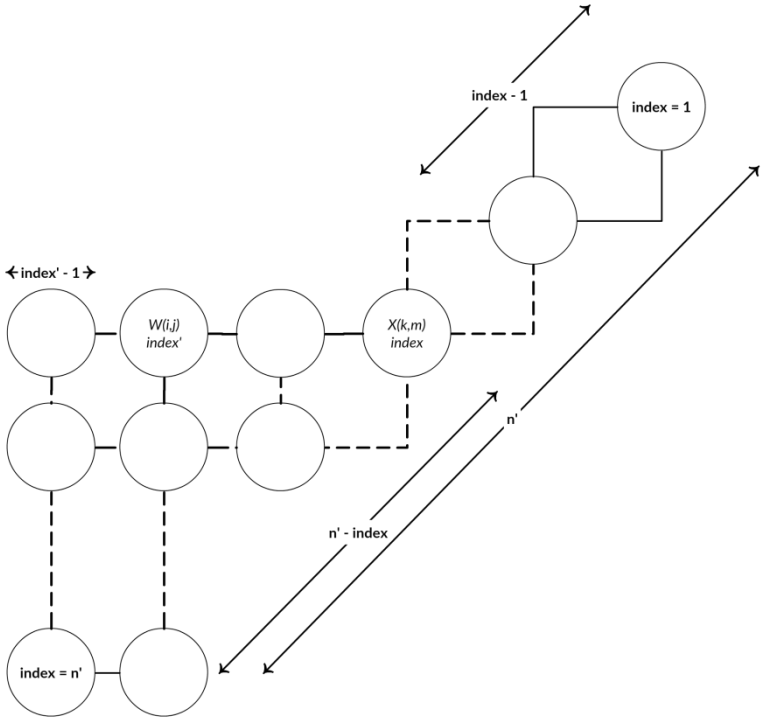

Figure 23: Location of *W*(*i*,*j*) in 2D mesh

$1 \leq index_{Diagonal} \leq n'$; and $index_{Row}$ is the position of *W*(*i*,*j*) in the row originating traffic, $1 \leq index_{Row} \leq L$.

If diagonal is below main diagonal i.e. $n+1-Node_y < Node_x$

$$L = n - index_{Diagonal}$$

Else

$$L = n' - index_{Diagonal}$$

Where $Node_x$ is the *X* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) row; and $Node_y$ is the *Y* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) row.

Proof: We consider increasing diagonal for the proof; however, the same proof applies to decreasing diagonal. In this type, we follow the same steps as in Type 6. However, in Type 11, we focus on the effect of the same row nodes communication with the diagonal nodes.

In this type, as shown in Figure 23, switch *X*(*k*,*m*) belongs to a 2D mesh diagonal with n' nodes at position $index_{Diagonal}$ (denoted *index* in Figure 23). We have $(index_{Diagonal} - 1)$ diagonal nodes before node *X*(*k*,*m*) and $(n' - index_{Diagonal})$ diagonal nodes after *X*(*k*,*m*). Also, there are $(n - index_{Diagonal})$ on the same row as *X*(*k*,*m*) before node *X*(*k*,*m*). Let switch *W*(*i*,*j*) belongs to one of these $(n - index_{Diagonal})$ nodes at position $index_{Row}$ (denoted $index'$ in Figure 23) with $(index_{Row} - 1)$ nodes before it on the same row.

For this type, under up traffic, each of the $(n - index_{Diagonal})$ nodes belonging to the same row as *X*(*k*,*m*) send packets to the (*index* - 1) diagonal nodes before node *X*(*k*,*m*). Since *W*(*i*,*j*) is one of these nodes, then each of the packets sent by the $(index_{Row} - 1)$ before *W*(*i*,*j*) in the same row passes through *W*(*i*,*j*). Since each node sends only one packet to each of the NoC nodes (check the assumptions), the

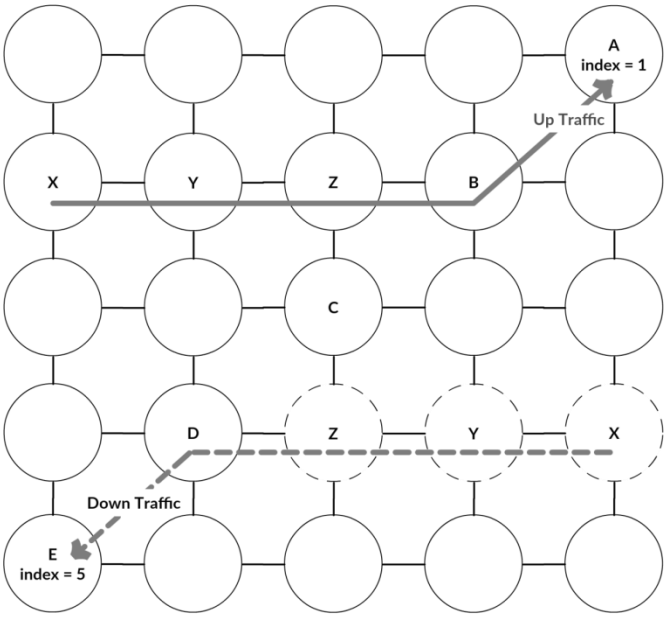

Figure 24: Type 11 example for an increasing diagonal under both up and down traffics

Table 16: Up traffic passing through switch Z_{Solid}

Source → Destination	Pass $W(i,j)$ or Not
$X_{Solid} \rightarrow A$	Pass
$Y_{Solid} \rightarrow A$	Pass

overall count for the packets passing through $W(i,j)$ is $(index_{Row} - 1)$ * $(index_{Diagonal} - 1)$ packets.

■

In order to illustrate the *Count* calculations, in Figure 24, we show the up traffic from the nodes on the same row as switch Z_{Solid}($index_{Diagonal}$ = 2, $index_{Row}$ = 3, n = 5, n' = 5) i.e. traffic from (X_{Solid}, Y_{Solid}) to A. Table 17 lists all the communication from (X_{Solid}, Y_{Solid}) to A and whether the packets to A will pass switch Z_{Solid} or not.

Table 17 states that two packets pass switch Z_{Solid} under up traffic (*Count* = 2). Also, applying the *Count* equation, the number of packets passing through switch Z_{Solid} is two (*Count* = (3 – 1) * (2 – 1) = 2 * 1 = 2) which matches to the value deduced from Table 17.

3.2.11.2. Type 11 (b)

Description: Packets passing through switch $W(i,j)$ as a result of same row nodes communicating with nodes with higher index on increasing diagonal i.e. down traffic.

Count: $(index_{Row} - 1)(n' - index_{Diagonal})$

Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $index_{Diagonal}$ is the position of switch $W(i,j)$ row in the destination diagonal,

$1 \leq index_{Diagonal} \leq n'$; and $index_{Row}$ is the position of $W(i,j)$ in the row originating traffic, $1 \leq index_{Row} \leq L$.

If diagonal is above main diagonal i.e. $n+1-Node_y > Node_x$

$$L = index_{Diagonal} - 1 + n - n'$$

Else

$$L = index_{Diagonal} - 1$$

Where $Node_x$ is the X index of the node belonging to both the destination diagonal and $W(i,j)$ row; and $Node_y$ is the Y index of the node belonging to both the destination diagonal and $W(i,j)$ row.

Proof: The proof for this type is similar to the proof of Type 11(a).

3.2.11.3. Type 11 (c)

Description: Packets passing through switch $W(i,j)$ as a result of same row nodes communicating with nodes with lower index on increasing diagonal i.e. up traffic.

Count: $(index_{Row} - 1)(index_{Diagonal} - 1)$

Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $index_{Diagonal}$ is the position of switch $W(i,j)$ row in the destination diagonal, $1 \leq index_{Diagonal} \leq n'$; and $index_{Row}$ is the position of $W(i,j)$ in the row originating traffic, $1 \leq index_{Row} \leq L$.

If diagonal is above main diagonal i.e. $Node_y > Node_x$

$$L = n - index_{Diagonal}$$

Else

$$L = n' - index_{Diagonal}$$

Where $Node_x$ is the X index of the node belonging to both the destination diagonal and $W(i,j)$ row; and $Node_y$ is the Y index of the node belonging to both the destination diagonal and $W(i,j)$ row.

Proof: The proof for this type is similar to the proof of Type 11(a).

3.2.11.4. Type 11 (d)

Description: Packets passing through switch $W(i,j)$ as a result of same row nodes communicating with nodes with higher index on decreasing diagonal i.e. down traffic.

Count: $(index_{Row} - 1)(n' - index_{Diagonal})$

Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $index_{Diagonal}$ is the position of switch $W(i,j)$ row in the destination diagonal, $1 \leq index_{Diagonal} \leq n'$; and $index_{Row}$ is the position of $W(i,j)$ in the row originating traffic, $1 \leq index_{Row} \leq L$.

If diagonal is below main diagonal i.e. $Node_y < Node_x$

$$L = index_{Diagonal} - 1 + n - n'$$

Else

$$L = index_{Diagonal} - 1$$

Where $Node_x$ is the *X* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) row; and $Node_y$ is the *Y* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) row.

Proof: The proof for this type is similar to the proof of Type 11(a).

3.2.12. Type 12 Packets

Description: Packets passing through *W*(*i*,*j*) as a result of communication between node (*k*,*j*) from same column as node *W*(*i*,*j*) and nodes on node (*m*,*j*) diagonal where $1 \leq k, m \leq n$ and $i \neq k \neq m$.

We divide the discussion of Type 12 into four separate sub-types to ease the calculation for each of them. The sub-types are listed in the following sub-sections.

3.2.12.1. Type 12 (a)

Description: Packets passing through switch *W*(*i*,*j*) as a result of same column nodes communicating with nodes with lower index on increasing diagonal i.e. up traffic.

Count: $(index_{Column}-1)(index_{Diagonal}-1)$

Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $index_{Diagonal}$ is the position of switch *W*(*i*,*j*) column in the destination diagonal, $1 \leq index_{Diagonal} \leq n'$; and $index_{Column}$ is the position of *W*(*i*,*j*) in the column originating traffic, $1 \leq index_{Column} \leq L$.

If diagonal is above main diagonal i.e. $n+1-Node_y > Node_x$

$$L = n - index_{Diagonal}$$

Else

$$L = n' - index_{Diagonal}$$

Where $Node_x$ is the *X* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) column; and $Node_y$ is the *Y* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) column.

Proof: The proof for this type is similar to the proof of Type 11(a).

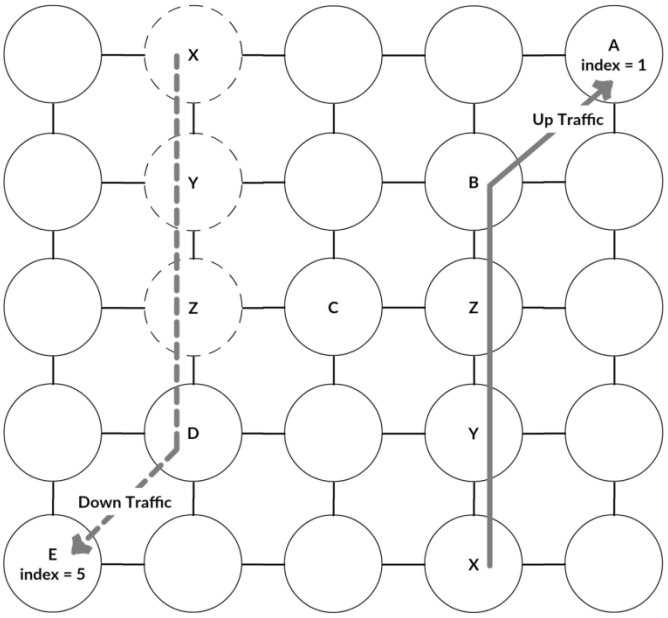

Figure 25: Type 12 example for an increasing diagonal under both up and down traffics

Table 17: Up traffic passing through switch Y_{Solid}

Source → Destination	Pass $W(i,j)$ or Not
X_{Solid} → A	Pass

In order to illustrate the *Count* calculations, in Figure 25, we study the up traffic from the nodes on the same column as switch Y_{Solid}($index_{Diagonal}$ = 2, $index_{Column}$ = 2, n = 5, n' = 5) i.e. traffic from X_{Solid} to *A*. Table 18 lists all the communication from X_{Solid} to *A* and whether the packets to *A* will pass switch Y_{Solid} or not.

Table 18 states that only one packet passes switch Y_{Solid} under up traffic (*Count* = 1). Also, applying the *Count* equation, the number of packets passing through switch Y_{Solid} is one (*Count* = (2 – 1) * (2 – 1) = 1 * 1 = 1) which matches to the value deduced from Table 18.

3.2.12.2. Type 12 (b)

Description: Packets passing through switch $W(i,j)$ as a result of same column nodes communication with nodes with higher index on increasing diagonal i.e. down traffic.

Count: $(index_{Column}-1)(n'-index_{Diagonal})$

Where n' is the number of nodes in the diagonal originating traffic, $1 \le n' \le n$; $index_{Diagonal}$ is the position of switch $W(i,j)$ column in the destination diagonal, $1 \le index_{Diagonal} \le n'$; and $index_{Column}$ is the position of $W(i,j)$ in the column originating traffic, $1 \le index_{Column} \le L$.

If diagonal is below main diagonal i.e. $n+1-Node_y < Node_x$

$$L = index_{Diagonal} - 1 + n - n'$$

Else

$$L = index_{Diagonal} - 1$$

Where $Node_x$ is the *X* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) column; and $Node_y$ is the *Y* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) column.

Proof: The proof for this type is similar to the proof of Type 11(a).

3.2.12.3. Type 12 (c)

Description: Packets passing through switch *W*(*i*,*j*) as a result of same column nodes communication with nodes with lower index on decreasing diagonal i.e. up traffic.

Count: $(index_{Column} - 1)(index_{Diagonal} - 1)$

Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $index_{Diagonal}$ is the position of switch *W*(*i*,*j*) column in the destination diagonal, $1 \leq index_{Diagonal} \leq n'$; and $index_{Column}$ is the position of *W*(*i*,*j*) in the column originating traffic, $1 \leq index_{Column} \leq L$.

If diagonal is below main diagonal i.e. $Node_y < Node_x$

$$L = n - index_{Diagonal}$$

Else

$$L = n' - index_{Diagonal}$$

Where $Node_x$ is the *X* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) column; and $Node_y$ is the *Y* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) column.

Proof: The proof for this type is similar to the proof of Type 11(a).

3.2.12.4. Type 12 (d)

Description: Packets passing through switch *W*(*i*,*j*) as result of same column nodes communication with nodes with higher index in decreasing diagonal i.e. down traffic.

Count: $(index_{Column} - 1)(n' - index_{Diagonal})$

Where n' is the number of nodes in the diagonal originating traffic, $1 \leq n' \leq n$; $index_{Diagonal}$ is the position of switch *W*(*i*,*j*) column in the destination diagonal, $1 \leq index_{Diagonal} \leq n'$; and $index_{Column}$ is the position of *W*(*i*,*j*) in the column originating traffic, $1 \leq index_{Column} \leq L$.

If diagonal is above main diagonal i.e. $Node_y > Node_x$

$$L = index_{Diagonal} - 1 + n - n'$$

Else

$$L = index_{Diagonal} - 1$$

Where $Node_x$ is the *X* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) column; and $Node_y$ is the *Y* index of the node belonging to both the destination diagonal and *W*(*i*,*j*) column.

Proof: The proof for this type is similar to the proof of Type 11(a).

Table 18: Formulas for different traffic types

Type	Formula
1	Ejection $n^2 - 1$
2	Injection $n^2 - 1$
3	Row $2(index-1)(n-index)$
4	Column $2(index-1)(n-index)$
5	Up Traffic $(index-1)\left\lfloor \frac{n'-index}{SS} \right\rfloor$
	Down Traffic $(n'-index)\left\lfloor \frac{index-1}{SS} \right\rfloor$
6	Up Traffic $(index-1)\sum_{x=index}^{n'} Q \times (1-\left\lceil \frac{(x-index) \bmod SS}{SS} \right\rceil)$ if diagonal below main diagonal $Q = n - x$ else $Q = n'-x$
	Down Traffic $(n'-index)\sum_{x=1}^{index} Q \times (1-\left\lceil \frac{(index-x) \bmod SS}{SS} \right\rceil)$ if diagonal above main diagonal $Q = x - 1 + n - n'$ else $Q = x - 1$
7	Up Traffic $(index-1)\sum_{x=index}^{n'} Q \times (1-\left\lceil \frac{(x-index) \bmod SS}{SS} \right\rceil)$ if diagonal above main diagonal $Q = n - x$ else $Q = n'-x$
	Down Traffic $(n'-index)\sum_{x=1}^{index} Q \times (1-\left\lceil \frac{(index-x) \bmod SS}{SS} \right\rceil)$ if diagonal below main diagonal $Q = x - 1 + n - n'$ else $Q = x - 1$

<table>
<tr><td>8, 9, 10</td><td>Procedure Count
Inputs: i, j, SS, X_{src}, X_{dst}
Outputs: Count
// Nodes on diagonal
for i = 1 to n'
// Nodes affected by nodes on diagonal from i to n'
for j = 1 to SS
// Nodes originating the traffic
for index = A to B
if (ΔX mod SS) ϵ (0, j)
Count += (i – j) * Multiplier
else if (ΔX > j)
Count += Multiplier</td></tr>
<tr><td rowspan="2">11</td><td>Up Traffic
$(index_{Row}-1)(index_{Diagonal}-1)$
if diagonal below main diagonal
$L = n - index_{Diagonal}$
else
$L = n' - index_{Diagonal}$</td></tr>
<tr><td>Down Traffic
$(index_{Row}-1)(n' - index_{Diagonal})$
if diagonal above main diagonal
$L = index_{Diagonal} - 1 + n - n'$
else
$L = index_{Diagonal} - 1$</td></tr>
<tr><td rowspan="2">12</td><td>Up Traffic
$(index_{Column}-1)(index_{Diagonal}-1)$
if diagonal above main diagonal
$L = n - index_{Diagonal}$
else
$L = n' - index_{Diagonal}$</td></tr>
<tr><td>Down Traffic
$(index_{Column}-1)(n' - index_{Diagonal})$
if diagonal below main diagonal
$L = index_{Diagonal} - 1 + n - n'$
else
$L = index_{Diagonal} - 1$</td></tr>
</table>

3.2.13. Summary of Packets Count Calculations[15]

In this section, we summarize the number of the packets passing through a switch. Table 19 shows the number of packets caused by all the types with a step size of *SS*. The variables used in Table 19 are described in Table 20.

[15] The listed equations and pseudo code are for the increasing diagonals only, but the same applies for the decreasing diagonals.

Table 19: Common variables used in Table 19

n	Number of row/column nodes	-
n'	Number of diagonal nodes	$1 \le n' \le n$
$index$ / $index_{Diagonal}$	Index in diagonal	$1 \le index / index_{Diagonal} \le n'$
$index_{Row}$ / $index_{Column}$	Index in row/column	$1 \le index_{Row} / index_{Column} \le L$

Table 20: A and B values for up and down traffic

	Up Traffic	**Down Traffic**
A	i	1
B	n'	i

Table 21: Multiplier value for Type 8, Type 9 and Type 10

Type	**Up Traffic**	**Down Traffic**
8	1	1
9	if diagonal below main diagonal $n - index$ else $n' - index$	if diagonal above main diagonal $index - 1 + n - n'$ else $index - 1$
10	if diagonal above main diagonal $n - index$ else $n' - index$	if diagonal below main diagonal $index - 1 + n - n'$ else $index - 1$

Table 22: Values for Type 9 up traffic communication

A	i
B	n'
Multiplier	if diagonal below main diagonal $n - index$ else $n' - index$

For Type 8, Type 9, and Type 10, we devised a pseudo code to calculate the count instead of using formulas it is more readable that way. Table 19 shows a generic code for the number of packets passing through a switch $W(i,j)$ for Type 8, Type 9 and Type 10 based on the values in Table 21 and Table 22. For example, Table 23 shows the values for Type 9 up traffic.

3.3. Proof of Packet Types Completeness

In this section, we prove that any communication between two nodes falls under one of the mentioned 12 types.

Table 23: First category cases

Case #	Condition	Description
1	$\Delta X = \Delta Y$	*P* moves on the diagonal from *S* to *D*. This case causes the pattern defined in Type 5.
2	$\Delta X = 0$	*P* moves on a column from *S* to *D*. This case causes the pattern defined in Type 4.
3	$\Delta Y = 0$	*P* moves on a row from *S* to *D*. This case causes the pattern defined in Type 3.
4	$\Delta X > \Delta Y$	*P* moves on a row till $\Delta X = \Delta Y$ then follows Case 1. This case causes the pattern defined in Type 6.
5	$\Delta X < \Delta Y$	*P* moves on a column till $\Delta X = \Delta Y$ then follows Case 1. This case causes the pattern defined in Type 7.

Table 24: Second category cases

Case #	Condition	Description
6	$\Delta X = \Delta Y$	*P* moves on the diagonal from *S* to *D*. The movement on the diagonal leads the packet to pass through switches on nearby diagonals. This case causes the pattern defined in Type 8.
7	$\Delta X > \Delta Y$	*P* moves on a row till $\Delta X = \Delta Y$ then follows Case 6. This case causes two patterns; moving on row causes the pattern defined in Type 11 and moving on diagonal causes the pattern defined in Type 9.
8	$\Delta X < \Delta Y$	*P* moves on a column till $\Delta X = \Delta Y$ then follows Case 6. This case causes two patterns; moving on column causes the pattern defined in Type 12 and moving on diagonal causes the pattern defined in Type 10.

Lemma In an $n \times n$ mesh, under MMaxFlex, any packet going from a source node to a destination node falls under one of the mentioned twelve traffic types.

Proof Here, we differentiate the patterns going through *W*(*i*,*j*) into two main categories:

1) The patterns due to moving to nodes on same row, column, or diagonal as *W*(*i*,*j*)
2) The patterns due to moving to nodes on different diagonal than that of *W*(*i*,*j*) (i.e. the effect on *W*(*i*,*j*) caused by category one)

Concerning the first category, consider the possible values for ΔX w.r.t ΔY. We list the cases in Table 24.

Now we consider the second category. Beside the patterns in first category, packets may pass through a switch as a result of other diagonal communication. This is because adjacent diagonal affects nodes other than its own nodes as there is no direct link between diagonal nodes. Thus, moving on diagonal will lead to move right-up or left-down. The effect differs based on the value of ΔX w.r.t ΔY as shown in Table 25.

Beside the two categories, we have two special cases not related to MaxFlex work; Node *W*(*i*,*j*) injecting to all other nodes (this case causes the pattern defined in Type 2), and Node *W*(*i*,*j*) receiving from all other nodes (this case causes the pattern defined in Type 1).

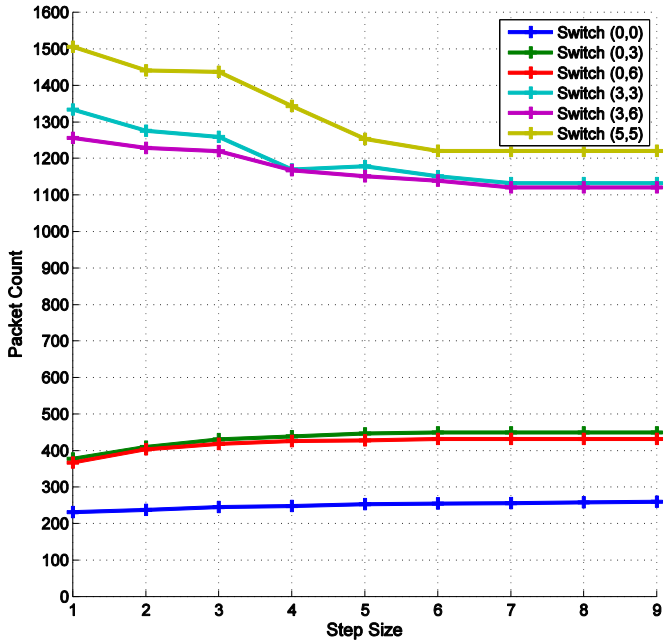

Figure 26: Number of packet passing through sample border and core switches over different fixed step size values

Thus, the above cases cover the 12 mentioned types proving the lemma.

■

3.4. Packets Distribution Analysis Results

In this section, we calculate the number of packets passing through each switch in a 10x10 2D mesh network using the count equations presented in Section 3.2 using different step sizes. We choose some representative switches based on their location in the network to represent border switches and core switches. We choose Switch (0, 0), Switch (0, 3) and Switch (0, 6) as border switches and Switch (3, 3), Switch (3, 6), Switch (5, 5) as core switches.

Figure 26 shows the number of packets passing through each of the mentioned switches with different step sizes. From the figure, we notice different trends; for the border nodes, the number of packets passing through the switch increases as the step size increases, while for the core switches, the number of packets decreases as the step size increases. In other words, the concentration in the central part of network bisection is relaxed.

This is because, as the step size increases, the packet moves in one dimension for more steps before alternating the dimension. This movement enables the packet to reach farther switches (i.e. switches away from the diagonals) which allows some relaxation for the core diagonal switches.

3.5. Experimental Setup

In this section, we present the method used to evaluate MMaxFlex. Also, we present the model of the used bufferless NoC. Finally, we define the performance

metrics used to evaluate the proposed approach. In the next section, we evaluate MMaxFlex selection function using different step sizes in terms of the used performance metrics. In addition, we calculate an approximate value for the optimal step size given a certain dimension.

3.5.1. Experimental Methodology

We evaluate the network performance of bufferless NoCs using the General purpose Simulator gpNoCsim [39]. The simulator is an open-source, component based simulation framework for NoC architectures that is developed entirely in Java. In gpNoCsim, we have either a processing node (a message generation or consumption points) or a switch connected through bidirectional links. Each switch has a router and a controller. gpNoCsim uses the wormhole switching technique. Processing nodes clock is synchronized with the switches.

3.5.2. Interconnection Network Model

We use the 2D mesh topology of varying size to model the network. Each switch has 5 input ports and 5 output ports, including the injection ports. Each of the switch latency and link latency is 1 cycle. In our configuration, we assume that each link is 128-bit wide and each data packet consists of 8 flits, each of which is assumed to have 128 bits. All packets are of fixed length. For comparing the effect of increasing the step size, we use a 10x10 mesh. On the other hand, for calculating the optimal step size given the 2D mesh dimension, we use a mesh size varying from 5x5 to 12x12.

We use synthetic traces to evaluate MaxFlex. Synthetic traces are used for various sensitivity analyses, as well as for comparing the different step sizes among each other and with other baseline selection functions. Each switch is associated with a processor and the destination address of a packet is determined by the statistical process of the uniform traffic pattern. Within each simulation there is a warm-up period of 100,000 cycles. The simulation terminates when 1000,000 packets are received.

3.5.3. Evaluation Metrics

Our main performance metrics for system performance evaluation are the average packet latency and the average flit deflection count. Packet latency is calculated as the time the packet takes to reach the destination (Last Flit Ejection Time – First Flit Generation Time) including source queuing time. Flit deflection count is the number of times the flit was forced to go through a non-productive port i.e. misroute.

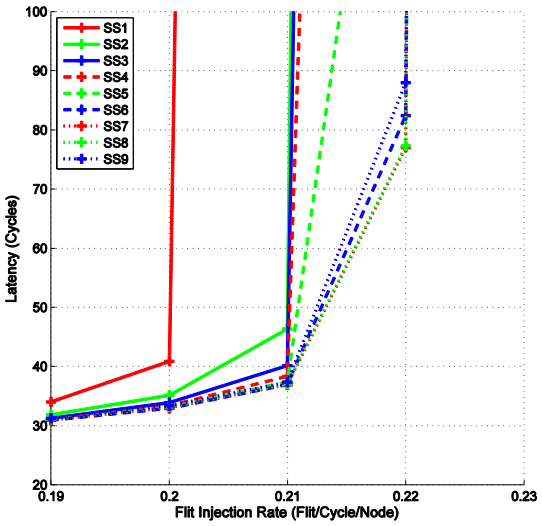

Figure 27: Average packet latency for different fixed step size values

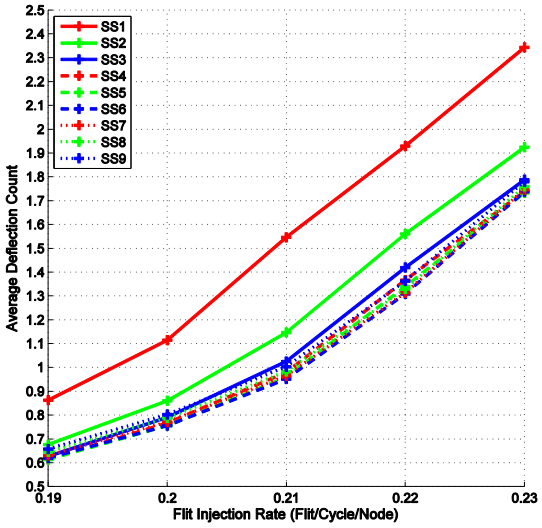

Figure 28: Average deflection count for different fixed step size values

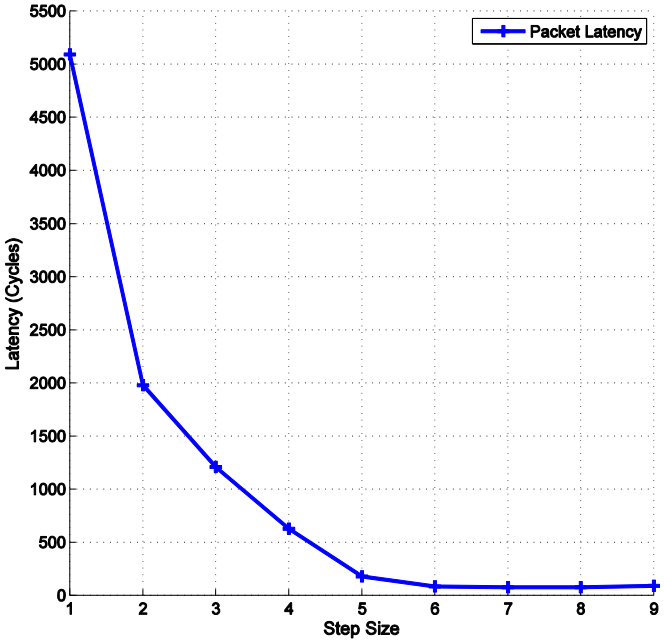

Figure 29: Average packet latency for different fixed step sizes at flit injection rate = 0.22 flit/cycle/node

3.6. Simulation Results

Here we show the results of increasing the step size under MMaxFlex and the MaxFlex performance compared with other selection functions. Figure 27 and Figure 28 show that as the step size increases, both the average packet latency and the average deflection count decreases. These results matches the analysis results in Section 3.4, as the better traffic distribution showed in the analysis can lead to better link utilization which can lead to faster delivery for the packets and hence better packet latency and

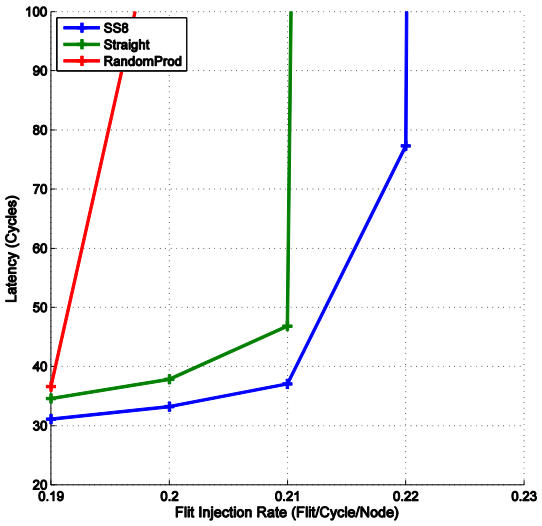

Figure 30: Average packet latency for fixed step size of 8 compared with different selection functions

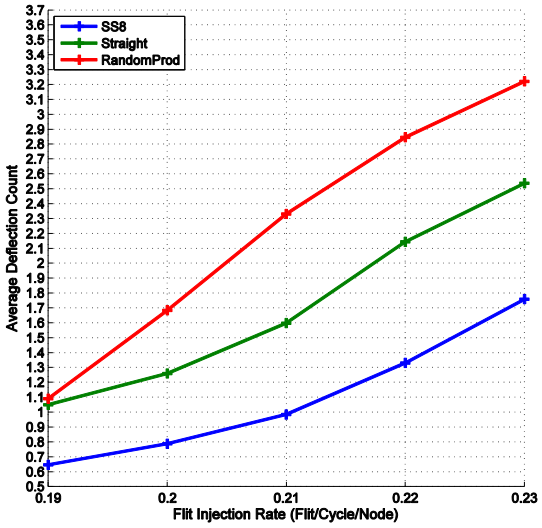

Figure 31: Average deflection count for fixed step size of 8 compared with different selection functions

less misrouting due to contention. Figure 29 focuses on the cut-off point of the flit injection rate of 0.22 flit/cycle/node i.e. the point after which the latency increases exponentially. The figure shows that for smaller step sizes, the average packet latency is very high (magnitude of thousands of cycles). While for larger step sizes the average packet latency is much smaller with the smallest packet latency achieved using step size of 8. The average packet latency using larger step sizes is almost equal as all these step size values lead to almost the same packet movements. For example, moving from node (0, 5) to (5, 10) under step size of 1 leads to moving one step in *X* then one step in *Y* till the destination is reached, while using a step size of 6 will lead to moving 6 steps in *X* then 6 steps in *Y*. Since the number of steps remaining in *X* is less than 6, the packet will move as if it uses dimension order routing (DO-XY). The same applies for step sizes larger than 6.

Now, we compare increasing the step size under MaxFlex with other selection functions, namely Straight Line selection function and random productive port selection function. In the Straight Line selection function, the flit favors the *X*-dimension movement till there are no steps remaining in *X*-dimension then moves in *Y*-dimension. In the random productive port selection function, the flit randomly chooses from the list of productive ports available at each step. Figure 30 and Figure 31 show that increasing the fixed step size under MaxFlex leads to better average packet latency and smaller deflection count. Specifically, using a fixed step size of 8 enhances the average packet latency by around 95% and 99% over using Straight Line selection function and random productive port selection function respectively. Also, the average deflection count decreases by 38% and 53% compared with Straight Line selection function and random productive port selection function respectively.

Table 25: Step size to mesh dimension percentage

Mesh Size	Best Step Size	Percentage
5x5	4	80
6x6	4	66.67
7x7	5	71.43
8x8	6	75
9x9	6	66.67
10x10	8	80
11x11	8	72.73
12x12	9	75

3.7. Estimation of the Value of the Step Size

In this section, given an $n \times n$ mesh, we estimate the value of the step size. In order to do this, we simulated the MaxFlex under different 2D mesh sizes varying from 5x5 to 12x12 and within each network we used step sizes ranging from 1 to $n - 1$. For example, for 7x7 mesh network, we used step sizes ranging from 1 to 6. The results are shown in the Table 26. Column 1 represents the mesh size, column 2 represents the best step size achieved, and column 3 represents the percentage of the step size to the dimension of the mesh.

Table 26 shows that using a step size with a value ranging from 60% to 80% of the 2D mesh dimension leads to better network performance. Based on the fixed step size analysis and simulation results, we conclude that using a larger value for the step size leads to better network performance. This is due to the better distribution of traffic among the network switches.

3.8. Concluding Remarks

In this chapter, we presented the idea of increasing the used step size under MaxFlex selection function. We started by analyzing the uniform traffic distribution under MaxFlex. We found that the traffic is divided into 12 different types. We studied how increasing the used fixed step size value can affect the overall traffic distribution among the NoC switches and links. Our analysis showed that increasing the step size helps in relaxing the traffic load on the NoC bisection. To back up our analysis, we simulated a 10x10 mesh under different step sizes and other selection functions. Our results showed that increasing the step size can lead to an enhancement of 95% and 38% in both average packet latency and average deflection count respectively. Additionally, we simulated 2D meshes of different sizes to get estimation for the value of the step size given only the mesh dimension. We found that using 60-80% of the mesh dimension leads to better performance in terms of both packet latency and deflection count.

Chapter 4 : Variable Step Size Maximum Flexibility Selection Function

In Chapter 3, we showed that the value of the step size greatly affect the overall performance of the bufferless 2D NoC. As a result, we proposed MMaxFlex selection function. MMaxFlex uses step size values greater than one for all the packets in order to push the traffic to the NoC borders as a way to increase the links utilization. Also, we proposed estimation for the appropriate step size. However, the selection of the step size is done at the compilation time. In other words, the value is selected based on the user input, and used for all the packets.

In this chapter, we investigate the effect of using a variable step size under MaxFlex selection function. First, we explain the idea behind using variable step size values and why it is appealing. Then, we propose different approaches on how to calculate the value of the variable step size. Finally, we provide the simulation results and explain how the results are related to the fixed step size results.

The chapter is organized as follows; Section 4.1 provides the motivation behind the variable step size idea. In Section 4.2, we explain the proposed approaches and their operation. We present the simulation environment and results in Section 4.3. Finally, Section 4.4 concludes the chapter.

4.1. Motivation

The use of fixed step size MMaxFlex with step size greater than one was shown to be effective in redistributing the traffic away from the central part of NoC switches and move more towards the border switches. This redistribution had a direct effect on decreasing the flits deflection count and thus decreasing the overall average packet latency.

Generally speaking, the idea is to utilize the NoC switches and links more in a way that enhances the traffic distribution even better. As a way to change the traffic distribution, we assign a different step size for each packet instead of assigning the same step size value to all the packets. How to calculate the value of a different step size for each packet differs based on the criteria used. We explain the different approaches in the next section.

4.2. Proposed Variable Step Size Approaches

In this section, we list and explain the different approaches used to calculate the variable step size value for $n \times n$ bufferless mesh. The approaches basically falls under two categories; the first one deals with the NoC nodes as a standalone modules, while the second category divides the NoC into a number of rectangular regions and assign each node to a specific region. In other words, we distribute the nodes of the NoC to a group of non-interleaving rectangular regions such that each region contains a group of nodes (at least one node and up to $n \times n$ nodes). Also, we assign indices to each region

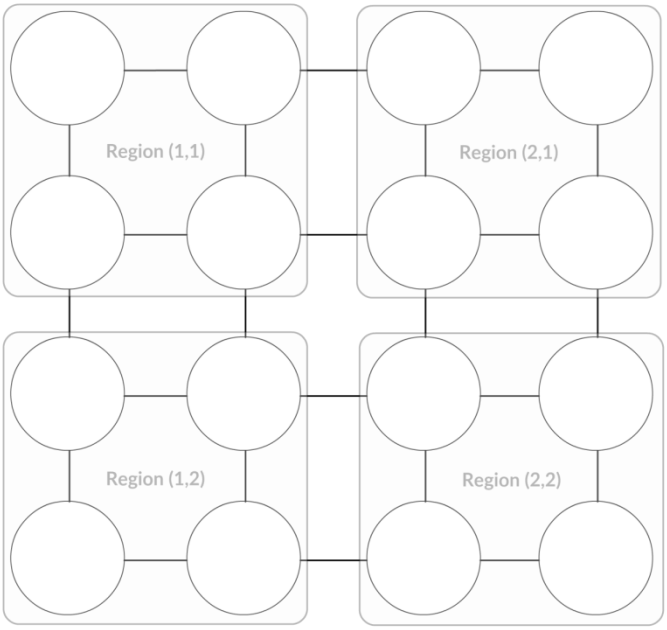

Figure 32: 4x4 mesh divided into four 2x2 regions

in a similar manner to the 2D NoC switches. In Figure 32, we show an example of 4x4 mesh divided into four regions along with the assigned indices.

In the following sub-sections, we explain and evaluate five approaches to calculate the variable step size. The first approach falls under the first category where we deal with the standalone nodes, while the rest of the approaches belong to the second category dividing the NoC into regions. The first approach calculates the step size based on the distance between the source and destination nodes of the flit. The second approach calculates the step size based on the distance between the source and destination regions of the flit. The third approach uses MMaxFlex with independent variable step sizes for routing inside the region and for routing between regions (i.e. in/out region routing). Finally, the fourth approach incorporates the in/out region routing with the distance between the source and destination nodes to calculate the step size.

4.2.1. Using the Manhattan distance between NoC nodes (NMDVS)

This approach aims to assign small step size to near nodes and large step size to nodes far from each other. By this approach, we use the information gained from the fixed step size analysis to better distribute the traffic by using smaller step size to the traffic between nearby nodes.

In NMDVS, we use the Manhattan distance between the source and destination nodes. Specifically, we calculate the variable step size as a percentage of the calculated Manhattan distance.

$$SS = minimum(Percentage \times d, n)$$

Where d is the Manhattan distance between source and destination nodes; and $Percentage$ is a customizable variable, $1 \leq Percentage \leq 100$.

Using large step size value for the traffic between nearby nodes is not effective. This can be explained by the following; in case of nearby nodes, the distance between the source and destination nodes is small, so the difference between the source and destination *X*-dimension or *Y*-dimension is also small (maximum value is equal to the distance between source and destination incase same row or column). Thus, using large step size leads to moving similar to using Straight Line selection function which leads to losing the freedom granted by MaxFlex. Given this insight and the analysis given in Chapter 3, we use smaller step size for the near nodes and larger step for the far nodes leading to the diversity we want in the traffic distribution.

4.2.2. Using the Manhattan distance between NoC regions (RMDVS)

As in NMDVS, this approach aims to assign small step size to near nodes and large step size to nodes far from each other. In RMDVS, we apply the regions concept. We divide the NoC into group of regions, and then assign each node to one of the regions.

To calculate the step size, RMDVS approach uses the Manhattan distance between the source and destination regions. Specifically, it calculates the step size based on the difference between regions indices i.e. X_{Region} and Y_{Region}. If the nodes are in the same region then the difference is zero and the step size is one. Otherwise, if the nodes are in different regions, then the step size is calculated based on how near or far are the regions.

$$SS = \Delta X_{Region} + \Delta Y_{Region} + 1$$
$$\Delta X_{Region} = \left| X_{Src\ Region} - X_{Dst\ Region} \right|$$
$$\Delta Y_{Region} = \left| Y_{Src\ Region} - Y_{Dst\ Region} \right|$$

Where $X_{Src\ Region}$ is the X index of the source node region; $Y_{Src\ Region}$ is the Y index of the source node region; $X_{Dst\ Region}$ is the X index of the destination node region; and $Y_{Dst\ Region}$ is the Y index of the destination node region.

Near regions most probably leads to smaller difference in the X_{Region} and Y_{Region} indices which leads to smaller step size. On the contrary, far regions lead to larger difference and hence larger step size. Also, this approach matches the analysis presented in Chapter 3.

4.2.3. Using In-Region and Out-Region routing (IORVS)

In this approach, we use the regions concepts in a different way. Similar to the RMDVS, we divide the NoC into regions and assign nodes to each region. However, in IORVS, we differentiate between the traffic between nodes belonging to the same region (in-region routing), and the traffic between nodes from different regions (out-region routing). In case of in-region routing, we consider each region to be a separate smaller NoC that can route the traffic between its own nodes using a step size that fits its characteristics. While in out-region routing, we look at the region as a whole unit and route the data between the regions using a step size that is tailored to the inter-region traffic.

$$SS_{In\ Region} = Fixed$$
$$SS_{Out\ Region} = Fixed$$

Based on the value of both in-region and out-region step sizes, the performance of the MaxFlex varies. Thus, using the freedom granted by IORVS, we study the different behavior between the near nodes traffic and the far nodes traffic under different in-region and out-region step sizes. Also, we study the effect of the region size on the overall performance.

4.2.4. Using the Manhattan distance between NoC nodes for Out-Region routing (ORMDVS)

In this approach, we mix between using the regions concepts as in IORVS with using the Manhattan distance between NoC nodes approach as in NMDVS. Specifically, we use a fixed step size customized for the in-region routing, and use the Manhattan distance between NoC nodes for calculating the out-region step size.

$$SS_{In\ Region} = Fixed$$
$$SS_{Out\ Region} = Percentage \times d_{Region} \times Size_{Region}$$

Where d_{Region} is the Manhattan distance between source and destination regions; $Percentage$ is a customizable variable, $1 \leq Percentage \leq 100$; and $Size_{Region}$ is the number of row (or column) nodes in a region.

In other words, ORMDVS uses the idea of assigning the step size as a percentage of the distance between the source and destination nodes mentioned in NMDVS, but in order to calculate such distance, it uses the Manhattan distance between the regions and the region's size instead of using the Manhattan distance between source and destination nodes. It aims to get the advantage of NMDVS and the flexibility of IORVS.

4.3. Simulation Results

In this section, we adapt the same experimental setup used in Chapter 3 to examine the use of the variable step size proposed approaches. First, we evaluate the NMDVS approach separately to get an estimate for the value of the percentage to use. Then, we evaluate the RMDVS approach and compare it with another formula that performs the opposite functions of RMDVS. IORVS is evaluated to study the effect of the region size, in addition to differentiate between the traffic between near nodes versus the traffic between far nodes. Finally, we present the ORMDVS approach performance results.

To evaluate NMDVS, we assigned a different step size for each packet based on the Manhattan distance between the packet's source and destination. For packet P, let the Manhattan distance between the source and destination is distance *d*, the value of the step size for P is a percentage of *d*. We examined different percentage value ranging from 10% to 90%.

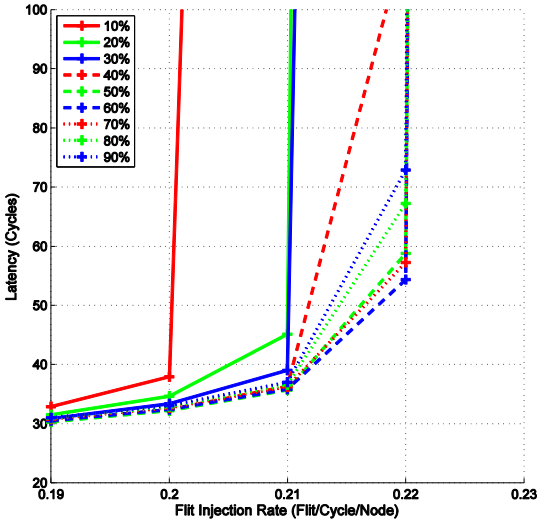

Figure 33: Average packet latency for NMDVS using different % values

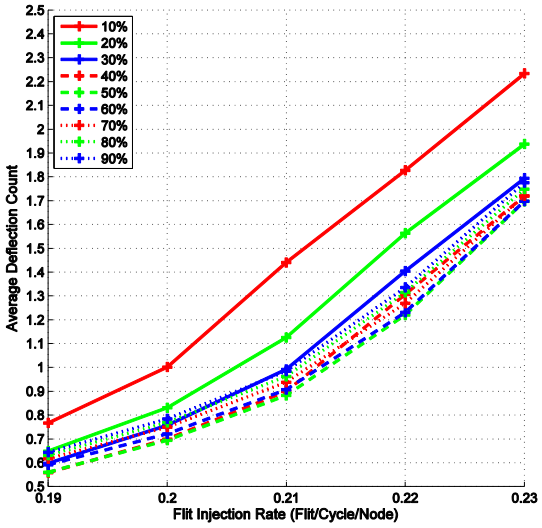

Figure 34: Average deflection count for NMDVS using different % values

Figure 33 and Figure 34 shows that as the percentage value increases, the average packet latency decreases. The best percentage value is about 60% of the distance. Also, Figure 33 and Figure 34 show that using higher percentage values degrades the performance as it leads to step sizes that can be similar to using a large fixed step size. These results matches the results for the fixed step size, as using the percentage value of 60% leads to larger step size value for the packets with long distance to go and smaller step size for the packets with short distance to go.

For RMDVS evaluation, we started by presenting another formula, RMDVS` that performs the exact opposite of RMDVS. In other words, RMDVS assigns small step size for the near nodes communication and large step size for the far nodes communication; however, in RMDVS`, by subtracting the differences between the X and Y dimensions of the NoC regions, we tend to generate small step size for the far nodes traffic and large step size for the near nodes traffic. Specifically, RMDVS` uses the following formula to calculate the step size.

$$SS = \left|\Delta X_{Region} - \Delta Y_{Region}\right| + 1$$
$$\Delta X_{Region} = \left|X_{Src\ Region} - X_{Dst\ Region}\right|$$
$$\Delta Y_{Region} = \left|Y_{Src\ Region} - Y_{Dst\ Region}\right|$$

Where $X_{Src\ Region}$ is the X index of the source node region; $Y_{Src\ Region}$ is the Y index of the source node region; $X_{Dst\ Region}$ is the X index of the destination node region; and $Y_{Dst\ Region}$ is the Y index of the destination node region.

Also, to study the effect of changing the region size under the RMDVS approach, we simulated both RMDVS and RMDVS` using 2x2 region size and 5x5 region size under 10x10 mesh. We expect RMDVS` to not perform well as it does not conform to the aforementioned fixed step size analysis in Chapter 3.

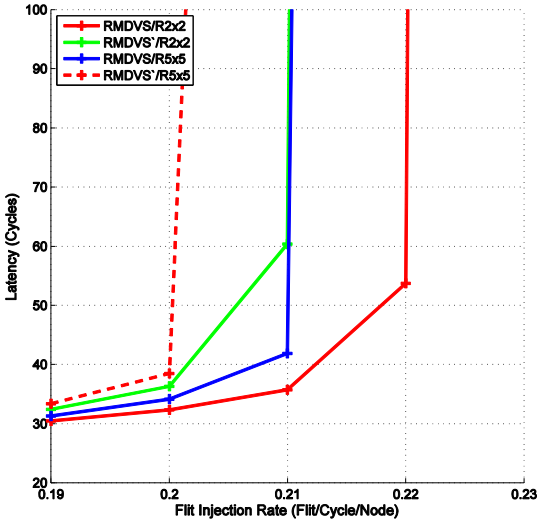

Figure 35: Average packet latency for RMDVS compared with RMDVS`

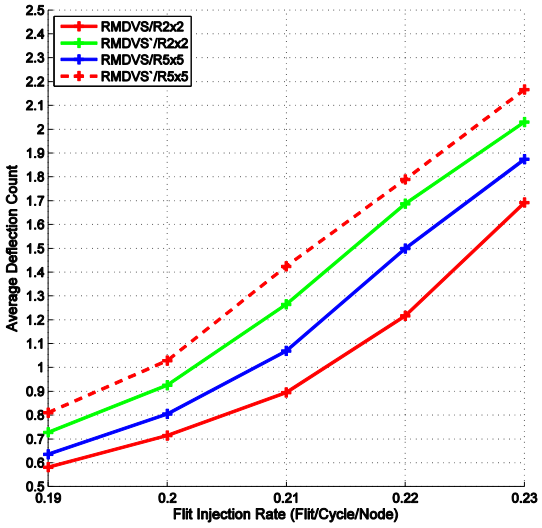

Figure 36: Average deflection count for RMDVS compared with RMDVS`

As shown in Figure 35 and Figure 36, RMDVS performance exceeds the performance of its opposite formula, RMDVS`, in terms of both average packet latency and average deflection count respectively. The superior performance is accounted for how RMDVS step size calculation conforms to the analysis presented in Chapter 3. RMDVS calculates a large step size for the far node communication, while RMDVS` calculates a small step size. As a result, given the analysis in Chapter 3, assigning a large step size decreases the concentration on the NoC central switches and moves part of the traffic to the borders. Also, RMDVS calculates a small step size in case of near nodes communication which produces diversity in distributing the NoC traffic leading to better link utilization, thus better packet latency and deflection count.

Concerning the region size, as shown in both figures, using 2x2 regions resulted in better performance than using 5x5 regions. This is because using 2x2 region size resulted in 25 regions, while using 5x5 region size resulted in 4 regions only. Increasing the number of regions resulted in more fine control in the step size calculation, thus better distribution for the values of the calculated step size.

In IORVS, we divide the NoC into regions, and differentiate between nodes communication in the same region and nodes communication between regions in order to study the difference between the near nodes traffic and the far nodes traffic, and to study the effect of the region size on the overall performance. To evaluate IORVS, we simulated 10x10 mesh using 2x2 regions and 5x5 regions. Also, as the performance is affected by the in-region step size and out-region step size, we simulated all the possible combinations for the in-region and out-region step sizes, In other words, for every in-region step size value ranging from one to nine, we used out-region step size value ranging from one to nine. Thus, for each region size, we simulated 81 experiments to cover all the cases (i.e. 162 for both 2x2 and 5x5 regions).

From Figure 37 to Figure 72, we show the average packet latency and average deflection count for each of the 162 experiments. From these figures, concerning far nodes traffic, we noted that under any in-region step size value, using a large step size for out-region communication leads to better performance under both region sizes. Specifically, step size of seven or eight leads to the best performance under the used in-region step size. This conforms to the analysis and step size estimation done in Chapter

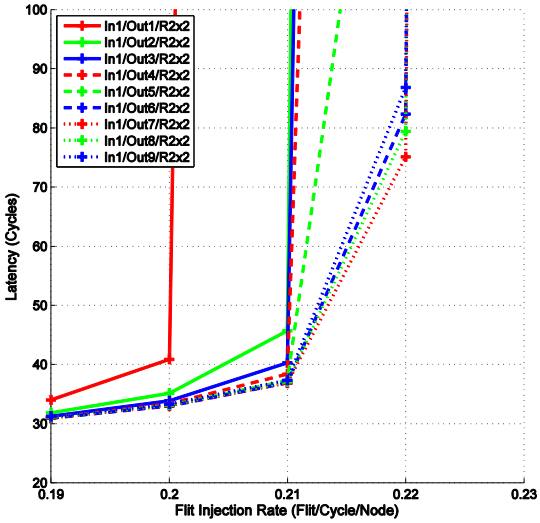

Figure 37: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 1$ using 2x2 region size

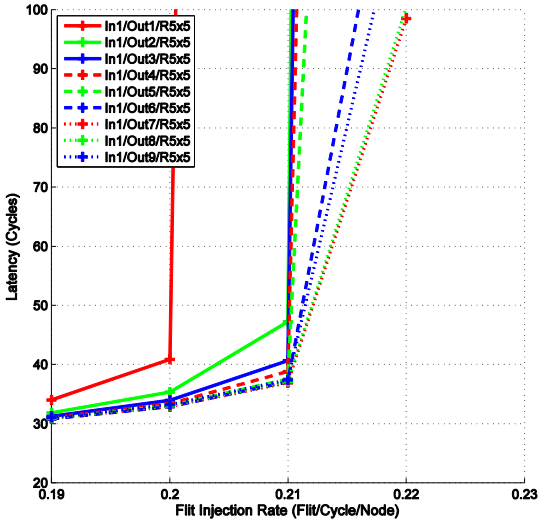

Figure 38: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 1$ using 5x5 region size

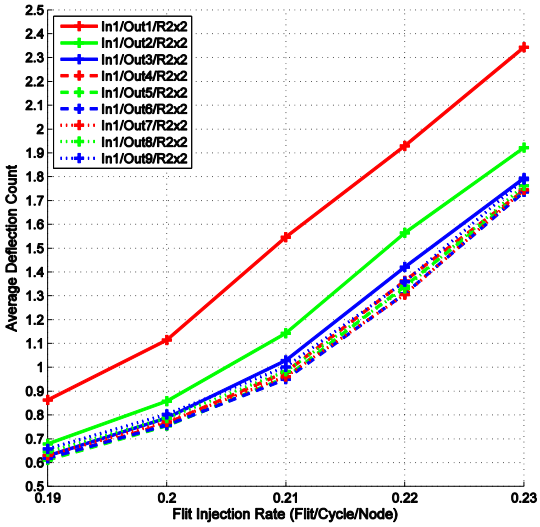

Figure 39: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 1$ using 2x2 region size

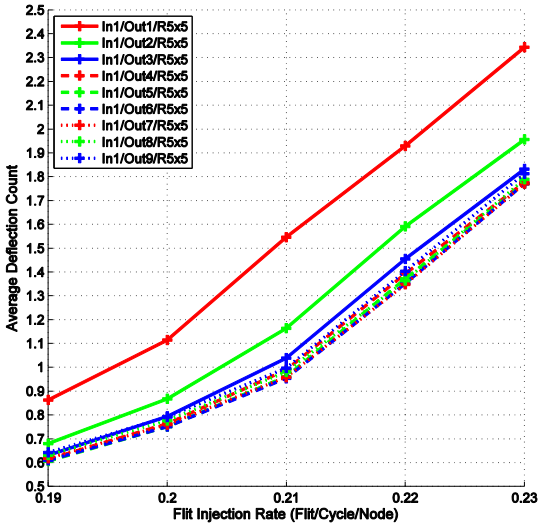

Figure 40: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 1$ using 5x5 region size

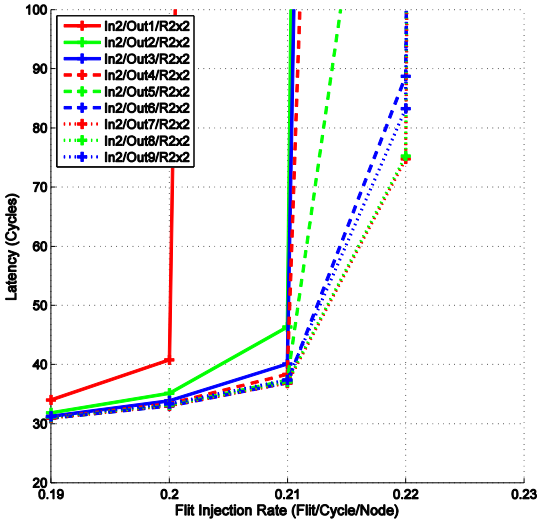

Figure 41: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 2$ using 2x2 region size

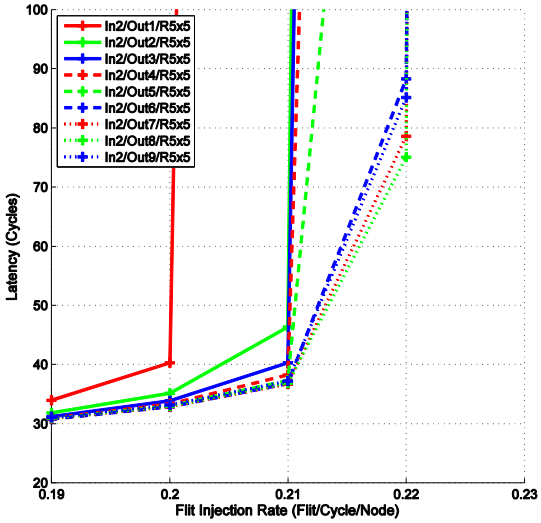

Figure 42: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 2$ using 5x5 region size

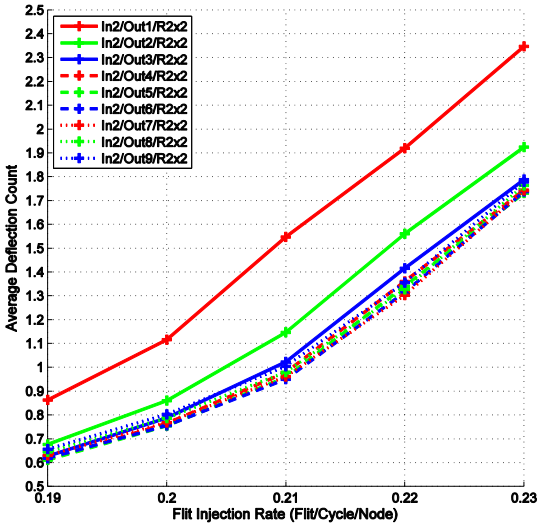

Figure 43: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 2$ using 2x2 region size

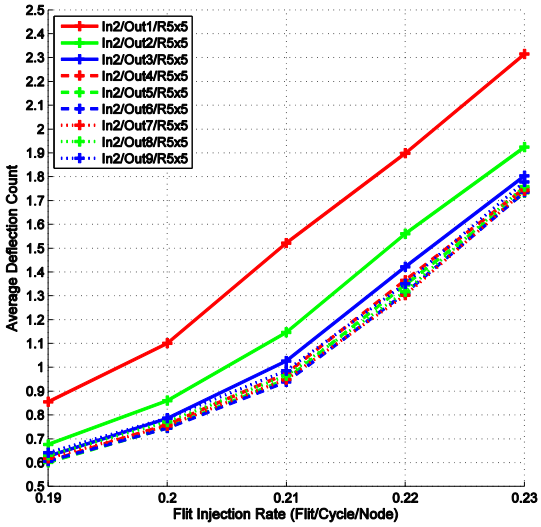

Figure 44: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 2$ using 5x5 region size

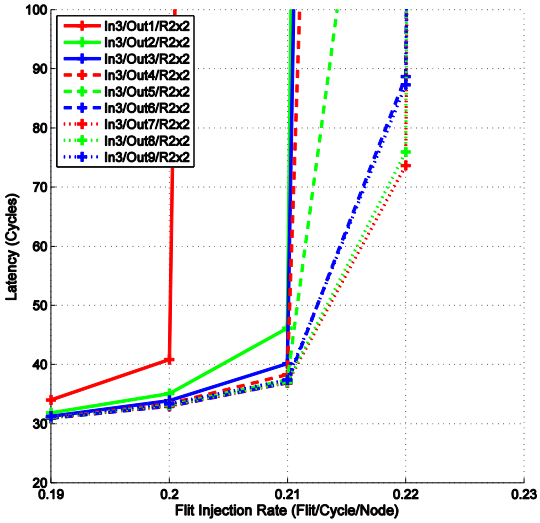

Figure 45: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 3$ using 2x2 region size

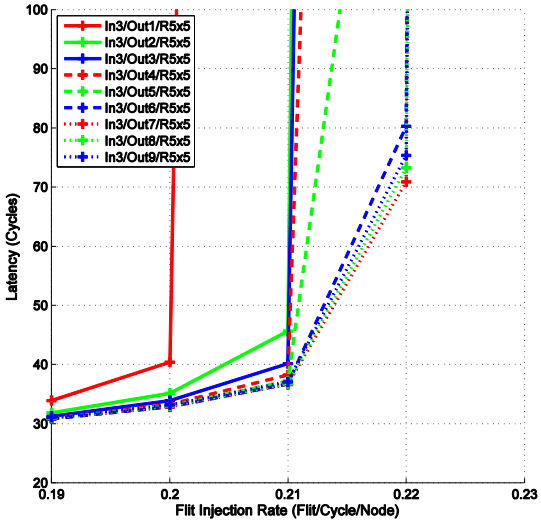

Figure 46: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 3$ using 5x5 region size

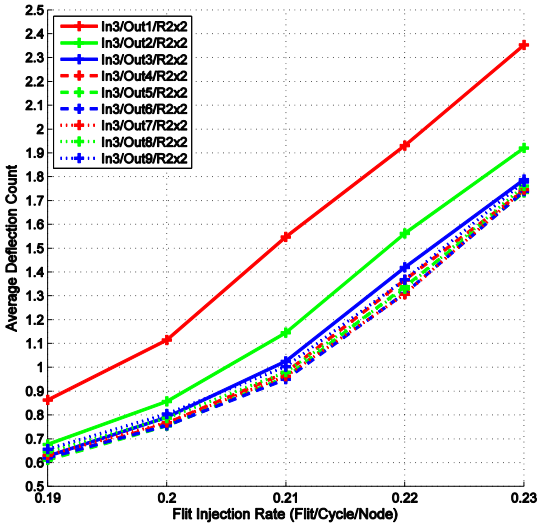

Figure 47: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 3$ using 2x2 region size

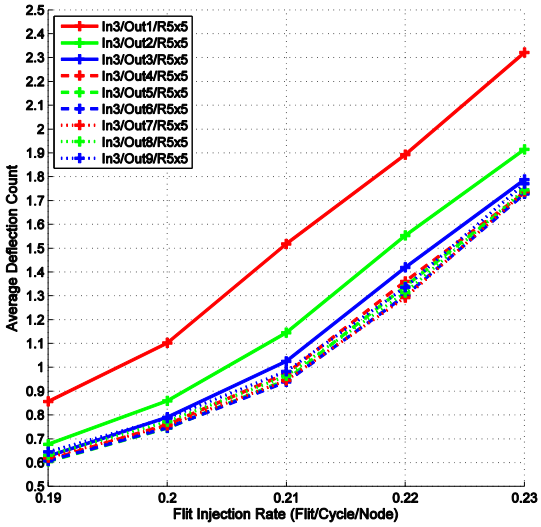

Figure 48: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 3$ using 5x5 region size

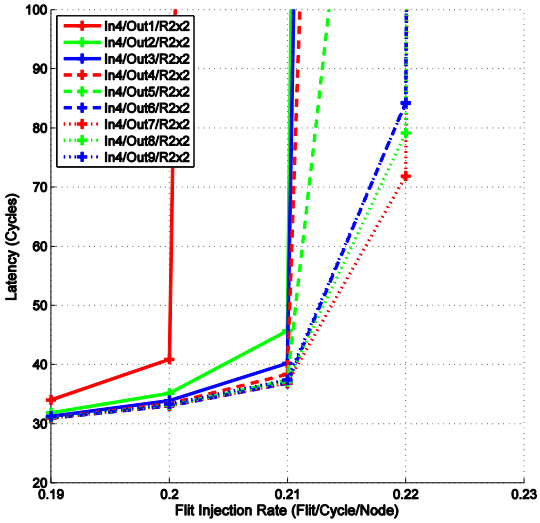

Figure 49: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 4$ using 2x2 region size

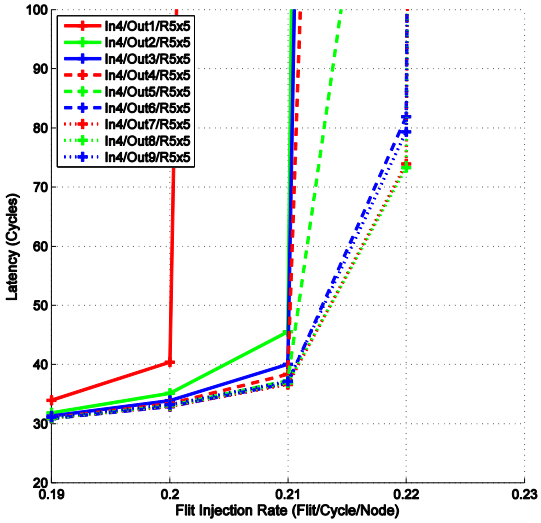

Figure 50: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 4$ using 5x5 region size

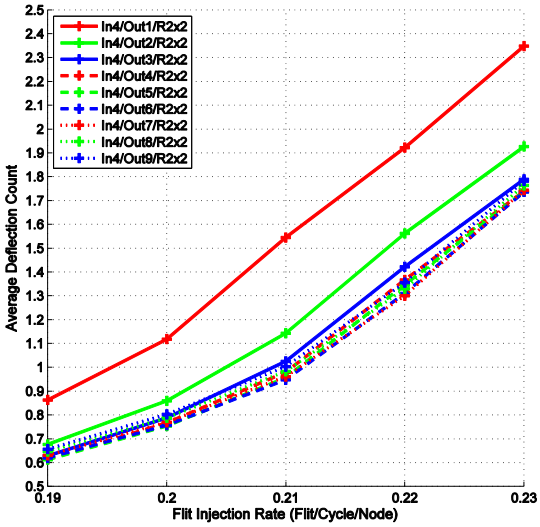

Figure 51: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 4$ using 2x2 region size

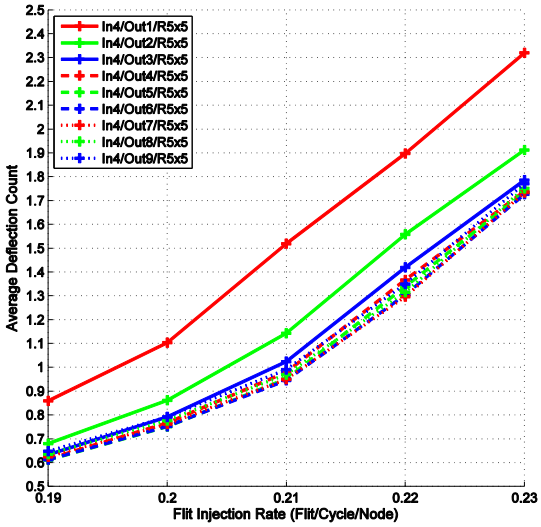

Figure 52: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 4$ using 5x5 region size

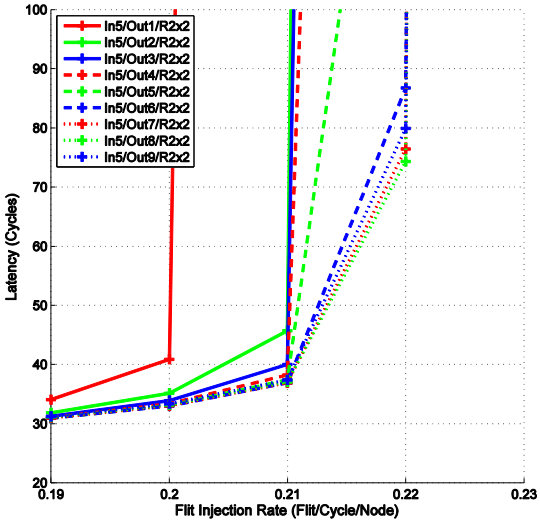

Figure 53: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 5$ using 2x2 region size

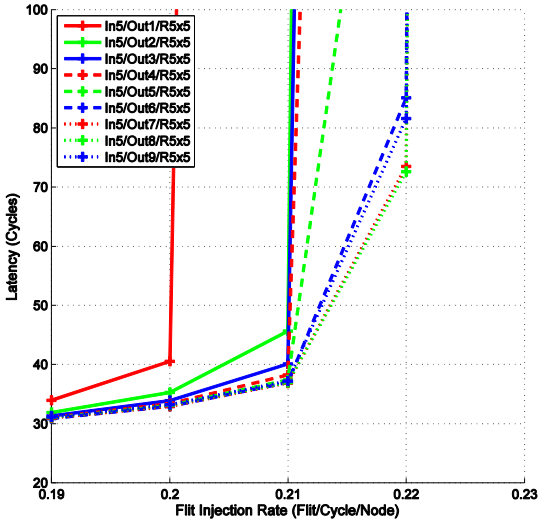

Figure 54: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 5$ using 5x5 region size

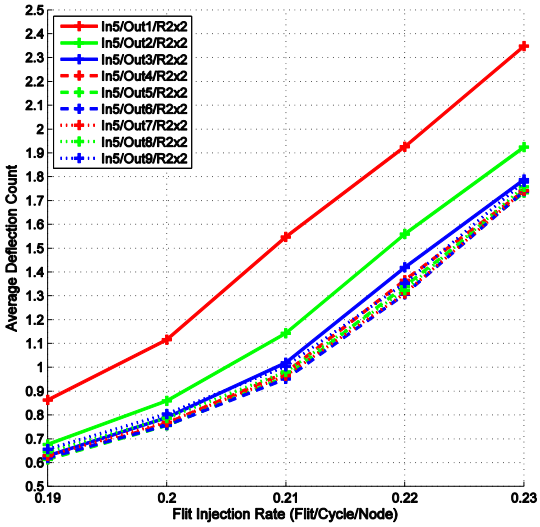

Figure 55: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 5$ using 2x2 region size

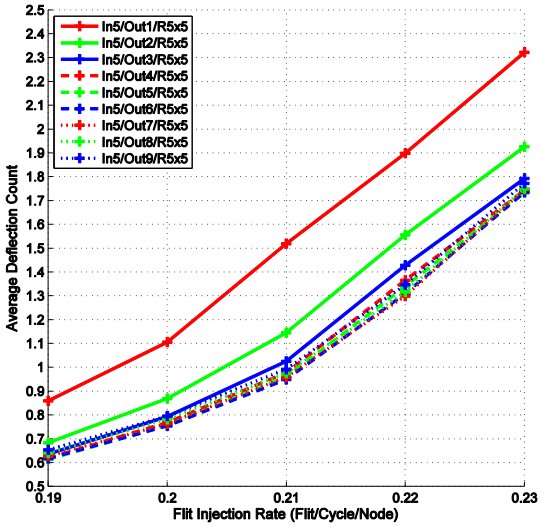

Figure 56: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 5$ using 5x5 region size

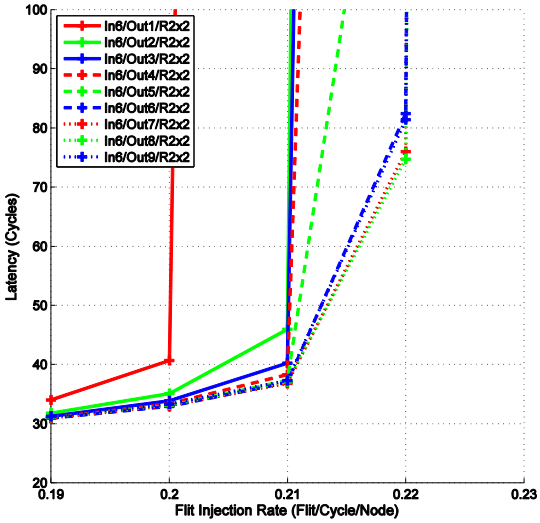

Figure 57: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 6$ using 2x2 region size

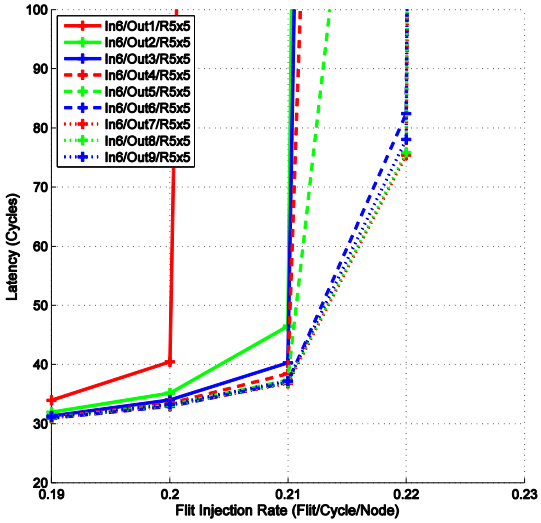

Figure 58: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 6$ using 5x5 region size

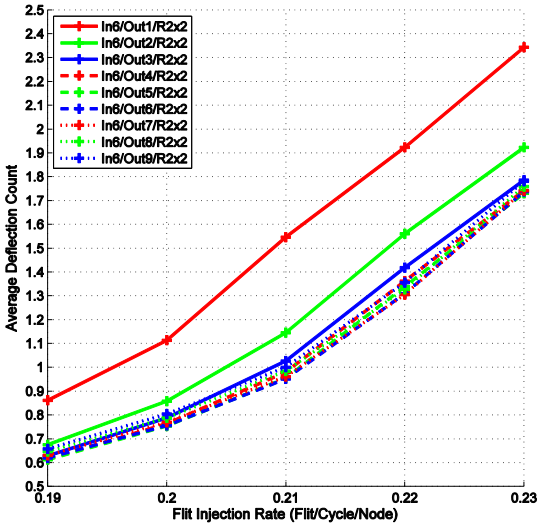

Figure 59: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 6$ using 2x2 region size

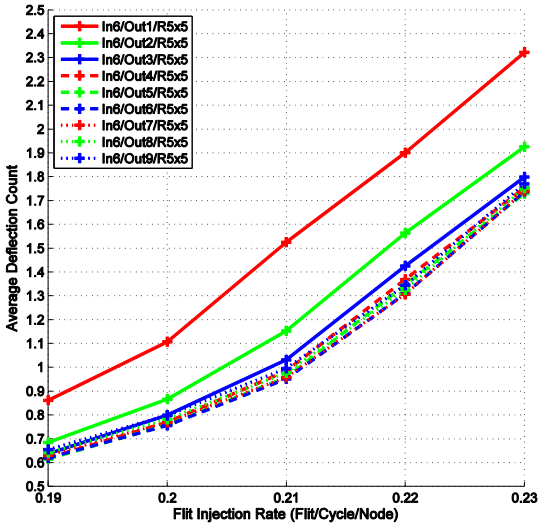

Figure 60: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 6$ using 5x5 region size

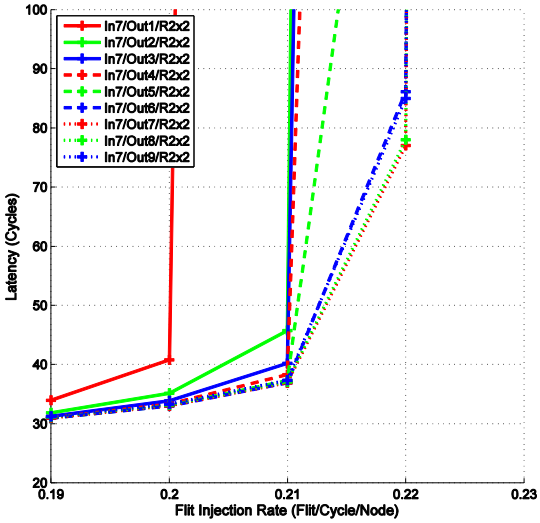

Figure 61: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 7$ using 2x2 region size

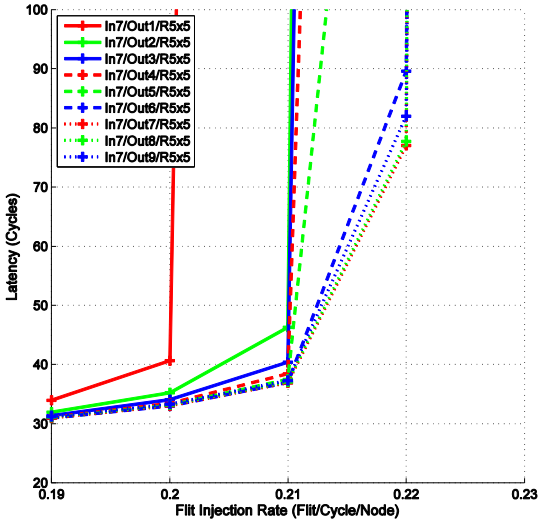

Figure 62: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 7$ using 5x5 region size

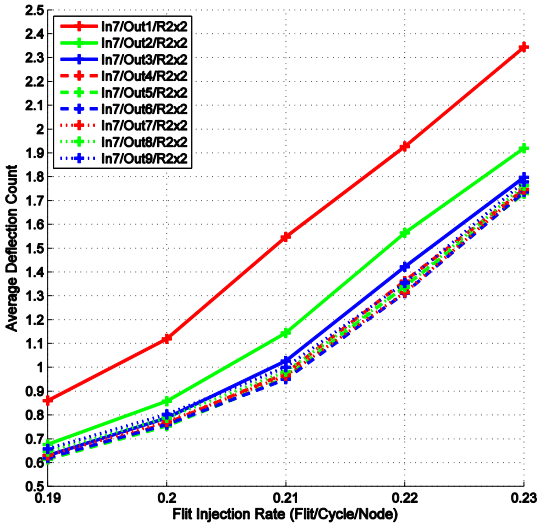

Figure 63: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 7$ using 2x2 region size

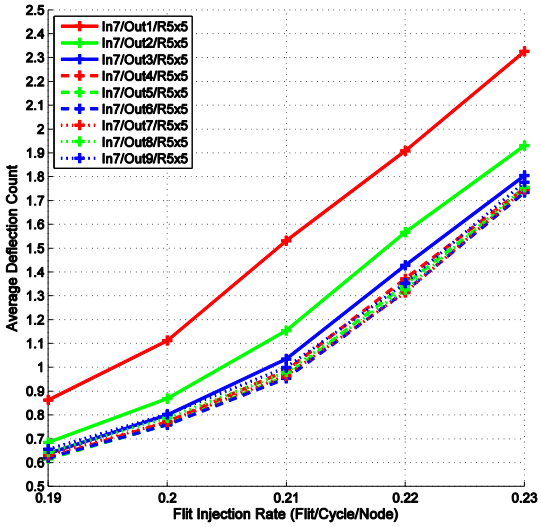

Figure 64: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 7$ using 5x5 region size

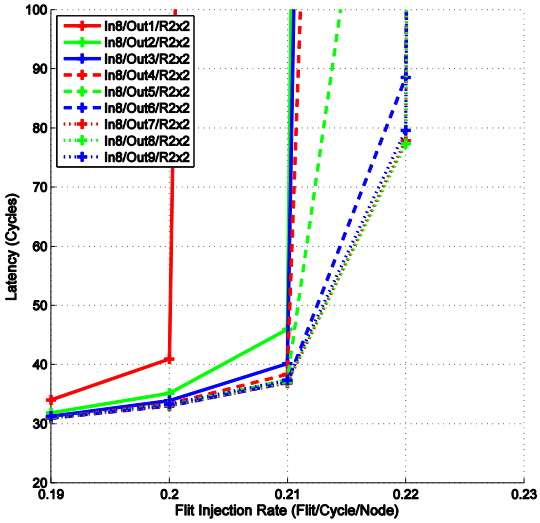

Figure 65: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 8$ using 2x2 region size

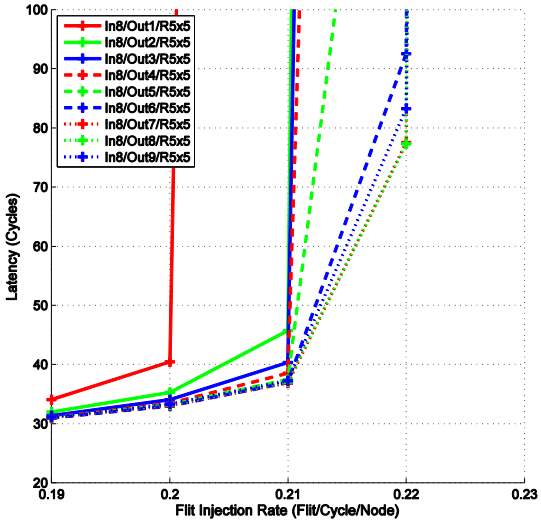

Figure 66: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 8$ using 5x5 region size

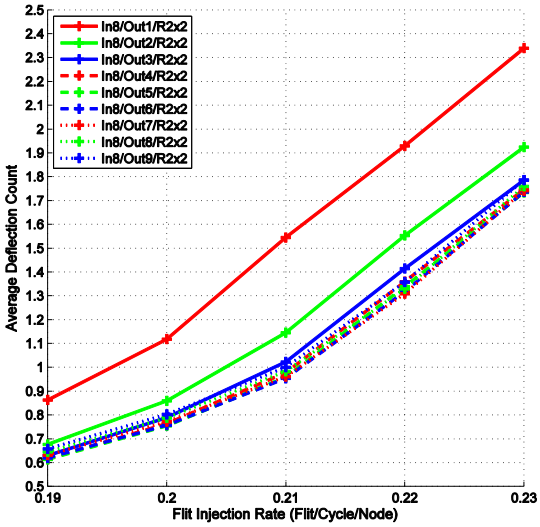

Figure 67: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 8$ using 2x2 region size

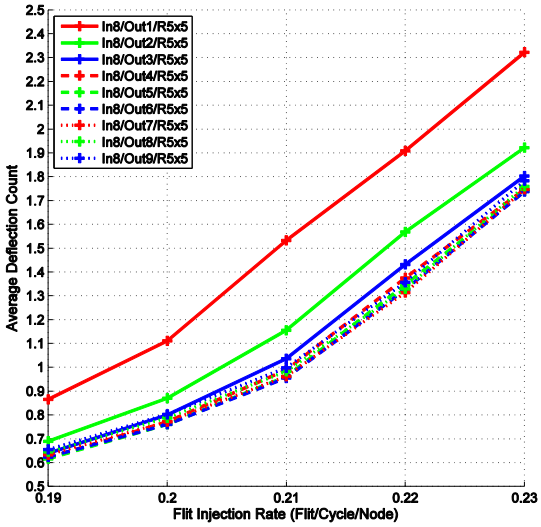

Figure 68: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 8$ using 5x5 region size

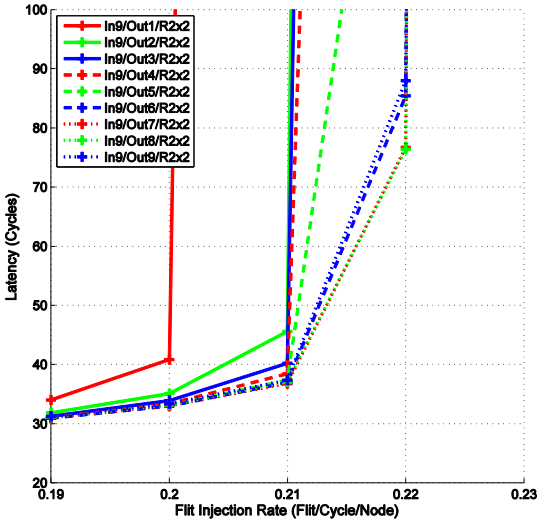

Figure 69: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 9$ using 2x2 region size

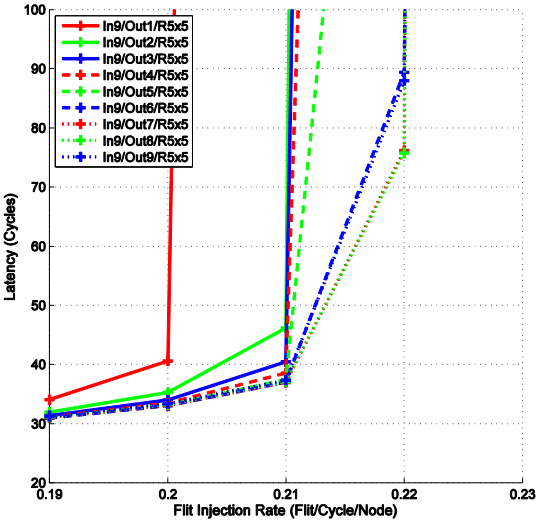

Figure 70: Average packet latency for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 9$ using 5x5 region size

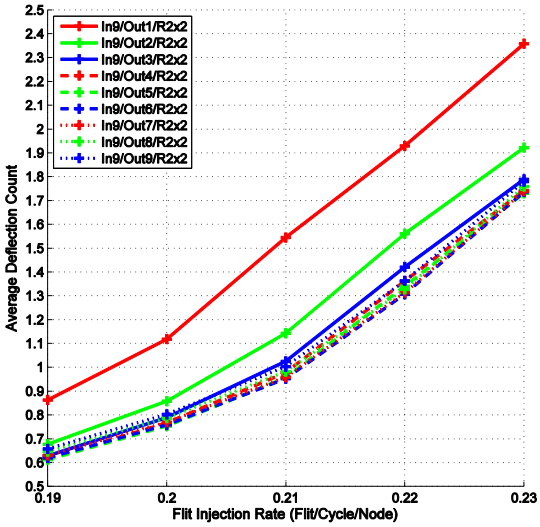

Figure 71: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 9$ using 2x2 region size

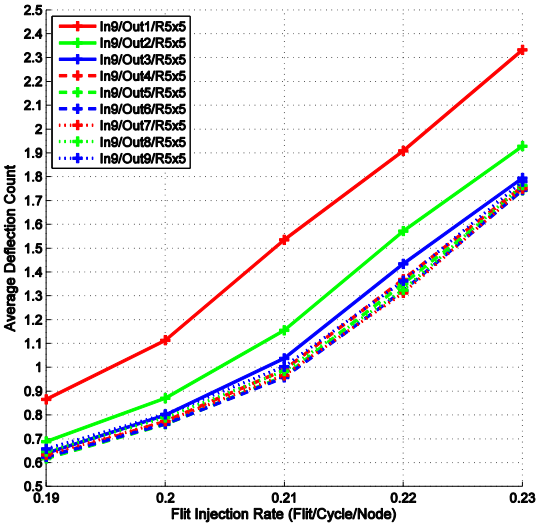

Figure 72: Average deflection count for different $SS_{OutRegion}$ values under in $SS_{InRegion} = 9$ using 5x5 region size

3. It was estimated that using 60% to 80% of the NoC dimension *n* as a step size performs the best under MMaxFlex ($7/10 \equiv 70\%$ and $8/10 \equiv 80\%$).

As for the near region traffic, from the figures, the performance varies based on the used in-region step size value, and the used region size value. For example, the best performance under region size 5x5 is achieved using in-region step size of three. This in-region step size for the 5x5 regions also conforms to the estimation done in Chapter 3 ($3/5 \equiv 60\%$). As for 2x2 regions, the best value is achieved using in-region step size of four. However, the performance of all the in-region step size values is almost similar as the used region size is small (2x2 regions). As a result, using any in-region step size value, ranging from one to nine, leads to a behavior similar to DO routing inside 2x2 region. For the same reasons, using 5x5 regions, any value for in-region step size larger than three leads to similar performance.

As for the effect of the region size, in the figures (from Figure 37 to Figure 72), the size of the region doesn't have a clear cut effect on IORVS approach. This is due to the fact that the calculation of the in-region or out-region step size is not function in the region size or the number of regions as was in RMDVS. In other words, for any region size used and following the work done in Chapter 3, we can estimate a value for the in-region step size, and use large step size for out-region step size to achieve the best possible performance under the used region size.

Finally, in ORMDVS approach, we combine the calculation of the variable step size in NMDVS approach, and the flexibility of IORVS approach. Specifically, we use divide the NoC into regions, use a step size customized for the in-region routing behavior, and calculate the out-region step size using a formula similar to what was used in NMDVS.

We simulated 10x10 mesh using 2x2 regions and 5x5 regions. For the percentage value, we used 60% as it achieved the best performance under NMDVS. For the in-region step size, we used different values ranging from one to nine to evaluate the effect of changing the in-region step size. The results for 2x2 regions are shown in Figure 73 and Figure 74, while the results for 5x5 regions are in Figure 75 and Figure 76.

For 2x2 regions, in Figure 73 and Figure 74, the performance under any in-region step size is similar with a slight advantage for in-region step size of one. This is due to using small region size (2x2 regions). As a result, using any in-region step size value, ranging from one to nine, leads to a behavior similar to DO routing inside 2x2 region. On the other hand, in Figure 75 and Figure 76, using 5x5 regions leads to worse performance than using 2x2 regions. The best performance for 5x5 regions is achieved using in-region step size of four due to the step size estimation presented in Chapter 3.

Under ORMDVS, using 2x2 regions is better than using 5x5 regions as 2x2 regions generates more regions than using 5x5 regions (25 regions versus 4 regions). More regions means for flexibility in calculating the out-region step size. For example, using 5x5 regions (4 regions), the distance between regions can be one or two only. Thus, the distance estimated between the communicating nodes, based on the used formula, has a two values only (five or ten) leading to out-region step size values of three and six only. On the other hand, using more regions under 2x2 region size, gives more values for the distance between the regions, thus leading to more variability in the calculated out-region step size.

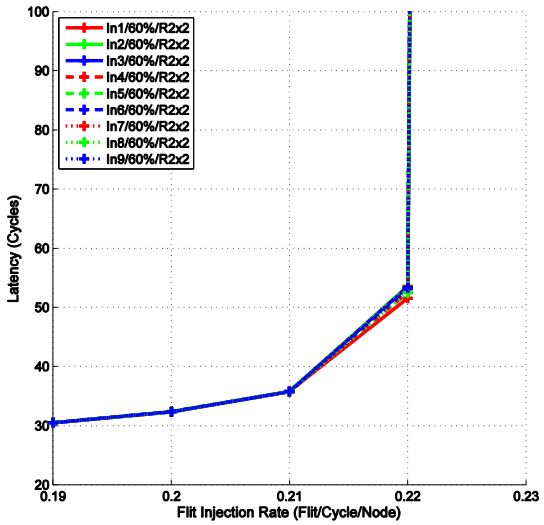

Figure 73: Average packet latency using different $SS_{InRegion}$ values and 60% under 2x2 region size

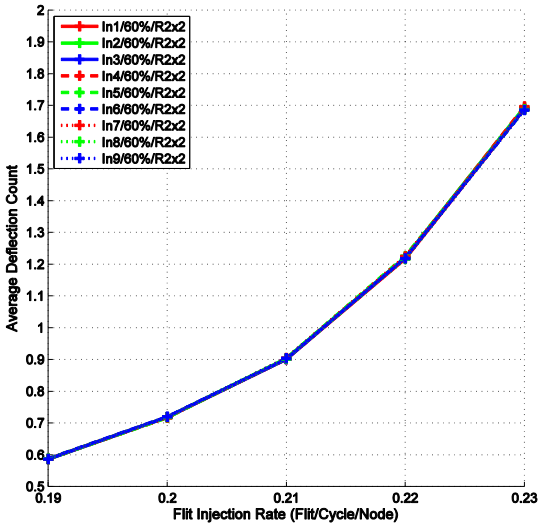

Figure 74: Average deflection count using different $SS_{InRegion}$ values and 60% under 2x2 region size

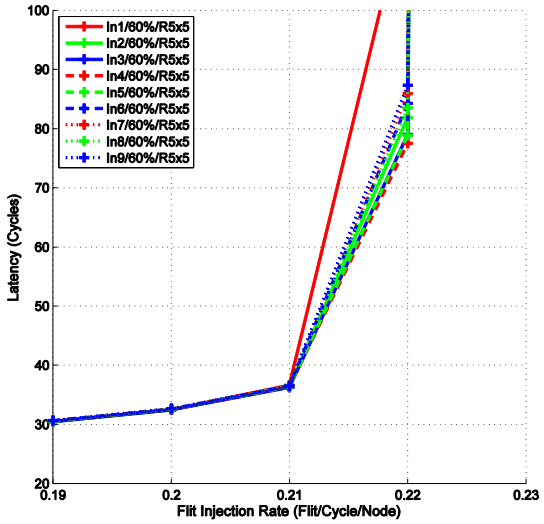

Figure 75: Average packet latency using different $SS_{InRegion}$ values and 60% under 5x5 region size

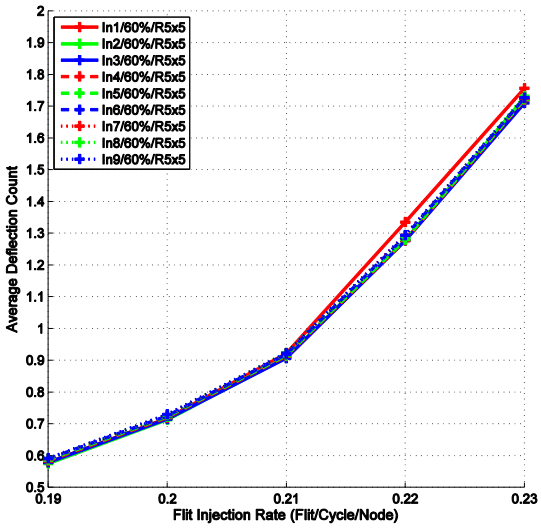

Figure 76: Average deflection count using different $SS_{InRegion}$ values and 60% under 5x5 region size

Till now, we presented each approach results separately. To evaluate the different approaches, we selected the best result achieved under each approach, and compared these results with using fixed step size of eight under MMaxFlex. For NMDVS, we used 60% as the percentage. For RMDVS, we used 2x2 region size. As for IORVS, we selected in-region step size of four and out-region step size of seven under 2x2 region size, and in-region step size of three and out-region step size of seven under 5x5 region size. Finally for ORMDVS, we used 60% as the percentage and in-region step size of one under 2x2 region size.

As shown in Figure 77 and Figure 78, all the proposed approaches enhances the performance over using a fixed step size of eight under MMaxFlex in terms of both

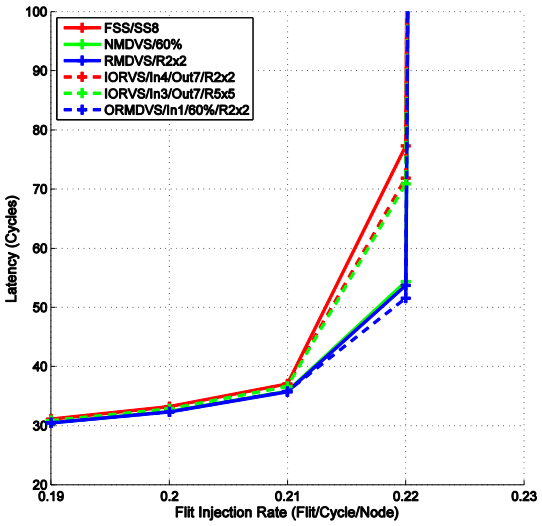

Figure 77: Average packet latency for different variable step size formulas

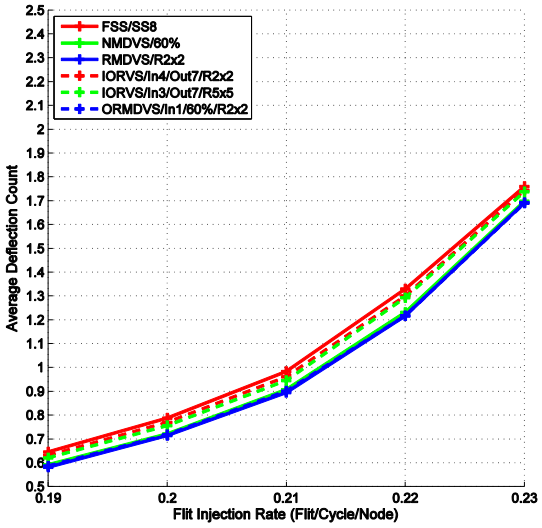

Figure 78: Average deflection count for different variable step size formulas

average packet latency and average deflection count. This is due to using different step size values for the packets instead of fixing the value for all the packets. This variability leads to better traffic distribution thus better utilization for the bufferless NoC links.

From the figures, we note that IORVS approach achieves the least enhancement over the fixed step size. Specifically, using 2x2 regions, the enhancement is 7.03% and 2.23% in terms of average packet latency and average deflection count respectively. While using 5x5 regions enhances by 8.3% and 2.79% in terms of average packet latency and average deflection count respectively. This small enhancement is due minimum variability used in IORVS. IORVS can be seen as an update for using fixed step size; however, instead of fixing the step size for all the packets, we use two separate fixed values for in-region and out-region routing.

Also, observing Figure 77 and Figure 78, the performance of NMDVS, RMDVS, and ORMDVS is almost similar with the best performance achieved by ORMDVS using 2x2 regions. ORMDVS enhances over fixed step size under MMaxFlex by 33.28% and 8.49% in terms of average packet latency and average deflection count respectively. The superiority of these approaches can be seen as a result of the higher variability achieved in calculating the step size. Additionally, ORMDVS superior enhancement is due to mixing NMDVS and IORVS. Using IORVS granted the flexibility in separating the in-region and out-region routing. While using NMDVS granted better distribution and variability for calculating the out-region step size.

4.4. Concluding Remarks

In this chapter, we presented the idea of varying the used step size under MMaxFlex selection function. We started by presenting different approaches for calculating the variable step size value. We presented approaches that used the distance between the source and destination nodes for calculating the step size. Other approaches divided the NoC into smaller regions and separated the in-region and out-region routing. To test the performance of the proposed formulas, we simulated a 10x10

mesh. Our results showed that using any of the proposed approaches achieves better results than using fixed step size under MMaxFlex. Specifically, one of the approaches lead to an enhancement of 33.28% and 8.49% in terms of average packet latency and average deflection count respectively compared with fixed step size of 8 under MMaxFlex.

Chapter 5 : New Flit Ranking Policies for Deflection-based Bufferless NoCs

In Chapter 3 and Chapter 4, we enhanced the bufferless NoC performance via selection functions. We investigated increasing and varying the used step size under MaxFlex as a way to enhance the links utilization which affects the performance.

In this chapter, we study the role of using different flit ranking policies on the Bufferless NoC performance. Ranking policies determine the order by which the flits are served. By changing the order of serving the packets/flits, the performance can change in a drastic way.

First, we explain the importance of ranking policies and why it worth studying. Then, we present different policies for ranking the flits. Finally, we experimentally evaluate the proposed policies.

The chapter is organized as follows; Section 5.1 provides the motivation behind studying ranking policies. In Section 5.2, we propose new ranking policies. Section 5.3 simulates and evaluates the proposed ranking policies. Finally, Section 5.4 concludes the chapter.

5.1. Motivation

During the NoC operation, a 2D mesh NoC switch can receive up to five flits; four from the ports connected to its neighboring switches, in addition to one flit injected from the node connected to it. Each of these flits needs an output port to reach its required destination. As a result, a conflict may arise due to different flits requiring the same output port. In order to solve the contention between the different flits, a flit ranking policy is used. A flit ranking policy applies a criterion to determine the order of serving the incoming flits. In other words, it determines which flit chooses an output port first.

Different ranking policies employ different criteria to order the flits. Subsequently, the order of serving the flits differs leading to different arrival patterns for the NoC flits. A good ranking policy results in a pattern that minimizes the average latency among all the NoC packets.

In buffered NoCs, if a flit fails to get its required output port, it enters the buffer waiting for its turn to pass. Thus, even in case of a weak ranking policy, the flit can still wait till its shortest path is free. However, in bufferless NoC, the ranking policies have greater effect due to the buffers elimination. If a flit fails to get its productive port, it is deflected through a non-productive port as the links are the only buffering resource. This unnecessary detours increase the overall packet latency.

In the next section, we propose new ranking polices and an enhancement tailored for bufferless NoCs and MaxFlex. We evaluate the proposed approaches with two well-known ranking policies discussed in the following sub-sections.

5.1.1. Oldest First Ranking Policy (OF)

The OF ranking policy chooses the age of the flit as its criteria. The age of the flit is the number of cycles passed since its generation. OF ensures that there is a total age

order among flits and prioritizes older flits. In other words, OF tends to direct the flit with higher age to its destination as to not increase the average latency.

At a certain cycle *t*, let *A* be a flit with age $Age_{(A,t)}$, and priority $Priority_{(A,t)}$. Also, let *B* be a flit with age $Age_{(B,t)}$, and priority $Priority_{(B,t)}$. If $Age_{(A,t)} > Age_{(B,t)}$ then $Priority_{(A,t)} > Priority_{(B,t)}$.

5.1.2. Most Deflection First Ranking Policy (MDF)

MDF ranking policy chooses the deflection count of the flit as the ranking criteria. The deflection count of the flit is number of times the flit takes a non-productive port as its output port. MDF prioritizes the flits with more deflections. In other words, MDF tends to direct the flit with higher deflection count to its destination as to not increase the average latency.

Let *A* be a flit with deflection count $Deflection_{(A,t)}$, and priority $Priority_{(A,t)}$. Also, let *B* be a flit with deflection count $Deflection_{(B,t)}$, and priority $Priority_{(B,t)}$. If $Deflection_{(A,t)} > Deflection_{(B,t)}$ then $Priority_{(A,t)} > Priority_{(B,t)}$.

5.2. Proposed Flit Ranking Policies

Based on the results from the fixed/variable step size study in Chapter 3, and from a recent bufferless NoC study that discusses the effect of deflections on the overall performance [22], we propose ranking policies that tend to decrease the deflection count of the NoC flits. The proposed policies favor the flit with more deflections as extra detouring for this flit leads to extra delay thus increasing the overall packet latency.

In the following sub-sections, we propose updating the Most Deflections First (MDF) policy to use the deflection count of the flit along with its age, and the distance between its source and destination. Also, we propose an enhancement that can work with the any of the policies. It should be noted that even though the proposed ranking policies in this chapter are intended for bufferless NoCs, these policies can also be applied to buffered NoCs.

5.2.1. Deflection Age Ratio Ranking Policy (DAR)

DAR ranking policy chooses the deflection/age ratio as its criteria. DAR prioritizes the flits with higher ratio. OF and MDF policies favor the oldest and most deflected respectively, however, the flit may be old or deflected many times because the distance between its source and destination is large. Thus, DAR takes into consideration both the time the flit has been in the NoC and its deflection count. DAR favors the flits that have suffered more deflections during its lifetime in the NoC.

Let *A* be a flit with age $Age_{(A,t)}$, deflection count $Deflection_{(A,t)}$, and priority $Priority_{(A,t)}$. Also, let *B* be a flit with age $Age_{(B,t)}$, and delfeciton count $Deflection_{(B,t)}$, and priority $Priority_{(B,t)}$. If $Deflection_{(A,t)}/Age_{(A,t)} \geq Deflection_{(B,t)}/Age_{(B,t)}$ then $Priority_{(A,t)} \geq Priority_{(B,t)}$.

5.2.2. Deflection Distance Ratio Ranking Policy (DDR)

DDR ranking policy chooses the deflection/distance ratio as its criteria. The distance is the Manhattan distance between the source and destination of the flit. DDR prioritizes the flits with higher ratio. Following the same idea as in DAR, DDR favors the flits that have suffered more deflections during the path from its source and destination.

Let A be a flit with distance between its source and destination $Distance_A$, deflection count $Deflection_{(A,t)}$, and priority $Priority_{(A,t)}$. Also, let B be a flit with distance between its source and destination $Distance_B$, delfeciton count $Deflection_{(B,t)}$, and priority $Priority_{(B,t)}$. If $Deflection_{(A,t)}/Distance_A \geq Deflection_{(B,t)}/Distance_B$ then $Priority_{(A,t)} \geq Priority_{(B,t)}$.

5.2.3. Last Dimension Ranking Policy (LD)

LD is an enhancement that can work with any of the ranking schemes. It is designed to work specifically with bufferless NoCs and MaxFlex selection function. In case of competing flits, LD favors the flit that has hops in only one direction. In case of a draw, LD uses other ranking policies to break the draw. For example, it two flits are competing and one of the flits has only moves left in the X direction, while the other still has moves in both X and Y directions, then LD favors the first flit.

The motivation behind favoring the flit stuck in one direction is that any deflection for this flit leads to extra unnecessary detour. This detour needs at least two cycles to correct the path of the flit. Thus, if we choose not to deflect this flit, we enhance the overall packet latency as we decrease the overall deflection count.

Here, we present the usage of LD along with MDF and DDR ranking policies as draw breakers.

5.3. Simulation Results

In this section, we adapt the same experimental setup used in Chapter 3 to evaluate the approaches mentioned in the previous section. First, we present the experimental results concerning the updated approaches DAR and DDR in contrast to the baseline approaches OF and MDF. Then, we evaluate the LD enhancement compared with MDF and DDR.

Figure 79 and Figure 80 compare between the presented ranking policies in terms of average packet latency and average deflection count respectively. As shown in both figures, all the deflection based policies have a superior performance over the OF ranking policy in addition to operating under higher injection rates. Also, in Figure 79, the proposed policies DAR and DDR exceed MDF performance in terms of packet latency. That is because MDF only focus on the deflections without considering the time spent in the NoC or the distance to be covered. DDR has the best performance in terms of both packet latency and deflection count as it considers the shortest distance between the source and destination of the flit. The shortest distance between the source and destination is known and can be calculated upfront. As a result, if a flit suffered high deflection count while travelling short distance, it is favored over the flit that was deflected the same number of times but while travelling long distance. Thus, factoring the distance differentiates between the two flits even though they have the same

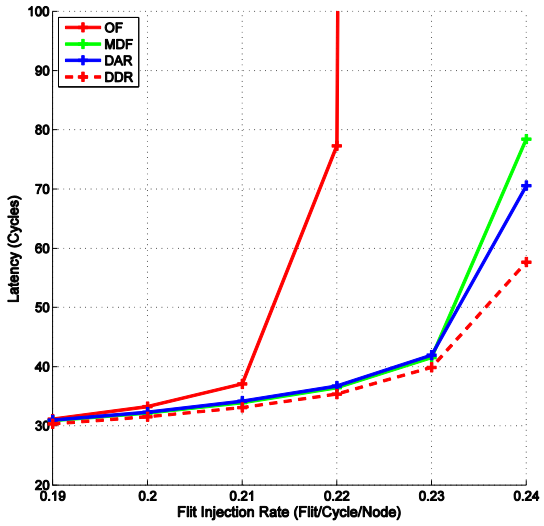

Figure 79: Average packet latency for different ranking policies

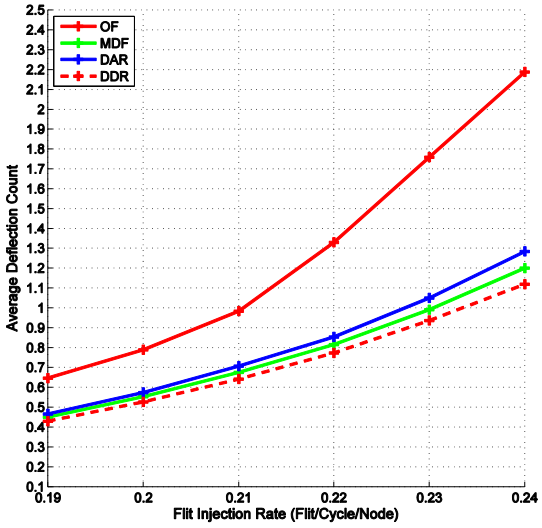

Figure 80: Average deflection count for different ranking policies

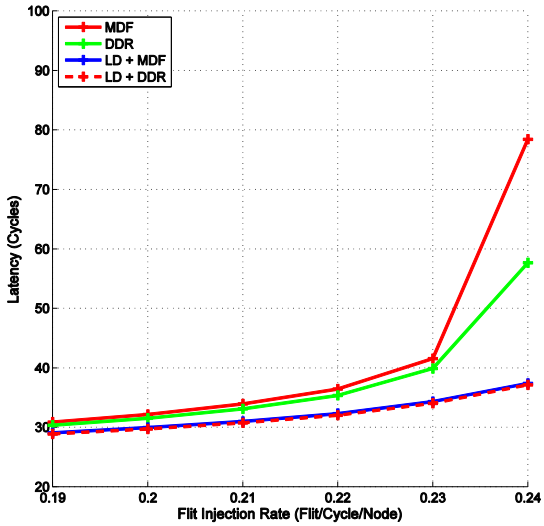

Figure 81: Average packet latency for LD enhancement over other ranking policies

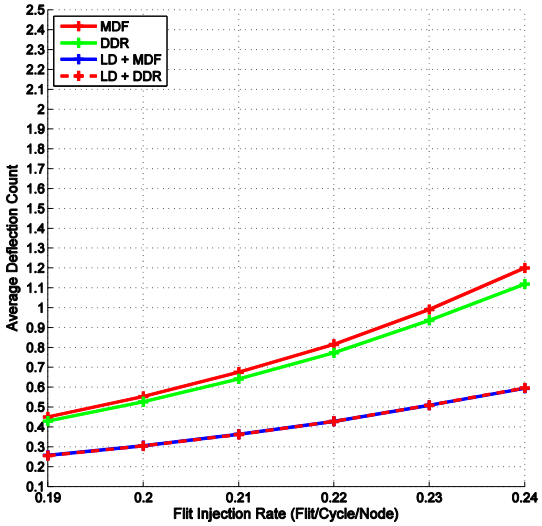

Figure 82: Average deflection count for LD enhancement over other ranking policies

deflection count. Also, the deflection count performance shown in Figure 80 matches the packet latency results.

In order to show how the LD enhancement affects the performance, we simulated LD with MDF and DDR as draw breakers. We compared LD performance in contrast with MDF and DDR respectively. As shown in Figure 81 and Figure 82, the LD enhancement greatly boosts the performance under higher injection rates. Specifically, using LD along with MDF under injection rate of 0.24 flit/cycle/node enhances the packet latency and the deflection count over MDF by 52.3% and 50.4% respectively. While using LD along with DDR enhances the packet latency and the deflection count over DDR by 35.6% and 46.7% respectively.

To explain this superior performance, we refer to Figure 82. As shown in Figure 82, the average deflection count for LD along with either MDF or DDR dramatically decreases as LD removes any unnecessary detours for the flits. Decreasing the deflection count for the flits directly affects the overall packet latency.

5.4. Concluding Remarks

In this chapter, we presented new deflection-based flit ranking policies. We first explained how bufferless NoCs are more affected by flit ranking polices more than buffered NoCs. Also, we explained the idea behind choosing the deflection count as our criterion. Then, we updated the MDF ranking policy by incorporating the age and the distance between the flit's source and destination along with the deflection count. In addition to updating MDF ranking policy, we proposed the LD enhancement that can be used along with other ranking policies to decrease the deflection count and hence improve the performance. Finally, we provided an experimental study for the proposed polices and the enhancement on a 10x10 mesh versus other well-known ranking polices.

Chapter 6 : Time-Sensitive Congestion Management Mechanisms

In the previous chapters, we investigated the use of output port selection functions and flit ranking policies to enhance the bufferless NoC performance. However, none of the proposed approaches directly targets the main roadblock facing bufferless NoC, namely the congestion problem.

In this chapter, we investigate the role of using proper congestion management mechanisms on bufferless NoC performance. Congestion can quickly develop under bufferless NoCs due to the lack of buffers. By managing the congestion, the performance is boosted in a drastic way.

First, we explain the importance of congestion management and why we choose the prevention approach. Then, we present different congestion prevention mechanisms. Finally, we simulate and evaluate the proposed approaches.

The chapter is organized as follows; Section 6.1 discusses the importance of managing the congestion specifically in bufferless NoCs. In Section 6.2, we propose two different prevention mechanisms. The updated experimental setup and the experimental results are presented and discussed in Section 6.3. Finally, Section 6.5 concludes the chapter.

6.1. Motivation

Due to lack of buffers, congestion can quickly develop in bufferless NoC preventing it from competing with the buffered NoCs performance especially under high injection rates. As mentioned earlier, combining high injection rate with the deflection behavior of the bufferless NoC leads to increased traffic volume which results in more contention between the flits. As the contention increases, the deflection rates increases and the starvation at the source nodes also increases (the source nodes are not able to inject new flits). This leads to a collapse in the performance of the NoC.

Various approaches exist for managing the NoC congestion. These approaches falls under one of two categories: detect and control the congestion, or prevent the congestion from developing. The first category approaches apply heuristics and monitor the NoC performance to detect the congestion once it arises. If congestion is detected, these approaches apply a control mechanism to relieve the congested areas. The problem with the first category approaches is that if the heuristics used to monitor the performance or the actions taken to relieve the congestion are biased or excessive, the overall performance of the system is affected.

On the other hand, the prevention approaches uses extra resources to decrease the probability of developing the congestion. The idea is to use the extra resources to provide other options for the flits in case of contention under high traffic volume. For example, a buffered NoC can use extra buffers to host the flits in case of increased traffic volume. In bufferless NoCs, we don’t have the luxury of using buffers, so we investigate how to prevent the congestion with the only buffering resource available i.e. the NoC links.

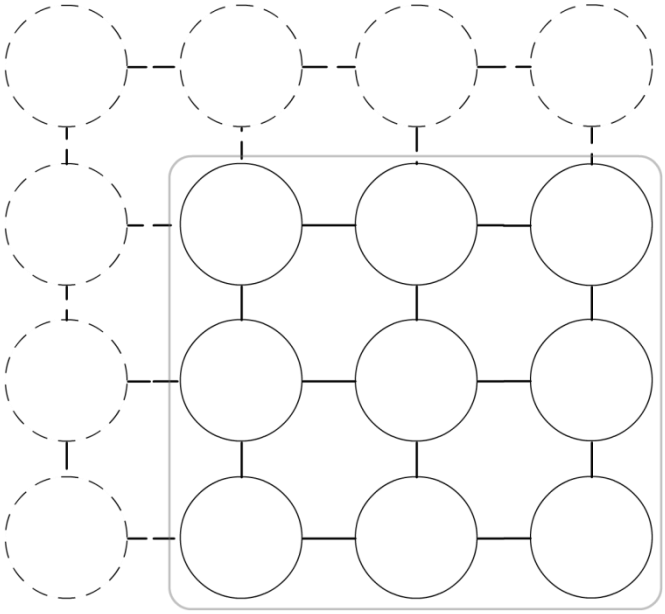

Figure 83: Using 4x4 mesh instead of 3x3 mesh

6.2. Proposed Approaches

In this section, we investigate how to relieve the traffic volume under bufferless NoC thus preventing the congestion from developing in the first place. Our goal is to operate latency-sensitive applications on bufferless NoCs under high injection rates without inducing extra power or chip area usage.

To be able to do that, we provide more links bandwidth to the flits so that they have more freedom in their movement towards their destinations. We propose two mechanisms to achieve this freedom. The first approach runs the application mix on larger NoCs, while the second approach divides the application mix to smaller subsets to be run sequentially.

6.2.1. Using Larger NoCs (LNoC)

In the LNoC approach, we propose running the application mix on a larger NoC with more nodes, switches, and links. For example, as in Figure 83, instead of running the application mix on a 3x3 mesh, we run it on a 4x4 mesh. Specifically, instead of running a given application mix on an $n \times m$ mesh and quickly reach congestion at injection rate R_1, we run the same application mix on $k \times l$ mesh and operate under injection rate R_2 where $n \times m \ < k \times l$ and $R_1 < R_2$.

The idea behind LNoC is to take advantage of the extra links provided as a result of using the larger NoC thus providing extra space for the flits to move with less competition with the other flits. Figure 83 shows the extra nodes (switches) and links as dotted circles and line respectively.

6.2.2. Using Sequential Injection (SI)

In the SI approach, we propose dividing the application mix into smaller subsets where only a subset of the NoC nodes is allowed to inject it. Then, instead of running

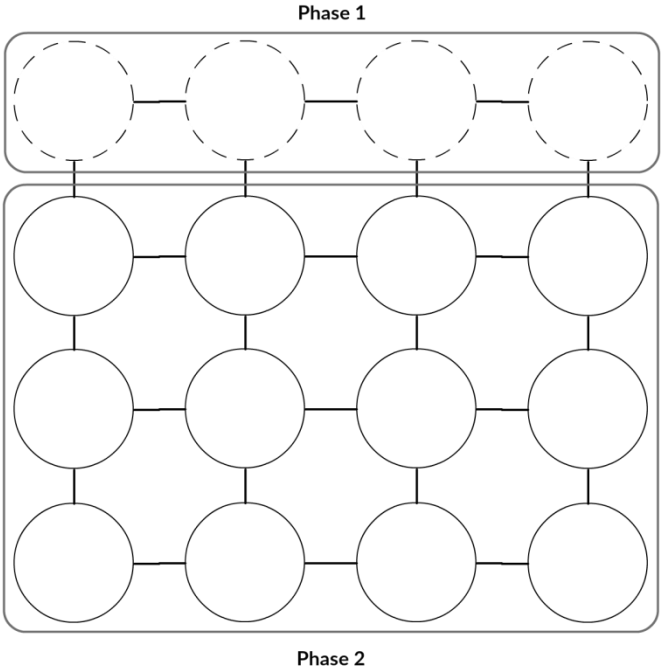

Figure 84: Example of two phase sequential injection

and injecting all the applications traffic at the same run, we divide the injection into sequential runs. In other words, we run the smaller application subsets sequentially on the whole NoC. Figure 84 shows an example for two phase injection. In the example, four nodes inject their traffic during the first phase. After receiving phase one injected traffic, the rest of the nodes (twelve nodes) inject their traffic into the NoC.

By doing that, we basically divide the problem of running the given application mix to a group of smaller application mixes that we can run in sequence. The smaller application mix, which results in smaller traffic volume, in combination with the sequential operation leads to injecting less data into the NoC in each smaller run which directly affects the deflection count and the packet latency in a positive way.

6.3. Simulation Results

In this section, we adapt the same experimental setup in Chapter 3; however, we change the termination condition for each run. We simulate a 10x10 mesh; however, instead of having a warm-up period of 100,000 cycles, and termination after receiving 1000,000 packets, we remove the warm-up period, inject 10,000 packets per node and terminates when all these packets are received.

We evaluate each of the proposed prevention mechanisms separately. We start by evaluating the LNoC approach in two ways. First, we compare the performance of running fifteen nodes in different mesh sizes, specifically, 3x5 mesh, 5x3 mesh, and 4x4 mesh with one extra node (switch). Second, we evaluate the effect of placing the extra nodes by simulating 10x10 mesh and change the number and the position of the extra nodes (switches). Concerning the SI approach evaluation, we simulate 10x10 mesh to study the effect of the number of nodes in each phase and their position in the NoC.

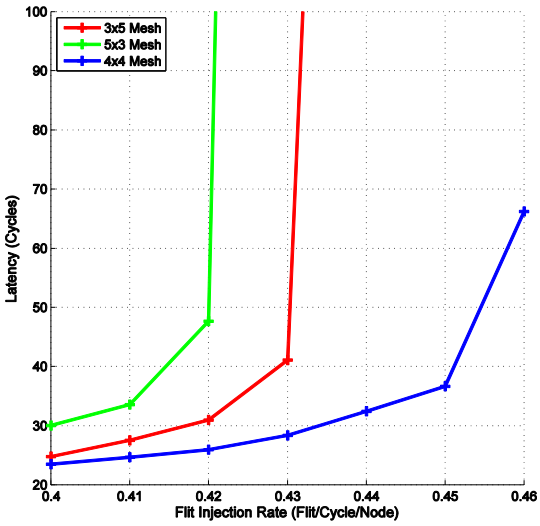

Figure 85: Average packet latency for fifteen nodes in different mesh sizes

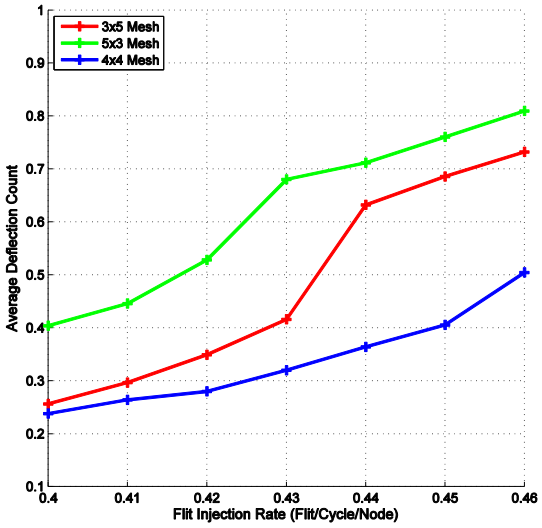

Figure 86: Average deflection count for fifteen nodes in different mesh sizes

To evaluate the LNoC approach, we considered an application that uses fifteen nodes only and we arrange the nodes in three different mesh sizes: 3x5 mesh, 5x3 mesh, and 4x4 mesh with one extra node. We used MMaxFlex with step size of one to study the effect of using different arrangements and extra node(s). As shown in Figure 85 and Figure 86, using 4x4 mesh resulted in better performance in both average packet latency and average deflection count. The enhancement is accounted for the use of extra node (switch) and the links connected to it which provided extra freedom for the flits to reach their destinations. Specifically, using an extra node instead of the required fifteen nodes in 3x5 mesh enhances the average packet latency and the average deflection count at flit injection rate 0.48 flit/cycle/node by 98.85% and 31.07% respectively. Also, from both figures, we notice that using 3x5 mesh is better that using 5x3 mesh in both performance metrics. This is due to the default behavior of MaxFlex, namely, moving on *X*-dimension first then on *Y*-dimension. Thus, as the number of columns in 5x3 mesh is less than the number of columns in 3x5 mesh (three versus five), the flits have more freedom to move in the *X*-dimension in case of 3x5 mesh than in case of 5x3 mesh.

The previous experiment did not study the number of the extra nodes used and their placement in the NoC, so we simulated 10x10 mesh and varied the number of extra nodes and changed their location from border nodes to central nodes. We compared using all the nodes in 10x10 mesh with the following: 90 nodes with 10 extra nodes placed as border nodes, 90 nodes with 10 extra nodes placed as central (core) nodes, 80 nodes with 20 extras nodes as central nodes, and 50 nodes with 50 extra nodes placed in the even columns of the 10x10 mesh. All of the previous experiments were simulated under MMaxFlex with step size of eight.

As shown in Figure 87 and Figure 88, using any extra nodes enhanced the performance over using all the 10x10 mesh nodes. This is also a result of the extra space provided for the flits in case of using extra nodes. For example, using only 90 nodes for injecting traffic instead of the provided 100 nodes leaves 10 switches in addition to their links to help in forwarding the traffic. The extra links works as extra roads for the flits to move.

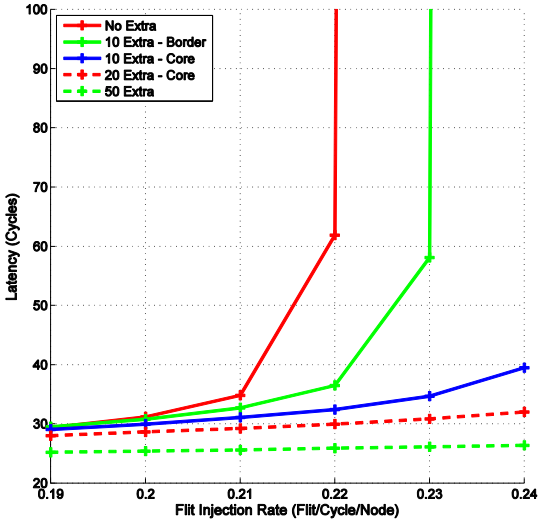

Figure 87: Average packet latency for different number of extra nodes in different locations in 10x10 mesh

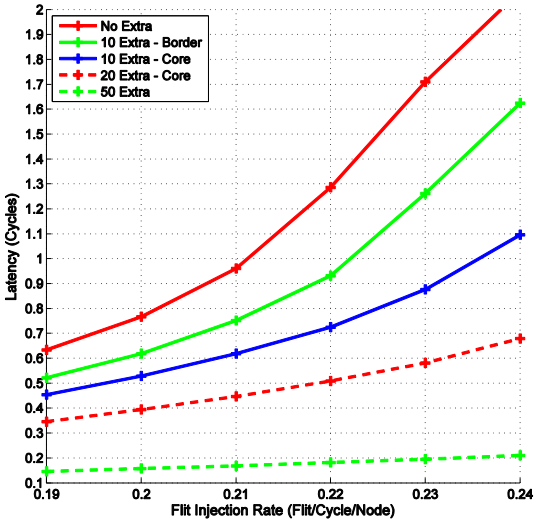

Figure 88: Average deflection count for different number of extra nodes in different locations in 10x10 mesh

Also, Figure 87 and Figure 88 presented the effect of the number of extra nodes and their placement. As the number of extra nodes increases, both the average packet latency and the average deflection count decreases. This is because using more extra nodes leads to more space for the flits to reach their destination. Concerning the placement of the extra nodes, Figure 87 and Figure 88 shows the difference between using 10 extra nodes placed on the border of the NoC and using 10 extra nodes placed in the center of the NoC. As in both figures, placing the extra nodes in the center of the NoC enhanced the performance over placing them on the border in terms of average packet latency, average deflection count, and the flit injection rate. The enhancement is due to the fact that the central switches are responsible for more traffic forwarding and handling than the border switches, thus placing the extra nodes in the center frees the central switches for forwarding only and leaves the injection for the rest of the nodes.

Concerning the SI approach evaluation, we used two phase sequential injection with different number of nodes at each phase. Also, we changed the location of the nodes in each phase to study the effect of the nodes placement. By two phase sequential injection, we mean that we divide the NoC nodes into two groups that take turn in injecting their traffic. For evaluation, we compared injecting the traffic from all the nodes in 10x10 mesh as one phase with the following: two phase with 90 nodes in the first phase and 10 nodes placed as border nodes in the second phase, two phase with 90 nodes in the first phase and 10 nodes placed as central nodes in the second phase, and two phase with 80 nodes in the first phase and 20 nodes placed as central nodes in the second phase. All of the previous experiments were simulated under MMaxFlex with step size of eight.

As shown in Figure 89 and Figure 90, using two phase SI injection enhances the performance over using one phase injection in terms of the used performance metrics. Specifically, in Figure 90, the average deflection count decreases as ratio between the number of nodes in each phase increases. This can be explained as in LNoC approach, namely, dividing the nodes evenly between the phases lead to less nodes injecting in each phase which lead to less competition between the flits, hence less deflections. As for the packet latency, increasing the ratio between the number of nodes in each phase

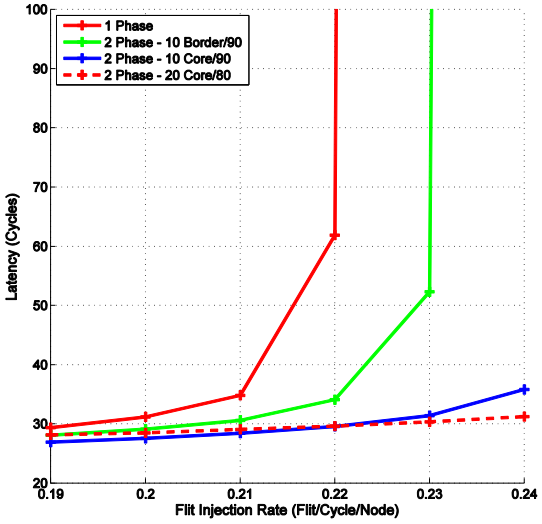

Figure 89: Average packet latency for two phase SI using different number of nodes in different locations in 10x10 mesh

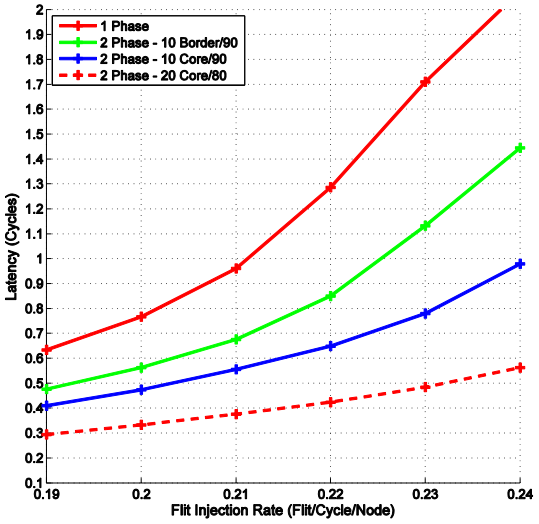

Figure 90: Average deflection count for two phase SI using different number of nodes in different locations in 10x10 mesh

resulted in better average latency and achieves higher flit injection rates as shown in Figure 89.

As for the nodes placement, changing the location of nodes from the border of the NoC to the center of the NoC decreased the average packet latency and the average deflection count by 98.36% and 32.2% respectively. This enhancement is accounted to the same reasons as in LNoC. Specifically, the central switches forward and handle more traffic than the border switches, thus separating the center nodes injection in different phase frees the center of the NoC to only forward the traffic of the rest of the nodes.

6.4. Concluding Remarks

In this chapter, we presented the idea of using proper congestion prevention mechanisms in bufferless NoCs. Also, we presented two prevention mechanisms, LNoC and SI, and idea behind each of them. Each of the two approaches provided more space for the flits to move in the NoC thus less contention between the flits. To test the performance of the proposed approaches, we simulated a 10x10 mesh. Our results showed that our proposed mechanisms resulted in better performance in terms of both average packet latency and average deflection count compared with fixed step size of 8 under MMaxFlex.

Chapter 7 : Discussion and Conclusion

In this thesis, we were concerned with pushing the boundaries of using bufferless NoCs. In other words, how bufferless NoCs can achieve a performance, packet latency and deflection count, similar to buffered NoCs under higher injection rates but with the added benefit of less power and area. We first focused on using the selection functions to achieve our goal. Specifically, we investigated using larger and variable step sizes under MaxFlex selection function to enhance the traffic distribution and hence the performance. Our analytical and experimental work showed that using larger step size values led to better performance figures. Also, using variable step size for each packet instead of fixing the value for all packets led to better traffic distribution which resulted in enhanced performance. Then, we shifted to investigate the usage of different ranking policies under MaxFlex to boost the performance enhancement. We tailored our proposed policies to focus on decreasing the flits' deflections as enhancing the deflection count should result in better packet latency. Finally, we looked into easing the congestion problem in bufferless NoCs. We wanted to prevent the congestion instead of detecting and controlling it later. Our prevention mechanisms allowed the flits to have more link bandwidth while moving to their destinations. We achieved that by using extra resources and/or organizing the injection of the running latency-sensitive applications. Our work in this part showed a huge enhancement in both the packet latency and the deflection count.

7.1. Future Work

We can extend our work in different directions. First, we can investigate the proper size for the regions based on the overall NoC size as we only investigated the usage of regions in determining the variable step size value. Also, we can look into other formulas to determine the variable step size. Additionally, the concept of dividing the NoC into regions can be extended to other aspects in NoC not only for the variable step size. For example, regions can be used to enhance the performance on the application level by assigning different applications to different regions and based on each application we can customize each region. Second, we can extend our congestion mechanisms to consider throughput-sensitive applications like GPGPUs in addition to latency-sensitive applications. Finally, we want to investigate the effect of absorbing and re-injecting the NoC traffic via "Sink Nodes" as an approach to ease congestion instead of using source throttling as most of the presented work in the literature proposed.

Beside the proposed extensions, we can investigate the bufferless NoCs usage in other hot topics. One of the current hot topics related to NoCs is the usage of die stacking technologies to incorporate memory stacks inside the chip. Currently, instead of using 3D stacking, researchers are investigating the usage of 2.5D stacking i.e. silicon interposer. In 2.5D stacking, instead of adding the memory or other processor die on the top of the base processor die, the silicon interposer is built to be large enough to hold the processor die and the memory stacks surrounding the die. The interposer is a layer rich in communication resources which can be harvested to connect several components in the chip with extra cost. Recent works proposed the usage of the silicon interposer instead of 3D stacking. The 2.5D stacking presents several challenges in

designing the NoC to support the higher memory bandwidth required. We can look into using the bufferless NoC in the design to harvest the underlying rich interposer without the need to add extra buffers. Also, both 3D and 2.5D technologies can be investigated to see how using the bufferless NoC can enhance the overall design.

Also, recent works investigated the usage of random topologies for NoCs. They showed that random topologies provide better scalability in terms of network diameter and provide inherent load balancing. We can look into using the bufferless NoC design with these random topologies.

References

[1] L. Benini and G.De. Micheli, "Networks on Chips: A New SoC Paradigm," *Computer*, pp. 70-78, 2002.

[2] W. J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," in *Design Automation Conference*, Las Vegas, 2001, pp. 684-689.

[3] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*. San Francisco, California: Morgan Kaufmann, 2002.

[4] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, California: Morgan Kaufmann, 2003.

[5] S.R. Vangal et al., "An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS," *IEEE Journal of Solid-State Circuits*, pp. 29-41, 2008.

[6] M.B. Taylor et al., "Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams," in *International Symposium on Computer Architecture*, Munich, 2004, pp. 2-13.

[7] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz Mesh Interconnect for a TeraFLOPS Processor," in *International Symposium on Microarchitecture*, Chicago, 2007, pp. 51-61.

[8] P. Gratz, C. Kim, R. McDonald, S.W. Keckler, and D. Burger, "Implementation and Evaluation of On-Chip Network Architectures," in *International Conference on Computer Design*, San Jose, 2006, pp. 477-484.

[9] C. Gómez, M. E. Gómez, P. López, and J. Duato, "Reducing Packet Dropping in a Bufferless NoC," in *International European Conference on Parallel and Distributed Computing*, Las Palmas de Gran Canaria, 2008, pp. 899-909.

[10] T. Moscibroda and O. Mutlu, "A Case for Bufferless Routing in On-Chip Networks," in *International Symposium on Computer Architecture*, Austin, 2009, pp. 196-207.

[11] M. Hayenga, N.E. Jerger, and M. Lipasti, "SCARAB: A Single Cycle Adaptive Routing and Bufferless Network," in *International Symposium on Microarchitecture*, New York, 2009, pp. 244-254.

[12] G. Michelogiannakis, D. Sanchez, W.J. Dally, and C. Kozyrakis, "Evaluating Bufferless Flow Control for On-Chip Networks," in *ACM/IEEE International Symposium on Networks-on-Chip*, Grenoble, 2010, pp. 9-16.

[13] S. Badr and P. Podar, "An Optimal Shortest-Path Routing Policy for Network Computers with Regular Mesh-Connected Topologies," *IEEE Transactions on Computers*, pp. 1362-1371, 1989.

[14] W. J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels," *IEEE Transactions on Parallel and Distributed Systems*, pp. 466-475, 1993.

[15] C. Gomez, M. E. Gomez, P. Lopez, and J. Duato, "BPS: A Bufferless Switching Technique for NoCs," in *Workshop on Interconnection Network Architectures*, 2008, pp. 1-6.

[16] A. Lankes, T. Wild, S. Wallentowitz, and A. Herkersdorf, "Benefits of Selective Packet Discard in Networks-on-Chip," *ACM Transactions on Architecture and Code Optimization*, 2012.

[17] J. Lin, X. Lin, and L. Tang, "Making-a-Stop: A New Bufferless Routing Algorithm for On-Chip Network," *Journal of Parallel and Distributed Computing*, pp. 515-524, 2012.

[18] C. Fallin, C. Craik, and O. Mutlu, "CHIPPER: A Low-complexity Bufferless Deflection Router," in *International Symposium on High Performance Computer Architecture*, San Antonio, 2011, pp. 144 - 155.

[19] S.A.R. Jafri, Yu-Ju Hong, M. Thottethodi, and T.N. Vijaykumar, "Adaptive Flow Control for Robust Performance and Energy," in *International Symposium on Microarchitecture*, Atlanta, 2010, pp. 433 - 444.

[20] C. Fallin et al., "MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect," in *International Symposium on Networks-on-Chip*, Lyngby, 2012, pp. 1 - 10.

[21] J., Nayak, B. Jose, K. Kumar, and M. Mutyam, "DeBAR: Deflection Based Adaptive Router With Minimal Buffering," in *Conference on Design, Automation and Test in Europe*, Grenoble, 2013, pp. 1583 - 1588.

[22] Y. Li, K. Mei, Y. Liu, N. Zheng, and Y. Xu, "LDBR: Low-Deflection Bufferless Router for Cost-Sensitive Network-on-Chip Design," *Microprocessors and Microsystems*, vol. 38, no. 7, pp. 669-680, October 2014.

[23] T. Weller and B. Hajek, "Comments on "An Optimal Shortest-Path Routing Policy for Network Computers with Regular Mesh-Connected Topologies"," *IEEE Transactions on Computers*, pp. 862-863, 1994.

[24] W. Feng and K. Shin, "Impact of Selection Functions on Routing Algorithm Performance in Multicomputer Networks," in *International Conference on Supercomputing*, Vienna, 1997, pp. 132-139.

[25] M. Koibuchi, A. Jouraku, and H. Amano, "MMLRU Selection Function: A Simple and

Efficient Output Selection Function in Adaptive Routing," *IEICE Transactions*, pp. 109-118, 2005.

[26] F. Gilabert, M. E. Gómez, P. López, and J. Duato, "On the Influence of the Selection Function on the Performance of Fat-trees," in *International European Conference on Parallel and Distributed Computing*, Dresden, 2006, pp. 864-873.

[27] A. Farouk and H.M. El-Boghdadi, "On the Influence of Selection Function on the Performance of Fat-Trees under Hot-Spot Traffic," in *IEEE/ACS International Conference on Computer Systems and Applications*, Sharm El-Sheikh, 2011, pp. 120-127.

[28] A. Farouk and H.M. El-Boghdadi, "A Cost-efficient Congestion Management Methodology for Fat-trees using Traffic Pattern Detection," *The Journal of Supercomputing*, vol. 71, no. 4, pp. 1249 - 1276, April 2015.

[29] J. Jose, B. M. Jacob, and H. P. Kamal, "An Energy Efficient Load Balancing Selection Strategy for Adaptive NoC Routers," in *International Workshop on Network on Chip Architectures*, Cambridge, 2014, pp. 31 - 36.

[30] Z. Lu, M. Zhong, and A. Jantsch, "Evaluation of On-Chip Networks using Deflection Routing," in *Great Lakes Symposium on VLSI*, Philadelphia, 2006, pp. 296 - 301.

[31] G. Nychis, C. Fallin, T. Moscibroda, and O. Mutlu, "Next Generation On-Chip Networks: What Kind of Congestion Control Do We Need?," in *Workshop on Hot Topics in Networks*, Monterey, 2010.

[32] R. Ausavarungnirun, K. K.-W. Chang, C. Fallin, and O. Mutlu, "Adaptive Cluster Throttling: Improving High-Load Performance in Bufferless On-Chip Networks," Computer Architecture Lab (CALCM) Carnegie Mellon University, SAFARI Technical Report 6, 2011.

[33] K. K.-W. Chang, R. Ausavarungnirun, C. Fallin, and O. Mutlu, "HAT: Heterogeneous Adaptive Throttling for On-Chip Networks," in *International Symposium on Computer Architecture and High Performance Computing*, New York, 2012, pp. 9 - 18.

[34] J. Yan, G. Lai, and X. Lin, "A Novel Distributed Congestion Control for Bufferless Network-on-Chip," *The Journal of Supercomputing*, vol. 68, no. 2, pp. 849 - 866, May 2014.

[35] Tilera Corporation. Tilera announces the world's first 100-core processor with the new tile-gx family. [Online]. http://www.tilera.com/news_&_events/press_release_091026.php

[36] D. Wentzlaff et al., "On-Chip Interconnection Architecture of the Tile Processor," in *International Symposium on Microarchitecture*, Chicago, 2007, pp. 15 - 31.

[37] Intel Corporation. Single-Chip Cloud Computer. [Online]. http://techresearch.intel.com/articles/

[38] M. Taylor, J. Kim, J. Miller, and D. Wentzlaff, "The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs," in *International Symposium on Microarchitecture*, Istanbul, 2002, pp. 25 - 35.

[39] H. Hossain, M. Ahmed, A. Al-Nayeem, T.Z. Islam, and M.M. Akbar, "gpNoCsim - A General Purpose Simulator for Network-on-Chip," in *International Conference on Information and Communication Technology*, Dhaka, 2007, pp. 254-257. [Online]. http://www.buet.ac.bd/cse/research/group/noc/index.htm

Appendix A: 2D Mesh Terminologies

In this appendix, we explain the concept of main diagonal, and how to differentiate between the increasing and decreasing diagonals in $n \times n$ mesh. Also, we explain how to determine if a certain node is above or below the main diagonal.

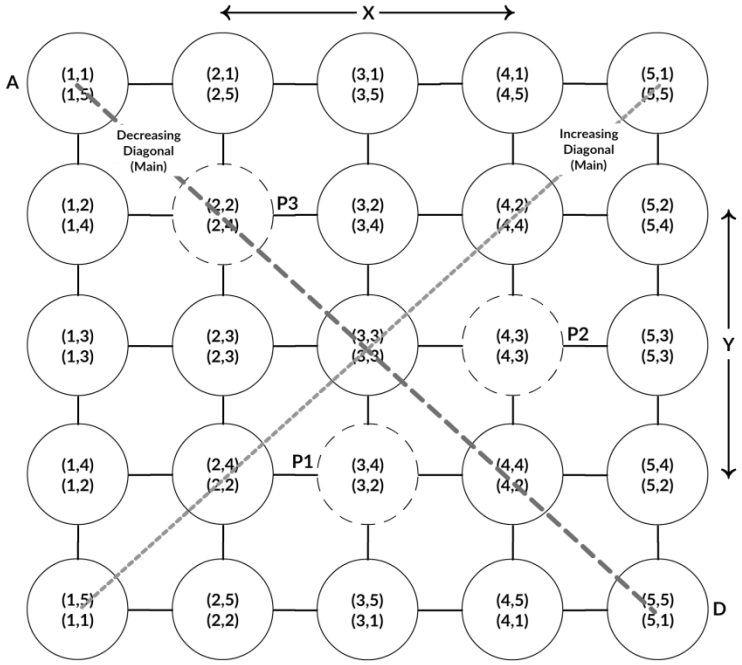

Figure 91: Main increasing and decreasing diagonals in 5x5 mesh

We start by defining the main diagonal concept in $n \times n$ mesh. The *main diagonal* is the longest diagonal in a given $n \times n$ mesh. In other words, it is the diagonal with n nodes on it. All other diagonals in $n \times n$ mesh contains less than n nodes. Figure 91 shows an example of main diagonals in 5x5 mesh.

Also, we differentiate between increasing and decreasing diagonals in $n \times n$ mesh. Figure 91 shows both of the diagonal types. In the *decreasing diagonal*, both the *X* and *Y* indices increases for each node along the diagonal. In contrast, the *X* index increases while the *Y* index decreases for each node along the *increasing diagonal*. A typical 2D mesh node belongs to an increasing diagonal as well as a decreasing diagonal but not necessarily of same size. For example, node *P3* in Figure 91 belongs to the main decreasing diagonal and to an increasing diagonal with three nodes.

To determine if a node is above or below the main diagonal, we study the slope of a virtual line on which the node lays. Also, we shall differentiate between the increasing and decreasing diagonal cases. For example, in Figure 91, to determine if nodes *P1* and *P2* are above the main decreasing diagonal, we compare between the slopes of lines *AD* and *AP1*, and *AP2*.

$$AD_{Slope} = \frac{5-1}{5-1} = 1$$

$$AP1_{Slope} = \frac{4-1}{3-1} = 1.5$$

$$AP2_{Slope} = \frac{3-1}{4-1} = 0.67$$

Since $AP1_{Slope} > AD_{Slope}$, then node *P1* is below the main decreasing diagonal (i.e. line AD). Also, since $AP2_{Slope} < AD_{Slope}$, then node *P2* is above the main decreasing diagonal.

Also, we can use the node *X* and *Y* indices to determine if a node is above or below the main diagonal. As in the previous method, we differentiate between increasing and decreasing diagonals. Figure 91 shows the nodes indices in both (X, Y) and $(X, n+1-Y)$ formats. For the decreasing diagonal, if $X < Y$, then the node is below the main decreasing diagonal; else the node is above the main decreasing diagonal. For the increasing diagonal, $X > n+1-Y$, then the node is below the main increasing diagonal; else the node is above the main increasing diagonal. For example, node $P1(X = 3, Y = 4)$ has $X < Y$, then node P1 is below the main decreasing diagonal. Also, node $P3(X = 2, n+1-Y = 4)$ has $X < n+1-Y$, then the node is above the main increasing diagonal.

Publications

[1] M. A. Abd ElMohsen and H. M. El-Boghdadi, "Investigating the Viability of Maximum Flexibility Selection Function in Bufferless 2D Meshes," in *International Workshop on Many-core Embedded Systems*, Portland, 2015, pp. 52-55.

الملخص

مع وصول الأنظمة متعددة المعالجات، بدأت الشبكة–على–رقاقة تشكل العمود الفقري للاتصال داخل رقاقة المعالج الدقيق. و لكن يحد أداء الشبكة–على–رقاقة استهلاكها العالى للطاقة و لمساحة الرقاقة. كحل للحد من استهلاك الطاقة والمساحة، ظهرت الشبكة–على–رقاقة الغيرُ مُخَزِّنة. ازالت الشبكة–على–رقاقة الغيرُ مُخَزِّنة عناصر تخزين تستخدم لتوجيه حزم البيانات و/أو التحكم في تدفقها وتتعامل مع التنافس على مخارج التوجيه باستخدام إسقاط الحزمة من الشبكة أو تغيير مسار الحزمة بعيدا عن اقصر مسار.

في هذه الأطروحة، نحن نركز على تحسين أداء الشبكة–على–رقاقة الغيرُ مُخَزِّنة للتطبيقات ذات الحساسية للوقت عن طريق تقليص الوقت اللازم للوصول وتقليص عدد الانحرافات. تم تقسيم الأطروحة لتركز على ثلاث محاور. أولاً، كيفية اختيار مخارج توجيه حزم البيانات. ثانياً، كيفية ترتيب حزم البيانات في حالة التنافس على مخرج توجيه. أخيراً، كيفية تخفيف الازدحام في الشبكة في ظل ارتفاع معدل تدفق حزم البيانات.

أولاً، نقدم دراسة لإختيار مخارج توجيه حزم البيانات بقدر عالي من المرونة باستخدام قيم ثابتة و متغيرة للخطوة المستخدمة. تستخدم هذه الطريقة ذات المرونة العالية لانها تعمل على زيادة الاختيارات المتاحة لحزم البيانات. في هذا الجزء، نقدم دراسة تحليلية لحركة مرور حزم البيانات في الشبكة–على–رقاقة الغيرُ مُخَزِّنة التى تختار مخارج توجيه حزم البيانات بقدر عالى من المرونة باستخدام قيم مختلفة للخطوة المستخدمة. تشير نتائج المحاكاة أنه مع قيم معينة للخطوة، يمكن تخفيض وقت وصول حزم البيانات بنسبة 97٪ مقارنة باختيار مخارج التوجيه بشكل مستقيم. الدراسة التحليلية المقدمة توضح تفوق النتائج التجريبية.

ثانياً، نقدم طرق مختلفة لترتيب خدمة حزم البيانات التي تركز على تخفيض عدد انحرافها. و تبين نتائج المحاكاة أن بعض الطرق يمكن أن تقلل من وقت وصول حزم البيانات بنسبة تصل إلى 58٪ مقارنة بسياسة اختيار الاقدم.

و أخيراً، نوجه اهتمامنا للتخفيف من تأثير الازدحام في الشبكة في ظل ارتفاع معدل تدفق حزم البيانات. نقترح اسلوبين لمنع اسباب الازدحام. و تهدف أول طريقة لتشغيل التطبيقات على الشبكة باستخدام موارد إضافية. الطريقة الثانية تقسم حزم البيانات المتدفقة لسلسلة من الأحمال الأخف. وتبين نتائج المحاكاة أن الطرق المقترحة تعزز وقت وصول حزم البيانات بنسبة تصل إلى 61٪، بالإضافة إلى رفع معدل تدفق حزم البيانات.

مهندس: محمد عاصم عبد المحسن ابراهيم

تاريخ الميلاد: 1988\07\07

الجنسية: مصرى

تاريخ التسجيل: 2010\10\01

تاريخ المنح: ....\....\2016

القسم: هندسة الحاسبات

الدرجة: ماجستير العلوم

المشرفون: أ. د. حاتم محمود البغدادى

الممتحنون: أ. د. محمد زكي عبدالمجيد (الممتحن الخارجي)
- استاذ بكلية الهندسة - جامعة الأزهر

أ. م. د. عمرو جلال الدين وصال (الممتحن الداخلي)

أ. د. حاتم محمود البغدادى (المشرف الرئيسي)

عنوان الرسالة:

عن تحسين أداء الشبكة–على–الرقاقة الغيرُ مُخَزِّنة

الكلمات الدالة:

شبكة–على–رقاقة غيرُ مُخَزِّنة، طريقة اختيار مخرج للتوجيه، طريقة اختيار ذات مرونة قصوى، ترتيب حزم البيانات، التعامل مع الاحتقان

ملخص الرسالة:

مع وصول الأنظمة متعددة المعالجات، بدأت الشبكة–على–رقاقة تشكل العمود الفقري للاتصال داخل رقاقة المعالج الدقيق. و لكن يحد أداء الشبكة–على–رقاقة استهلاكها العالى للطاقة و لمساحة الرقاقة. كحل للحد من استهلاك الطاقة والمساحة، ظهرت الشبكة–على–رقاقة الغيرُ مُخَزِّنة. ازالت الشبكة–على–رقاقة الغيرُ مُخَزِّنة عناصر التخزين وتتعامل مع التنافس على مخارج التوجيه باستخدام إسقاط الحزمة من الشبكة أو تغيير مسار الحزمة بعيدا عن اقصر مسار. في هذه الأطروحة، نحن نركز على تحسين أداء الشبكة–على–رقاقة الغيرُ مُخَزِّنة عن طريق تقليص الوقت اللازم للوصول وتقليص عدد الانحرافات.

أولا، نقدم دراسة تحليلية لحركة مرور الحزم في الشبكة–على–رقاقة الغيرُ مُخَزِّنة التى تختار مخارج توجيه الحزم بقدر عالى من المرونة باستخدام قيم مختلفة للخطوة المستخدمة. تشير نتائج المحاكاة أنه مع قيم معينة للخطوة، يمكن تخفيض وقت وصول الحزم بنسبة 97٪ مقارنة باختيار مخارج التوجيه بشكل مستقيم. الدراسة التحليلية المقدمة توضح تفوق النتائج التجريبية. ثانيا، نقدم طرق مختلفة لترتيب خدمة الحزم التي تركز على تخفيض عدد انحرافها. و تبين نتائج المحاكاة أن بعض الطرق يمكن أن تقلل من وقت وصول الحزم بنسبة تصل إلى 58٪ مقارنة بسياسة اختيار الاقدم. و أخيرا، نوجه اهتمامنا للتخفيف من تأثير الازدحام في الشبكة في ظل ارتفاع معدل تدفق الحزم. نقترح اسلوبين لمنع اسباب الازدحام. و تهدف أول طريقة لتشغيل التطبيقات على الشبكة باستخدام موارد إضافية. الطريقة الثانية تقسم الحزم المتدفقة لسلسلة من الأحمال الأخف. وتبين نتائج المحاكاة أن الطرق المقترحة تعزز وقت وصول الحزم، بالإضافة إلى رفع معدل تدفق الحزم.

عن تحسين أداء الشبكة-على-الرقاقة الغيرُ مُخَزّنة

إعداد
محمد عاصم عبد المحسن ابراهيم

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة ماجستير العلوم
في
هندسة الحاسبات

يعتمد من لجنة الممتحنين:

الدكتور: حاتم محمود البغدادى — المشرف الرئيسى
- استاذ بكلية الهندسة - جامعة القاهرة

الدكتور: عمرو جلال الدين وصال — الممتحن الداخلي
- استاذ مساعد بكلية الهندسة - جامعة القاهرة

الاستاذ الدكتور: محمد زكي عبدالمجيد — الممتحن الخارجي
- استاذ بكلية الهندسة - جامعة الأزهر

كليــة الهندســة - جامعــة القاهـرة
الجيـزة - جمهوريـة مصـر العربيــة
2016

عن تحسين أداء الشبكة-على-الرقاقة الغيرُ مُخَزِّنة

إعداد
محمد عاصم عبد المحسن ابراهيم

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة ماجستير العلوم
في
هندسة الحاسبات

تحت اشراف

د. حاتم محمود البغدادى
استاذ
قسم هندسة الحاسبات
كلية الهندسة جامعة القاهرة

كليــة الهندســة - جامعــة القاهــرة
الجيـزة - جمهوريـة مصـر العربيــة
2016

عن تحسين أداء الشبكة-على-الرقاقة الغيرُ مُخَزِّنة

إعداد

محمد عاصم عبد المحسن ابراهيم

رسالة مقدمة إلى كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة ماجستير العلوم
في
هندسة الحاسبات

كليــة الهندســة - جامعــة القاهـرة
الجيـزة - جمهوريـة مصـر العربيــة
2016