Design and Evaluation of Force-Directed (FD) Routing Algorithm in 2D Mesh Network on Chip

by

Kasem Shwiegi
B.Sc., Computer Engineering, University of Zawia, 2007

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Applied Science

in the Department of Electrical and Computer Engineering

Design and Evaluation of Force-Directed (FD) Routing Algorithm in 2D Mesh Network on Chip

by

Kasem Shwiegi
B.Sc., Computer Engineering, University of Zawia, 2007

Supervisory Committee

Dr. Fayez Gebali, Supervisor
(Electrical and Computer Engineering)

Dr. Atef Ibrahim, Departmental Member
(Electrical and Computer Engineering)

Supervisory Committee

Dr. Fayez Gebali, Supervisor
(Electrical and Computer Engineering)

Dr. Atef Ibrahim, Departmental Member
(Electrical and Computer Engineering)

ABSTRACT

System on Chip (SoC) connects several IP cores i.e., memories, processors, DSPs etc., in one chip. After the rapid progress in electronic systems, the number of cores needed in SoC has been increased to improve the performance and efficiency in the system. However, raising the number of IP cores in SoC has led to many issues in the chip, such as poor scalability, high complexity, more power consumption, high latency etc. The Network on Chip (NoC) is an interconnection network which provides a perfect architecture to resolve the limitations in SoC. The routing algorithm in NoC plays the main role in improving system performance and efficiency. In this thesis, a full-adaptive routing algorithm is proposed for 2D mesh NoC, which takes into consideration the network conditions for routing a packet and provides more flexibility to route a packet to any destination in the network. The proposed routing algorithm, named the Forced-Directed (FD) routing algorithm, directs a packet to its destination through a number of routes in the network. The FD routing algorithm presents an important improvement in throughput and efficiency and reducing the average packet loss in 2D mesh NoC when compared to the adaptive routing algorithm DyXY. The comparison between FD and DyXY routing algorithms is done in two simulation setup phases: the first phase is based on different input traffic loads, and the second is based on different 2D mesh NoC dimensions. In the first phase (with different traffic loads and the network dimension 5×5 mesh NoC), the FD routing algorithm has improved the network throughput by 84%, efficiency by 76% and has reduced the average packet loss by 52% compared to DyXY routing. In the second phase (with traffic load 0.7 and different network dimensions), throughput, efficiency and the

average packet loss have been enhanced by 90%, 92% and 59%, respectively, relative to the DyXY routing algorithm.

Contents

List of Tables

List of Figures

List of Acronyms

2D	Two Dimension
BS	Buffer Status
CMOS	Complementary Metal Oxide Semiconductor
DSP	Digital Signal Processor
FIFO	First in First out
FLITs	FLow controL digITs
FD	Force Directed
HoL	Head of Line
IC	Integrated Circuits
IP	Intellectual Property
NI	Network Interfaces
NoC	Network on chip
SoC	System on Chip
VLSI	Very Large Scale Integration

ACKNOWLEDGEMENTS

Foremost, all praises belong to Allah the Merciful and Gracious for giving me the guidance, blessings, strength and patience to be able to finish this work. I would like to thank:

My parents, brothers and sisters, for their prayers, support and motivation to achieve my scientific ambition.

My Supervisor, Dr. Fayez Gebali, for the mentoring and skillful guidance that he gave me during the period of my working under his supervision.

University of Victoria, for providing a decent environment that helps grad student to achieve their research objectives.

DEDICATION

I would like to dedicate this work to my parents and my supervisor Dr. Fayez Gebali.

Chapter 1

Introduction

In the last few decades, there has been rapid evolution in the field of microelectronics. Moreover, using Very Large-Scale Integration (VLSI) for creating Integrated Circuits (ICs) has resulted in integrating millions of transistors on a single chip [1] [2]. Nowadays, up to one billion transistors can be integrated into one chip using CMOS technology [1]. However, this huge increase in the number of transistors has led to increased complexity and decreased performance in the chip. To overcome the problem of complexity, SoC has been proposed as an efficient architecture that is useful for simplifying and improving the performance in the chip.

SoC is a complete system on one chip which consists of interconnected cores called Intellectual Property (IP) cores. An IP core is a functional component that has its own role inside the chip [3]. These IP cores can be general-purpose processors, DSP, embedded memories or I/O blocks, each having its own function and capacity. The ability to integrate a number of IP cores in one SoC helps in improving the electronic system's performance and decrease the complexity in the chip. However, an increase in the number of IP cores in a SoC results in greater communication complexity between these IP cores. The issue of communication complexity in SoCs has become the most important and considerable challenge in SoC design [4]. Therefore, several approaches and methods have been proposed to overcome this issue in SoCs.

The first approach for resolving the communication complexity in SoC is called point-to-point connection; it provides direct point-to-point communication between the IP cores in SoC. In this approach, the IP cores can directly communicate with each other without any priority or arbitration unit. However, increasing the number of IP cores in one chip increases the connectivity complexity and wiring density inside the chip. Therefore, a larger wiring area is needed to implement a complete system in

SoC. As a result, this approach includes many disadvantages i.e. a large wiring area, underutilization of resources, high complexity and poor scalability [5]. The direct point-to-point connection technique in SoC is shown in Fig. 1.1.

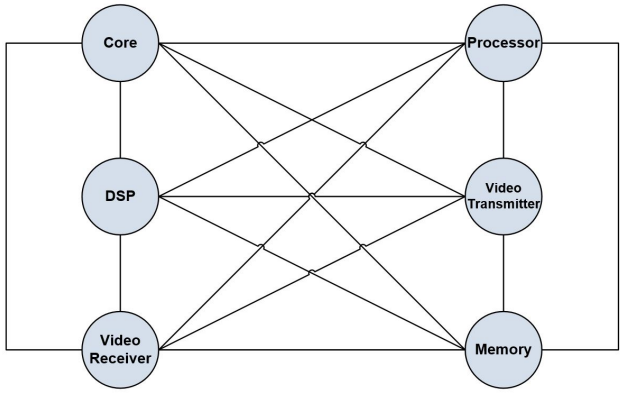

Figure 1.1: Point to Point Connection System

Another solution to the problem of communication inside SoCs is bus-based communication systems. There are several existing SoCs that are bus-based [3]. In bus-based communication, a shared communication bus is provided that connects multiple IP cores as shown in Fig. 1.2. Each IP core in the system can be connected to single or multiple buses through interfaces. In addition, a bus-arbiter controls the communication and contention in the system. In this communication system, the communication delay is reduced, but the bus-access time depends upon the number of IP cores connected to the bus. A larger number of IP cores connected to the bus leads to an increase in bus-access time. Moreover, arbitration operation in the system could affect the data movement. Also, the scalability can be scaled up only to a limited degree; otherwise, the systems efficiency will be impacted [6].

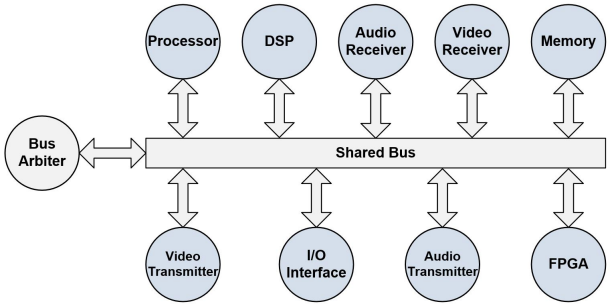

Figure 1.2: Bus-Based System

As discussed above, both communication systems have disadvantages in terms of resource underutilization, high complexity and less scalability. For this reason, Network-on-chip communication was proposed by Dally [7] to solve the mentioned problems. NoC is an interconnection network system which enables a decrease in the communication complexity and improve the performance in multi-core SoCs. NoC has three main components i.e., Router, Network Interface and Link as shown in Fig. 1.3. The Communication between the IP cores is done using routers that are distributed across the whole network and serve as the communication backbone of the NoC. The main purpose of the router in an NoC is to route the packet towards its destination either directly or through other routers. Routing packets in NoC depends on the routing algorithm used, which decides the approach and paths of packets moving through the network. NoC provides high-scalability, less resource underutilization, less complexity and high-performance compared to other communication techniques [8] [9].

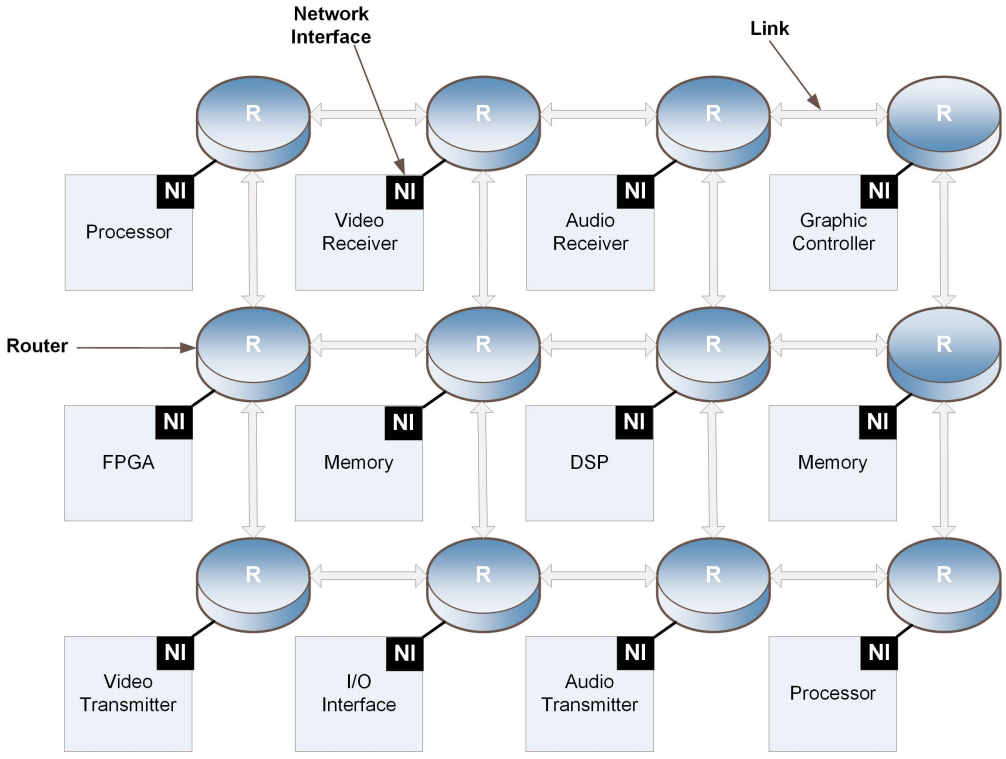

Figure 1.3: Network on Chip (NoC)-Based System

1.1 Research Goal and Objectives

Routers in NoC play a vital role in establishing communication between the IP cores using different protocols called routing algorithms [10]. The routing algorithm is the most significant part of the router because it is mainly responsible for determining the way of moving the packets inside NoC. Developing and enhancing routing algorithms has a direct effect on NoC performance and efficiency. For this reason, the main objective and goal of this thesis is to propose a novel full-adaptive routing algorithm, named Force-Directed (FD), to enhance the performance, throughput, efficiency in 2D mesh NoC. FD routing provides a full adaptability to route a packet from the source to the destination. Also, FD routing gives more optional paths to route a packet with a full consideration of network condition. Based on this, FD provides high throughput, efficiency and reduces the average packet loss in 2D mesh NoC compared to the DyXY and other non-full adaptive routing algorithms.

1.2 Contributions

The main contribution in this thesis is proposing a new full adaptive-routing algorithm for 2D mesh NoC. The proposed routing algorithm decreases the number of packet lost in 2D mesh NoC system and increases the throughput and efficiency result in from the network. The contributions can be summaries as follow:

1. Proposing a new full-adaptive routing algorithm named Force-Directed (FD) for 2D mesh NoC with full adaptability.

2. Implementing the proposed routing algorithm using object-oriented programming in Matlab which was not used before for implementing 2D mesh NoC routing algorithms.

1.3 Thesis Organization

In Chapter 1, we introduce a general overview of on-chip systems evolution, research goals and objectives, contributions as well as the thesis organization. In Chapter 2, we present a general Network-on-Chip background and the main concepts of NoC i.e., NoC topology, switching techniques and routing algorithms. Also, in the same chapter, the routing algorithm types and taxonomy have been demonstrated and

discussed. In Chapter 3, we illustrate an introduction for the proposed 2D mesh NoC routing algorithm, called Force-Directed (FD). Router architecture for FD Routing Algorithm, FD routing methodology, Packet routing scenario, and FD routing Pseudo code are detailed in the same chapter. In Chapter 4, FD routing algorithm evaluation and simulation are discussed with a comparison of its results such as throughput, efficiency, packet loss, and average latency with DyXY routing. The conclusions and future works are discussed in Chapter 5.

Chapter 2

Network on Chip Background

NoC consists of three main components i.e., routers, links and Network Interfaces (NIs). Figure 2.1 shows NoC components for 3×3 2D mesh topology network. The data is transmitted from router to router through bidirectional links and each router in NoC is connected to one IP core via the network interface.

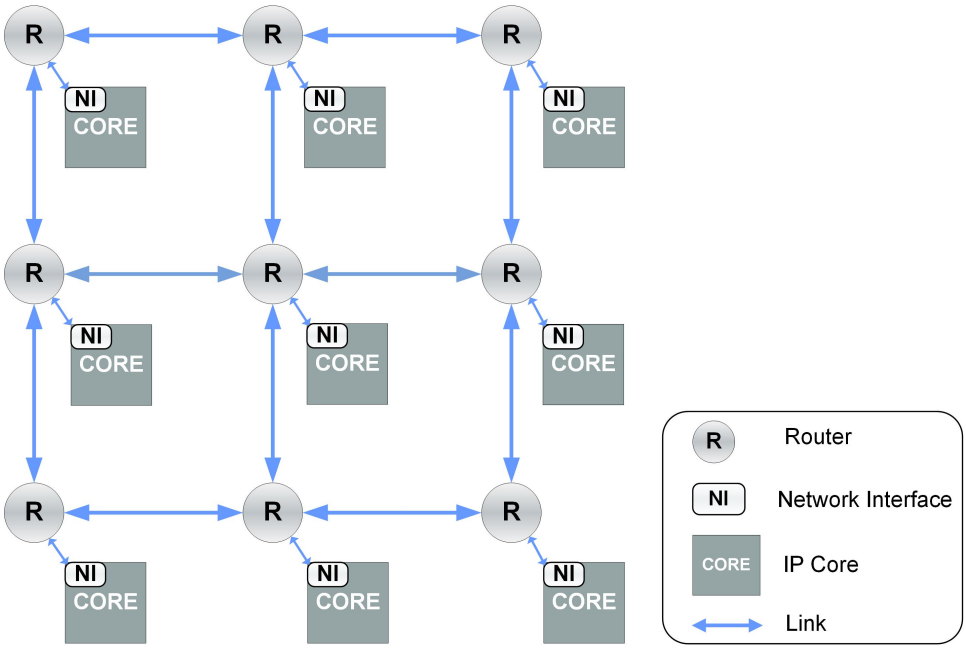

Figure 2.1: 3×3 2D mesh NoC components

- **Router**

  The router is the most important component in any network and represents the backbone of communication in NoC. Routers in the center nodes in NoC have

five ports: East, West, North, South and a Local. The first four ports (East, West, North and South) are connected to the adjacent routers and the Local port is connected to the IP core as shown in Fig. 2.2. The main role of the router in NoC is to receive packets and determine the direction that each packet should take towards the destination [11]. This is determined by using routing protocols that are already implemented inside the router.

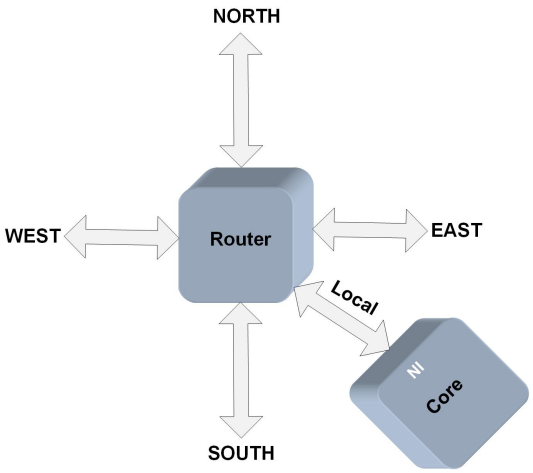

Figure 2.2: Router connections in NoC

- **Links**

  The routers in NoC are connected to each other through channels called links. The links also connect each router to one local IP core as shown in Fig. 2.1. The links represent the main way to transmit the packets from router to router or from routers to IP cores. In NoC, the links are bidirectional which provide full-duplex communication between the connected nodes [12] [13].

- **Network Interface**

  The network interface is the part of NoC that provides the connection between the routers and IP cores. This interface coordinates the data moving between routers and IP cores in NoC via a full-duplex communication link. When the IP core transmits the data to the network, the data will be first received in the network interface, which packetizes the data and adds destination information to the packets. After that, the routers will read the packet information to route it to the right direction based on the routing algorithm policy in the router [12].

2.1 Network Topology

Network topology is the connecting and arranging of a number of nodes and links in the network. In NoC, several factors, i.e., performance, scalability, power consumption, design complexity, etc., rely on the topology [14]. Moreover, network topology helps to determine the number of nodes inside NoC and decides the length of the links between the nodes. The topology in NoC can be classified into two main categories, i.e., direct and indirect topology. In the direct topology, each router is connected directly to its neighbour routers and to its local IP core [12] [8]. In indirect topology, however, not all routers are connected to the IP cores because some of these routers are used only to propagate the packets inside the network. Also, in indirect topology, the number of routers is larger than the IP cores. In contrast, the number of routers and cores in the direct topology are the same. Examples of the direct topology are mesh, torus, and ring along with fat-tree network topology represents one of the most common examples of indirect topology. These examples of direct and indirect topology is the most popular commercial types in NoC [15] which will be demonstrated in detail in the section below.

2.1.1 Mesh Topology

Mesh topology is the most widely used and simplest direct topology in NoC which consists of $R \times C$ nodes, where R is the number of rows and C is the number of columns in the network. Each router in the nodes is connected to its IP core as shown in Fig. 2.3. Each node in this topology is represented by one IP core and one router. Furthermore, in this topology, the number of cores and the numbers of routers are the same. Figure 2.3 also shows that the intermediate nodes are connected to four neighbouring nodes, the nodes on the edges are connected to three nodes, and the corner nodes are connected to only two neighbouring nodes. The advantages of mesh topology are that it is easy to implement in NoC since it assumes the same length between the nodes [16]. Because of these advantage in mesh topology, we have applied and tested the proposed routing algorithm in 2D mesh NoC.

2.1.2 Torus Topology

Torus topology is also a type of direct network topology and is similar to mesh topology [7]. However, Torus topology is more complex than mesh topology in terms

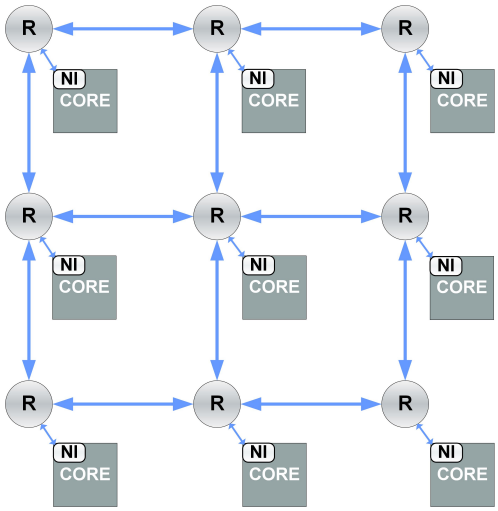

Figure 2.3: Mesh Topology

of implementation because it uses wrap-around links to connect the routers at the edges with the other routers at the opposite edges as shown in Fig. 2.4. The main disadvantage in Torus topology is that the wrap-around links can result in increase packet propagation delay. In addition, the repeaters which are required through the long links increase the area needed in the network architecture [17].

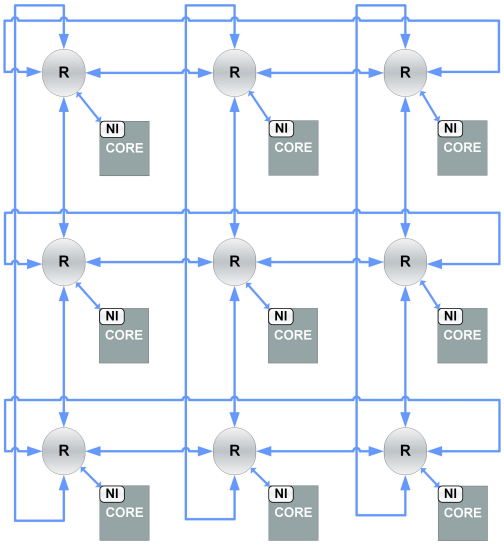

Figure 2.4: Torus Topology

2.1.3 Ring Topology

Direct topology can also be in a ring scheme in which every node is connected to only two nodes. In this topology, the first node and last node are connected to each other to make a ring, as depicted in Fig. 2.5. The packets in the ring topology traverse through all the nodes that precede the destination node during their routing [18]. This topology is feasible for small networks, but if the diameter of the ring increases, there is an increase in the number of hops that packets need to pass to reach the destination.

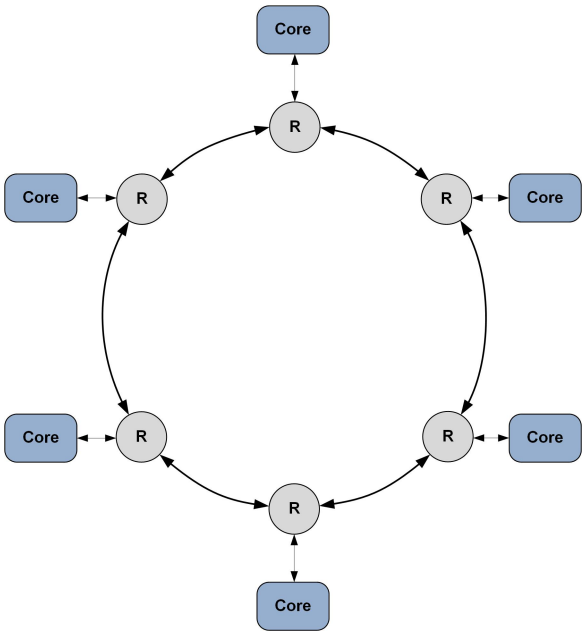

Figure 2.5: Ring Topology

2.1.4 Fat-Tree Topology

This is an indirect network topology and, as the name suggests, it depicts a tree as shown in Fig. 2.6. In fat-tree network topology, there is a root node which expands into the branch nodes, which are also known as child leaf nodes. The cores are only connected to the child leaf nodes. The number of downwards links to the leaf nodes is the same as that of upwards links to the root node [19].

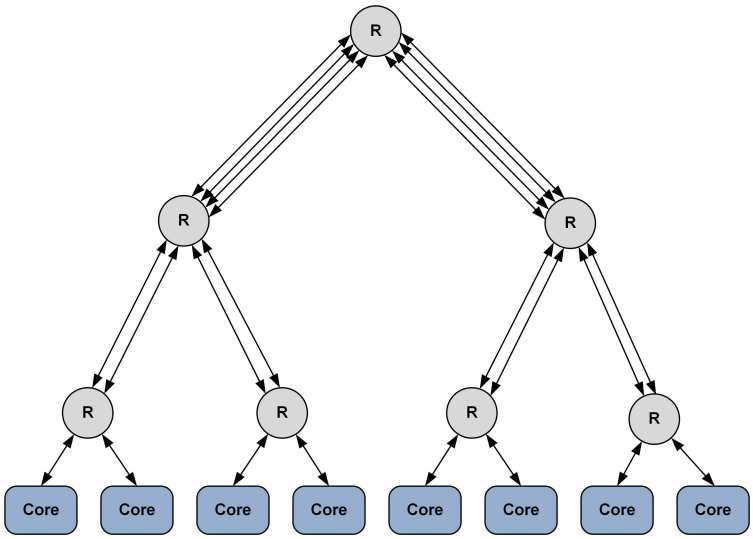

Figure 2.6: Fat-Tree Topology

2.2 Switching technique

The switching technique defines the approach that determines the data follow through the routers in NoC. Circuit switching and packet switching are the most common switching techniques in NoC [20]. There is only one category of circuit switching technique, but the packet switching technique has been classified into three categories: store-and-forward switching (packet switching), virtual cut-through switching and wormhole switching.

2.2.1 Circuit Switching Technique

The circuit switching technique is a type of NoC switching in which the communication between source and destination takes place through a dedicated path [21]. This path will be reserved prior to the start of communication between the source and the destination. Once this connection is established, all the data are transferred using this dedicated path until the communication is completed. The main advantages of this switching technique are that it has less latency and guaranteed bandwidths; however, it takes a long time to initiate the connection, there is no efficient channel utilization, the network throughput is low.

2.2.2 Packet Switching

This type of switching technique is based on transmitting message in the form of packets as shown in Fig. 2.7. It is also considered as the enhanced approach of the circuit switching technique. In packet switching, all the data are divided into chunks or packets. Each packet in this technique is independent of the other packets and can take any suitable path to reach its destination [22]. At the destination, the packets arrive in random times depending on the path taken by each packet. It is the responsibility of the destination node to rearrange the packets into their right order to form the original message. The main advantages of packet switching are: improved channel utilization and network throughput.

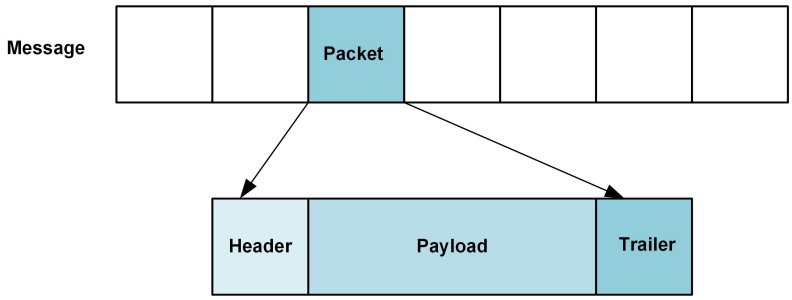

Figure 2.7: Packet Structure

- **Store-and-Forward**

  According to [17], it is also can be named as packet switching. It is the simplest switching mechanism and, as the name suggests, it stores and forwards the packets. This mechanism stores the entire packet in its place in the input buffer until the time of routing [23]. Then, the packet will be forwarded to the adjacent router if this router has enough buffer space to store the entire packet. The prominent advantage of this approach is that it is completely free of deadlock and livelock problems. This advantage lead to increase the number of packets that arrive successfully to their destinations which results in high throughput and efficiency. In addition, Store and Forward switching is very simple to implement but it requires large buffer space in NoC as well as the average latency is high. Store and forward switching has been utilized in our work for FD routing algorithm implementation.

- **Virtual Cut-Through**

  This approach addresses the problem of latency in the store-and-forward switching mechanism by decreasing the packet time latency at each router along the routing path. In this mechanism, routing of the entire packet starts as soon as the downstream router receives the packet header [24] [25]. In other words, the current router can make a decision to route the packet as next router receives the packet header, which is relatively small compared to the size of the whole packet. However, if the buffer of the next router is not free, Virtual Cut-Through could require a large buffer to store the entire packet and this is the main disadvantage of the approach.

- **Wormhole**

  This is similar to the virtual-cut-through approach, but it depends on chunks called FLITs (FLow-control digITs) rather than packets. The buffer size problem in store-and-forward switching and virtual-cut-through switching has been resolved using wormhole switching technique [22] [26]. In this approach, the packet is divided into a set of FLITs [27]. The first FLIT portion reserves the channel of the next router and is known as the header FLIT. The second FLIT (the body of the FLIT), follows the header to the reserved channel and is known as the payload FLIT. The last FLIT which releases the reserved channel, is known as the tail FLIT. Similar to Virtual Cut-Through, the router does not need to wait until the whole packet is received, and the routing decision is made on the basis of the header FLIT. The main advantage of this switching approach is that it does not require much buffer space; however, its is more vulnerable to deadlock and livelock issues in NoC.

2.3 Routing Algorithms

Routing algorithms determine the paths that a packet should take during its journey from the source to the destination. In NoC, the routing algorithms have an essential role to decide the approach for moving the packets between the nodes inside the network. Also, routing algorithms have a direct effect on NoC's performance,

throughput and latency. There are several challenges that may appear with routing algorithms such as deadlock, livelock and starvation. A deadlock is a situation that arises when all the packets routed in the network must wait for each other to release specific channels in a cyclic manner [28]. Consequently, none of the packets can move further and will be kept in a deadlock situation. The problem of livelock occurs when all the packets are moving around the destination but cannot reach it [29]. A starvation problem can happen when certain priorities are assigned to particular packets in NoC. High-priority packets can reach the destination and reserve the resources, making them unreachable by low-priority packets in the network. In the past two decades, many routing algorithms have been proposed to avoid these challenges and improve the performance as well as develop the scalability in NoC. NoC routing algorithms can be classified based on our work as shown in Fig. 2.8. Figure 2.8 illustrates that based on the number of destinations, NoC routing algorithms can be classified into unicast and multicast routing [30]. Also, On the basis of path routing decision, unicast routing can be: source routing (decides the whole path in the source node prior routing the packet) or distributed routing (determine the path during routing the packet). Moreover, according to the adaptability, distributed routing is also categorized into either deterministic (non-adaptive), partial and full-adaptive routing algorithm [31]. The proposed routing algorithm FD is considered as full-adaptive routing algorithm which will be explained in details in Chapter 3.

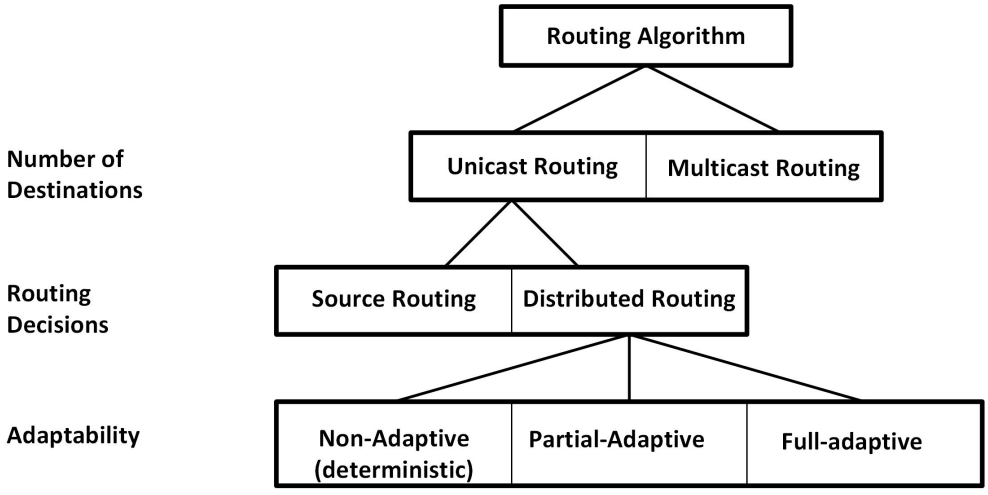

Figure 2.8: NoC Routing Algorithms Classification

2.3.1 Unicast and Multicast Routing Algorithm

Routing algorithms in NoC can be unicast or multicast. Unicast routing algorithms can be operated as one-to-one, whereby the communication will be established between only one source and one destination [32]. However, multicast NoC routing algorithms can establish the connection between one source and many destinations [33]. In this case, the routing algorithms send the packet from one source to multiple destinations in the network [34].

2.3.2 Source and Distributed Routing

The unicast routing algorithm can be a source or distributed routing. In the source routing algorithms, the routing decision and the path are decided at the source node before routing the packet. Afterwards, the path and routing information will be embedded into the packet, which causes increase the packet size [35] [36]. In this case, the packet will not be allowed to change the path during the routing time. Meanwhile, the routing paths in the distributed routing algorithms are decided during the time of routing the packets between the nodes. In the distributed routing algorithm, the packets only need to hold the destination information during their routing in the network, which results in less packet size compared to source routing [37]. In addition, distributed routing is comparatively easier to be implemented in the network, and it has the ability to detect and avoid failed and congested paths.

2.3.3 Deterministic and Adaptive Routing

Distributed routing algorithm can be classified as deterministic (non-adaptive) or adaptive routing algorithms [38]. Deterministic routing algorithms route a packet through only one path based on its destination information without considering the network conditions [39]. On the other hand, in adaptive routing algorithms, the packet can be routed through any paths based on network conditions and the used routing algorithm policy. Therefore, the routing paths in adaptive routing algorithms are not determined only by the source and destination information but they also depend on the network condition at the time of routing the packet. Adaptive routing has an ability to avoid congested and fault paths in the network. However, the deterministic routing can not avoid or change the routing path even if it is congested.

2.4 Classifying NoC routing algorithms

Routing algorithms in NoC can be classified based on Adaptability. The Adaptability is the ability of routing algorithm to change its routing decisions based on the network conditions. Based on adaptability, routing algorithms can be classified into three types: non-adaptive (deterministic), partial-adaptive and full-adaptive [40]. The non-adaptive routing algorithm has only one potential path to route a packet through it from the source to the destination. In non-adaptive routing algorithm, the packets cannot change their routing paths based on network conditions. One of the most common non-adaptive routing algorithms in NoC is XY routing which was proposed by Zhang et al. [41]. Partial-adaptive routing algorithms allow some paths and prohibit others to route packets in the network. Network conditions are not always considered in the partial-adaptive routing algorithms. An example of this kind of routing algorithm is the DyXY routing algorithm which was proposed by Li et al. [42]. In full-adaptive routing algorithms, the whole paths in the network can be used to route the packets to their destinations. This routing algorithm always considers the network conditions in each node during routing the packets to the destinations. Our proposed FD routing algorithm is a perfect example of a full adaptive routing algorithm which will be elaborated in Chapter 3.

2.4.1 Non-Adaptive Routing Algorithm

Non-adaptive or deterministic routing algorithm is a routing algorithm which allows only one potential path for routing a packet. This path is determined based on source and destination information of the packet. Also, non-adaptive routing algorithm does not consider the networks condition which means that even if the selected path was congested, the routing algorithm will not change the path. As a result, the packet in this path will be blocked and can not reach the destination, then will be considered as packet lost. For this reason, the throughput and efficiency of non-adaptive routing algorithms is very low. For example, XY routing algorithm routes the packet in only one potential path between X and Y directions. The packet will travel first through the X direction (in the same row of the source node) and then to Y direction when it reaches the column of the destination. Figure 2.9 shows how the packet can be routed using XY routing algorithm between the source node (0,3) and destination node (2,0) in 5×5 2D-mesh NoC.

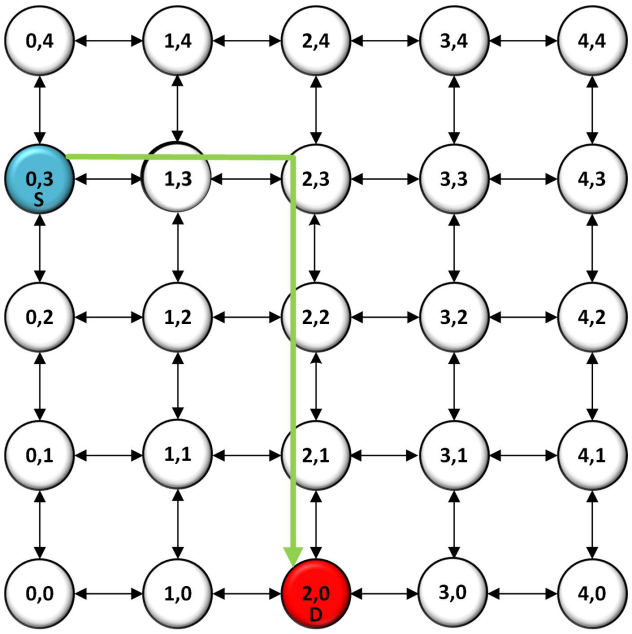

Figure 2.9: Routing a packet using XY routing algorithm

First, the packet traverses from the source node (0,3) horizontally into the node (1,3) toward the destination. Then, the routing algorithm will check if the y-coordinate of the current source node (1,3) is the same as the y-coordinate of the destination node (2,0). Because the y-coordinates of both nodes are not the same, the packet will continue moving horizontally to the next node (2,3). Then, the routing will again check the y-coordinates of the new current source node (2,3) and destination node (2,0). Because the y-coordinates of the current source node and the destination node are identical, the packet will change its movement to moving vertically (without changing its path) until reaching the destination node (2,0). If node (2,2) or node (2,1) are congested or the link between (2,1) and (2,0) fails, changing the route will not be permitted in this routing algorithm. As a result, the packet will not reach the destination and will be considered as a lost packet which will badly affect the resultant throughput in NoC.

2.4.2 Partial-Adaptive Routing Algorithm

Partial-adaptive routing algorithm allows routing a packet through more optional paths than in deterministic. However, this type of routing algorithm prohibits some paths in the network based on the routing algorithm policy. Also, partial-adaptive

routing algorithm considers networks conditions in some stages of routing a packet and ignore them in some cases based on the routing algorithm policy. For example, in the partial-adaptive routing algorithm DyXY, the packet will be routed based on the network conditions until it arrives to one of the nodes whose the same X or Y coordinate as the destination node. At that point, the packet will be routed horizontally or vertically toward the destination in the same path regardless of the network status of that path. This can be summarized that DyXY behaves as a full-adaptive routing algorithm but changes to non-adaptive routing behaviour when the X or Y coordinates of current source node and destination nodes are the same. Figure 2.10 illustrates an example of a DyXY routing algorithm by assuming node (1,3) as a source node and node (3,2) as a destination node.

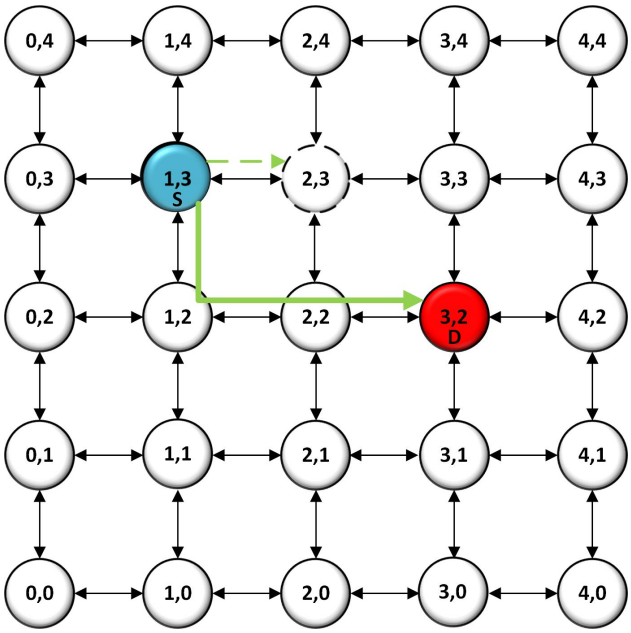

Figure 2.10: Routing a packet using DyXY routing algorithm

First, the routing starts comparing the X and Y coordinates of node (1,3) and node (3,2). Because the X and Y coordinates of (3,1) and (3,2) nodes are not the same, the packet will be moved to node (2,3) or node (1,2). If we assume that the buffer occupancy (the number of packets in the buffer) in node (1,2) is less than in (2,3) then the packet in node (1,3) will be moved to node (1,2). Then, node (1,2) will become the current source node which holds the packet. Because node (1,2) has the same X-coordinate as destination node (3,2), the routing will keep moving the packet horizontally (to node (1,2) then (2,2)) until reaching the destination node (3,2). Even

if (1,2) or (2,2) is congested, the routing will not be able to change the path. This would mean that the DyXY routing algorithm has selected the wrong path. At that point, the congestion can not be avoided and may lead to blocking this packet. This issue represents the main drawback in DyXY routing thereby decreasing the average throughput and efficiency in the network.

2.4.3 Full-Adaptive Routing Algorithm

When the all paths are not prohibited to route a packet and selecting the path is completely relying on network conditions, that means this type of routing algorithm is a full-adaptive. The full-adaptive algorithms provide more optional paths to route a packet than non and partial-adaptive routing algorithm in the network. By increasing the number of optional paths to route a packet, this will decrease the number of probability to loss the packet before reaching the destination. This because the full-adaptive routing helps to avoid the congested and failure paths. As a result, full-adaptive routing algorithms raise the average throughput and efficiency in NoC as well as decrease the average packet loss in the network. However, the latency in this type of routing algorithms is rationally high comparing with non and partial-adaptive routing algorithms. Our proposed FD routing algorithm is an example of full-adaptive routing algorithm which routes the packets based on the distance between the source and destination nodes as well as based on network conditions. FD routing algorithm methodology will be explained with an example in Chapter 3. Also, the FD routing algorithm's simulation and results will be analyzed and compared with the partial-adaptive routing algorithm DyXY in Chapter 4.

Chapter 3

Proposed Routing Algorithm

3.1 Introduction

The proposed routing algorithm is named Force-Directed (FD) algorithm. It is a unicast routing algorithm since it sends a packet from one source to one destination. The unicast FD routing also is classified as distributed routing algorithm because it takes the routing decision during the time of moving a packet from node to node. Also, it is full-adaptive routing algorithm because it considers the network conditions for routing a packet. The FD routing algorithm routes a packet based on two main factors. The first factor is the location of the destination node according to the source node which is summarized in sixteen cases in Table 3.1. These cases help to define the direction that should be used to route the packet from the source node to one of the neighbouring nodes. The second factor is the network conditions which relies on the input buffer status of the surrounding neighbour nodes of the source node. The FD routing algorithm improved the network throughput, efficiency and decreased the average packet loss in the network compared to DyXY routing. This because DyXY routing provides partial adaptability and does not allow the packet to change its path when Y or X coordinates of the current source node and the destination node are the same even if that path is congested as it is explained in Section 2.5.2. However, FD routing allows the packet to change its path based on the current network condition during the time of routing. For this reason, FD routing avoids any congested paths during routing the packet which ensure that packet will not be blocked or lost before reaching the destination. As a result, the number of blocked packets will be decreased in FD routing and increase the throughput and efficiency in the network.

3.2 Router Architecture for FD Routing Algorithm

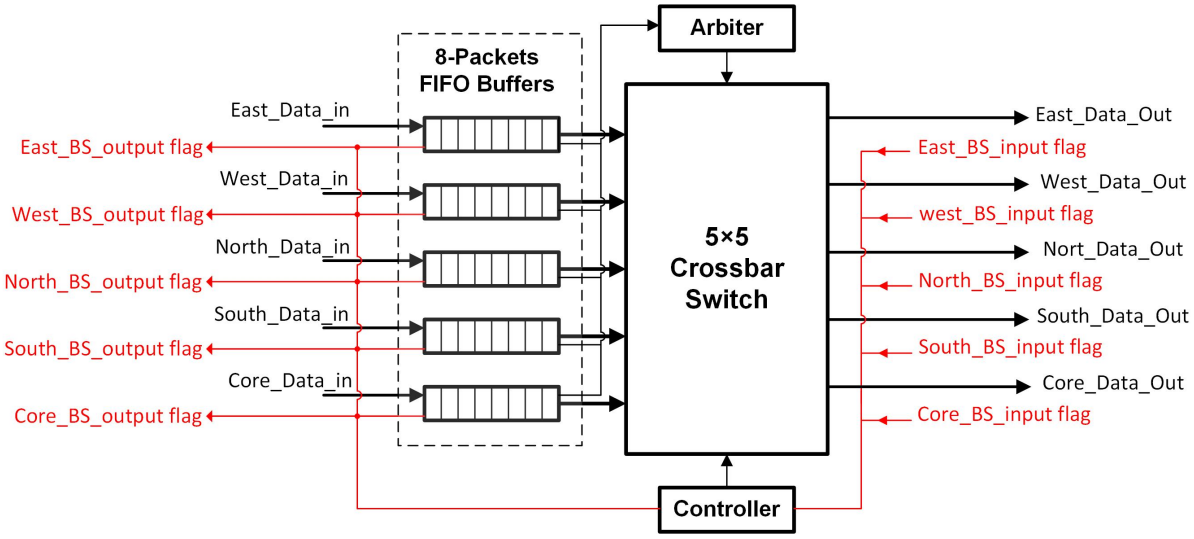

Figure 3.1: Router Architecture for FD Routing Algorithm

The NoC router architecture which is used for FD routing algorithm is shown in Fig. 3.1. The router contains five data input ports (East_Data_in, West_Data_in, North_Data_in, South_Data_in, and Core_Data_in) and five data output ports (East _Data_out, West_Data_out, North_Data_out, South_Data_out and core_Data_out). Also, the router has five First-in First-out (FIFO) input buffers, a crossbar switch circuit, a controller, and an arbiter. The assumed size of FIFO input buffer in each port is eight packets. The full and busy buffer status are represented in one flag named Buffer Status (BS) flag. The BS flag shows the status of the input buffers of the routers in current source node and downstream node if they are available to receive the packets or not. BS flag can be an output to the upstream routers to update them with the status of the router in the current source node. For example, East_BS_output flag updates the upstream router with the status of the east input buffer in the current source node as shown in Fig. 3.1. Also, the BS flag can be input to the router of the current source node to update it with the downstream router's buffers status. The controller in FD router determines the routing direction for the incoming packet from the downstream to the upstream router based on FD routing algorithm. For example, when the router in the current source node receive a packet from one of the upstream nodes, the controller will chose the best optional directions to route the packet toward one of neighboring nodes. This decision is based on the packet's destination

information, FD routing algorithm policy and network condition. The arbiter is that part in FD router which receives the requests from the input buffers and grants one of the output ports based on controller decision. [43]. There are different types of arbitration protocols which are explained in detail in [6]. Equal-priority logic arbiter is used in our work because we have assumed that all packets have the same priority. Also, the router consists of 5×5 crossbar which provides the connection between the input data port to the outputs data port which both are selected by the arbiter.

3.3 FD Routing Algorithm Methodology

The proposed FD routing algorithm relies on the location of the destination node according to the source node as well as it considers the network condition during routing a packet. The FD checks the horizontal and vertical distances between the source and destination nodes and routes the packet horizontally when the horizontal distance is greater than the vertical distance and vice versa. If the horizontal and vertical distances between the source and destination node are the same, FD will route the packet horizontally or vertically based on the network condition in the neighbouring nodes of the source node. To explain FD routing, 5×5 2D mesh NoC as shown in Fig. 3.2 is considered.

Figure 3.2 demonstrates that each node in 5×5 2D mesh NoC has X and Y coordinates (X,Y). The X coordinate of the nodes increases when we move to the right from node to node in the network. Also, the Y coordinate of the nodes increases when moving from the bottom node to the top node. The communication link between the nodes is a bidirectional link which allows each node to send and receive the packets at the same time (duplex communications). The FD routing algorithm routes the Head of Line (HoL) packet in the input buffers. HoL is the first packet in FIFO input buffers inside each node. FD routing transmits the HoL packet through a number of nodes to the destination node or directly to the destination if the source node is connected to the destination node. When the HoL packet arrives at the destination, the router inside the destination node will move the packet to its local core as a final terminal. The vertical and horizontal distance is the number of links between the nodes. The horizontal distance between two nodes (ΔX) can be defined by subtracting X-coordinates of the nodes. Also, the difference between two node's Y coordinates represents the vertical distance (ΔY). By calculating the summation

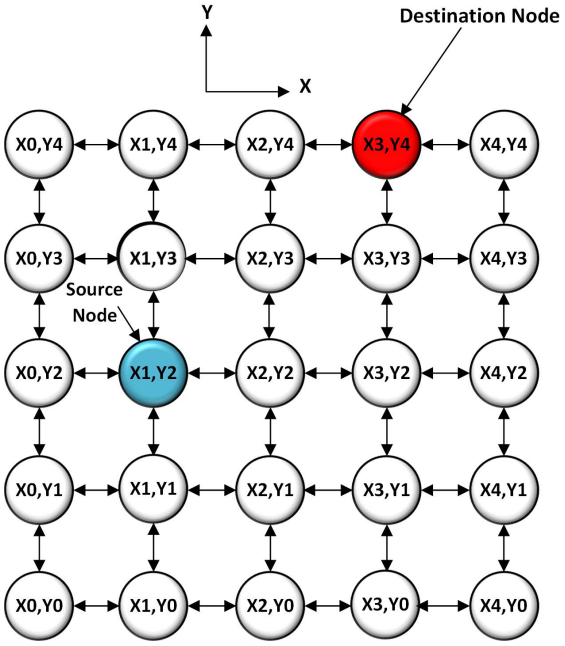

Figure 3.2: 5×5 2D mesh NoC Architecture

of ΔX and ΔY, we can find the minimum distance between the two nodes. In Fig. 3.2, the horizontal and vertical distance between the source node (X1,Y2) and the destination node (X3,Y4) can be calculated as following:

Horizontal distance:

$$\Delta X = X3 - X1 \tag{3.1}$$

Vertical distance:

$$\Delta Y = Y4 - Y2 \tag{3.2}$$

From equations (3.1) and (3.2), the minimum distance (D) between the source node (X1,Y2) and the destination node (X3,Y4) can be found from the formula:

$$D = |\Delta X| + |\Delta Y| \tag{3.3}$$

Table 3.1 shows the cases of routing a packet (horizontally or vertically) in FD routing algorithm based on the destination location according to the source node.

Case#	Destination location according to source node	First option	Second option	Third option
1	$\Delta X > 0$ and $\Delta Y = 0$	East Node	North Node	South Node
2	$\|\Delta X > \Delta Y\|$ and $\Delta X > 0$ and $\Delta Y > 0$	East Node	North Node	South Node
3	$\|\Delta X = \Delta Y\|$ and $\Delta X > 0$ and $\Delta Y > 0$	North Node	East Node	South Node
4	$\|\Delta X < \Delta Y\|$ and $\Delta X > 0$ and $\Delta Y > 0$	North Node	East Node	West Node
5	$\Delta X = 0$ and $\Delta Y > 0$	North Node	East Node	West Node
6	$\|\Delta X < \Delta Y\|$ and $\Delta X < 0$ and $\Delta Y > 0$	North Node	West Node	East Node
7	$\|\Delta X = \Delta Y\|$ and $\Delta X < 0$ and $\Delta Y > 0$	North Node	West Node	East Node
8	$\|\Delta X > \Delta Y\|$ and $\Delta X < 0$ and $\Delta Y > 0$	West Node	North Node	South Node
9	$\Delta X < 0$ and $\Delta Y = 0$	West Node	North Node	South Node
10	$\|\Delta X > \Delta Y\|$ and $\Delta X < 0$ and $\Delta Y < 0$	West Node	South Node	North Node
11	$\|\Delta X = \Delta Y\|$ and $\Delta X < 0$ and $\Delta Y < 0$	West Node	South Node	East Node
12	$\|\Delta X < \Delta Y\|$ and $\Delta X < 0$ and $\Delta Y < 0$	South Node	West Node	East Node
13	$\Delta X = 0$ and $\Delta Y < 0$	South Node	West Node	East Node
14	$\|\Delta X < \Delta Y\|$ and $\Delta X < 0$ and $\Delta Y > 0$	South Node	West Node	East Node
15	$\|\Delta X = \Delta Y\|$ and $\Delta X < 0$ and $\Delta Y > 0$	South Node	East Node	West Node
16	$\|\Delta X > \Delta Y\|$ and $\Delta X < 0$ and $\Delta Y > 0$	East Node	South Node	North Node

Table 3.1: Routing Cases in FD Routing Algorithm

3.4 Packet Routing Scenario

As mentioned above, the FD routing algorithm mainly depends on the locations of the destination nodes according to source nodes in 2D mesh NoC. These locations are summarized in sixteen cases in Table 3.1. When a packet in specific node needs to be routed, FD routing will check the destination information of this packet. FD routing then calculate the horizontal distance (ΔX) and vertical distance (ΔY) between the source and destination nodes, after that decides which case in Table 3.1 can be used to route the packet. Each case provides different potential directions (East node, West node, North node or South node) to route the packet. These directions are organized as first, second, third optional directions in each case. FD routing attempts to route the packet through the first node direction if its input buffer is available. Otherwise, FD routing will check the input buffer of the other two nodes directions which are provided in the same case. If all the options in one case are not available, FD routing will keep the packet in the same node in the current simulation time and check the three options again in the next simulation time until the packet can be routed.

An example for routing a packet from source to destination node is shown in Fig. 3.3 where node (1,1) is a source node and node (3,4) is a destination node. FD routing will check the location of the destination node (3,4) according to the source node (1,1) in the 5×5 2D mesh NoC. This match case number 4 in Table 3.1.

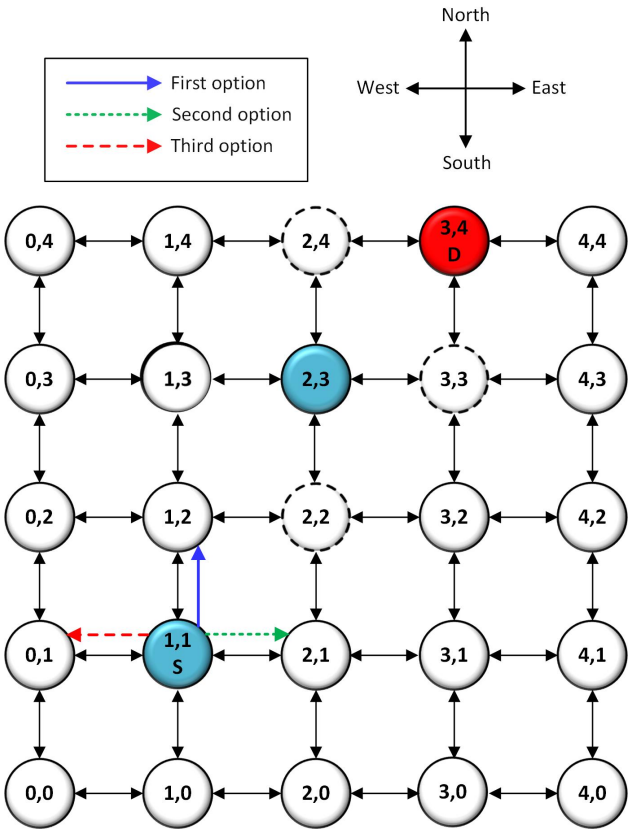

Figure 3.3: Routing a packet from node (1,1) toward the destination node (3,4)

Because the vertical distance (ΔY) is greater than the horizontal distance (ΔX), the first optional direction for routing the packet is the north node (1,2). This is because the FD routing algorithm is based on decreasing the greatest horizontal or vertical distance between the source and destination nodes. If the input buffer of the north node (1,2) is not available, FD will check the input buffer of the second optional direction which is the east node (2,1). If the input buffer of the east node (2,1) is available, the packet will be sent to the east node; otherwise, FD routing will do the same check with the input buffer of the west node (0,1). If all three options are not available, FD routing will keep the packet in the same node and repeat the check at the next simulation time until it is able to route the packet. This could increase the packet latency, but at the end the packet will be routed to its destination. For this

reason, FD routing provides high throughput with relatively high latency.

Assuming in the above figure that the input buffer of the north node (1,2) is available, the packet will be routed to node (1,2) toward the destination. Then node (1,2) has become the new source node which holds the packet needed to be routed to the destination node (3,4) as shown in Fig. 3.4. FD routing will repeatedly check the location of the destination node (3,4) according to the new current source node (1,2) which is the same as shown in case number 3 in Table 3.1. In this case, the horizontal distance (ΔX) is equal to the vertical distance (ΔY). For this reason, routing the packet through the east node (2,2) or north node (1,3) has the same significance, but FD routing will choose the node with a less busy input buffer. If the input buffer of the east and north nodes is not available, FD routing will check the third optional direction which is the south node (1,1). The same scenario as in the previous case will be taken if the three nodes are not available.

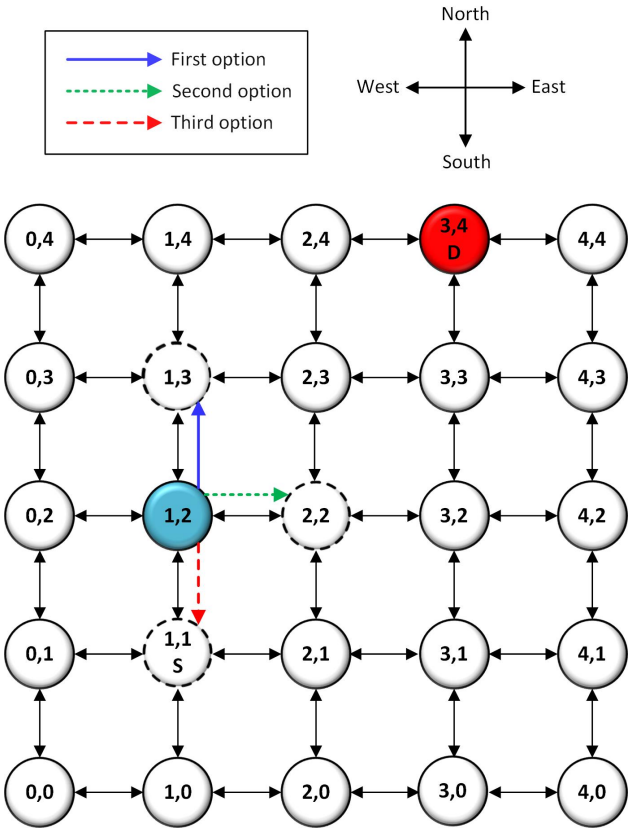

Figure 3.4: Routing a packet from node (1,2) toward the destination node (3,4)

Assuming that the input buffer of the north node (1,3) has a fewer number of packets than the input buffer of the east node (2,2), FD will route the packet to the north node (1,3). In this case, the new source node which holds the packet will be node (1,3) as shown bellow in Fig. 3.5. FD will then check the location of the destination node (3,4) according to the new source node (1,3) which is the same as case 2 in Table 3.1. In this case, because the horizontal distance (ΔX) between the new source node (1,3) and destination node (3,4) is greater than the vertical distance (ΔY), the FD routing algorithm will prefer to route the packet horizontally to decrease the horizontal distance between the source and destination nodes. For this reason, the first option to route the packet is to the east node (2,3). If the input buffer of the east node (2,3) is busy or full, FD routing will check the other two routing options using the same procedure as in the previous cases.

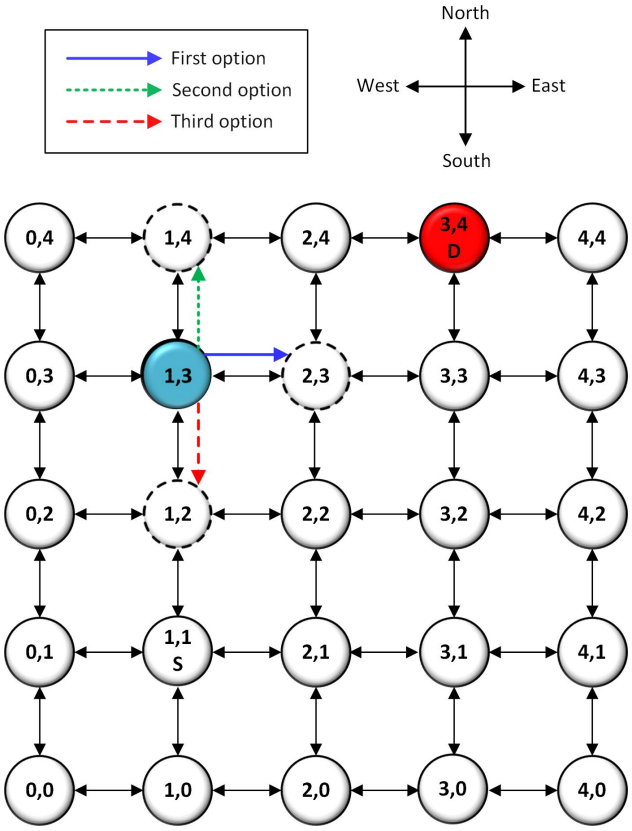

Figure 3.5: Routing a packet from node (1,3) toward the destination node (3,4)

If the input buffer of the east node (2,3) is available, FD routing will send the packet horizontally to that node. As a result, node (2,3) will be the new current source node as shown in Fig. 3.6. From Fig. 3.6, the vertical and horizontal distance between the new source node (2,3) and the destination node (3,4) is the same. Then, the FD routing algorithm will route the packet based on case 3 in Table 3.1. This case will be the same as the case in Fig. 3.4. FD routing will do the same check to find the proper direction to route the packet.

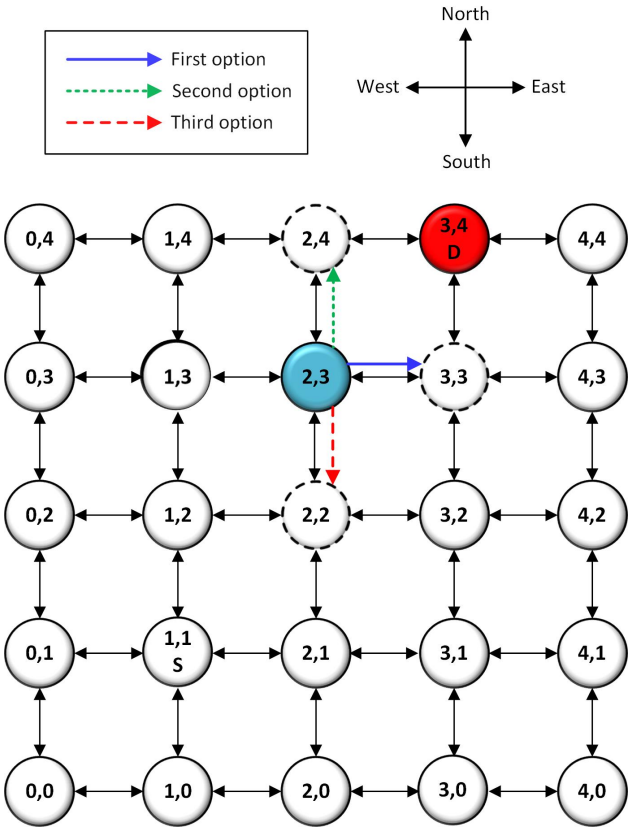

Figure 3.6: Routing a packet from node (2,3) toward the destination node (3,4)

Presuming that the input buffer of node (2,4) has more packets than the input buffer of node (3,3), the packet will be routed to node (3,3), and node (3,3) will become the new current source node as shown in Fig. 3.7. Case number 5 in Table 3.1 matches the situation of the destination node (3,4) and the new current source node (3,3) in Fig. 3.7. In this case, the vertical distance ($\Delta Y = 1$) is greater than the horizontal distance ($\Delta X = 0$). On that basis, one of three optional node directions listed in case 5 in Table 3.1 can be used to route the packet. FD will do the same check that has been done in the previous cases.

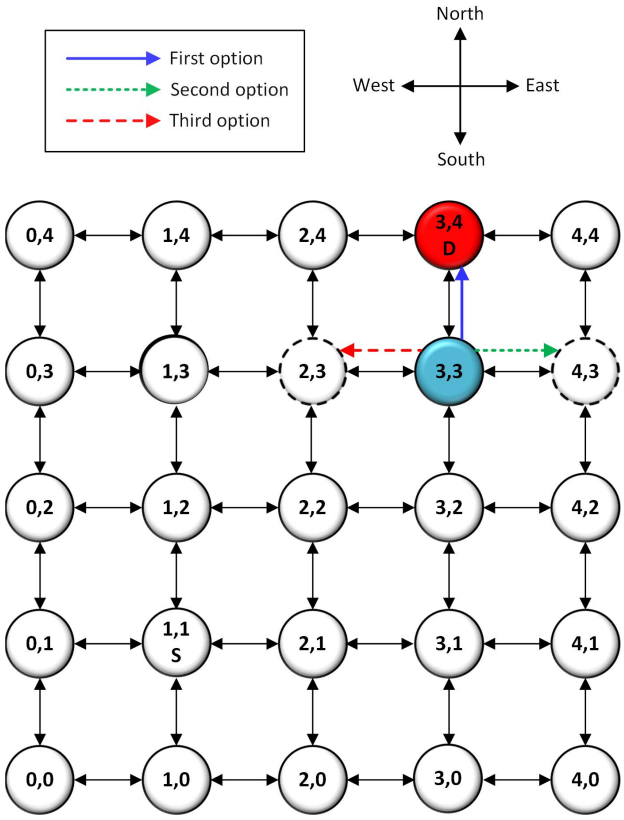

Figure 3.7: Routing a packet from node (3,3) toward the destination node (3,4)

If the input buffer in the north node (3,4) is available, the packet will be routed to node (3,4) which is the final terminal node of the packet. At this point, the router inside node (3,4) will move the packet to the core and, then the packet has arrived at its destination. Figure 3.8 shows the entire path (Undashed line) that is taken to route the packet from the source node (1,1) to the destination node (3,4) in the previous example. Also, the dashed lines number 1 and 2 show the other two potential paths that can be used to route a packet from node (1,1) to node (3,4). In addition, there are still many optional paths to route a packet from node (1,1) to node (3,4) but we only demonstrate the three paths as examples.

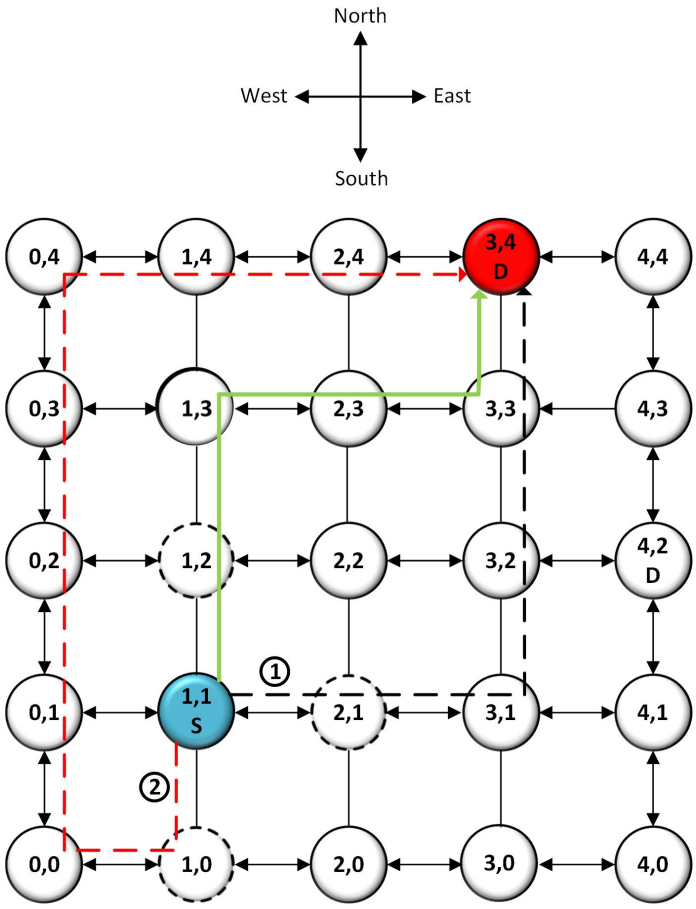

Figure 3.8: The three potential paths of routing a packet form the source node (1,1) to the destination node (3,4) using FD routing algorithm

It is clear from the above example that the FD routing algorithm is very reliable in terms of the throughput rate and efficiency because it provides a lot of optional paths to route packet with fewer restrictions. Also, the full adaptability of the FD routing algorithm allows the packet to avoid the congested links in the network. All these factors contribute to the high throughput and efficiency rates as well as decrease the average of packet lost in the FD routing algorithm when compared to the DyXY routing algorithm. However, allowing a packet to move in different paths toward the destination without banning some of these paths will cause a relative increase in its latency but it helps to ensure delivering the packet with less possibility of blocking or losing it.

3.5 FD Routing Pseudo Code

The methodology above of the FD routing algorithm can be summarized in the following Pseudo code:

Algorithm 1 FD Algorithm

Input:

$SourceNode\ (Sx, Sy)$

$DestinationNode\ (Dx, Dy)$

Output: Path

1: $\Delta x\ = Dx - Sx$

2: $\Delta y\ = Dy - Sy$

3: $L = [L1\ \ L2\ \ L3]$ See Table 3.2

4: Path = Route (L)

5: **return** Path

Case	$L1$	$L2$	$L3$
$\Delta X > 0$	1		
$\Delta X = 0$	2		
$\Delta X < 0$	3		
$\Delta Y > 0$		1	
$\Delta Y = 0$		2	
$\Delta Y < 0$		3	
$\lvert\Delta X > \Delta Y\rvert$			1
$\lvert\Delta X < \Delta Y\rvert$			2
$\lvert\Delta X = \Delta Y\rvert$			3

Table 3.2: L Vector Values in FD Routing Pseudo Code

L vector in Algorithm 1 helps to define the destination node location of the packet according to its source node in NoC. The value of L vector is represented by $L1$, $L2$ and $L3$ as shown in Table 3.2. Each L vector value represents one case from the cases in Table 3.1. Based on L vector value, FD routing algorithm will decide the proper direction (East or West or North or South) that should be selected to route the packet based on the equivalent case in Table 3.1. From Fig. 3.9, the location of the destination node (4,3) according to the source node (2,2) is represented by the vector value $L = [\ 1\ 1\ 1\ ]$. This vector value is equivalent to case number 2 ($\Delta X > 0$,

$\Delta Y > 0$, $|\Delta X > \Delta Y|$) in Table 3.1. Based on this case, the packet will be routed to the east node (3,2) or north node (2,3) or south node (2,1) with respecting to the order and their availability to receive the packet from the source node (2,2). As we mentioned in the previous section, when the packet move to the next node, FD routing algorithm will repeat the same procedure until reaching to its destination.

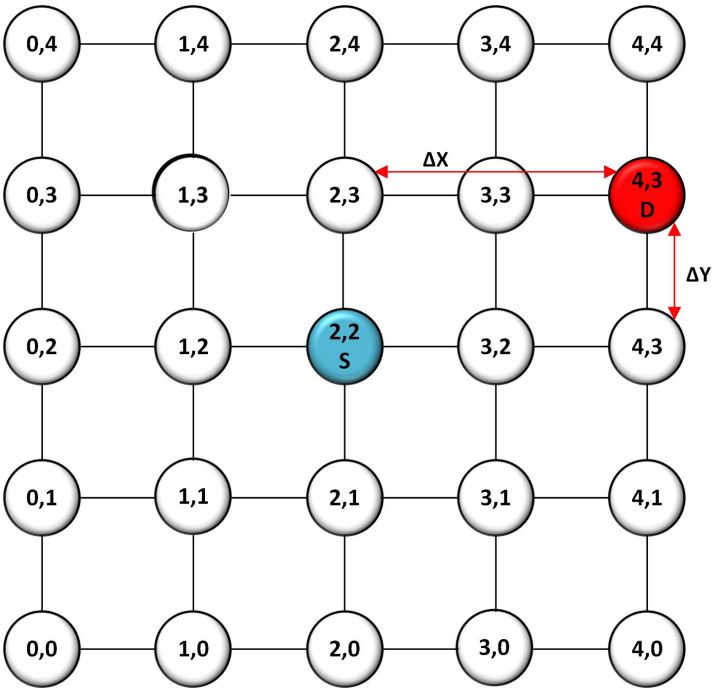

Figure 3.9: The location of the destination node (4,3) according to the source node (2,2) in 5×5 2D Mesh NoC

Chapter 4

Simulation and Results

This chapter presents the evaluation of the proposed routing algorithm (FD) compared to the DyXY routing algorithm. The implementation of the FD and DyXY routing algorithms has been programmed using object-oriented programing in Matlab in which the nodes and packets are built as classes. In addition, Matlab simulation has been applied in two simulation setup scenarios. The first simulation setup is done with different input traffic loads and with the network dimension 5×5 2D mesh NoC. In the second simulation setup, the input traffic loads are fixed to 0.7, and the network dimension is changed from 3×3 to 8×8 2D mesh NoC. The input traffic loads represent the probability of generating a number of packets in the network. In each simulation setup, the resultant throughput, efficiency, packet loss, and latency of FD routing are compared with the DyXY routing algorithm. The Matlab program starts generating the nodes and packets based on the information that is entered in their classes. The network dimension entered into the program defines the number of nodes that should be generated to create 2D mesh NoC. Matlab will assign X and Y coordinates for each node inside the network. In each node, five FIFO input buffers will be added to store the incoming packets. Because we are using the Store and Forward switching technique, the entire packet should be stored in the buffer before routing it to the next node. The maximum number of packets that can be stored in each input buffer is eight. The input traffic load, which is applied in our work, is binomial distribution traffic. At the time of routing, the routing algorithm will read the information of the first packet queued in the input buffers in each node and route it either to the local core in the same node or to one of the four neighbouring nodes. The performance measures and evaluation concepts which are used in our work will be explained in the next section.

4.1 Measures Concepts

- **Total Packets** (P_{total})

  The total number of packets which are successfully generated in the network and have the source and destination information.

- **Packet Generation Time** (T_g)

  The time zero of any packet in the network which can be used to calculate the packet latency and the actual arrival time of the packet.

- **Maximum Arrived Packets** (P_{max})

  The calculated maximum number of packets that expected to reach their destinations in the network. It can be found from the number of rows (R), columns (C) and the number of simulation cycles (T)

$$P_{max} = R \times C \times T \tag{4.1}$$

- **Total Arrived Packets** ($P_{Arrived}$)

  The actual number of packets that have arrived at the destination successfully in the network. It is very significant in calculating the throughput and efficiency in the network.

- **Minimum Arrival Time** (T_m)

  The time that a packet should spend to reach the destination without delay. It happens when the network condition is ideal.

- **Actual Arrival Time** (T_a)

  The actual time that a packet spends during its journey from the source to its destination. It includes the minimum arrival time (T_m) of the packet and the packet extra delay (D_e)

4.2 Performance Measures

- **Throughput** (T_h)

  The throughput represents the average number of packets delivered from the source to destination in a given time period. The total throughput can be calculated as follows:

$$Th = \frac{P_{Arrived}}{P_{max}} \tag{4.2}$$

- **Packet Latency** (L_c)

  Packet latency is the time required to transfer the packet from its source to the destination. The packet latency can be calculated by finding the difference between the actual arrival time and the packet generation time of the packets as follows:

$$L_c = T_a - T_g \tag{4.3}$$

- **Packet Extra Delay** (D_e)

  The extra time that a packet spends during its routing to the destination because of the imperfect network conditions. Because of these network conditions, the packet will wait in the same node for a specific time until it finds a suitable direction toward the destination. The packet may face this scenario many times before reaching its destination. The extra delay of a packet can be calculated using the following equation:

$$D_e = T_a - T_m \tag{4.4}$$

- **Efficiency** (E)

  The efficiency represents the ratio of the total arrived packets to the total packets generated in the network.

$$E = \frac{P_{Arrived} \times 100}{P_{total}} \tag{4.5}$$

- **Packet Loss** (P_{loss})

  This is the percentage of blocked packets in the network which occur when buffers do not have enough space to receive the packets. The average packet loss can be calculated using the total number of lost packets in the network (NP_{lost}) and the total packets (P_{total}) as follows:

$$P_{Loss} = \frac{NP_{lost} \times 100}{P_{total}} \tag{4.6}$$

4.3 Results Analysis and Comparison

In this section, the results of the proposed routing algorithm will be demonstrated and compared with the DyXY routing algorithm. The performance comparison is performed in five measures i.e., 2D-mesh NoC throughput, efficiency, packet loss, latency and packet extra delay. All these measures will be monitored and analyzed for FD and DyXY routing algorithms using two aspects: the first is by employing different traffic loads injected in the network, and the second is based on network dimension changes.

4.3.1 Different Traffic Loads With 5×5 2D Mesh NoC

This simulation setup has been applied for 5×5 2D mesh NoC with the changing of traffic loads for the proposed routing algorithm (FD) and the DyXY routing algorithm. Figure 4.1 depicts the comparison between FD and DyXY routing algorithms in terms of throughput. The throughput in the proposed routing algorithms is exponentially increased as the input traffic load increases in the network. However, in DyXY routing algorithm, the throughput increases linearly till the end of the simulation. The throughput in the FD routing algorithm rises sharply after the lowest input traffic load point 0.1 in the network and continues toward the peak value of the throughput (0.53) when the applied traffic load is 0.8. At this point (0.8), the DyXY routing algorithm reaches only 0.05 which very low compared to 0.53 in the FD routing algorithm. After this point, FD throughput will decrease slightly to reach 0.51 with traffic load 0.9, but DyXY throughput will continue to increase to reach maximum throughput 0.05. Clearly, the FD routing algorithm has improved the throughput in 2D mesh NoC by approximately 84% when compared to the DyXY routing algorithm.

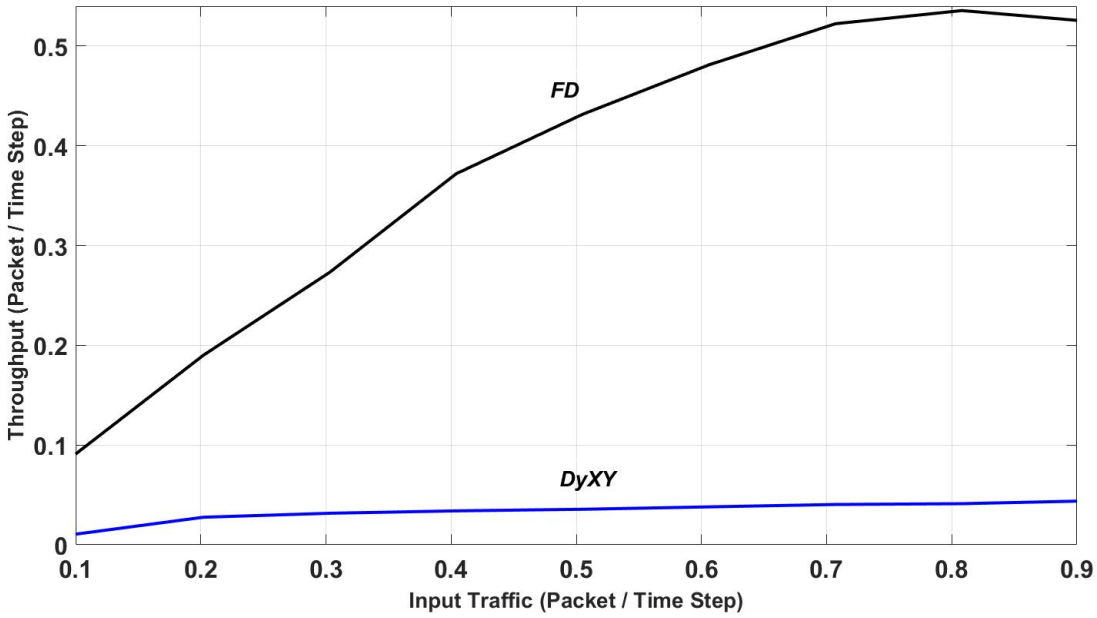

Figure 4.1: Network Throughput vs. Input Traffic on 5×5 2D Mesh

The efficiency obtained in the 5×5 2D mesh NoC for the proposed routing algorithm and the DyXY routing algorithm is shown in Fig. 4.2. The proposed routing algorithm's efficiency outperforms the efficiency of the DyXY routing algorithm. The proposed routing algorithm provides 100% efficiency when the traffic rate is between 0.1 and 0.72. However, the maximum network efficiency of the DyXY routing algorithm only hits about 39% when the input traffic rate is 0.1. After that point, the efficiency of DyXY starts decreasing as traffic load increases until it becomes approximately 6% at maximum traffic. In addition, FD routing algorithm efficiency goes down sharply after increasing the traffic load in the network to about 0.73 and continues decreasing until reaching 72% at the highest traffic load as shown in Fig. 4.2. Overall, the proposed routing algorithm has enhanced 5×5 2D mesh NoC efficiency by 76% compared to the DyXY routing algorithm.

Because the FD routing algorithm provides full adaptability and more optional paths for routing a packet in the network, the average number of packets lost in this routing is less than in the DyXY routing algorithm as shown in Fig. 4.3. The FD routing algorithm has reduced the number of packets lost in 5×5 2D mesh NoC by 52% compared to the DyXY routing algorithm.

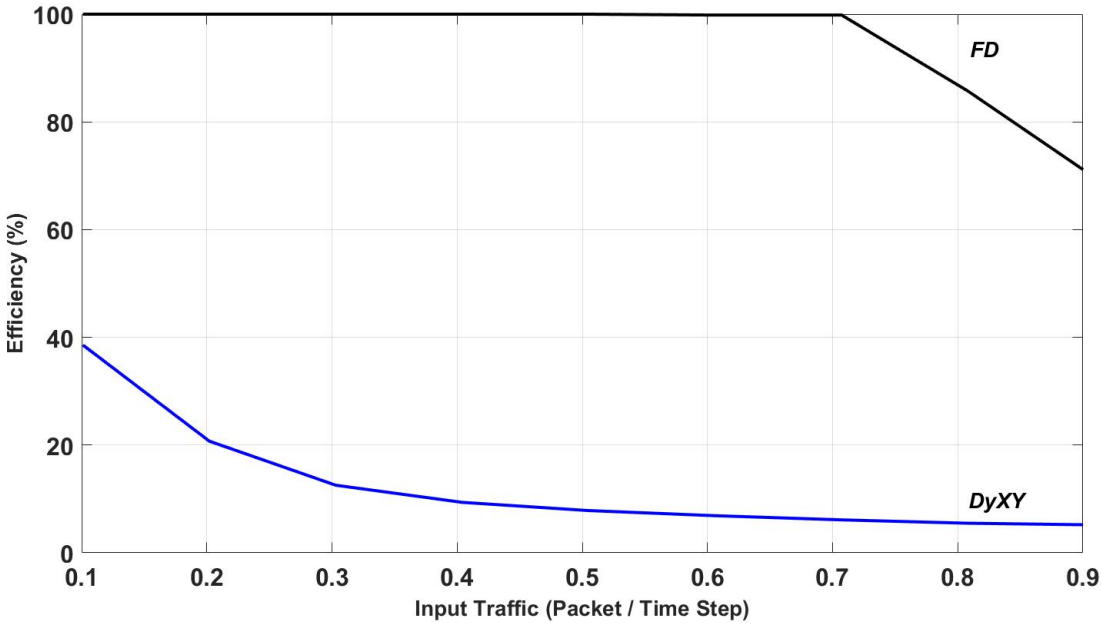

Figure 4.2: Efficiency vs. Input Traffic on 5×5 2D Mesh

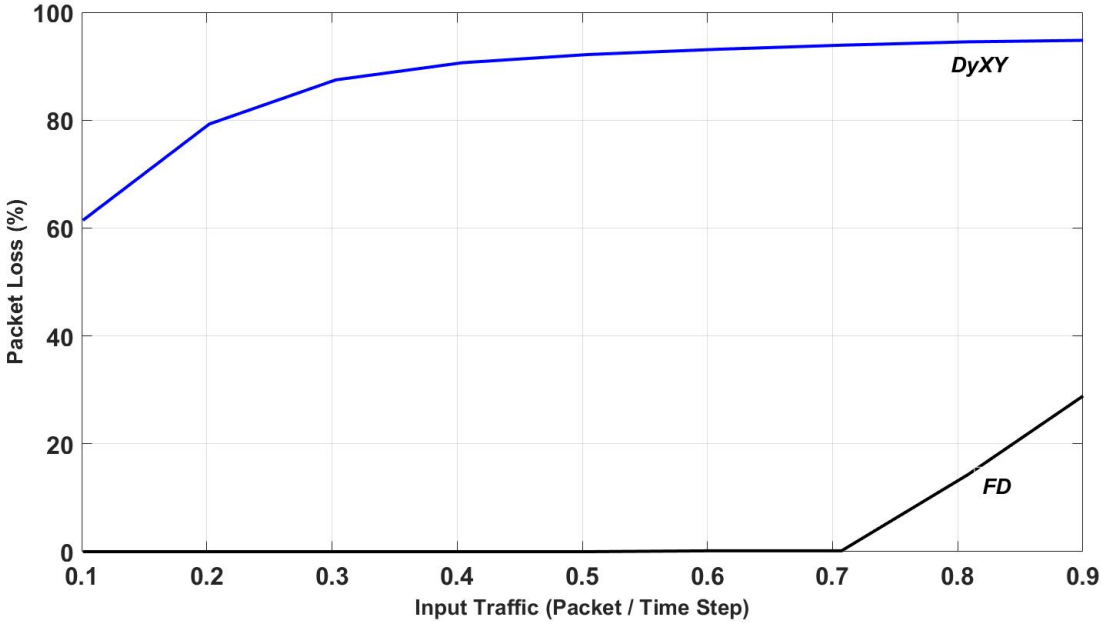

Figure 4.3: Packet Loss vs. Input Traffic on 5×5 2D Mesh

Although the FD routing algorithm has improved the throughput and efficiency as well as decreasing the average number of packets lost in 2D mesh NoC, the average

latency in the FD routing algorithm is high in relation to DyXY routing. This is because some of the packets in FD routing algorithm selected the long paths toward the destination in order to avoid the congested paths in NoC. Figure 4.4 illustrates that at input traffic 0.1, the average latency of the FD routing algorithm is slightly higher than the average latency of the DyXY routing algorithm. At the points 0.17 and 0.47, the average latency of FD and DyXY routing algorithms is the same. However, between these points, the average latency of FD routing algorithm is lower than the average latency of DyXY routing algorithm. After the input traffic load of 0.48, the average latency of FD routing climbs steadily as the input traffic increases, as shown in Fig. 4.4.

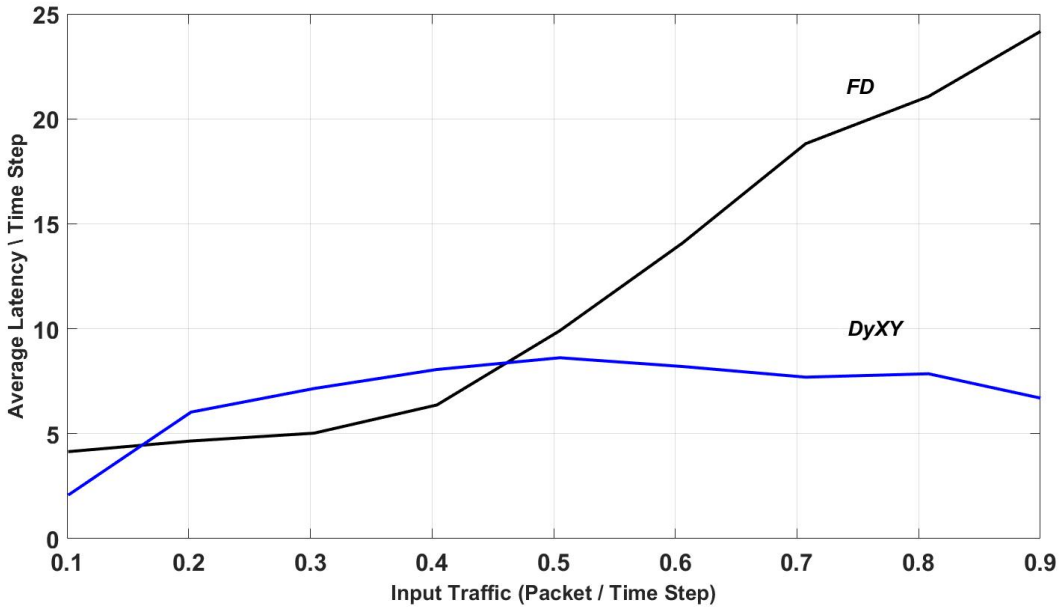

Figure 4.4: Average Latency vs. Input Traffic on 5×5 2D Mesh

Additionally, the average extra delay of FD routing algorithms is higher than DyXY routing algorithms because the packets in FD routing algorithm stay extra time in some nodes when the traffic is high. Also, some packets in the network may select longer paths to destinations which leads to extra time. Between the input traffic 0.2 and 0.4 as demonstrated in Table 4.1, the FD routing algorithm provides less average extra delay than DyXY routing algorithm because the input traffic still low. However, by increasing the input traffic load in the network, the average extra delay of FD sharply increased compared to DyXY routing algorithm. For example,

when the input traffic is 0.6, the resultant average delay in FD routing algorithm is 11. However, the corresponding value in DyXY is 5.35 which is almost the half value in FD routing algorithm.

Input Traffic	**Average extra delay in FD Routing Algorithm**	**Average extra delay in DyXY Routing Algorithm**
0.1	1.14	0
0.2	1.56	3.38
0.3	1.92	4.05
0.4	3.28	5.02
0.5	6.77	5.31
0.6	11.00	5.35
0.7	15.00	4.67
0.8	18.10	4.85
0.9	21.49	3.54

Table 4.1: Average Extra Delay in FD vs. DyXY Routing Algorithm

4.3.2 Different Mesh Network Dimensions with Fixed Traffic Load (0.7)

In this simulation setup, the comparison between FD and DyXY routing algorithms will be applied with a fixed traffic load of 0.7 and different 2D mesh NoC dimensions. Figure 4.5 shows that FD routing algorithm provides a significant improvement in throughput by 90% for different network dimensions compared to DyXY routing algorithm. The figure illustrates that in 3×3 2D mesh NoC, the average throughput of FD routing algorithm outperforms the throughput of DyXY routing algorithm with a difference of approximately 0.22. After that, the throughput of FD continues to increase as the 2D mesh dimension gets larger until reaching a peak throughput with 5×5 2D mesh; however, DyXY goes down to the lowest average throughput when using the same dimension. The average throughput in FD routing algorithm starts declining as the dimension of 2D-mesh NoC increases from 6×6 to 8×8. This result is because in larger 2D mesh dimensions, a number of packets will not be able to reach the destination during the assigned simulation time since they may use the longest paths in the network.

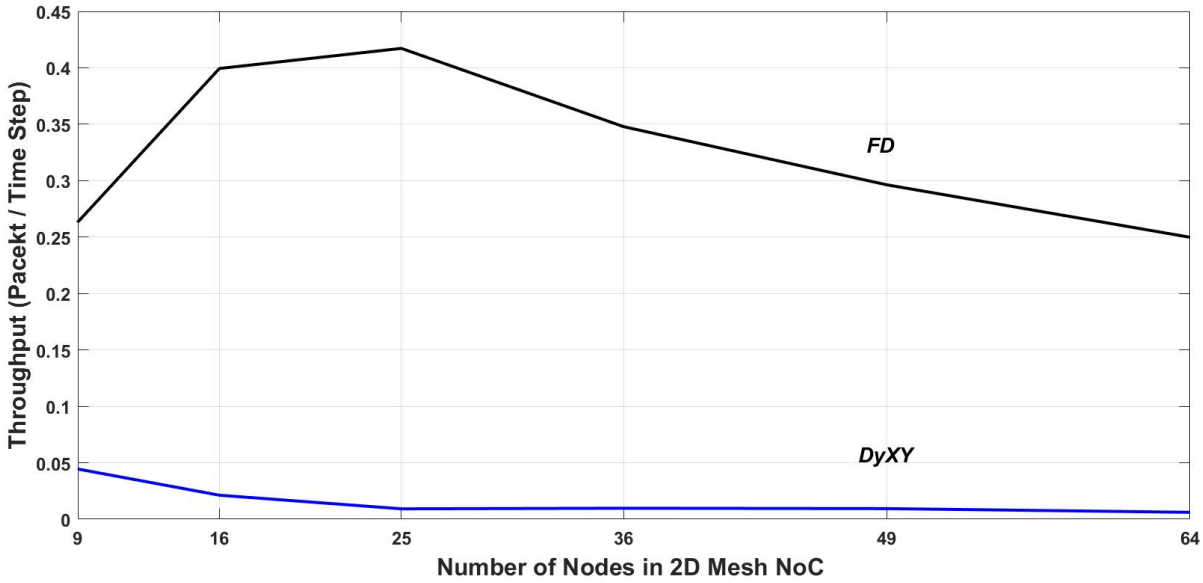

Figure 4.5: Network Throughput vs. Network Dimension with Traffic Rate 0.7

Figure 4.6 shows the comparison between FD and DyXY routing algorithms in terms of efficiency for different dimensions of 2D mesh NoC. FD routing algorithm provides important improvement of the efficiency for different dimensions in 2D mesh NoC by 92% compared to DyXY routing algorithm. The reason is because the number of arrived packets at their destinations in FD is more than in DyXY. For example, when network dimension is 8×8 2D mesh NoC, the efficiency of FD routing algorithm reaches about 85%. However, for the same network dimension, DyXY routing algorithm records very low efficiency of about 2%. This illustrates how FD routing algorithm provides an excellent guarantee to deliver the packets to their destination with less probability of loss. Table 4.2 shows the improvement of the efficiency that FD routing algorithm has presented in each network dimension compared to DyXY routing algorithm.

Figure 4.7 shows the packet loss percentage for FD and DyXY routing algorithms in 2D mesh NoC for the dimensions (from 3×3 to 8×8 2D mesh NoC). With 3×3 2D mesh NoC, the packet loss percentage in the proposed routing algorithm is less than DyXY with a difference in percentage of 43%. This difference increases with the larger dimensions (4×4, 5×5, 6×6 and 7×7) 2D mesh and reaches the highest difference with the percentage of 83% in 8×8 2D mesh NoC. The packet loss rises in DyXY routing algorithm because it restricts the packets movements when they

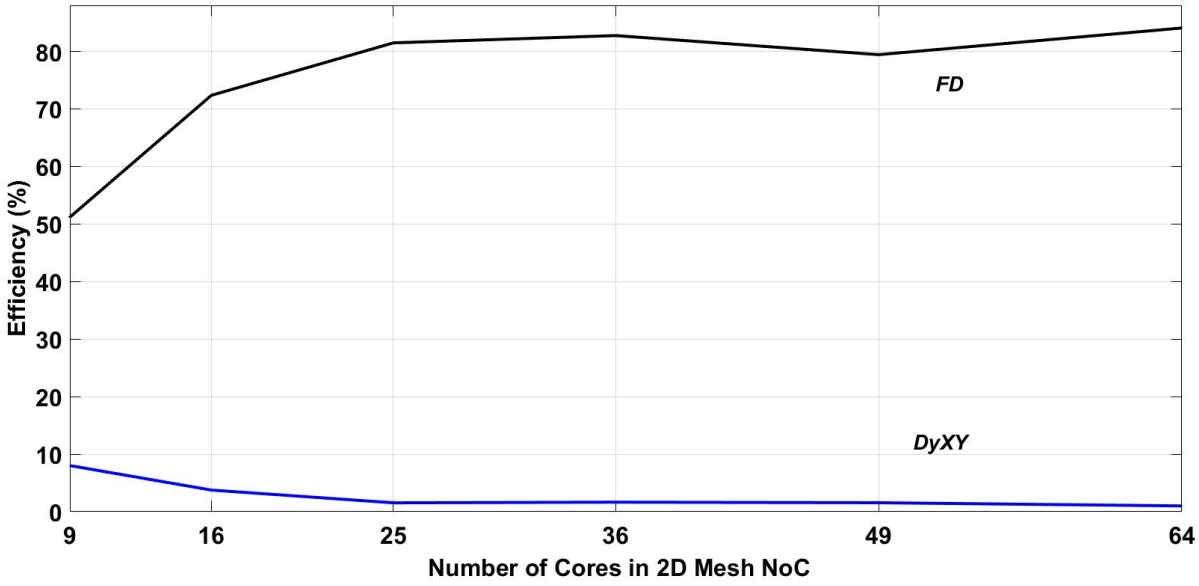

Figure 4.6: Efficiency vs. Network Dimension with Traffic Rate 0.7

Network Dimension	**The Efficiency of FD routing**	**The Efficiency of DyXY routing**	**The Improvement Rate by FD routing**
3×3	51.19	8.01	73%
4×4	72.37	3.76	90%
5×5	81.48	1.54	96%
6×6	82.75	1.65	96%
7×7	79.43	1.56	96%
8×8	85	2.00	97%

Table 4.2: Efficiency Improvement in FD Compared to DyXY Routing Algorithm

arrive at the same X or Y coordinates of the destinations as explained in Chapter 2. However, the average packet loss decreases in FD routing algorithm because the optional routing paths increase as the network dimension gets larger. In general, FD routing algorithm has improved the average packet loss for the 2D mesh NoC dimensions shown in Fig. 4.7 by 59% compared to DyXY routing algorithm.

Figure 4.8 shows that, in general, the average latency of FD routing algorithm is higher than DyXY routing algorithm. However, in the dimension 3×3 2D mesh NoC, the average latency of FD is lower than the DyXY routing algorithm latency.

When the 2D mesh NoC dimension gets bigger, the latency in FD routing algorithm increases. On the other hand, the latency in DyXY routing algorithm will be only slightly changed as network dimensions vary.

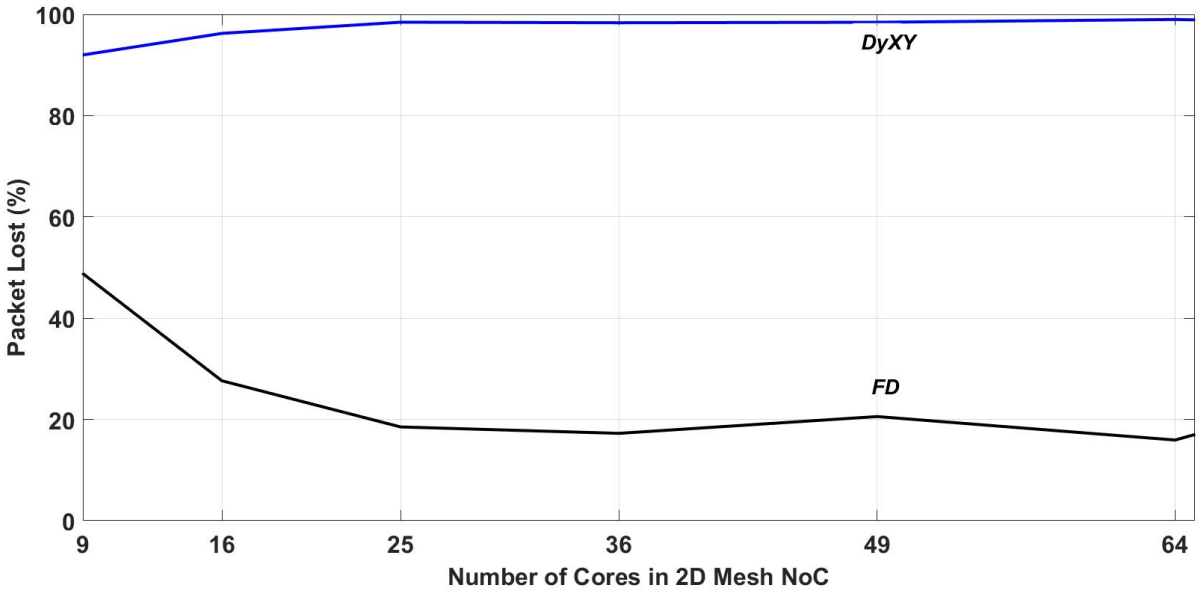

Figure 4.7: Packet Loss vs. Network Dimension with Traffic Rate 0.7

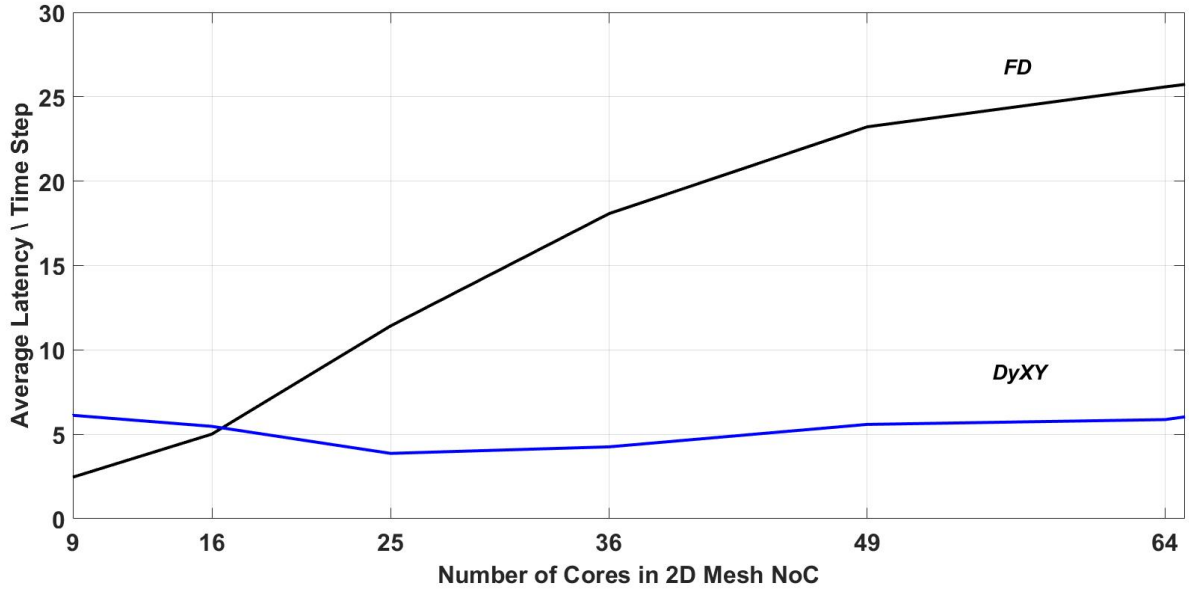

Figure 4.8: Average Latency vs. Network Dimension with Traffic Rate 0.7

Chapter 5

Conclusions

This thesis has presented a novel, full-adaptive routing algorithm for 2D mesh NoC called Forced- Directed (FD) routing algorithm. The reason for proposing FD routing algorithm is to enhance the network throughput and efficiency in 2D mesh NoC. In addition, FD routing introduces a unique and reliable technique to route the packets through the network toward the destinations with less average packet loss compared to DyXY routing algorithm. FD routing algorithm has achieved excellent results by providing full adaptability to allow the packet to avoid the congested paths in the network without prohibiting any paths in the network. The FD routing algorithm is evaluated in two simulation setup phases compared to the DyXY routing algorithm. In the first simulation, 5×5 2D mesh NoC dimension and different traffic loads are setup. In this simulation, compared to DyXY, the FD routing algorithm has improved the throughput by 84% and the efficiency by 76%. In addition, the average packet loss in FD has been reduced by 52% compared to the average in DyXY routing algorithm. The second simulation is based on a fixed traffic load (0.7) and different 2D mesh NoC dimensions (3×3, 4×4, 5×5, 6×6, 7×7, 8×8 2D mesh). The results from this simulation show that the FD routing algorithm has increased the resultant network throughput and efficiency by 90% and 92% respectively and reduced the average packet loss by 59% when compared to the DyXY routing algorithm.

Future Work

Our future work includes implementing FD routing algorithm in 3D NoC architecture and comparing the resultant throughput, efficiency, and average packet loss with one of the traditional routing algorithms in 3D NoC , such as the DyXYZ routing algorithm [42].

Bibliography

[1] S. Mubeen, "Evaluation of source routing for mesh topology network on chip platforms," Master of Science Thesis, Jönköping University, School of Engineering, Computer and Electrical Engineering, 2009.

[2] R. R. Schaller, "Technological innovation in the semiconductor industry: a case study of the International Technology Roadmap for Semiconductors (ITRS)," in *Portland International Conference on Management of Engineering and Technology, 2001*, (Portland, OR, USA), pp. 195–195, IEEE, Aug. 2001.

[3] J. K. Singh, A. K. Swain, T. N. K. Reddy, and K. K. Mahapatra, "Performance evalulation of different routing algorithms in Network on Chip," in *IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, (Visakhapatnam, India), pp. 180–185, Dec. 2013.

[4] R. Holsmark and M. Högberg, "Modelling and Prototyping of a Network on Chip," Master of Science Thesis, Rijksuniversiteit Groningen, 2002.

[5] M. Palesi and M. Daneshtalab, *Routing algorithms in networks-on-chip*. New York Heidelberg Dordrecht London: Springer, 2014.

[6] F. El Guibaly, "Design and analysis of arbitration protocols," *IEEE transactions on Computers*, vol. 38, pp. 161–171, Feb. 1989.

[7] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Design Automation Conference*, (Las Vegas, NV, USA), pp. 684–689, IEEE, Jun. 2001.

[8] L. Benini and G. De Micheli, "Networks on Chips: A new SoC paradigm," *computer*, vol. 35, pp. 70–78, Jan. 2002.

[9] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, and D. Lindqvist, "Network on chip: An architecture for billion transistor era," in *Proceeding of the IEEE NorChip Conference*, (Färögatan 6, 164 40 Kista, Sweden), pp. 1–11, Jul. 2000.

[10] A. Agarwal, C. Iskander, and R. Shankar, "Survey of Network on Chip (NoC) architectures & contributions," *Journal of engineering, Computing and Architecture*, vol. 3, pp. 21–27, Jan. 2009.

[11] N. E. Jerger and L.-S. Peh, "On-Chip Networks," *Synthesis Lectures on Computer Architecture*, vol. 4, pp. 1–141, Jul. 2009.

[12] É. Cota, A. de Morais Amory, and M. S. Lubaszewski, *Reliability, Availability and Serviceability of Networks-on-chip*. Springer Science & Business Media, 2011.

[13] S. Singh, S. Bhoj, D. Balasubramanian, T. Nagda, D. Bhatia, and P. Balsara, "Network interface for NoC based architectures," *International Journal of Electronics*, vol. 94, pp. 531–547, Aug. 2007.

[14] M. Moadeli, P. Maji, and W. Vanderbauwhede, "Quarc: A high-efficiency network on-chip architecture," in *Advanced Information Networking and Applications*, (Bradford, UK), pp. 98–105, IEEE, May 2009.

[15] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: an engineering approach*. 340 Pine Street, Sixth Floor San Francisco, USA: Morgan Kaufmann, 2003.

[16] K. Tatas, K. Siozios, D. Soudris, and A. Jantsch, *Designing 2D and 3D network-on-chip architectures*. Springer New York Heidelberg Dordrecht London: Springer, 2016.

[17] M. Atagoziyev, "Routing algorithms for on chip networks," Master of Science Thesis, The graduate school of natural and applied sciences, 2007.

[18] S. Gugulothu and M. Chawhan, "Design and implementation of various topologies for networks on chip and its performance evolution," in *International Conference on Electronic Systems, Signal Processing and Computing Technologies (ICESC)*, (Nagpur, India), pp. 7–11, IEEE, Jan. 2014.

[19] A. Q. Ansari, M. R. Ansari, and M. A. Khan, "Performance evaluation of various parameters of Network-on-Chip (NoC) for different topologies," in *Annual IEEE India Conference (INDICON)*, (New Delhi, India), pp. 1–4, IEEE, Dec. 2015.

[20] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *INTEGRATION, the VLSI journal*, vol. 38, pp. 69–93, Oct. 2004.

[21] Y. A. Sadawarte, M. A. Gaikwad, and R. M. Patrikar, "Comparative study of switching techniques for network-on-chip architecture," in *Proceedings of International Conference on Communication, Computing Security*, (New Delhi, India), pp. 243–246, ACM, Feb. 2011.

[22] W. J. Dally and C. L. Seitz, "The torus routing chip," *Distributed Computing*, vol. 1, pp. 187–196, Dec. 1986.

[23] Z. Shi, *Real-time communication services for networks on chip*. PhD dissertation, University of York, 2009.

[24] J. Duato, A. Robles, F. Silla, and R. Beivide, "A comparison of router architectures for virtual cut-through and wormhole switching in a NOW environment," *Journal of Parallel and Distributed Computing*, vol. 61, pp. 224–253, Feb. 2001.

[25] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, vol. 3, pp. 267–286, Sep. 1979.

[26] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *California Institute of Technology*, Jun. 1988.

[27] N. E. Jerger, L.-S. Peh, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," *35th International Symposium on Computer Architecture*, vol. 36, pp. 229–240, Jun. 2008.

[28] M. Singhal, "Deadlock detection in distributed systems," *Computer*, vol. 22, pp. 37–48, Nov. 1989.

[29] S. P. Adiga, "NoC characterization framework for design space exploration," Master of Science Thesis, Delft University of Technology, 2014.

[30] A. Jantsch and H. Tenhunen, *Networks on Chip*. Dordrecht: Kluwer Academic Publishers, 2003.

[31] J. Qi, *System-Level design automation and optimisation of network-on-chips in terms of timing and energy*. PhD dissertation, University of Southampton, 2015.

[32] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "HAMUM-A novel routing protocol for unicast and multicast traffic in MPSoCs," in *18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, (Pisa, Italy), pp. 525–532, Feb. 2010.

[33] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. 500 Sansome Street, Suite 400, San Francisco, CA : Elsevier, 2004.

[34] D. Masoud, *Exploring Adaptive Implementation of On-Chip Networks*. PhD Thesis, University of Turku, 2011.

[35] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, pp. 62–76, Feb. 1993.

[36] S. Mubeen and S. Kumar, "On source routing for mesh topology network on chip," in *9th Swedish System on Chip Conference (SSoCC 09)*, pp. 1–4, May 2009.

[37] R. Holsmark, *Deadlock Free Routing in Mesh Networks on Chip with Regions*. PhD Thesis, Linköping University Electronic Press, 2009.

[38] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *ACM Sigarch Computer Architecture News*, vol. 20, pp. 278–287, May 1992.

[39] V. Rantala, T. Lehtonen, and J. Plosila, *Network on chip routing algorithms*. University of Turku, Department of Information Technology Joukahaisenkatu 3-5 B, 20520 Turku, Finland: Citeseer, 2006.

[40] Y. Aydogan, C. B. Stunkel, C. Aykanat, and B. Abali, "Adaptive source routing in multistage interconnection networks," in *Proceedings of IPPS'96, The 10th International Parallel Processing Symposium*, (Honolulu, HI, USA), pp. 258–267, Apr. 1996.

[41] W. Zhang, L. Hou, J. Wang, S. Geng, and W. Wu, "Comparison research between xy and odd-even routing algorithm of a 2-dimension 3x3 mesh topology network-on-chip," in *WRI Global Congress on Intelligent Systems*, (Xiamen, China), pp. 329–333, Aug. 2009.

[42] M. Li, Q.-A. Zeng, and W.-B. Jone, "DyXY: a proximity congestion-aware deadlock-free dynamic routing method for network on chip," in *Proceedings of the 43rd annual Design Automation Conference*, (San Francisco, CA, USA), pp. 849–852, ACM, Jul. 2006.

[43] M. S. Sayed, A. Shalaby, M. El-Sayed, and V. Goulart, "Flexible router architecture for network-on-chip," *Computers / Mathematics with Applications*, vol. 64, pp. 1301–1310, Sep. 2012.