ROUTING ALGORITHMS FOR ON CHIP NETWORKS

MAKSAT ATAGOZIYEV

DECEMBER 2007

ROUTING ALGORITHMS FOR ON CHIP NETWORKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MAKSAT ATAGOZIYEV

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2007

Approval of the thesis:

ROUTING ALGORITHMS FOR ON CHIP NETWORKS

submitted by **MAKSAT ATAGOZIYEV** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Dr. Şenan Ece (Güran) Schmidt _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Semih Bilgen _____
Electrical and Electronics Engineering Dept., METU

Dr. Şenan Ece (Güran) Schmidt _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Hasan Güran _____
Electrical and Electronics Engineering Dept., METU

Asst. Prof. Dr. Cüneyt Bazlamaçcı _____
Electrical and Electronics Engineering Dept., METU

M.Sc. Murat Soysal _____
Network Technologies Dept., TÜBİTAK-ULAKBİM

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Maksat Atagoziyev

Signature :

ABSTRACT

ROUTING ALGORITHMS FOR ON CHIP NETWORKS

Atagoziyev, Maksat

M.Sc., Department of Electrical and Electronics Engineering

Supervisor: Dr. Şenan Ece (Güran) Schmidt

December 2007, 79 pages

Network-on-Chip (NoC) is communication infrastructure for future multi-core Systems-on-Chip (SoCs). NoCs are expected to overcome scalability and performance limitations of Point-to-Point (P2P) and bus-based communication systems. The routing algorithm of a given NoC affects the performance of the system measured with respect to metrics such as latency, throughput and load distribution. In this thesis, the popular Orthogonal One Turn (O1TURN) and Dimension Order Routing algorithms (DOR) for 2D-meshes are implemented by computer simulation. Investigating the effect of parameters such as packet, buffer and topology sizes on the performance of the network, it is observed that the center of the network is loaded more than the edges. A new routing algorithm is proposed and evaluated to achieve a more balanced load distribution. The results show that this goal is achieved with a trade off in latency and throughput in DOR and O1TURN.

Keywords: Networks on Chip, Dimension Ordered Routing, on-chip routing

ÖZ

ÇİP ÜZERİNDEKİ AĞLAR İÇİN YÖNLENDİRME ALGORİTMALARI

Atagoziyev, Maksat
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Danışmanı: Dr. Şenan Ece (Güran) Schmidt

Aralık 2007, 79 sayfa

Yonga üzeri Ağlar (YüA) gelecekte gerçekleştirilecek çok çekirdekli Yonga üzeri Sistemler (YüS) için haberleşme alt yapısı olacaklardır. YüA'ların noktadan noktaya ya da veriyolu tabanlı haberleşmenin daha önce karşılaşılmış olan ölçeklenme ve başarım kısıtlarının üstesinden gelmeleri beklenmektedir. Bir YüA için seçilen yönlendirme algoritması sistemin gecikme, veri iletme kapasitesi ve yük dağılımı gibi başarım değerlerini etkilemektedir. Bu tezde iki boyutlu örgüler için çok kullanılan Orthogonal One Turn (O1TURN) ve Boyut Sıralamalı Yönlendirme (BSY) algoritmaları bilgisayar benzetimi yolu ile gerçekleştirilmektedir. Paket, arabellek ve ağ büyüklükleri parameterelerinin ağ başarımına etkisi incelendiğinde ağın merkez kısımlarının kenarlardan daha çok yüklendiği görülmüştür. Buna göre yükü daha dengeli dağıtacak yeni bir yönlendirme algoritması tasarlanmış ve başarımı incelenmiştir. Elde edilen sonuçlar BSY ve O1TURN'e göre gecikme ve veri iletme kapasitesinde bir düşüşe karşılık istenen amaca ulaşıldığını göstermektedir.

Anahtar Kelimeler: Çip üzerindeki Ağlar, Boyut Sıralama Yönlendirme, çip üzerinde yönlendirme

To My Parents

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Dr. Şenan Ece (Güran) Schmidt for her continuous encouragement, patience and skillful guidance during the whole process of writing this thesis.

I would also like to thank Venera Adanova for her valuable support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

2D	Two Dimensional
AD	Address Decoding
BFT	Butterfly Fat Tree
CB	Crossbar
CLICHÉ	Chip-Level Integration of Communicating Heterogeneous Elements
DOR	Dimension Order Routing
FIFO	First In-First Out
HOL	Head of Line
I^2P	Infrastructure Intellectual Property
IP	Intellectual Property
MIN	Multistage Interconnection Network
MP-SoC	Multiprocessor System on Chip
NoC	Network on Chip
O1TURN	Orthogonal One Turn
P2P	Point-to-Point
PE	Processing Element
RA	Routing and crossbar Arbitration
ROMM	Randomized, Oblivious, Multi-phase, Minimal
SAF	Store-and-Forward
SoC	System on Chip
SPIN	Scalable Programmable Integrated Network
VC	Virtual Channel
VCID	Virtual Channel Identifier
VCT	Virtual Cut-Through
VLSI	Very Large-Scale Integration

CHAPTER I

INTRODUCTION

In 1990's more and more resources (e.g. processor cores and reusable components) have been integrated on a single silicon die. This approach has become known as System-on-Chip (SoC) paradigm. Buses and point-to-point (P2P) links were used as communication infrastructure for SoCs, but as silicon technology advances further, they are becoming too costly for this purpose because of their poor scalability, low and unpredictable performance, and high power consumption [1]. Emergence of SoC platforms containing a large set of processing elements (PE) is inevitable. One of the most important parts of such multiprocessor SoC (MP-SoC) is communication infrastructure [2]. Buses cannot provide efficient interconnect from performance point of view. They are capable of supplying the needed level of interconnect efficiency if the system size does not exceed a few tens of components, otherwise significant degradation of performance can be observed, at the same time P2P connections can efficiently connect even fewer numbers of elements [3]. Thus, communication is becoming a bottleneck in SoC design, and today chip design methodologies are much more communication rather than computation centric.

Around 1999 several research groups started investigation of communication part of SoCs [1]. As a result, in 2002, Networks-on-Chip (NoC) paradigm was proposed to overcome the limitations arising with bus and P2P connection based communication infrastructures [4]. NoCs are expected to be a response to the "interconnect showstopper". They have many in common with interconnect architectures of high-performance parallel computing systems. The key idea of NoC approach is using

packet-switched technique and so called infrastructure IPs (I^2Ps) [5] for communication between functional IP blocks. Ideas such as switch-based (routers) networks and packet-switched communication were borrowed from macroscopic communication networks, since over many years of rapid evolution computer networks demonstrated sustainable scalability, continuously improving performance and reliability [6]. Packet-switched communication not only provides high degree of scalability, but also gives an opportunity for standardization and reuse of communication architecture [3]. Using NoCs unlike buses makes possible sharing wiring resources between several communication flows, what provides better wire utilization. Another advantage of using networks instead of buses is that it has higher bandwidth and allows multiple concurrent communications [7].

The most important features that distinguish NoC architectures are network topology and routing algorithms [7] [8] [9]. However, such parameters as buffer size and area overhead should also be kept in mind while trying to improve network performance. Area overhead is directly proportional to buffer size; increasing buffer size of each physical channel of 4x4 NoC from 2 to 3 words leads to 30% increase of total router area [10]. Buffers are the most power-hungry components of on-chip routers. Thus, usage of buffers should be limited as much as possible and a large set of wiring resources available on silicon chip should be utilized instead. On-chip routing algorithms and switching techniques also should be chosen such that they do not make use of huge buffers and network state is available through control links [11]. At the same time, they must be simple enough in order not to extend routing time and implementation complexity. Often it becomes the reason of preferring deterministic routing algorithms to adaptive ones.

The main performance metrics considered in the literature [13][36][38] in evaluating NoC designs and routing algorithms are throughput and average packet delay. Another parameter we are considering in performance evaluation of routing algorithms in this study is the *load distributions.*

Our research was motivated by advances in VLSI technology and rapid evolution of NoCs. Despite a lot of research work has been conducted since proposal of NoC approach, much more work is to be done in future, since this field is still in its infancy. There are still many outstanding research problems in NoC design. In [12] the key problems and the suggested solutions were described. We mainly concentrate on routing algorithm design. Choosing the right type of routing is essential. It directly affects the network performance and power consumption [13] [14] [15] [16]. Implementation complexity and performance requirements are the main parameters affecting the choice of routing strategy. Deterministic routing requires less resources and guarantees correct packet arrival sequence, while adaptive routing is more complex in implementation but provides higher throughput and lower latency. Deterministic techniques have disadvantage of underutilization of network resources [4]. Adaptive algorithms are deadlock- and livelock-prone in comparison with deterministic ones and need extra precaution. However, applied deadlock/livelock detection mechanisms may lead to unpredictable degradation of performance [15]. The choice between adaptive and deterministic routing should be made very carefully, because it causes trade-offs between area and performance [12].

Dimension Order Routing (DOR) algorithm or its variants are often used in mesh-based NoCs, because they are minimal, fully deadlock- and livelock-free and simple in implementation. It is called *XY-routing* algorithm for 2D-meshes. However, XY-routing tends to send packets toward the center of the mesh when the contention is high [14]. Hence, the central part of the mesh becomes overloaded. It leads to early network saturation and performance degradation. XY-routing performs well just for uniformly distributed traffic.

One of the recently proposed algorithms based on DOR is Orthogonal One Turn (O1TURN) routing algorithm. According to [36] O1TURN achieves higher throughput than DOR when the network size increases.

This thesis explores performance of Dimension Order and O1TURN routing for pipelined wormhole routers. Also a deterministic routing algorithm based on Dimension Order and O1TURN routing and its performance evaluation is presented in this research. In the proposed algorithm an attempt to distribute the load in the center of 2D-mesh network was made. During the process of research a simulator for pipelined switch model was developed. Performance evaluation of the Dimension Order and O1TURN routing algorithm as well as the proposed algorithm is carried out using this simulator.

1.1 Thesis Organization

The thesis is organized as follows:

In Chapter 2, first, information on NoC design considerations is given, after that the most popular network on-chip topologies are introduced. Next, switching techniques for interconnection networks are reviewed. Then, on-chip routing algorithms are described and their performance characteristics are reviewed briefly. Finally, concise information on traffic models for NoCs is presented.

In Chapter 3, we present analysis of Dimension Order and O1TURN routing algorithms for different configurations of 2D-mesh network on chip. In the first part, the canonical router architecture is introduced then our simulator model is presented. In the next section we provide an examination of correctness of our simulator, after that discuss simulation results of Dimension Order routing for 2D-mesh topology. O1TURN algorithm and its performance evaluation are given in the last section of this chapter.

Chapter 4 introduces the algorithm we propose and its comparative performance evaluation in terms of throughput, average packet delay, and load distribution with Dimension Order and O1TURN algorithms.

Chapter 5 concludes the thesis and outlines future work.

CHAPTER II

BACKGROUND

SoCs consist of various IP blocks, routers, and physical links. IP block (in some literature terms such as processing element (PE) or virtual component are also used) is a general term used for representing such entities as DSPs, memory modules, MPEG decoders, processor cores and so on. Routers and links are employed for providing communication infrastructure for IP blocks in other words they organize a NoC. Routers have input and output ports which are connected to IP blocks and other routers in the system. The number of ports depends on the network topology. Physical links are actually copper wires utilized for data transmission between adjacent routers and IP blocks. The messages sent via physical links are divided into *packets* or *flits* depending on the used switching technique. Packet is a fixed length data block containing all the control and routing information, which gives the router an opportunity to decide where the incoming packet should be sent, in its header. Sometimes packets are further split into flits (*flow control units*). Unlike packets just the first flit called *header flit* involves routing information; its main task is reserving a path for other flits (*data flits*) following it, the last flit is a *tail flit* which releases all the resources reserved by header flit. Performance of NoC as defined in the previous chapter, depends on various factors such as network topology, routing strategy and switching technique. Hence, in this chapter, we give a brief theoretical background on these issues. Firstly, we introduce NoC design considerations. Subsequently, most popular topologies employed in NoCs are reviewed. Then, switching techniques used in NoCs are presented. In the next section we describe widely used routing

algorithms and review their performance characteristics briefly. Finally, we present brief information about traffic models for NoCs.

2.1 Network on Chip Design Considerations

All of the NoC designs have specific features, however, the same performance evaluation metrics apply to them such as latency, throughput, power consumption, area overhead, and implementation complexity. All of these metrics depend on each other, and cannot be considered separately. These performance metrics are divided into *performance parameters* and *cost factors* [44].

The performance parameters are throughput and latency. Throughput is defined as a fraction of packets delivered from sources to destinations in a given amount of time. Latency is defined as time taken to deliver a packet from source to destination [38][13]. For any of the NoC applications high throughput and low latency are desirable. Low latency and high throughput can be achieved by providing deadlock-, livelock-, and starvation-freedom (see Section 2.4 for definitions), better channel utilization, and using appropriate routing algorithms (see Section 2.4) and switching techniques (see Section 2.3).

Power consumption, area overhead, and implementation complexity are cost factors. Power consumption is one of the important NoC design constraints, especially for battery-operated mobile devices [1] because reduction of power consumption helps to extend the battery life. Area overhead is often depends on buffer size of an individual router. As we mentioned in previous chapter increasing buffer size significantly impacts total router area and its power consumption. Usage of large or small buffers depends on the requirements of switching technique and routing algorithm. Much more resources are required for implementing adaptive routing algorithms in comparison to deterministic ones. This leads to high implementation complexity of the router. In adaptive routing algorithms packets between the same

source destination pair may take different routes. Thus, they may arrive out of order. Huge buffers are needed for their reordering [45].

In Chapter IV we use another additional performance parameter for comparison of routing algorithms. This performance parameter is *load distribution.* We measure the *load* on a router as the number of flits passed through the router during the simulation time. The routers which are located closer to the center of the 2D mesh-based network experience higher loads compared to those located closer to the edges [46]. This leads to greater packet delays and lower throughput of a NoC system. If the load is distributed more uniformly better performance results can be obtained.

2.2 Network on Chip Topologies

Most interconnect architectures used for NoC came from the parallel computing field. However, SoC design paradigm introduces some constraints to those architectures, because there is a significant difference between on-chip and off-chip (on board) applications. An ideal architecture should provide high throughput, low latency, low power consumption, and have small area requirements and implementation complexity. Certainly, it is impossible to incorporate all of these features into the same system, because some of them contradict each other. That is why researchers always have to sacrifice some of the advantages of certain architecture for the sake of gaining another one. For this reason none of the existing architectures offer the desired performance. One of the essential parameters effecting performance of a NoC is topology. There were proposed different ones since the beginning of intensive research in this field. In the following, we overview the most popular ones.

2.2.1 SPIN

One of the proposed interconnect templates SPIN [17] (Scalable, Programmable, Integrated Network) uses fat-tree architecture (Figure 1. Black boxes are routers, white boxes are IP blocks). A fat-tree is a tree with routers on the nodes and IP blocks on the leaves. Every node has four leaves and the parent is replicated four times at any level of the tree. The number of parent port is equal to the number of child ports for every switch. The size of the network grows with a rate of ($N\log N$)/8 where N is the number of IP blocks. For N IP blocks, the number of switches converges to $S=3N/4$. There are as many parents as leaves so the network is non-blocking. The term *non-blocking* comes from the area of *Multistage Interconnection Networks* (MIN), it means that it is always possible to establish a connection between any idle pair of input and output ports having no effect on existing connections[13][37]. It is obvious that there must exist multiple paths between any given input and output ports in the network in order to be non-blocking. Supporting multiple paths leads to undesired growth of hardware complexity, power consumption, and high usage of on-chip space.

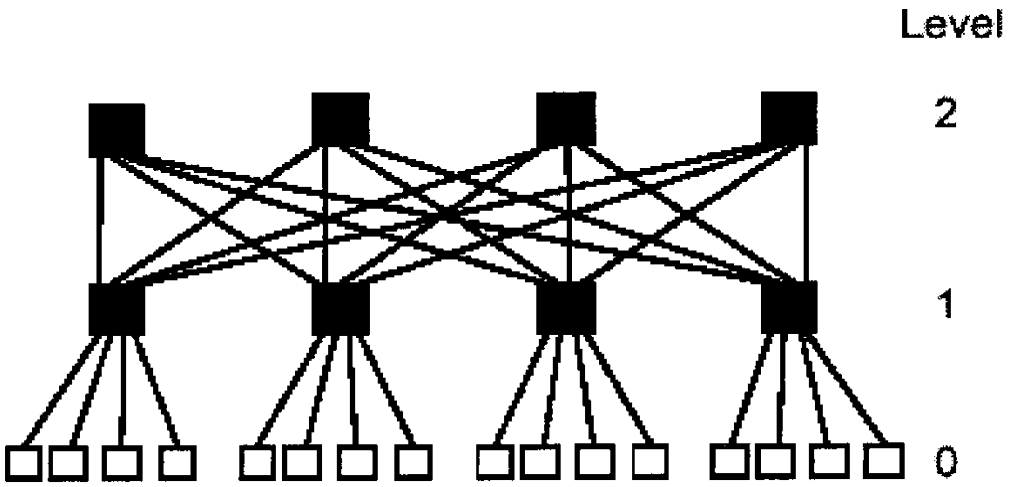

Figure 1 SPIN architecture [2]

2.2.2 CLICHÉ

CLICHÉ (Chip-Level Integration of Communicating Heterogeneous Elements) was proposed by Kumar et al. [8]. This architecture is the same as 2D mesh interconnection topology. There are as many switches as IP blocks. Every switch, except those at the edges, is connected to four neighboring switches and one IP block. Local interconnections between IP blocks and switches are independent of the size of the network. The simplicity of the architecture allows for the division of the chip into processing regions. Different protocols may be used in local regions. Routing is not complex, so the smaller size switches may be used and the network is scalable.

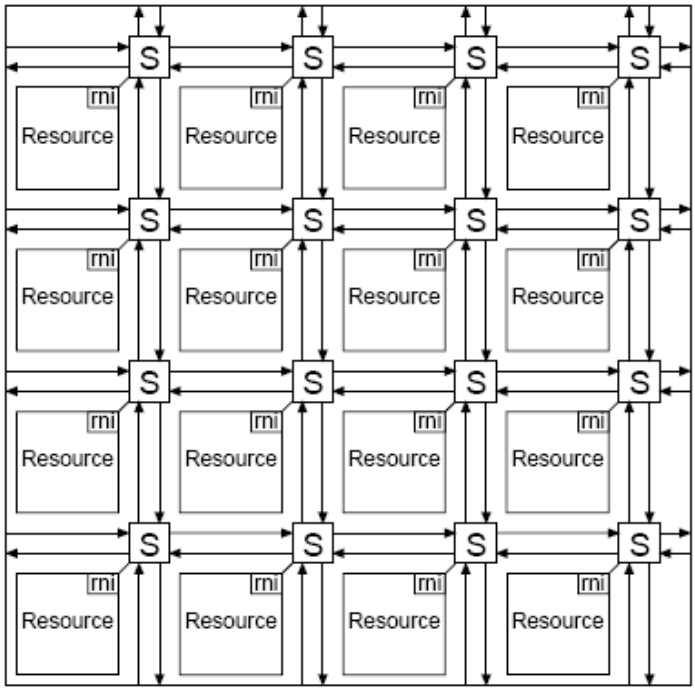

Figure 2 CLICHE [18]

2.2.3 Torus

Torus [7] architecture is the same as regular mesh. However, unlike the mesh, where edge switches are connected only to two neighboring switches, the torus architecture uses wrap-around channels in order to connect the switches at the edges to the switches at the opposite edge. The number of switches is equal to the number of IP blocks and every switch has five ports.

Due to the long wrap-around channels the packet transmission delay may become significantly long and require usage of repeaters. This can be avoided by folding the torus as it is shown in Figure 4. Folding is done by shifting all nodes in even rows to the right and all nodes in even positions of each row down, next connecting all the neighboring nodes in newly gained rows and columns then pair-wise connecting edge nodes in rows and columns. Now wraparound links are significantly shorter and link propagation delays fit within a single clock cycle [42].

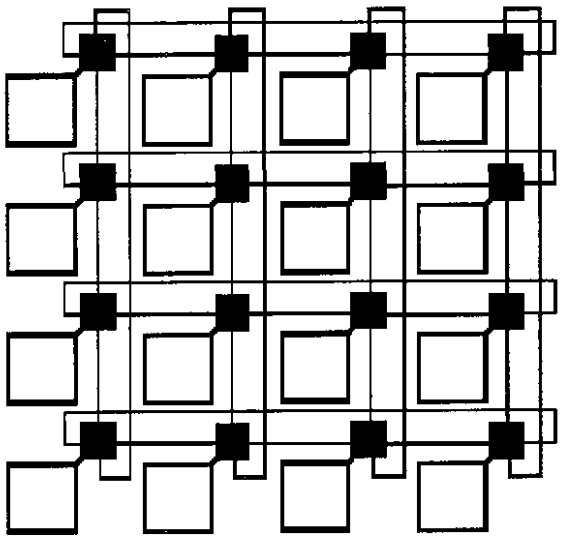

Figure 3 2D Torus [2]

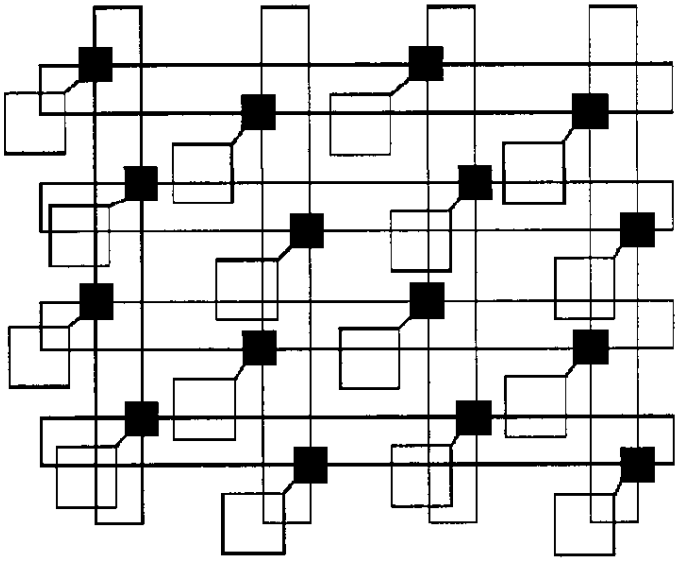

Figure 4 2D Folded Torus[2]

2.2.4 Octagon

The Octagon [19] architecture has its own advantageous properties. Every pair of nodes has a maximum two-hop path to communicate with each other. The basic model consists of eight IP blocks and 12 bidirectional links, as shown in Figure 5. The nodes are arranged in a ring and there is a central connection point in the center of a ring. Every node is also connected to the neighboring nodes. The node consists of IP block and a switch. Every switch has three connection ports. Usually the architecture uses a simple short-path algorithm. The throughput is higher than of the shared bus and crossbar interconnect if properly designed. This requires a development of good interconnection scheduler.

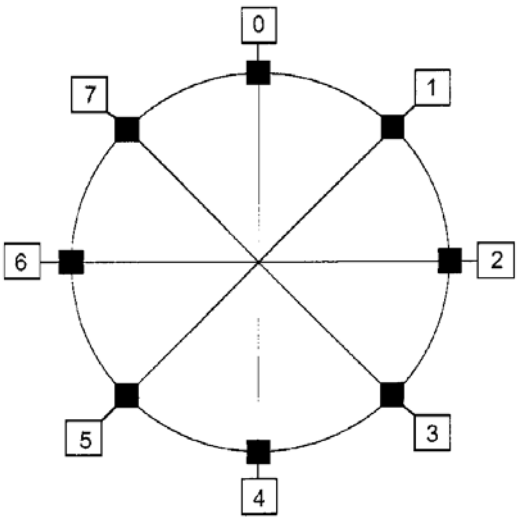

Figure 5 Octagon [2]

If we consider the scalability of the network then while increasing the network size, octagon is extended in multidimensional space. For large number of nodes this architecture may significantly increase the wiring complexity.

2.2.5 BFT

Another proposed interconnect template is BFT (Butterfly Fat Tree) [9], [20]. In this tree based architecture, IP blocks are placed at the leaves and switches are placed at vertices (Figure 6). Each switch has two parent ports and four child ports. In order to label the nodes, (l, p) coordinates are given to each node where l shows the level of the node and p shows the position of a node within that level. The address of a lowest

level is zero and the addresses of all IP's range from 0 to (N-1). There are N/4 switches at the first level and at the j^{th} level there are $N/2^{j+1}$ switches. The number of switches at each level reduces by 2. Unlike a simple mesh, where there is one switch for every four IP blocks, this topology requires one switch for every two IP blocks.

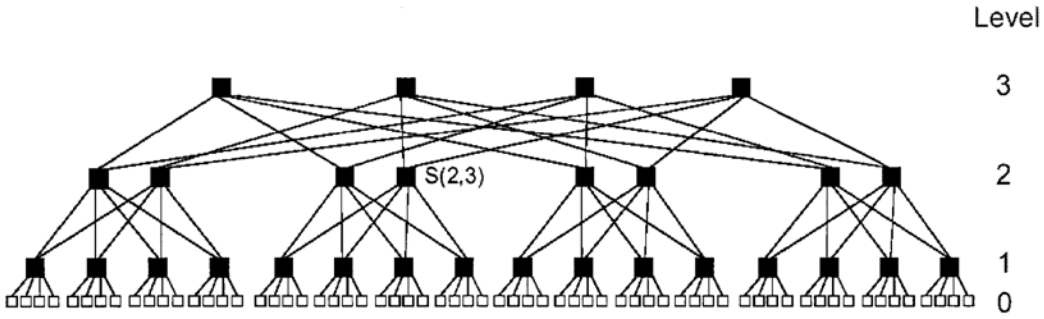

Figure 6 BFT Architecture [2]

2.3 Switching Techniques

Switching techniques define the way and time of connections between input and output ports inside a switch. There exist various switching techniques, but the most popular ones are Circuit Switching, Packet Switching, Virtual Cut-Through Switching and Wormhole Switching.

2.3.1 Circuit Switching

In circuit switching [13] the physical path from source to destination is reserved for the entire duration of data transmission. This is realized by injecting the header flit

into the network. The header flit contains the destination address and some additional control information. It moves toward the destination through intermediate routers reserving physical links it has passed. By the time it reaches the destination, the complete path has been reserved and the acknowledgement is sent back to the source. The reserved path may then be released by the destination or by the last bits of message itself.

This technique is advantageous when the messages are infrequent and long. In other words it is useful when the message transmission time is long compared to the pass set up time. However, it may block other messages while reserving the entire path, thus causing unnecessary delays.

2.3.2 Packet Switching

This technique is also called Store-and-Forward (SAF) [13] technique. The message is divided into fixed-length blocks, called packets. Unlike the circuit switching technique, which sets the path before sending a data, every packet is routed individually from source to destination. Every packet has routing and control information called packet header, which is used by intermediate routers to determine the packet's destination. Thus, the latency of a packet varies with the distance between source and destination. The longer the distance the greater is the latency.

Packet switching is advantageous when messages are short and frequent. It also fully utilizes the communication link, while the circuit switching may keep the reserved path idle for some time. However, the storage requirements at the individual routers can become extensive if packet size becomes large and multiple packets must be buffered at a node.

2.3.3 Virtual Cut-Through (VCT) Switching

In the packet switching technique, a packet must be received in its entirety before making any routing decisions. However, the size of a packet may be bigger than the *width* of a physical channel, so the transfer of a packet may take multiple cycles. The width of a physical channel is measured in bits and defines how many bits of information can be sent through the physical channel in parallel. The packet header is the first few bytes of a packet that can be received after first few cycles and it contains the routing information of a packet. Rather than waiting for an entire packet to be received, the router can start forwarding the packet header and following data as soon as the routing decision is made and the output buffer is free. In the absence of blocking, the packet does not have to be buffered in the output buffer and can cut through directly to the input buffer of the next router before the current router receives the complete packet. This switching technique is called virtual cut-through (VCT) [13]. The difference of this technique from packet switching is that the packets do not always have to be buffered in the intermediate routers; they are buffered only if the packet is blocked. That is why at high network loads the virtual cut-through switching behaves just like packet switching.

Only the packet header contains the routing information and the following data is simply forwarded along the same output channel as its predecessor. Therefore, transmission of different packets cannot be interleaved or multiplexed over the same physical channel.

Unlike circuit switching in VCT and SAF switching we do not have to reserve the whole path from source to destination since every packet contains routing information. A physical link between just two adjacent routers is reserved for the duration of packet transmission; it is released as soon as packet reaches the next router. In VCT and SAF switching each packet is routed independently from others and packets with the same source and destination may take different paths. In circuit switching links are underutilized, because if the header flit is stalled all the physical

links reserved so far cannot be used by anyone. VCT and SAF switching do not suffer from this kind of problems. As we said above circuit switching is good for light loaded networks with long messages while VCT and SAF switching are suitable for heavy loaded network configurations. It must be noted that we can make use of VCT's advantage over SAF only if the packet size is bigger than the width of the physical channel measured in bits, otherwise they perform similarly.

2.3.4 Wormhole Switching

In wormhole switching [13] the message is divided into flits. This is done in order to decrease the buffer size at routers and to achieve much faster routers. The input and output buffers of a router are large enough to contain a few flits.

A message is sent through the network at flit level in pipelined fashion. The header flit contains the routing information and builds a path in the network, which the other flits follow. In case of blocking VCT collects the following data at the current router, which requires bigger buffer size, while wormhole switching simply stops every flit in its current position. Thus, when a message is blocked it occupies several routers in the path constructed so far and a few flits need to be buffered at a router. As a result there is no need for a local processor memory to buffer messages, which significantly reduces average message latency. However, only the header flit contains the routing information and each incoming data flit is simply forwarded along the same output channel as the preceding data flit, which requires that the message must cross the channel in its entirety before it can be used by another message. This blocks the resources in case of stalled pipelines and makes the wormhole technique deadlock-prone. The problem can be solved by adding some control logic that splits the physical channel into several virtual channels. They will have their own buffers but share one physical medium, in other words each port of the router will have several queues for storing incoming flits instead of a single queue. In such an organization several buffers are associated with a single physical channel and form several virtual

channels, it is like adding lanes to a street. Each of the queues is allocated to a certain packet and contains flits of only that packet [31]. A physical channel is allocated to each of the virtual channels in flit-by-flit manner. The buffers sharing the same physical channel receive their flits alternately. If one of the packets is blocked the other one can advance further towards its destination.
In Table 1 a brief summary of switching techniques is given.

Table 2 Summary of switching techniques

Switching technique	**Communication entity**	**Path reservation**	**Buffer size**	**Resource utilization**
Circuit switching	Flit	Yes	Small	Poor
SAF switching	Packet	No	Large	Good
VCT switching	Packet	No	Large	Good
Wormhole switching	Flit	Yes	Small	Moderate

2.4 Routing Algorithms

A routing algorithm determines a path for a packet to reach its destination. It must be decided within each intermediate router which output channels must be selected for the incoming messages. There are various types of routing algorithms differentiated according to their key characteristics. In accordance with the place where the routing decision is made they may be grouped as centralized, source, and distributed routing

algorithms. If an algorithm is centralized the path is chosen by a centralized controller, if it is source routed then the route is determined by the source router prior to sending a packet, in distributed algorithms the path is chosen in a distributed manner at the intermediate routers. According to the way how they choose a path routing algorithms are broadly classified as deterministic and adaptive algorithms. Deterministic algorithms do not take into account network conditions when they take a decision that is why they always supply the same path from source to destination. But, it is not the case for adaptive ones in which network load, traffic conditions, information about available output channels are always taken into consideration.

Every algorithm has different impact on the network. Routing algorithms use a variety of metrics that affect the calculation of the optimal path for a message. Many properties of the interconnection network depend on the routing algorithm used because the complexity of an individual router has a significant impact on the complexity of the entire network. For example, if the routing algorithm is too complicated it will require extra hardware to realize the routing logic, moreover it may take much more time to make a decision about the direction where the message should be sent to. It will in turn lead to increase of packet latency. Deadlock, livelock and starvation freedom are also among those properties. This property shows the ability to guarantee that packets will not block or wander across the network forever or permanently stop and never reach its destination.

Deadlock: Deadlock is one of the situations that can postpone packet delivery indefinitely. It happens when a packet is requesting a resource that is held by another packet while holding the resource that is requested by other packet. There is a cyclic dependency between channels. Thus the packet may be blocked forever.
Deadlock is the most difficult problem to solve. There are three strategies that can cope with deadlock: deadlock prevention, deadlock avoidance and deadlock recovery.

Livelock: Livelock usually happens in adaptive routing schemes. It happens when a packet is running forever in circular motion around its destination, because the channels that are required to reach the destination are occupied by other packets.
Hot potato routing is an example for an algorithm that can cause a livelock. In this algorithm, whenever a desired channel is not available, a packet will pick any alternative available channel and move to the next switch instead of waiting. However, the alternative channels may misroute a packet around its destination.
In order to remove livelock several techniques have been proposed such as minimal path, restricted non-minimal path, probabilistic avoidance.

Starvation: Starvation may happen when a resource that was requested by a packet is always granted to other packets. Thus, the packet stops permanently in traffic, never getting the resource it needs. Starvation can be avoided by using correct resource assignment scheme.

2.4.1 Deterministic Routing Algorithms

Deterministic routing algorithms [13] always generate the same single routing path for a given pair of source and destination, usually choosing the shortest one. Only the addresses of current and destination nodes are used to compute the path. As messages with the same source and destination always use the same network path, they cannot use alternative paths to avoid blocked channels.

As it was said in the beginning of this section if source routing is used, the path is computed at the source node without considering any information about traffic. Otherwise, if distributed routing is used, routers make unique decisions at the intermediate nodes, based on current and destination node addresses. In both cases channel status is not considered while computing the output channel to be used. Deterministic algorithms should be progressive and profitable, which means that the header should move forward reserving a new channel at each routing operation,

under condition that the supplied channel always brings the packet closer to the destination. Thus deterministic routing algorithms use greedy algorithms, always choosing the shortest path.

The most popular deterministic algorithm is known as *dimension-order routing*. It is based on the idea that some topologies can be decomposed into several orthogonal dimensions, i.e. hyper cubes, meshes and tori. The distance between two nodes in these topologies is computed as the sum of the offsets in all dimensions. The algorithm reduces one of these offsets in each routing step. The offset of the current dimension must be equal to zero before the algorithm considers the offset of the next dimension.

Dimension–order routing is usually used for meshes and hypercubes. In 2D mesh it is called XY- or YX-routing depending on the dimension in which a packet travels first. The algorithm is deadlock-free for *n*-dimensional hypercubes and meshes, as their channel dependency graph (CDG) is acyclic. CDG is a directed graph where channels are represented by vertices and edges are pairs of channels connected by a routing function [28]. However, the CDG for some topologies has cycles. In order to remove cycles, physical channels may be split into virtual channels.

Most commercially available parallel machines usually use distributed deterministic routing as it is simple and fast. But distributed deterministic routing assumes that the traffic is uniform. In case of non uniform traffic the performance of distributed deterministic routing in terms of latency and throughput is very poor [13].

2.4.2 Adaptive Routing Algorithms

Adaptive routing algorithms do not restrict a message to a single path when traveling from the source to the destination. While making a decision the current network

conditions are considered. This makes the routing more flexible and reduces unnecessary waiting time delays, providing better fault tolerance.

Adaptive routing algorithms contain two functions: routing and selection. Routing function gives a set of output channels based on the current node and destination node, Selection function selects the most appropriate output channel from that set. The selection function always supplies with a free channel. Thus, messages can follow alternative ways instead of waiting for a busy channel. Non greedy algorithms, which can supply channels that send packets away from its destination, are also allowed.

Adaptive algorithms also allow backtracking technique, which enables the header to backtrack, releasing previously reserved channels, thus systematically searching for appropriate path. For deterministic algorithms backtracking technique is useless as the message will go by the same path again. Adaptive routing algorithms increase routing flexibility but the hardware becomes more complex and slower.

2.4.3 Minimal Adaptive Routing

Minimal routing algorithms always use the shortest path in order to reach destination. While being adaptive they restrict the routing direction in some level. One of the examples for minimal adaptive routing algorithms is double *Y*-channel routing algorithm [21]. This algorithm divides the network into several sub-networks. The packet is sent via particular sub-network according to the location of destination. The network is usually divided into $+X$ sub-network and $-X$ sub-network. Here *Y* dimension has a pair of channels and *X* dimension has unidirectional channel. If the location of the destination node is bigger than the location of source node, in other words if $d_x > s_x$, than the packet is sent through $+X$ sub-network. Otherwise $-X$ sub-network is used. If $d_x = s_x$ then either sub-network can be used.

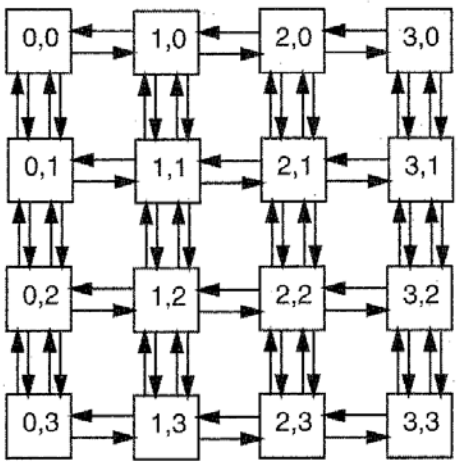

Figure 7 Double *Y*-channel 2D mesh [22]

The double *Y*-channel algorithm is minimal and fully adaptive, which means that the packet is delivered through any of the shortest paths. In order to avoid deadlock, channels should be ordered in appropriate manner [21]. The packet that is sent to the destination should path the channel in descending order, in other words, the channel label that was passed must be bigger than the channel label that is going to be passed. This algorithm is impractical when *n* is large (*n* is the number of dimensions), due to the additional channel requirement.

2.4.4 Non-minimal Adaptive Routing

Unlike the minimal adaptive routing algorithm, where a packet searches only for a shortest path, non-minimal adaptive routing allows the packet to take a longer path if there is no available shortest path.

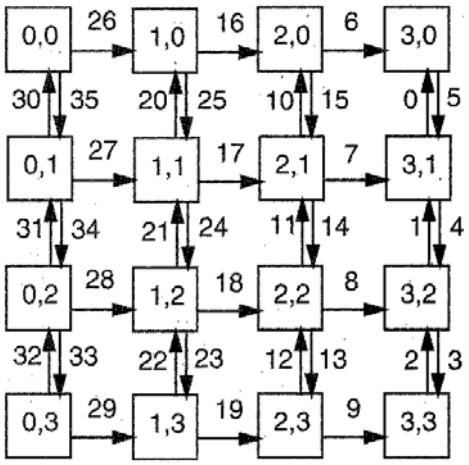

Figure 8 +*X* sub-network and labeling [22]

This technique is fully adaptive and can be achieved by few more additional channels. Two non-minimal routing algorithms were proposed by Dally et al. [23]. One of them is static dimension reversal routing algorithm, where every two adjacent nodes are connected by *r* pairs of channels. Here, the network is divided into *r* sub-networks and the i^{th} sub-network consists of all the i^{th} pair channels. Each packet header stores additional value *c* which is initially set to zero. The packet can move in any direction in its own sub-network, but when the packet moves from high dimensional channel to low dimensional channel the *c* field is increased by one. If *c* reaches the value *r*-1 then the packet must switch into the deterministic dimension-ordered routing algorithm. The packets can be routed also in reverse dimension order. Parameter *r* restricts the number of times it can happen.

Another algorithm is called dynamic dimension reversal routing algorithm. Here, the channels are divided into two classes: adaptive and deterministic. At first, packets are sent through adaptive channels, moving in any direction. But, when a packet reaches a node, where all output channels are busy by packets with values of *c* that is smaller

or equal to its own, it must switch to the deterministic channels. After entering the deterministic channels a packet cannot return to adaptive channels. The algorithm is deadlock-free.

2.4.5 Turn Model

This algorithm was proposed by Glass and Ni [14] for *n*-dimensional meshes. It suggests a systematic approach to the development of adaptive routing without additional channels. The algorithm supports both minimal and non-minimal adaptive routing and is partially adaptive.

The packet, while traveling through the network, switches from one dimension to another, in order to reach the destination. Switching from one dimension to another is called turn. If the packet changes its direction without moving to another dimension then this is 180-degree turn. Usually physical channels are split into virtual channels and the packet may move from one virtual channel to another. If it moves from one virtual channel to another and is still in the same dimension and direction then this is a 0-degree turn. Turns can form a cycle. It is known that, deadlock occurs when the packet routes contain turns that form a cycle. The algorithm is based on the idea that, if there is no cyclic dependency between channels then deadlock cannot occur. So, the concept of the algorithm is to prohibit the smallest number of turns such that cycles are prevented. There are several steps that can be useful in developing maximal adaptive routing algorithms:

- Classify channels according to the direction in which they route packets.
- Determine all possible turns that can occur between one direction and another. 0^0 and 180^0 turns are not considered.
- Identify the cycles that can be formed.
- Prohibit one turn in each cycle.

- 0^0 and 180^0 turns cannot be prohibited as they are needed for non-minimal routing algorithms or if there are multiple channels in the same direction.

One of the examples for this algorithm is west first routing algorithm, where it first routes a packet west, if necessary, and then adaptively south, east and north.

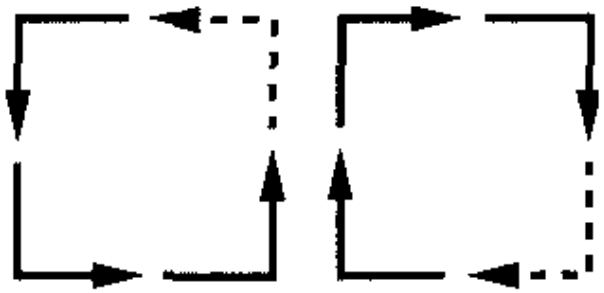

Figure 9 Six turns (solid arrows) allowed in west-first routing [22]

Figure 9 shows the turns that are prohibited. Thus, the packet can make only six turns and two turns to the west are not allowed. Therefore, to travel west, a packet must begin in that direction. For minimal routing, the algorithm is fully adaptive if the destination is on the right side of the source, otherwise, it is deterministic. For non-minimal routing the algorithm is adaptive in either case.

2.4.6 Randomized Routing Algorithms

DOR algorithm is considered to be one of the most popular deterministic routing algorithms due to its simplicity for implementation and good performance according

to average packet delay and throughput metrics [35]. As we can see in Figure 27 not the whole available bandwidth of the network is used. Sometimes minimality constraints are relaxed to some extend and randomization is added for more efficient usage of the network bandwidth.

Valiant and Brebner [34] proposed one of the best known randomized algorithms named Valiant. This algorithm has two phases. In both of the phases it uses dimension-order routing. In the first phase a random node is selected, and a packet is sent there. In the second phase, it routes the packet from that random node to its destination. Valiant is non-minimal and tries to avoid congestion in the network. Packets with the same source and destination are unlikely to take the same path, as they will have different intermediate nodes, which are selected randomly. Several sets of buffers are needed in order to avoid deadlocks. In a mesh topology, it requires two sets of independent buffers per communication link. In Valiant routing algorithm livelocks cannot occur, as the packet reaches its destination eventually, but it may take longer path than actually required. The path taken is a drawback of this algorithm, because its length increases according to the topology being used. For example, in a mesh, the packet's path may be doubled. This means that the Valiant routing results in longer routing times. Hence, the demand on network bandwidth is also doubled.

Another routing algorithm which uses randomization is Randomized, Oblivious, Multi-phase, Minimal (ROMM) [35] algorithm. It inherits minimality of DOR, and randomization of Valiant. Minimality assures that the path taken by a packet will be minimal. Randomization is used to assure that packets with same source and destination will not take the same path. These properties are combined to avoid congestion. This algorithm is oblivious, thus, making it easy to implement in contrast to adaptive algorithms which are more complex. As ROMM algorithm uses only minimal paths, it is constrained in randomization unlike Valiant which uses full randomization.

ROMM selects random nodes within the range of a minimal path a message is required to follow in order to reach the destination. It routes a packet in *p*-phases. A *p*-phase ROMM algorithm has *p-1* randomly selected nodes $Z_1, Z_2, \ldots, Z_{p-1}$ between source and destination, such that, those Z_i 's must be on minimal path from source to destination.

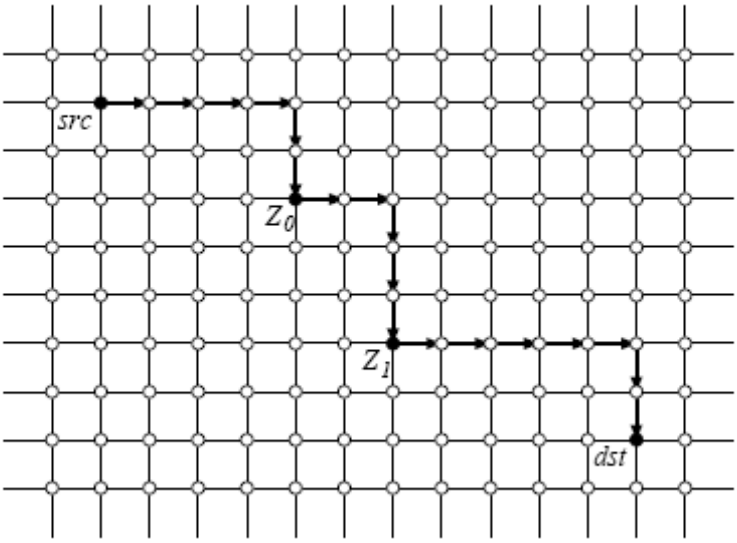

Figure 10 A possible path from *src* to *dst* using 3-phase ROMM on a 2D mesh

Figure 10 illustrates a possible path from source to destination in a 3-phase ROMM. Here, source node determines the path that would be taken by message using dimension-order routing and chooses a random node that lies within that path, in our case Z_0. Then message is routed from *src* to Z_0 using dimension-order routing. Now, Z_0 becomes a source node and chooses another random node that lies along the minimal path from Z_0 to dist, in this case Z_1, and send a message using dimension-

order routing. Z_1 in its turn routes a message to *dst* again using dimension-order routing.

If *p* phases are used then ROMM algorithm requires *p*-virtual channels for wormhole routed mesh network, in order to avoid deadlocks. The algorithm becomes very costly as *p* increases, as it increases memory usage adding virtual channels and complicates routing with additional logics required to manage virtual channels.

Another algorithm which allows several routes is O1TURN [36]. In O1TURN a network is partitioned into two virtual networks. One of them is XY-routed the other is YX-routed. It uses just one phase unlike ROMM and Valiant. However, it also provides more than one path, because when the flit is injected into the network it may choose one of the networks, more precisely it is randomly sent into one of the virtual networks. According to [36] O1TURN outperforms ROMM, Valiant and DOR under non-uniform traffic. But DOR is still better under uniform traffic pattern.

2.4.7 Comparison of Routing Algorithms

In this section we introduce general comparison of routing algorithms mentioned in previous sections as it is indicated in the literature. Deterministic algorithms in comparison to adaptive ones achieve higher throughput behavior under uniform traffic pattern in 2D mesh with the same number of VCs and have lower latencies at higher throughputs [13][14]. However, adaptive algorithms outperform them when used in 2D torus [13]. Deterministic algorithms suffer from channel underutilization while adaptive algorithms distribute the traffic more uniformly across the network and do not have such a shortcoming. Under non-uniform traffic pattern adaptive algorithms perform much better than deterministic ones in terms of throughput and latency. It is due to their ability to provide several routes between the same pair of source and destination.

Performance evaluation of O1TURN, DOR, and ROMM algorithms and their comparison were presented in [36]. DOR outperforms O1TURN under uniform traffic when the network size is small (4x4). It achieves lower latency and higher throughput behavior. When network size increases O1TURN behaves better. ROMM has the worst performance (greater latency, lower throughput) under uniform traffic. O1TURN is always better than DOR under non-uniform traffic pattern regardless of the network size. Under non-uniform traffic the performance of O1TURN is almost the same or better than the best of DOR or ROMM.

2.5 NoC Traffic Models

One of the main problems of NoC design is generation of realistic traffic pattern. Despite many researches are conducted very few of them are related to traffic models. Generally, Poisson packet arrival process is used for NoC traffic generation. However, it was shown that real-world NoC traffic is self-similar in nature [39]. Self-similar traffic can be modeled by aggregating many on/off message sources with Pareto distributed ($F(x) = 1-x^{-\alpha}$, with $1<\alpha<2$) on/off periods [43]. Most researchers assume uniform destination distribution. But, this is also not correct, because destination distribution depends on the application being modeled and how different functions of the application are mapped onto the SoC cores.

Marculescu proposed a traffic model for NoCs, and modeled MPEG-2 video applications, but his model is related to pair-wise traffic rather than the traffic over the entire network[40]. In [41] traffic traces of 30 applications were gathered and empirically derived and modeled. There, spatial and temporal variations of NoC traffic are captured by 3-tuple, where each component represents a statistical distribution (temporal burstiness, spatial hop distribution, and spatial injection distribution). However, nobody has introduced complete traffic model for NoCs, because NoC traffic is application-specific and it is very difficult to obtain realistic

traffic traces, it is even more difficult to predict what NoC traffic will look like in the future [41].

CHAPTER III

ANALYSIS OF NETWORK ON CHIP ROUTING ALGORITHMS

In this chapter we give description of the canonical router architecture and the simulator model assumed in this work, show the correctness of our simulator, present how XY routing algorithm performs under different configurations. Then performance evaluation of O1TURN routing algorithm is conducted. XY routing was chosen because of its wide popularity. The choice of O1TURN is explained by the fact that our implementation is based on its concept of dividing a single physical network into two virtual networks and it outperforms ROMM and Valiant routing algorithms in spite of their randomized features. However, O1TURN, ROMM , and Valiant are also based on XY routing.

3.1 Canonical Router Architecture

The performance of interconnection networks depends on the performance of the routers which they are constructed from. As routers are fundamental components of a network along with topology and flow control policies, the optimization of a router significantly increases network throughput. Several router delay models have been proposed. They show the performance of a router according to the implementation complexity.

The first router delay model was proposed by Chien [24], where he considered the implementation complexity of a router. In his model he defined the router with

several functions, which were on so called critical path. Those functions were address decoding (AD), crossbar (CB) and routing arbitration (RA), crossbar traversal, and virtual-channel (VC) allocation (Figure 11). Per-hop router latency was accepted as a total delay of these functions.

However, Chien did not consider the pipelining in his model, assuming that the entire critical path fits within a single clock cycle. He also assumed that the crossbar must provide each virtual channel with a separate port. This brings complications to crossbar design as the number of virtual channels increase. The extended form of Chien's model was proposed by Duato and Lopez [25]. In their model pipelining was taken into account. They proposed a three-staged pipeline: routing stage, switching stage and channel stage.

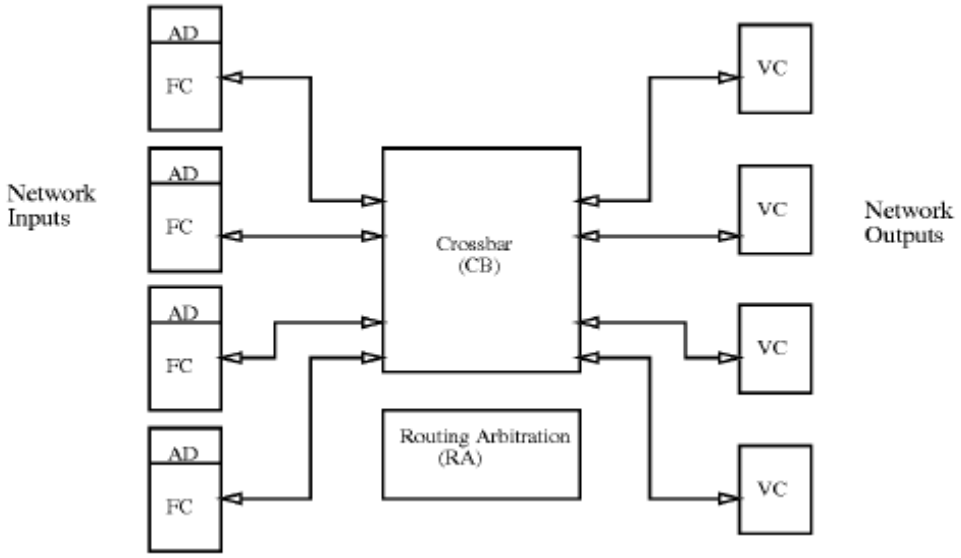

Figure 11 Canonical architecture proposed by Chien [24]

These proposed models assumed that the clock cycle time depends only on router latency. Peh et al. [26] suggested a more realistic router delay model, where the clock cycle time was fixed and the pipeline stages might vary. According to this delay model a router design consists of general and specific models. The overall view of a router and its pipeline stages were presented by general model while specific router model was used for thorough investigation of delays of each stage and determining the clock cycle time.

In this study, we define the general model only as this model will be considered in our implementation.

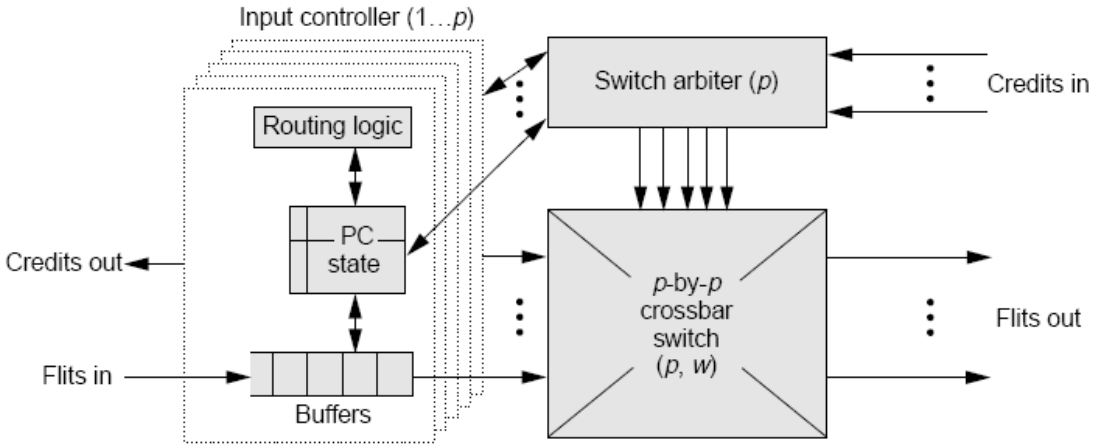

Figure 12 Canonical wormhole router architecture [26]

Figure 12 presents a canonical wormhole router architecture, where *p* is the number of physical channels and *w* is the channel width in bits as defined in the previous section. An incoming flit must pass through routing, switch arbitration and switch traversal stages. Each time a new flit arrives in a router the input controller identifies

its type: header, data or tail flit. If it is a header flit its destination field is forwarded to routing logic, the flit itself is placed into buffer, and since input controller is also responsible for setting states to a channel, it changes its state to routing. When the routing logic returns an output port for a packet (we say packet because the reservation is made for the duration of entire packet), input controller sets the channel state to switch arbitration and sends a request for that output port to the global switch arbiter. Global switch arbiter provides a requester with available output port and marks that output port as unavailable to other requests. As soon as an output port is granted to input controller it sets the input channel's state to switch traversal and sends the flit through crossbar if there is any available buffer space in the next node. Data flits only pass switch traversal stage since all the routing information is in the header flit and other flits simply follow the reserved path. The input controller buffers data flits in input queue and sends them to the output port reserved by the header flit. The tail flit sets the channel state to idle and signals to global switch arbiter to release the reserved output channel upon leaving the input queue.

Figure 13 illustrates canonical virtual-channel router architecture. It adopts additional parameter *v*, which is a number of lanes per physical channel. Crossbar ports are shared between lanes of a physical channel and allocated flit-by-flit. The architecture suggested in [26] uses a crossbar that has just *p* ports. In our implementation we assume that the crossbar has *p*x*v* input ports, which means that the number of ports is equal to the total number of virtual-channels.

Each lane has a separate buffer and a state. The packet passes through routing, virtual channel allocation, switch allocation and crossbar traversal stages. Each flit has a VC-identifier (VCID) field, which defines the lane where a flit is destined to. When a header flit arrives input controller decodes its VCID and injects the packet into virtual channel with associated VCID, where it is buffered. Virtual-channel enters a routing state and sends a flit's destination field to routing logic which returns the output virtual channel for a packet. Input controller then sets the virtual-channels state to virtual-channel allocation. Virtual-channel sends a request for those output

virtual-channels to global virtual-channel allocator, which in turn returns the available output virtual-channels and updates the states of those output virtual-channels to unavailable. Once an output virtual-channel is allocated the header flit sends a request to global switch allocator for output port. When a request is granted the header flit leaves for the next node, with its VCID field changed to the recently allocated virtual-channel's VCID.

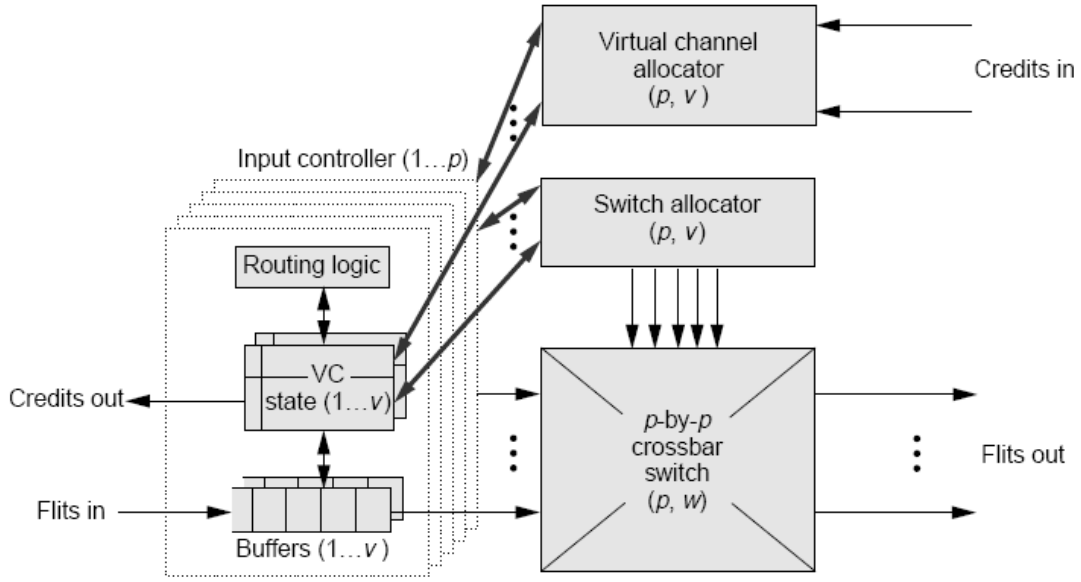

Figure 13 Canonical virtual channel router architecture [26]

When data flits arrive they inherit the output virtual-channel reserved by their header flit so they can immediately send a request to global switch allocator for output port. The tail flit signals the virtual-channel allocator to release the output virtual-channels reserved by its header flit after gaining access to crossbar passage.

In virtual channel router we have one extra pipeline stage for any kind of flits. So, wormhole and virtual channel routers contain three and four pipeline stages for header flits, and one and two stages for both data and tail flits.

3.2 Simulator Model

As it was mentioned before we have three pipeline stages in wormhole router model and four pipeline stages in virtual channel router model. We implemented in C++ the router models shown in Figures 12,13. The network is organized in a 2D-mesh (CLICHE) topology, used switching technique is wormhole switching. Also we consider the link delay between any two routers to be equal to one clock cycle and each pipeline stage is also assumed to take a single clock cycle.

In conducted simulations routers are exactly the same as for wormhole and virtual channel routers in [27]. All of the mentioned pipeline stages were implemented as separate functions. In our simulator all the generated packets are stored in packet buffers before being injected into the network. As long as free buffer space is available in injection queues a packet is split into flits and placed in there. In wormhole router when header flit arrives, input controller decodes its type, since the type field is set to “header”, its destination address is forwarded to routing logic, flit itself is placed into the buffer, and channel state (Figure 14) is set to *routing*. The routing logic returns output port. A request for that output port is sent to the switch arbiter and the channel state is set to *arbitration*. Switch arbiter resolves contention and assigns available outputs to inputs which bid for them. If the output port is granted for any of the input ports, it is marked as unavailable. When the grant is accepted the channel state is changed to *switch traversal* and header flit is sent over the crossbar to the next router over the link. When the next flit arrives the path is already reserved and it is directly forwarded to the output port. If the incoming flit is a tail flit, switch arbiter is signaled to release all the reserved resources and to set input channel state to *idle* on its departure [27]. The virtual channel router is a bit

different. We said that it has four pipeline stages. They are also implemented as separate functions. The difference from wormhole router is that after all the connections between input and output virtual channels are arranged, input VCs sharing the same output port send flits in a round-robin fashion. If one of the input queues is empty or its corresponding buffer in the next node is full it misses its turn. If any of output ports is reserved for a single input VC then that input VC can transmit each cycle, because it does not have to share the reserved output with others and a switch is always allocated to it.

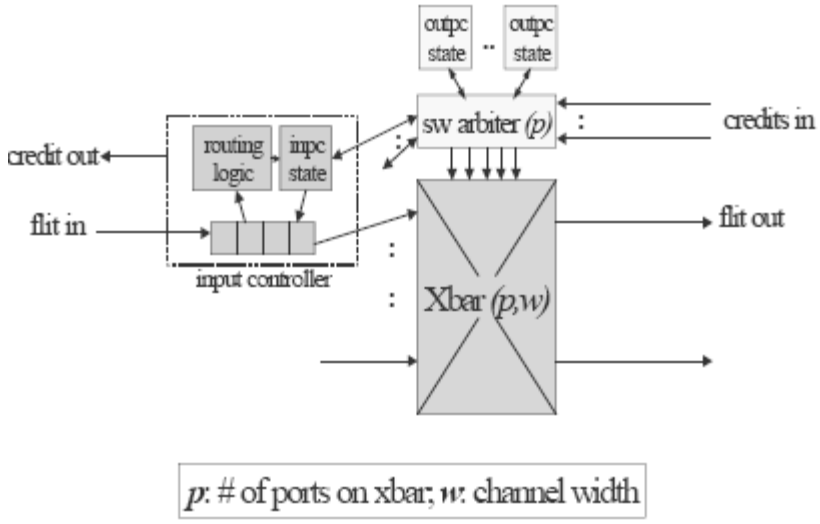

Figure 14 Canonical wormhole architecture [27]

We have the following entities in the simulator:

- Router – has 5 input and 5 output ports (4 input and 4 output ports are for communication with adjacent switches, left 2 ports are connected to local resource), priority matrix, routing function, arbitration function and crossbar

traversal function. Each input has its own FIFO-queue, for storing incoming flits.

- Link – connects output and input ports of two adjacent routers for flit transmission.
- Control Link – is used to transmit control data between two adjacent routers.

Each router except those at the edges has 4 neighbors (Northern, Southern, Eastern, and Western). We have packet generators connected to local port of each router. Packets are divided into fixed sized flits (flow control units).

We simulated our model for 4x4, 8x8, 12x12 and 16x16 networks with fixed sized packets and exponentially distributed packet lengths holding average packet lengths equal to 10-, 32- and 64-flits for each one, changing buffer sizes in the range of 3-10 flits per input port. Destinations are uniformly distributed. Each simulation is first run 3000 cycles for warm-up without gathering any statistics and then another 35 000 cycles collecting the network performance data. All the simulations were performed under Poisson traffic. However, as it was discussed previously (see Section 2.5), Poisson traffic model and uniform destination distribution are not suitable for generating realistic NoC traffic. They are still used because no other realistic traffic model has been proposed yet.

3.3 Correctness of the Simulator

In the following, we compare our simulation results of DOR and O1TURN algorithms with those in the literature in order to determine if our simulator works properly. We simulated these two algorithms under the same assumptions (packet size = 5 flits, buffer size = 8 flits per physical channel, network size = 8X8) as in the papers [26][27][36], and got the results given in Figures 15-19. In these figures throughput is given as a fraction of network capacity, which is defined as $4/k$

packets/node/cycle (packet size is assumed to be exactly one flit) [36] for *k*x*k* networks when *k* is even, where *k* is the number of nodes along each dimension. As we can see our results are approximately the same as in the original papers. There is a slight difference when DOR with 2 VCs is used. It is due to changes made to original algorithm, where routing stage is performed just one time. In our simulator we perform routing stage each time when contention for all the free VCs previously returned by a routing logic is lost. Source queuing delays (the time packets have to wait in packet buffers before being injected into the network) of packets are taken into account in Figures 15-19. This explains why the delays are different from those presented later in this thesis.

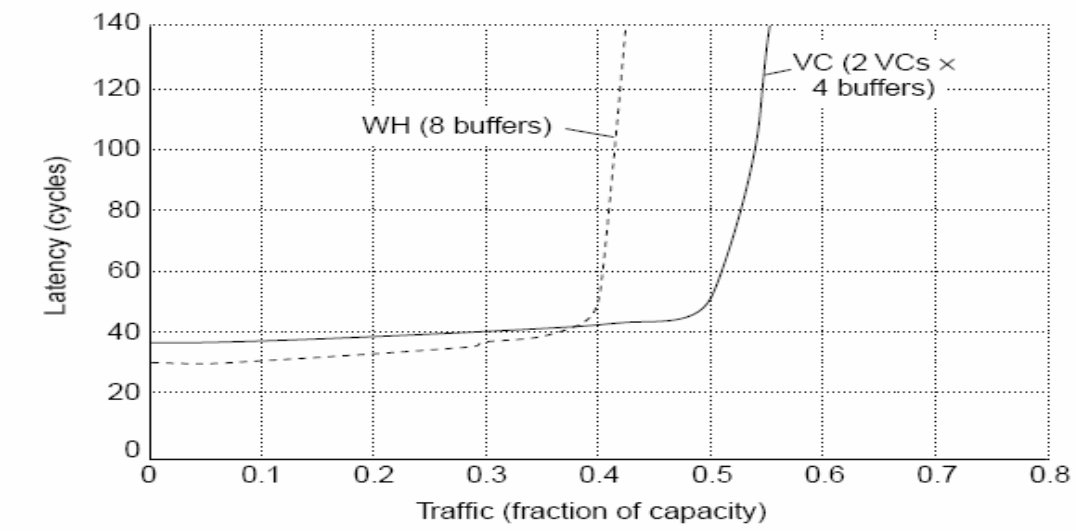

Figure 15 Latency-throughput curves of DOR algorithm with 1 and 2 VCs [26][27]

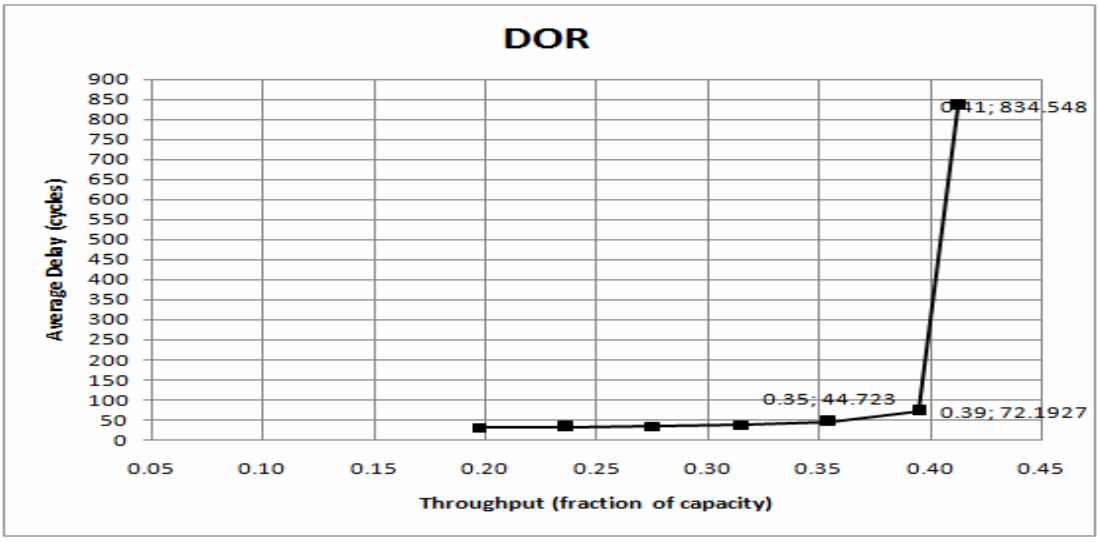

Figure 16 Latency-throughput curve of DOR algorithm with a single VC (obtained by our simulator)

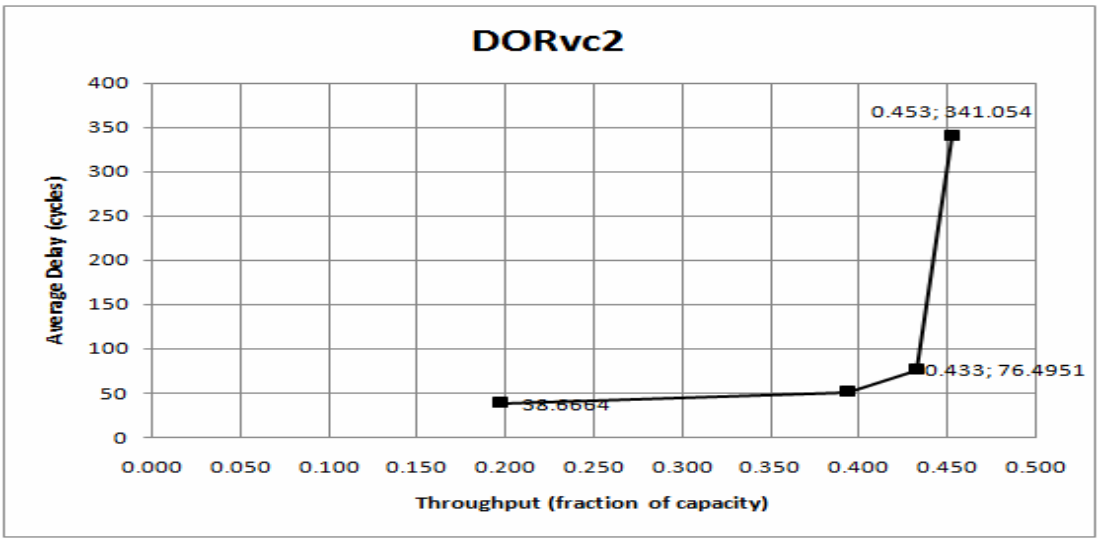

Figure 17 Latency-throughput curve of DOR algorithm with 2 VCs (obtained by our simulator)

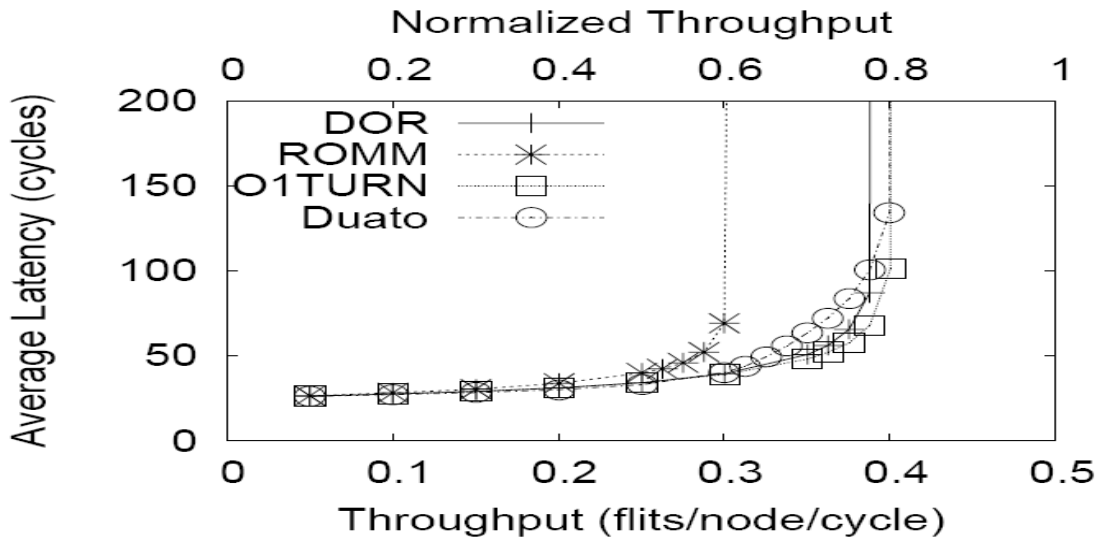

Figure 18 Latency-throughput curve of O1TURN algorithm [37]

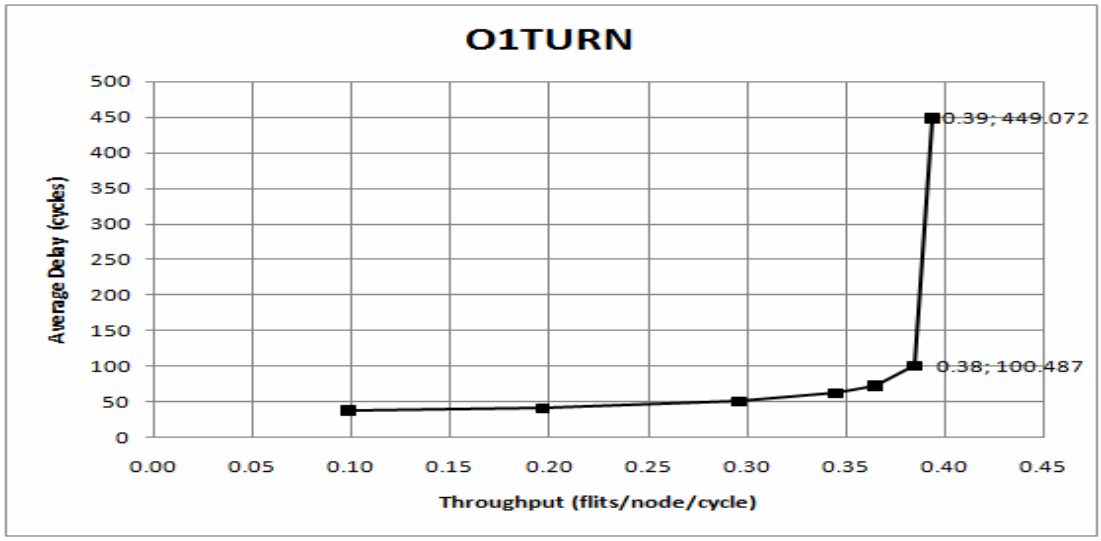

Figure 19 Latency-throughput curve of O1TURN algorithm (obtained by our simulator)

3.4 Performance Evaluation of XY Routing Algorithm

In this section we provide performance evaluation of XY routing algorithm. In Figure 20 we present its pseudocode. It is called if the incoming flit is a header flit. First the offsets in X and Y dimensions are calculated, then the appropriate output channel is determined and returned.

```
Algorithm: XY-Routing for 2-D Meshes
Inputs: Coordinates of current node (Xcurrent, Ycurrent)
        and destination node (Xdest, Ydest).
Output: Selected output Channel
Procedure:
Xoffset := Xcurrent − Xdest;
Yoffset :=Ycurrent −Ydest;
if Xoffset = 0 and Yoffset = 0 then
        Channel := Local;
else if Xoffset < 0 then
        Channel := East;
        else if Xoffset > 0 then
                Channel := West;
        else if Yoffset < 0 then
                Channel := North;
        else if Yoffset > 0 then
                Channel := South;
endif
```

Figure 20 XY routing algorithm for 2D-meshes [13]

3.4.1 Effect of Packet Size on Performance

In Figures 21-23 the effect of packet and buffer size on overall performance from throughput point of view is given. The network size is 4x4. For other network sizes

we observed the same trend in performance, the difference is that the larger the network size the earlier it saturates.

In all the figures throughout this thesis load is defined as follows: if our packet length is 10 flits we assume that load=1 means that our interarrival rate is one packet per 10 cycles, in other words 1 flit/cycle. If load=0.5, it gives us 0.5 X 1 / 10 = 1/20 packets/cycle. Delay of a packet is taken as the period of time elapsed since the generation of header flit at the source node and ejection of its tail flit at the destination node, the time spent by the packet in the packet buffer is not taken into account. When a flit reaches its destination it is immediately ejected from the network. The throughput is given as the ratio of received packets to sent packets.

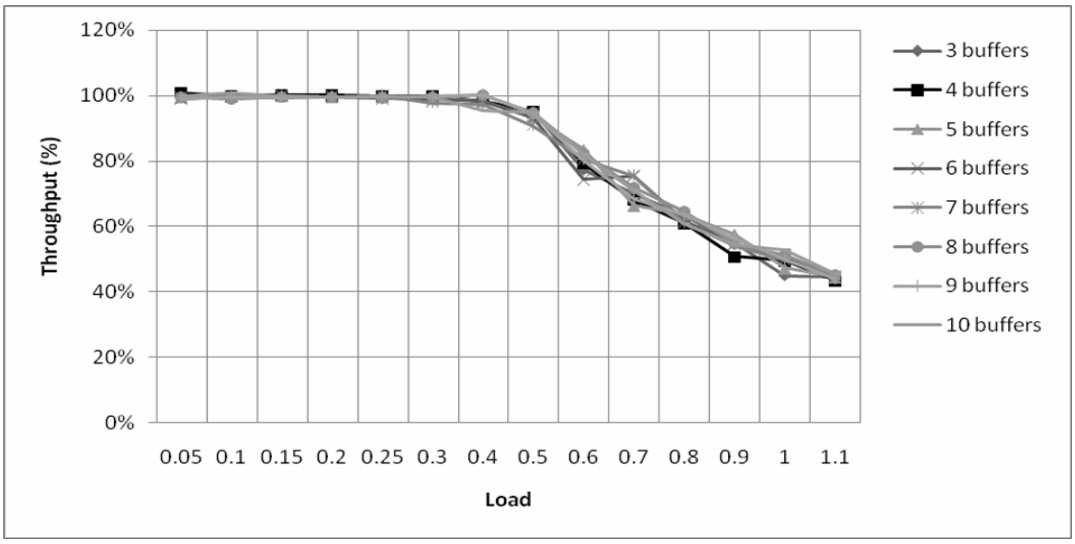

Figure 21 Effect of packet size on throughput (packet size = 10 flits)

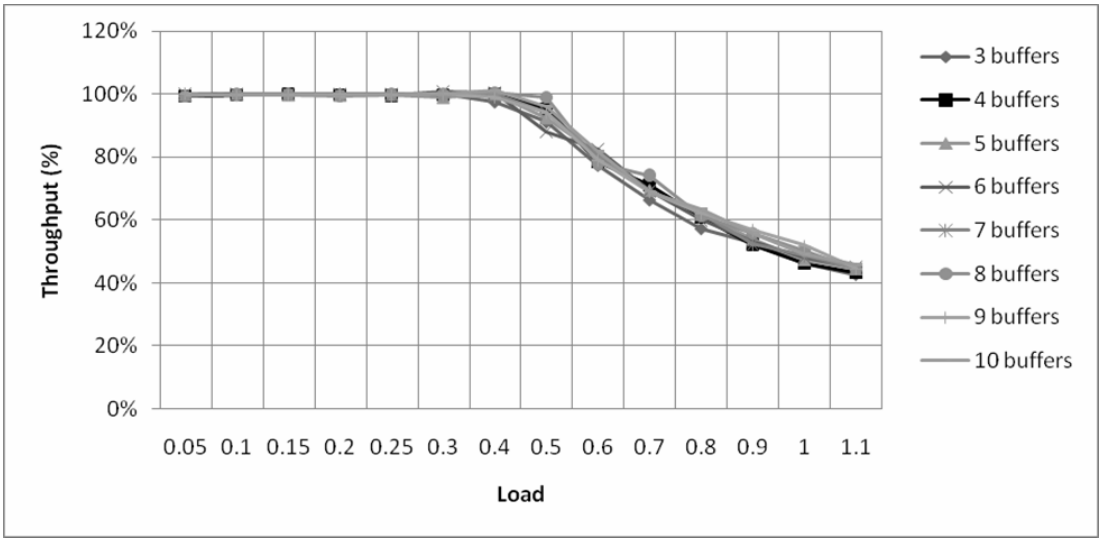

Figure 22 Effect of packet size on throughput (packet size = 32 flits)

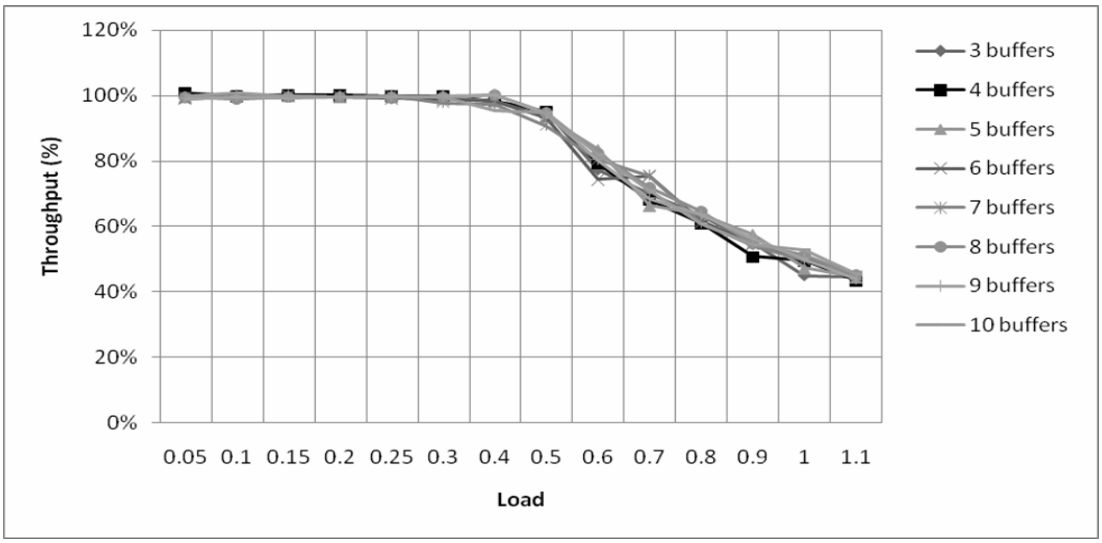

Figure 23 Effect of packet size on throughput (packet size = 64 flits)

As we can see in Figures 21-23 the buffer size does not affect the performance of the system [13], that is, the throughput behavior does not change much. When the network size is kept constant and the packet length is prolonged there cannot be seen any change in throughput since we measure it in percentage of received to sent packets in a fixed period of time. But, in terms of numbers of received packets, it would be completely different, because the larger the packet size the less its generation rate, in other words with growth of packet size its generation rate decreases but number of flits injected into the network remains unchanged. All the three figures above have the same throughput behavior, it means that the system has limited capacity and its utilization does not depend on the quantity of packets, it depends on the number of flits.

3.4.2 Effect of Network Size on Performance of XY-Routing Algorithm

In the following figures, the effect of network size on the average delay and throughput are depicted.

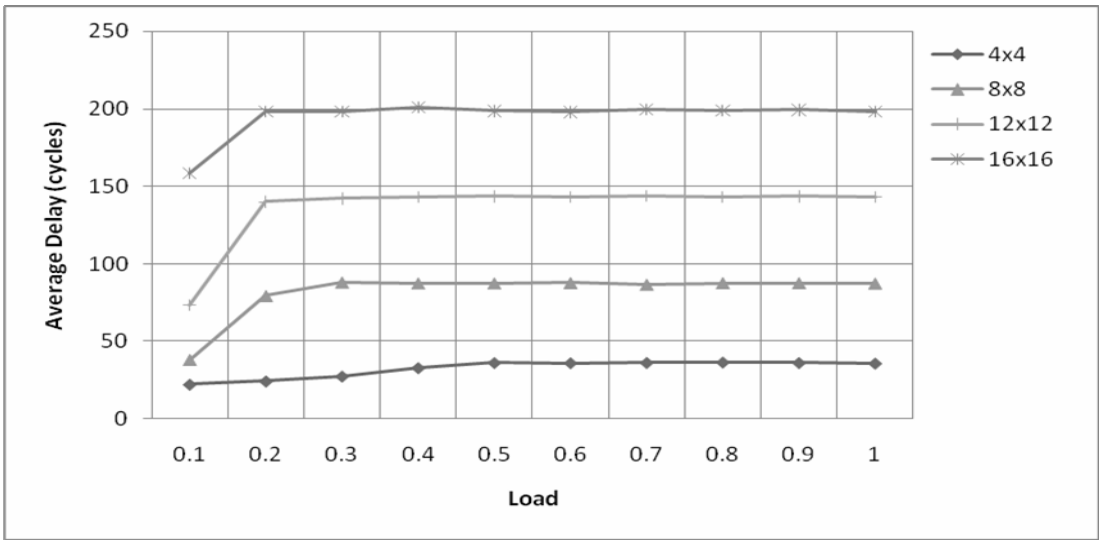

Figure 24 Effect of network size on average packet delay (packet size = 10 flits)

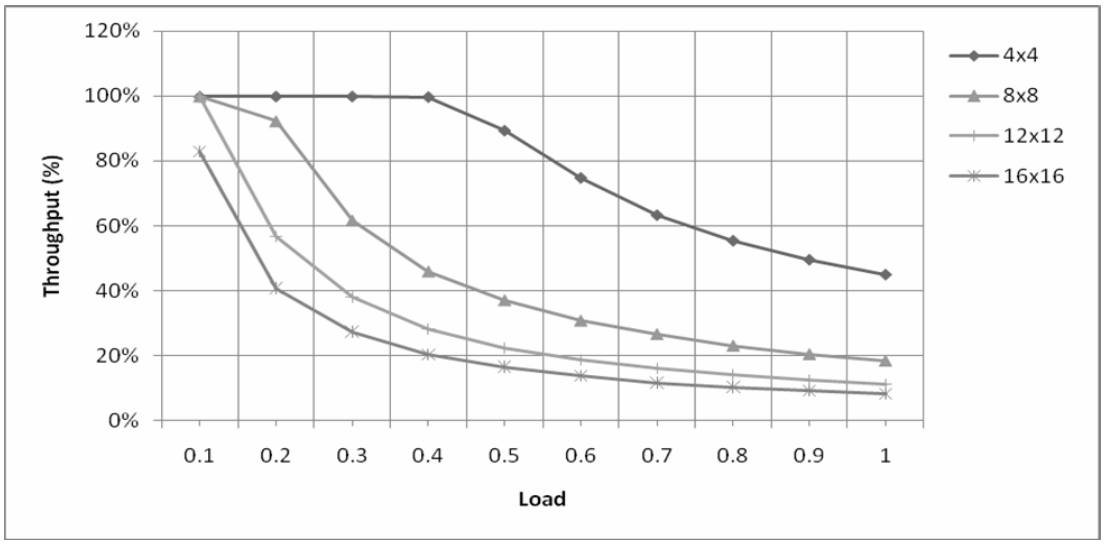

Figure 25 Effect of network size on throughput (packet size = 10 flits)

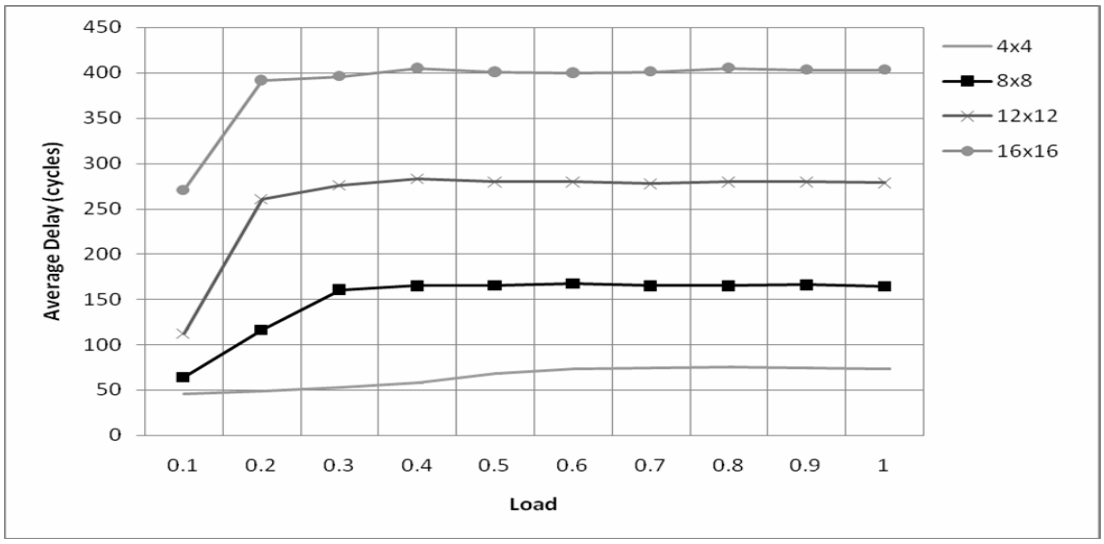

Figure 26 Effect of network size on average packet delay (packet size = 32 flits)

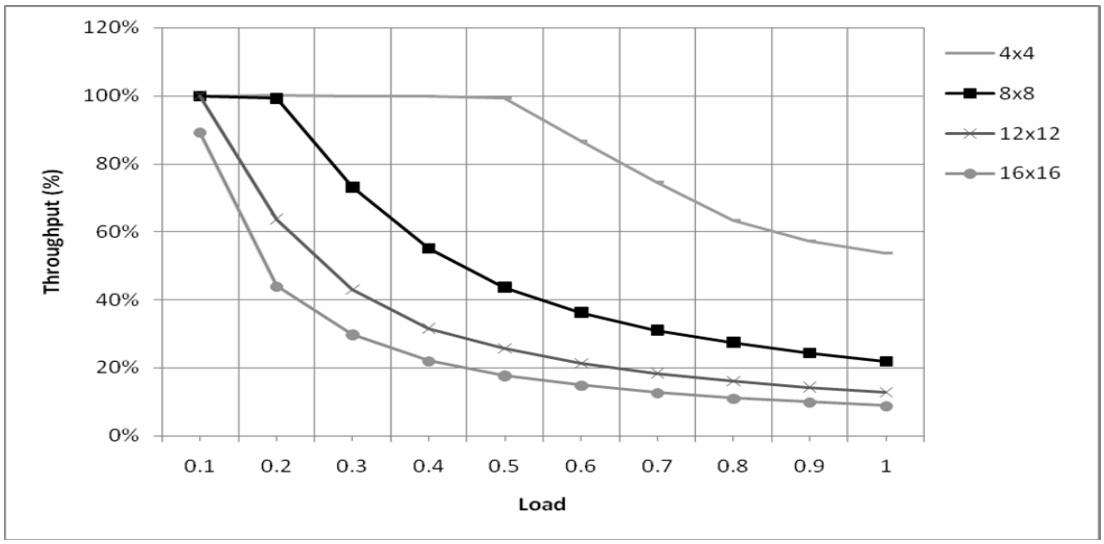

Figure 27 Effect of network size on throughput (packet size = 32 flits)

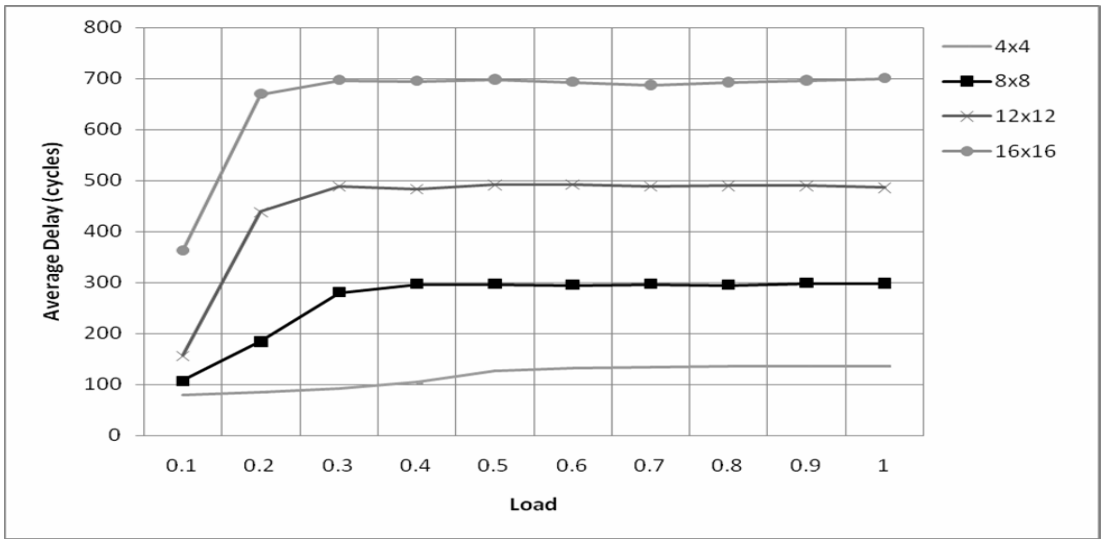

Figure 28 Effect of network size on average packet delay (packet size = 64 flits)

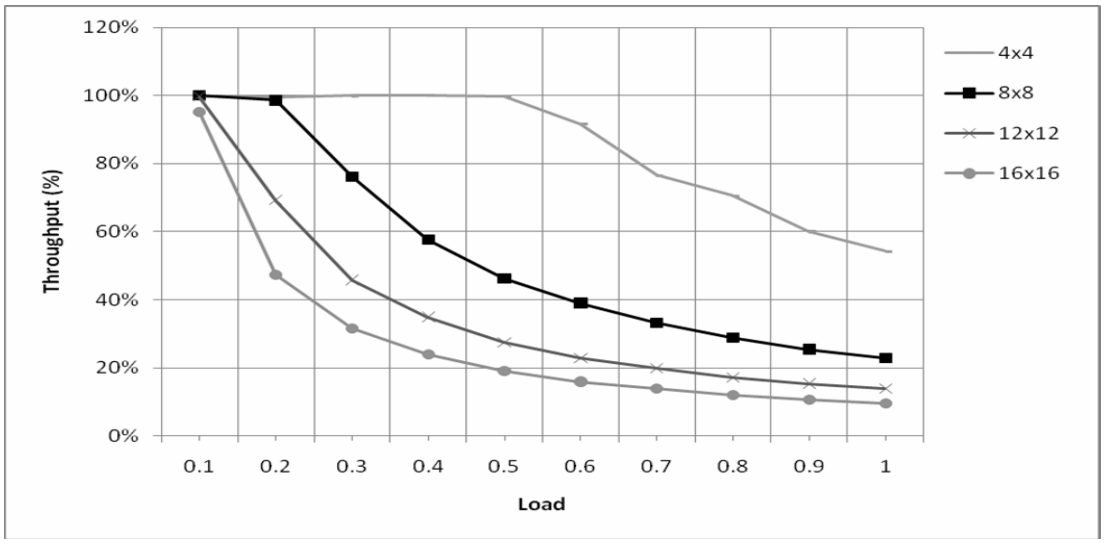

Figure 29 Effect of network size on throughput (packet size = 64 flits)

When the packet length is kept constant and network size is expanded the average packet delay grows (Figures 24, 26, 28), because the number of contending packets grows proportionally to network size. Another reason is that the average hop count of packets becomes greater. The throughput decreases because the saturation point of the network is reached earlier with increasing number of packets it has at any time instance (Figures 25, 27, 29). If we compare Figure 24 and Figure 25 with Figure 26 and Figure 27 respectively, we can notice that average packet delay grows with network and packet size, at the same time the throughput diminishes. It shows us that the impact of network and packet size on performance is significant.

3.5 Performance Evaluation of O1TURN Routing Algorithm

In Chapter 2 we gave brief information about O1TURN routing algorithm. In this section we provide its performance evaluation. But first, in Figure 30 we present its pseudo code:

```
Algorithm: O1TURN Routing for 2-D Meshes
Inputs: input port (In_port), coordinates of current node (Xcurrent, Ycurrent) and
destination node (Xdest, Ydest),
Output: Selected output Channel
Procedure:
Xoffset := Xcurrent − Xdest;
Yoffset :=Ycurrent −Ydest;
if Xoffset = 0 and Yoffset = 0 then
        Channel := Local;
else if In_port= Local_port
        randomly choose virtual_network
else define virtual_network according to information in the flit
if virtual_ network is XY_routed then
        if Xoffset < 0 then
                Channel :=East;
        else if Xoffset > 0 then
                Channel := West;
        else if Yoffset < 0 then
                Channel := North;
        else if Yoffset > 0 then
                Channel := South;
else
        if Yoffset < 0 then
                Channel := North;
        else if Yoffset > 0 then
                Channel := South;
        else if Xoffset < 0 then
                Channel := East;
        else if Xoffset > 0 then
                Channel := West;
endif
```

Figure 30 O1TURN routing algorithm for 2D Meshes

According to the pseudocode in O1TURN when a flit is coming from the local port, one of the virtual networks is chosen randomly; otherwise, virtual network is identified according to information extracted from the flit's header. Then one of the corresponding output channels is returned.

It is clear that this algorithm is more flexible in comparison to XY-routing because it provides two paths to the same destination. In [36] in performance evaluation section 8 VCs per physical port were used. In our model we only have 2 VCs per physical channel. Below performance evaluation of O1TURN under different conditions is presented.

3.5.1 Effect of Packet Size on Performance of O1TURN Routing Algorithm

Different sized O1TURN routed networks with different packet sizes were simulated. In all of them nearly the same results were observed (Figures 31-34). That is why we present just the results for 4x4 networks. Average delay of packets is directly proportional to their size, the longer the packet the higher its delay (Figures 31, 33). However, the throughput remains the same for all configurations of packet sizes. It was observed that the average delay of fixed sized packets is smaller than of exponentially distributed ones. Fixed sized packets always hold the exact amount of resources as the only reason for delay to increase is the level of contention in the network, which increases if the packet generation rate becomes higher. In case of exponentially distributed packet size it is not exactly the same. Even though all the nodes have the same packet generation rate the lengths of packets are different. Thus, they require different amount of resources, and the contention may vary in certain parts of the network. The trend in performance of DOR algorithm and O1TURN is the same since they use the same concept, but O1TURN provides some level of adaptivity. However, this adaptivity is not controlled.

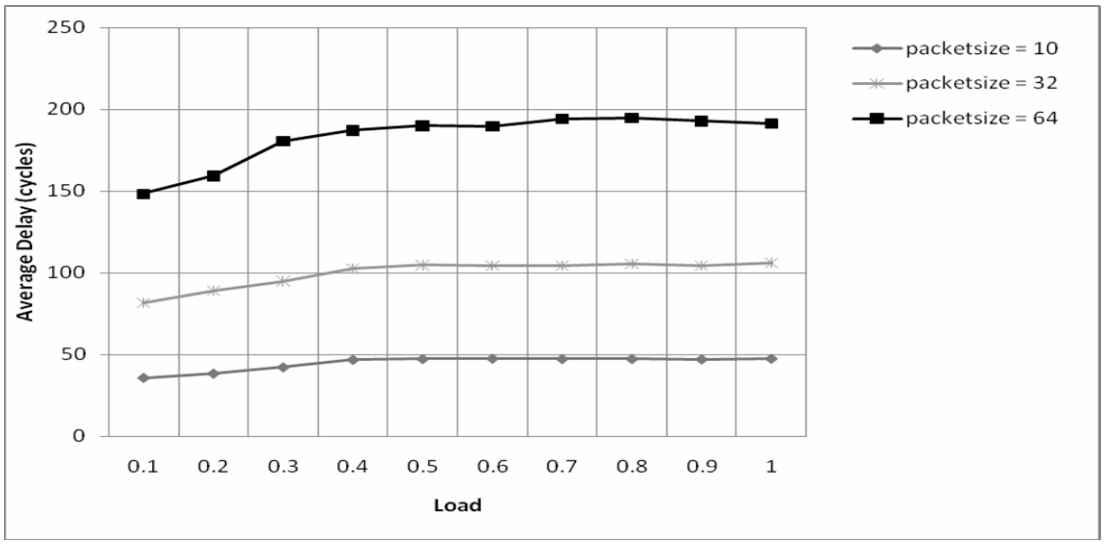

Figure 31 Effect of packet size on average delay in 4x4 network (fixed packet size)

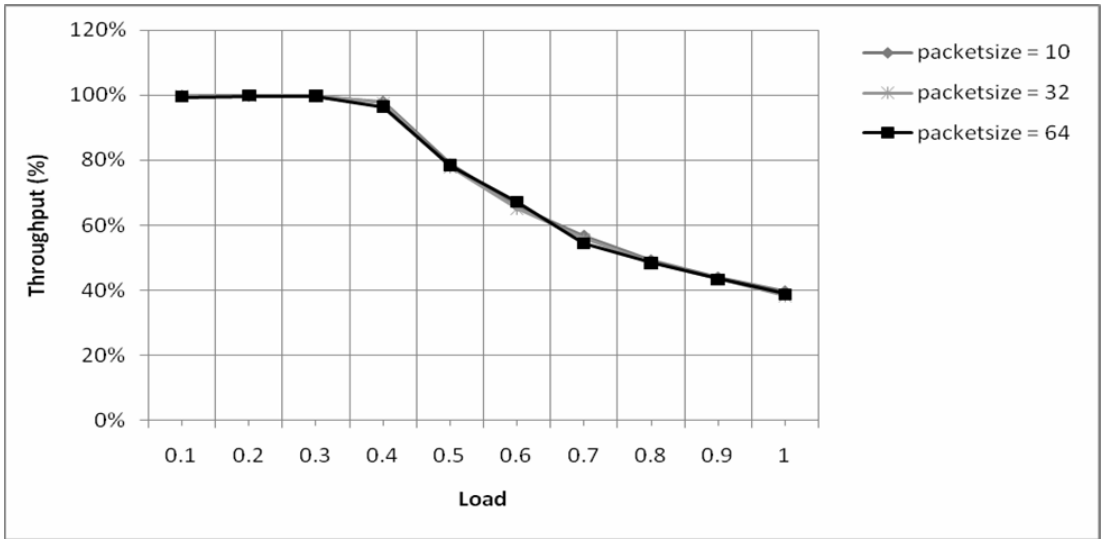

Figure 32 Effect of packet size on network throughput in 4x4 network (fixed packet size)

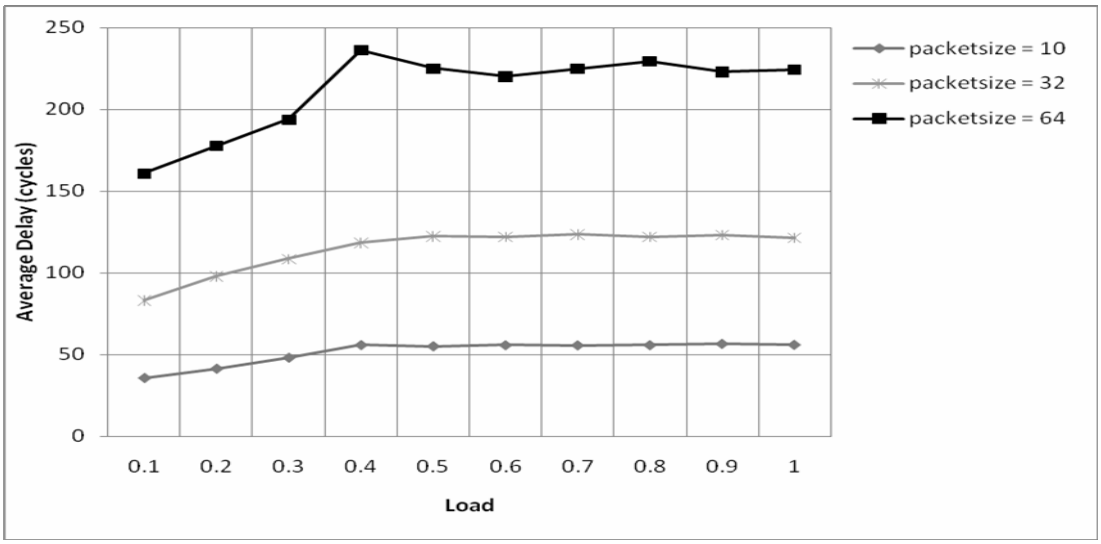

Figure 33 Effect of packet size on average delay in 4x4 network (exponentially distributed packet size)

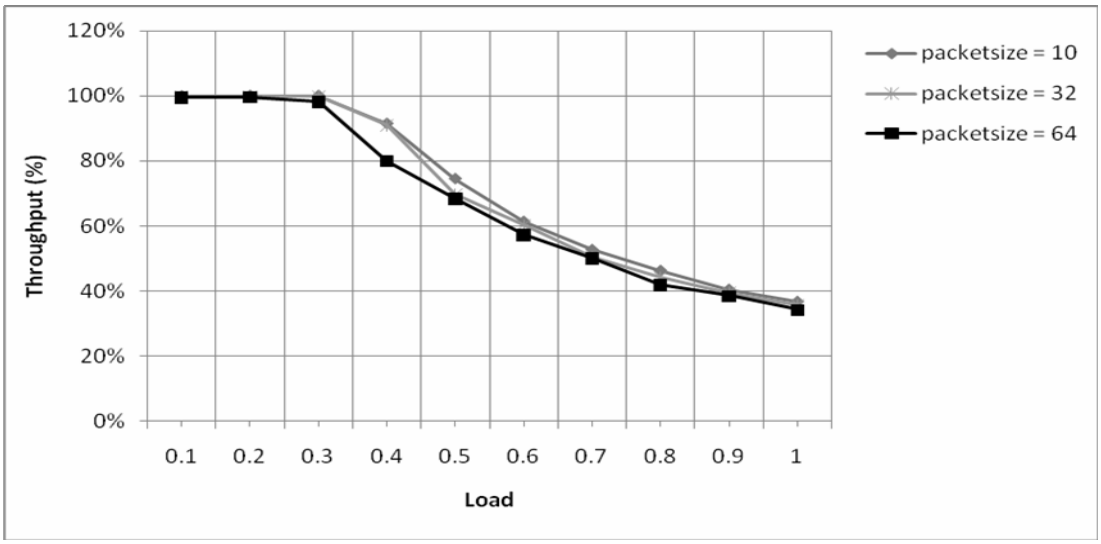

Figure 34 Effect of packet size on network throughput in 4x4 network (exponentially distributed packet size)

3.5.2 Effect of Network Size on Performance of O1TURN Routing Algorithm

Figures 35-40 illustrate effect of network size on throughput and average delay in O1TURN routing algorithm. There is no much difference from the results obtained for XY-routing algorithm except that for smaller sized networks XY-routing performs better, but as the network grows O1TURN shows higher performance in terms of throughput.

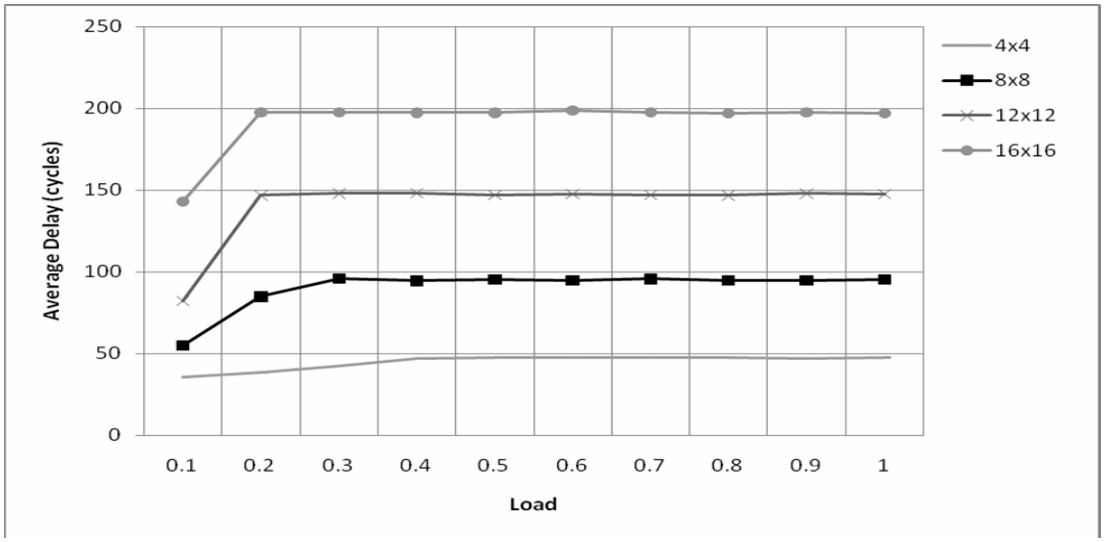

Figure 35 Effect of network size on average delay (packet size = 10 flits)

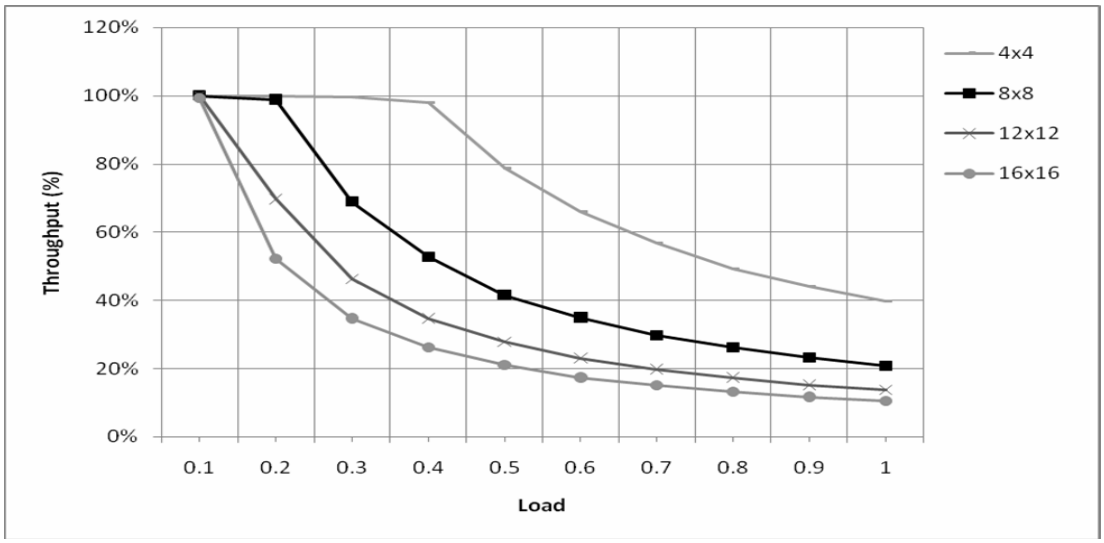

Figure 36 Effect of network size on throughput (packet size = 10 flits)

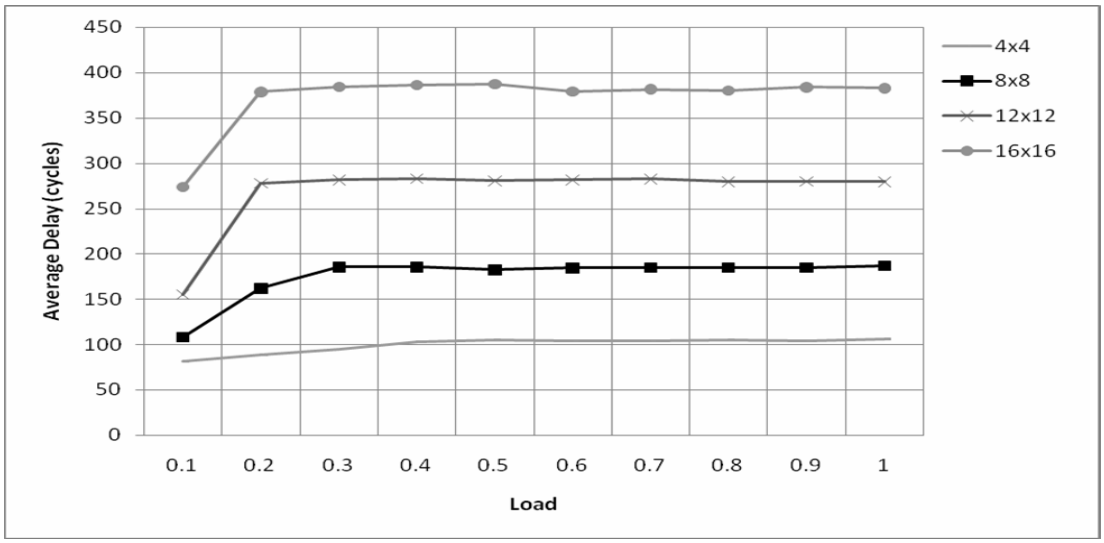

Figure 37 Effect of network size on average delay (packet size = 32 flits)

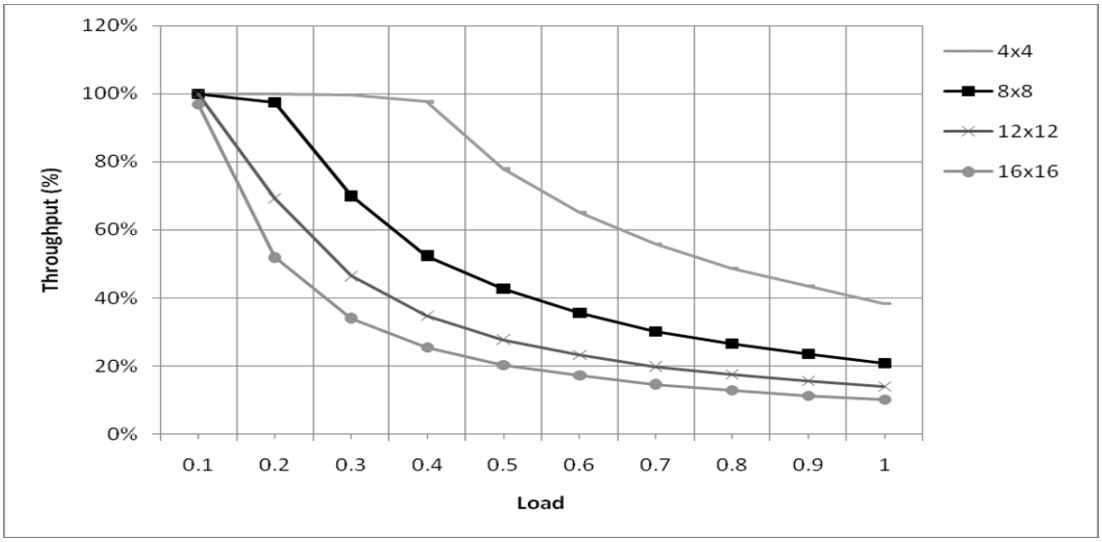

Figure 38 Effect of network size on throughput (packet size = 32 flits)

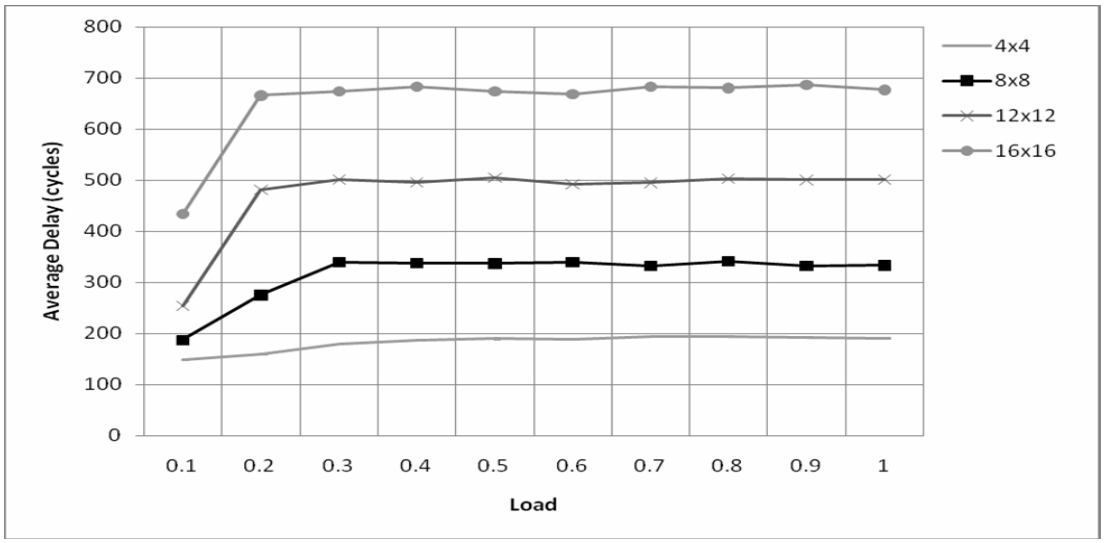

Figure 39 Effect of network size on average delay (packet size = 64 flits)

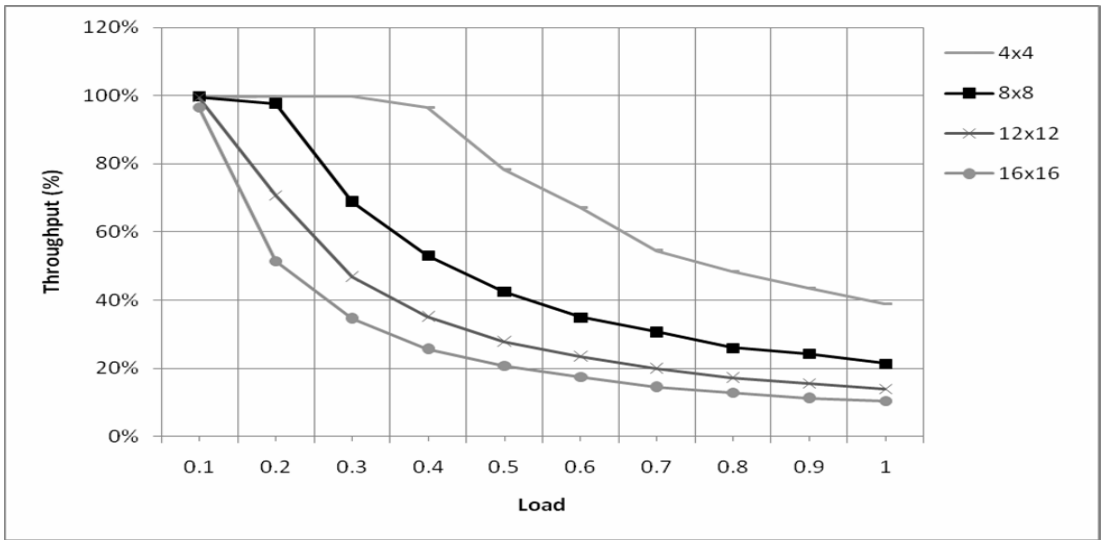

Figure 40 Effect of network size on throughput (packet size = 64 flits)

CHAPTER IV

DETERMINISTIC XY-YX ROUTING ALGORITHM

The first part of this chapter introduces deterministic XY-YX routing algorithm which is based on O1TURN routing algorithm. Then its performance evaluation and comparison with DOR and O1TURN algorithm is presented.

4.1 Deterministic XY-YX Routing Algorithm

According to our simulations when XY routing algorithm is used the central part of the network is more loaded (Figure 41). In this figure the load of all of the routers in a 16x16 network is shown where the incoming traffic is Poisson traffic with fixed packet sizes and the routing is done by XY routing algorithm. The greater the load of the node the darker is its colour. If this load is distributed more uniformly better performance results can be obtained.

Contention Look Ahead [11] routing was proposed for avoiding overloaded areas, but it does not guarantee deadlock-freedom. We propose to improve performance of a network by dividing the network into four equal quadrants and using XY and YX routing in those quadrants interchangeably. The same (XY or YX) routing algorithm is used in diagonally neighboring quadrants. Each quadrant is deadlock-free; problem arose just when the flit went from YX-routed quadrant to XY-routed one or vice versa, if it had to turn from Y(X)-direction to X(Y)-Direction. We tried to solve it by giving higher priorities to flits crossing YX-XY and XY-YX borders.

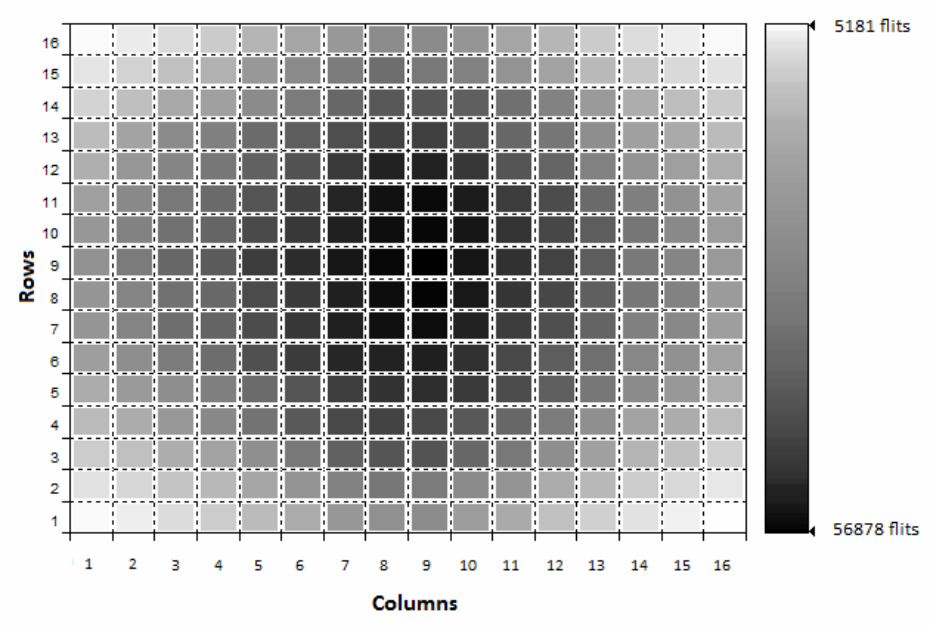

Figure 41 Distribution of network load over the routers in an 16x16 network

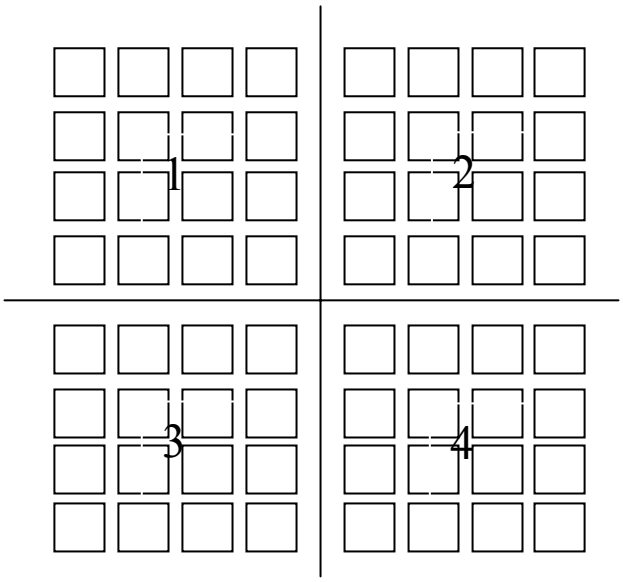

Figure 42 Network partitioning

However, we faced problems concerning deadlocks. In addition we realized that we cannot provide deadlock-freedom without VCs. So, we added additional VC and use two VCs and introduced some changes into our algorithm. We limit the number of VCs to minimum. This is dictated by the fact that additional buffer space significantly increases area overhead of a router and complexity of a switch allocator.

```
Algorithm: XY_YX Routing for 2-D Meshes
Inputs: input port (In_port), coordinates of current node (Xcurrent, Ycurrent)
and destination node (Xdest, Ydest),
Output: Selected output Channel
Procedure:
Xoffset := Xcurrent − Xdest;
Yoffset :=Ycurrent −Ydest;
if Xoffset = 0 and Yoffset = 0 then
        Channel := Local;
else
if In_port= Local_port
        if in quadrant 1or 4
                choose XY_routed virtual_network
        else choose YX_routed network virtual_network
else define virtual_network according to information in the flit
if virtual_ network is XY_routed then
        if Xoffset < 0 then
                Channel := East;
        else if Xoffset > 0 then
                Channel := West;
        else if Yoffset < 0 then
                Channel := North;
        else if Yoffset > 0 then
                Channel := South ;
else
        if Yoffset < 0 then
                Channel := North;
        else if Yoffset > 0 then
                Channel := South;
        else if Xoffset < 0 then
                Channel := East;
        else if Xoffset > 0 then
                Channel := West;
endif
```

Figure 43 XY_YX routing algorithm

Having two VCs per physical channel we divide our network into two virtual networks. As it was said above we partition the network into four quadrants as shown in Figure 42. If the source node is in quadrants 1 or 4 it injects the flits into the XY-routed virtual network, otherwise into the YX-routed one. So, we have two virtual networks where the number of packet sources is two times less than in the original physical network. Injected flits first travel in one dimension according to their routing algorithm and then do in the other. This approach was adopted from O1TURN algorithm. In Figure 43 we introduce the pseudocode of the proposed algorithm.

4.2 Performance evaluation of XY-YX Routing

In this section we conducted performance evaluation of XY-YX routing algorithm by comparing it with algorithms described in previous chapter. Additionally along with those algorithms we implemented DOR routing algorithm with two VCs. Performance evaluation and comparison is done by means of varying packet and network sizes.

4.2.1 Performance evaluation by means of different packet sizes

Here we present comparison of DOR, O1TURN and XY-YX routing algorithms. Figures 44-49 show the results of simulations with varying packet sizes (10-, 32-, 64-flits) and constant (4x4) network size.

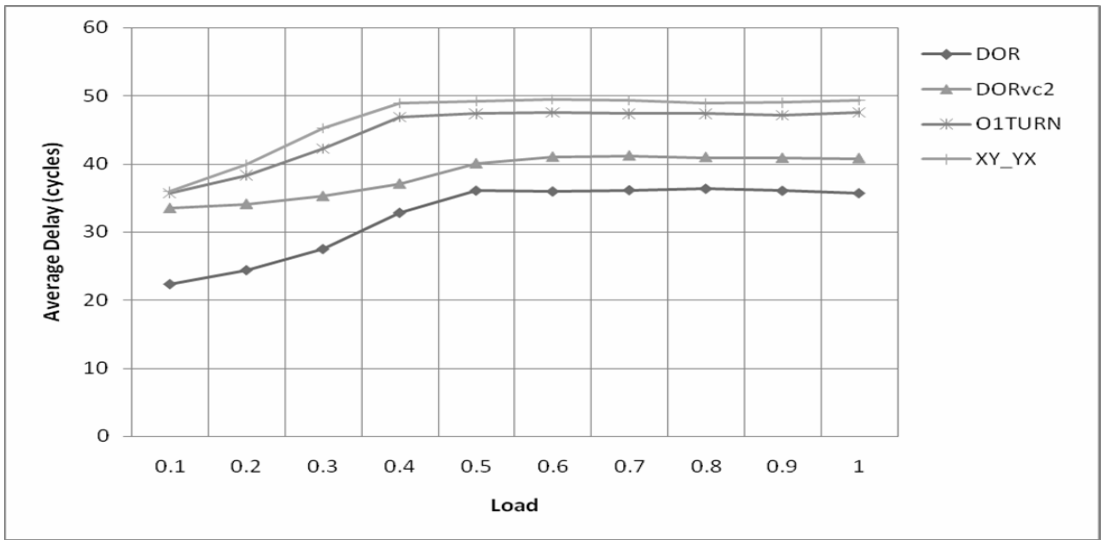

Figure 44 Effect of packet size on average delay (packet size = 10 flits)

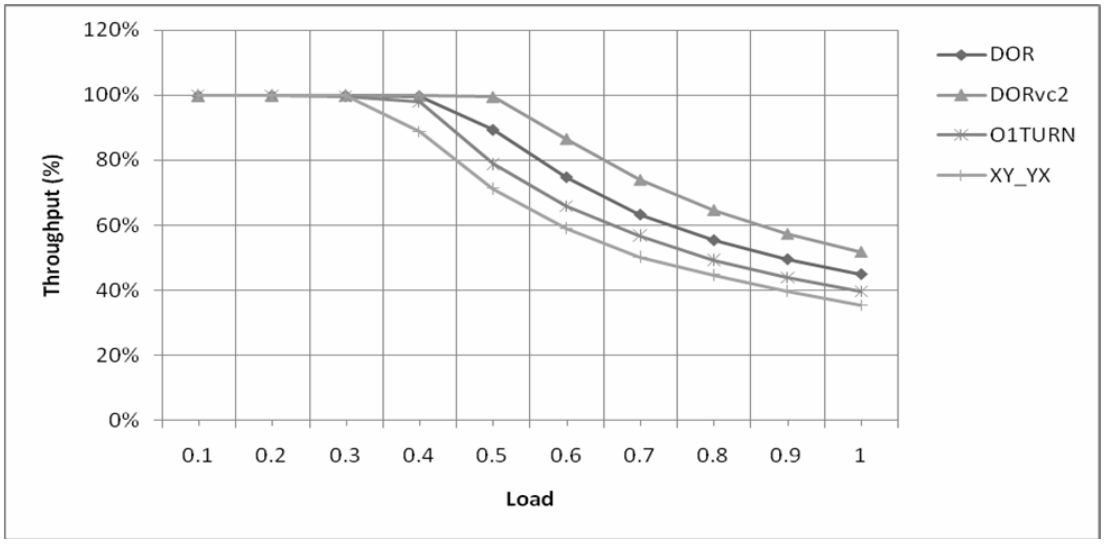

Figure 45 Effect of packet size on throughput (packet size = 10 flits)

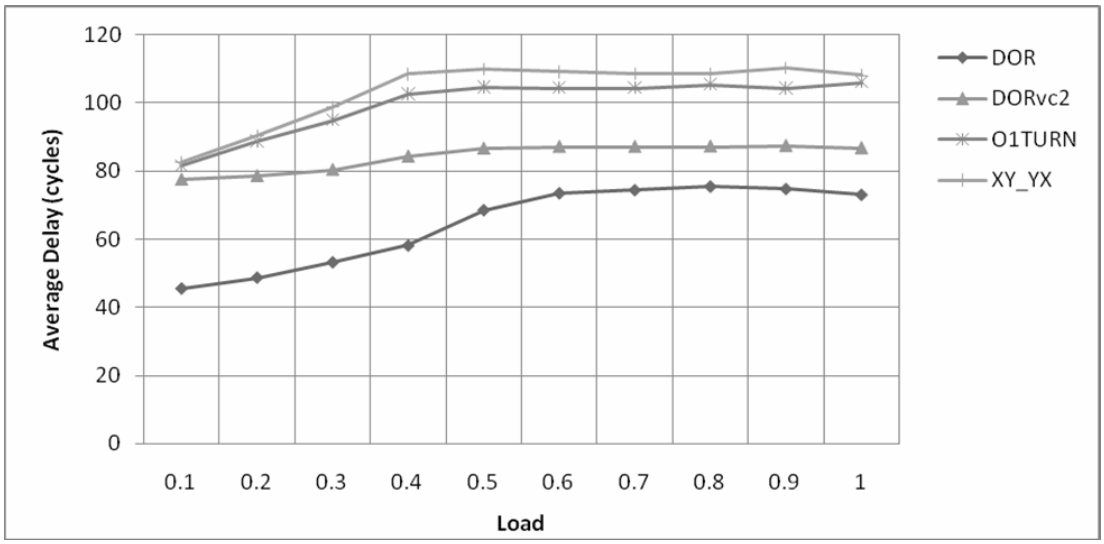

Figure 46 Effect of packet size on average delay (packet size = 32 flits)

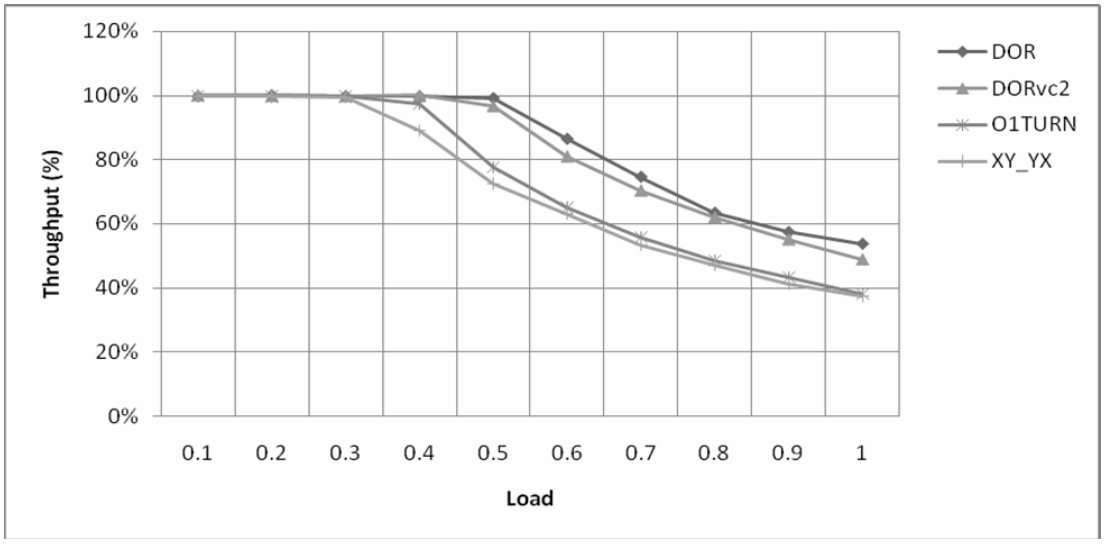

Figure 47 Effect of packet size on throughput (packet size = 32 flits)

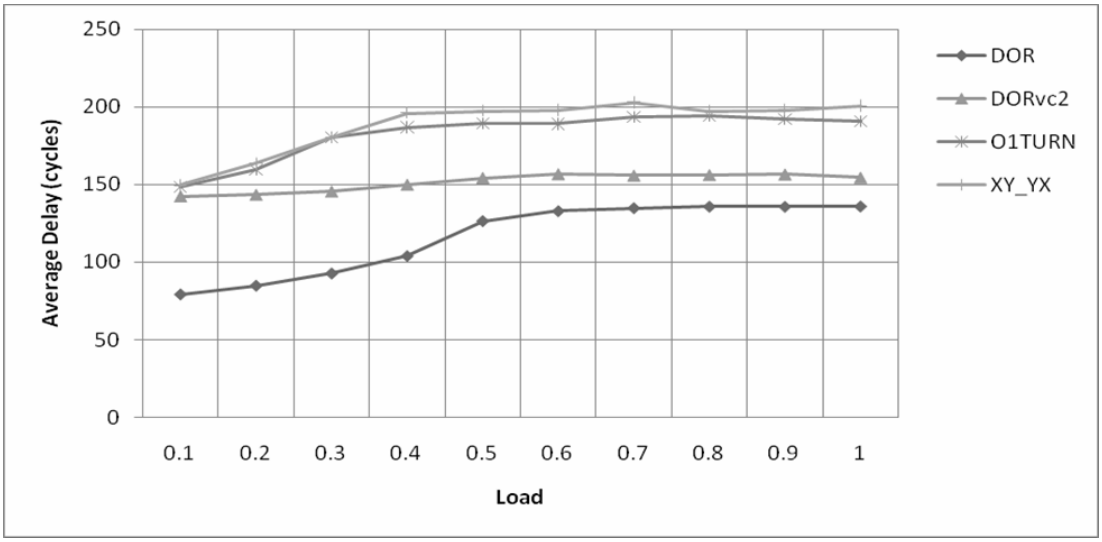

Figure 48 Effect of packet size on average delay (packet size = 64 flits)

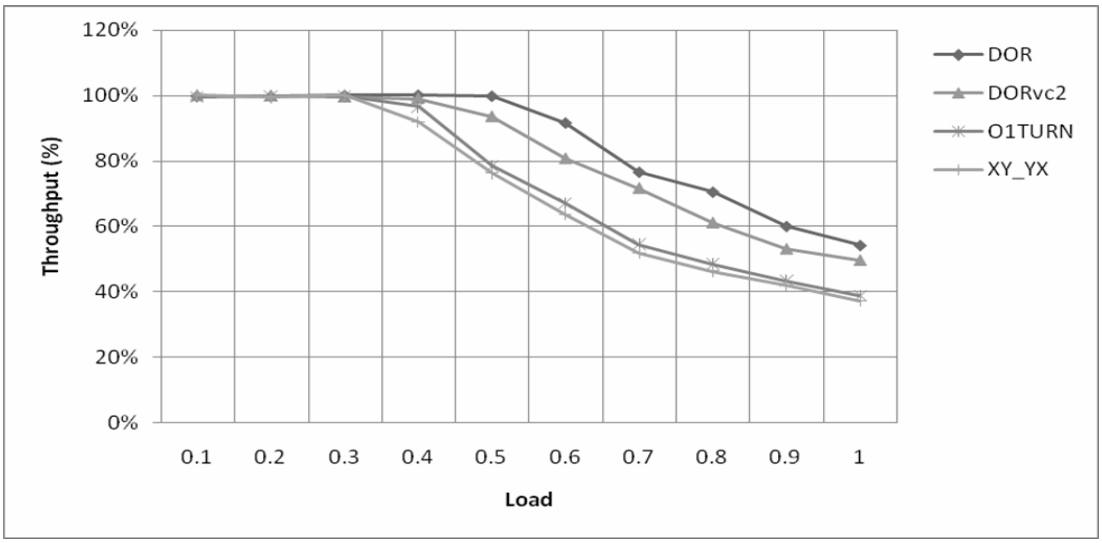

Figure 49 Effect of packet size on throughput (packet size = 64 flits)

It was expected that DOR with two virtual channels (DORvc2) would perform better than other algorithms, but it turned out that in a small (4x4) network DOR algorithm with a single virtual channel significantly outperforms other routing algorithms (Figures 47, 49). This is due to small amount of hops and larger packet sizes. In addition extra stages in the pipeline become a disadvantage since it increases router latency in such cases. Algorithms using extra VCs suffer from additional pipeline overhead in each router. In Figure 45 we see that DOR with two VCs beats all other algorithms. The reason is that 10-flits packet size is tolerable for showing its real performance in small networks. But, when the packet size increased we can observe significant degradation in its performance. O1TURN and XY-YX routing algorithms do not perform as good as the previous two. Because both of them have extra pipeline stages like DORvc2 and on the other hand they inherit the basic concept of DOR algorithm with a slight difference that normal DOR considers the network as a whole, while those two partition it into two virtual networks, in other words they inherit all the features which become disadvantage for DOR when the packet size is considerably small and DORvc2 when the packet size is large.

4.2.2 Performance evaluation by means of different network sizes

This section investigates throughput and delay characteristics of routing algorithms under different network sizes and constant packet size (10 flits). Figures 50-53 show variations of delay and throughput for larger (12x12, 16x16) network sizes.

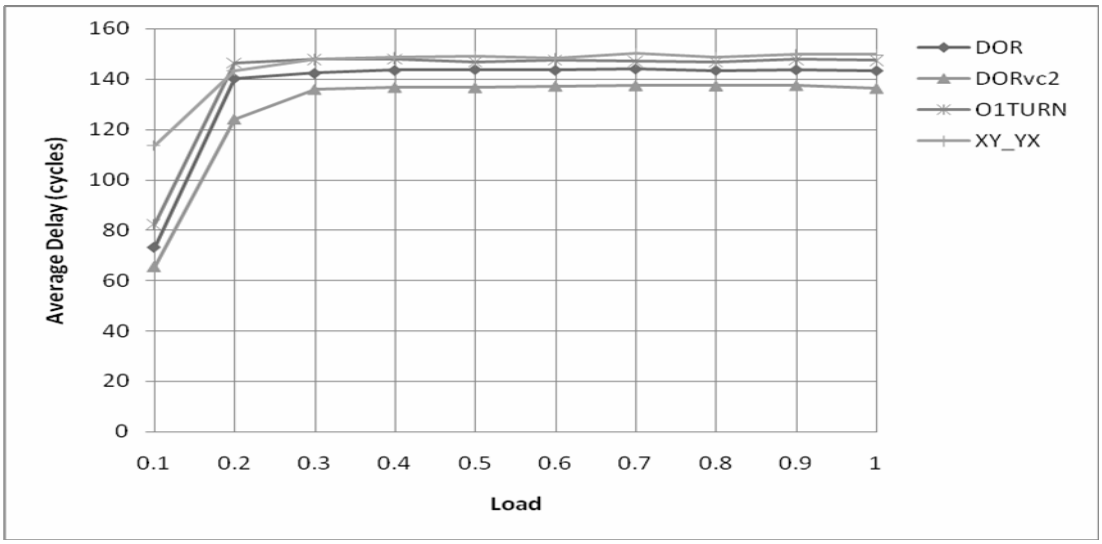

Figure 50 Effect of network size on average delay (network size 12x12)

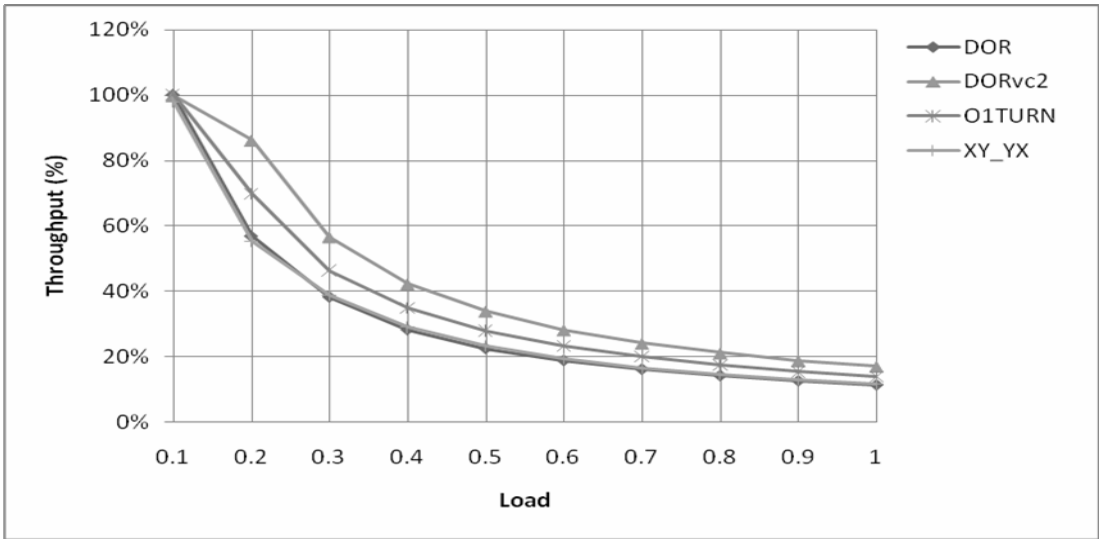

Figure 51 Effect of network size on throughput (network size 12x12)

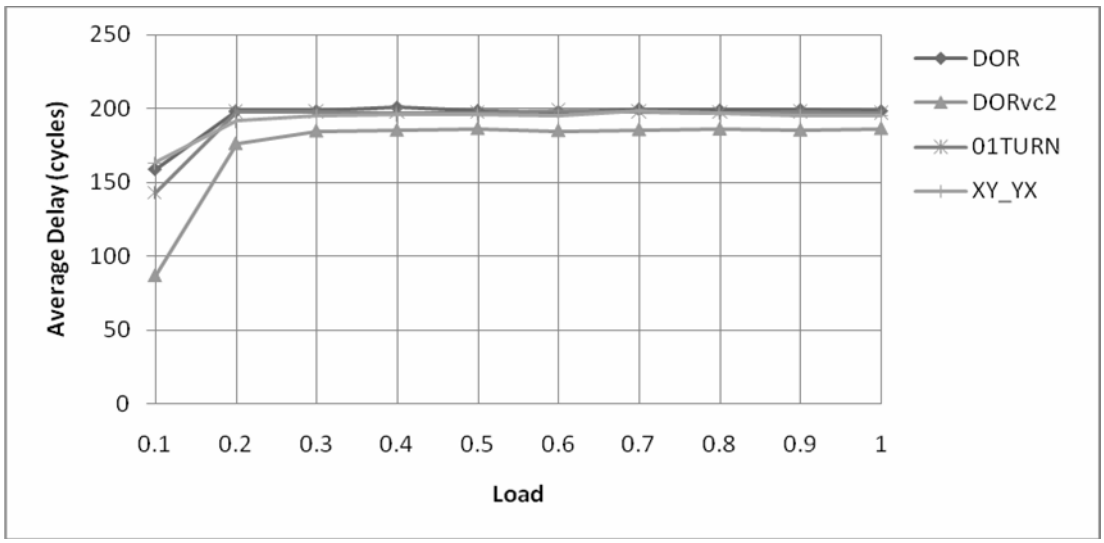

Figure 52 Effect of network size on average delay (network size 16x16)

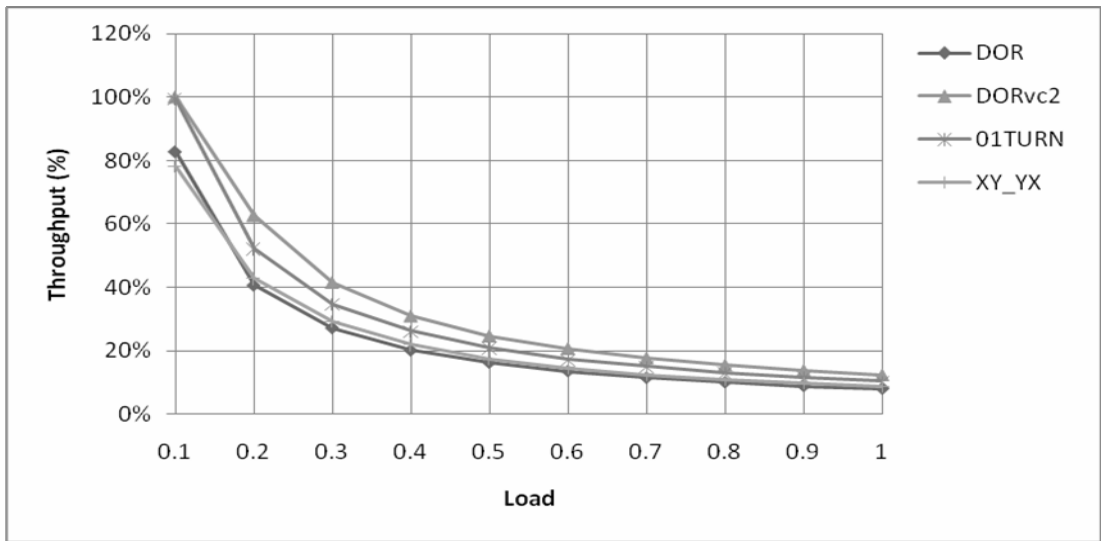

Figure 53 Effect of network size on throughput (network size 16x16)

This time packet size is kept constant (10 flits) while network size changes. In contrast with small networks O1TURN and XY-YX routing algorithms do not bate DOR and DORvc2 much. However, DORvc2 is the best. O1TURN turned out to be more tolerant to variations of network size. It is now better than DOR. XY-YX routing algorithm performs similar to DOR. We think O1TURN and XY-YX routing algorithms cannot perform as good as DORvc2 because despite they use virtual channels they still suffer from HOL-blocking since a flit is injected into one of the virtual networks it cannot leave it. In DORvc2 each flit may use any of the virtual channels if it is available. That is why DORvc2 is always better. DOR's performance degrades because it has no such flexibility. Despite O1TURN and XY-YX routing algorithms split the network into two virtual networks and they seem to exploit two separate networks they are still bounded by common physical channels, and they cannot make use of another virtual channel in the neighbouring virtual network for a particular packet since they cannot switch to it even it is not occupied. Thus, inspite of employing two virtual networks, they still behave like a single network. In addition they have extra pipeline stages in each router.

4.3 Load Distribution

In Figures 54-57 load distributions of differently routed networks are given. Load distributions were calculated according to the total amount of flits passed through the nodes. As it was said before when XY-routing is used the most loaded area is the center of the network (Figure 54). XY routing with two VCs does not introduce much difference to load distribution from that of XY routing with a single channel (Figure 55). When O1TURN routing is used due to injection of flits into two virtual networks the dark area has broadened towards the outside of network (Figure 56). It is more efficient than XY-routing algorithm from this point. This improvement is obtained by the adaptivity provided by additional YX routed virtual network. Figure 57 illustrates load distribution of XY-YX routing algorithm. Here significant change of the form of dark region can be observed. It took a shape of cross marking

the borders of differently routed areas. It occurred due to injection of packets into one of the virtual networks in accordance with location of the source node.

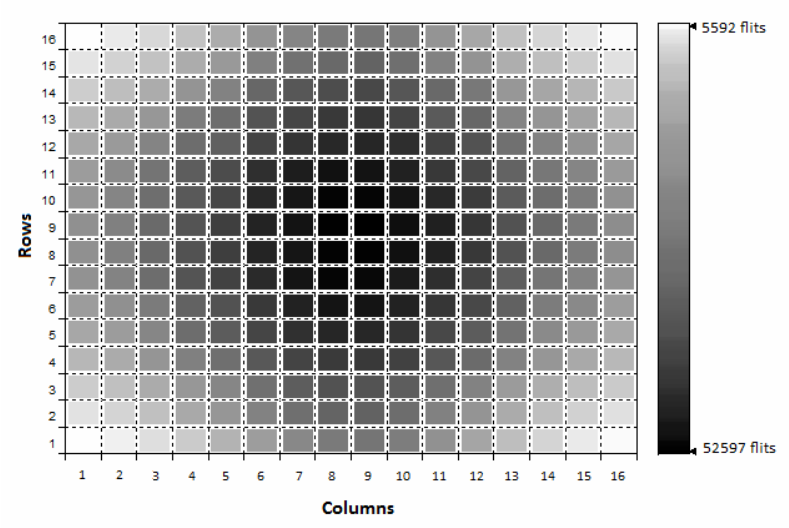

Figure 54 Load distribution of XY-routing algorithm

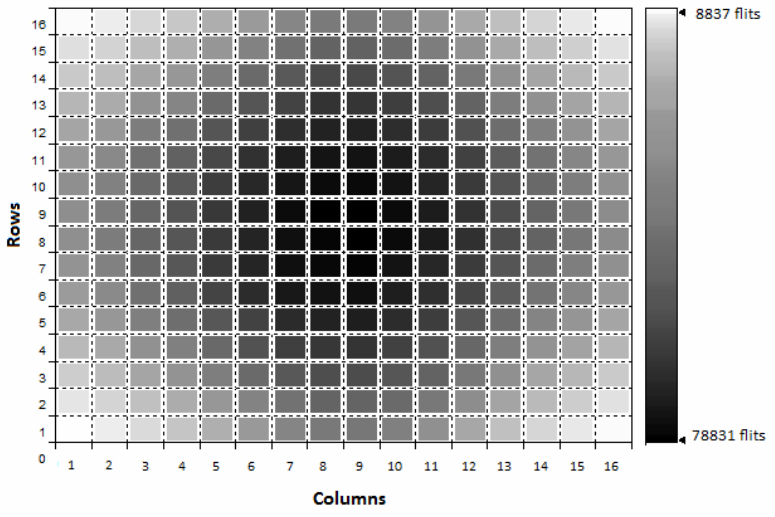

Figure 55 Load distribution of XY-routing algorithm with 2 VCs

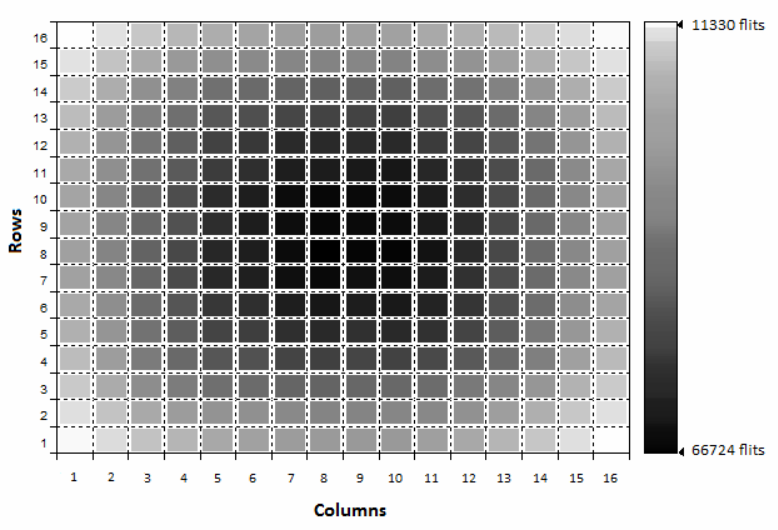

Figure 56 Load distribution of O1TURN routing algorithm

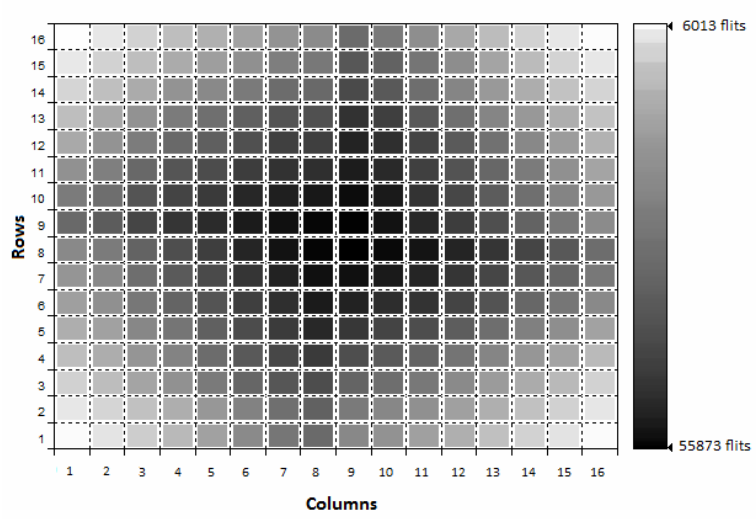

Figure 57 Load distribution of XY-YX routing algorithm

Figure 58 shows the distribution of the load over the nodes defined as the number of nodes with a certain amount of flits passed through them. Here, we find the least and the most loaded (in flits per node) nodes of the network, divide the range between their loads into 20 equal intervals and count the number of nodes in each interval. Each of the intervals is equal to 3750 flits/node. In the ideal case the load of all nodes of the network must be equal. However, it is not possible since the number of ports in each node is not the same (the corner nodes have 2 ports and the edge nodes have 3 ports, while the regular nodes have 4 ports), and the number of VCs also varies with routing algorithms used. Both of these parameters depend on the number of the neighboring nodes. There are four different colored groups of bars in Figure 59 corresponding to the 4 different routing algorithms that are evaluated. The first occurrence of bars of any color is not greater than five, the constituents of those bars are the nodes at the corners of 2D-mesh. They experience the lowest load because they have the fewest number of neighbors. The contributors of the rightmost bars are located in the center of the network. They are most efficiently utilized in terms of usage of full capacity of the routers when O1TURN and DOR with 2 VCs are exploited. When XY-YX routing is used there are not any nodes with loads higher than 57000 flits/node. Hence, it decreases the maximum amount of load that is observed on any given node in the network. However, there is a trade off with other performance metrics such as throughput and delay which is discussed above. The network is mostly loaded when DOR with 2 VCs is used, its peak load is about 79000 flits/node. In this figure we can observe the same trend for all four routing algorithms, the bars grow at the start, achieve their highest level in the middle, then decrease. We see that most of the routers when any of the mentioned routing algorithms is employed do not operate at their full capacities

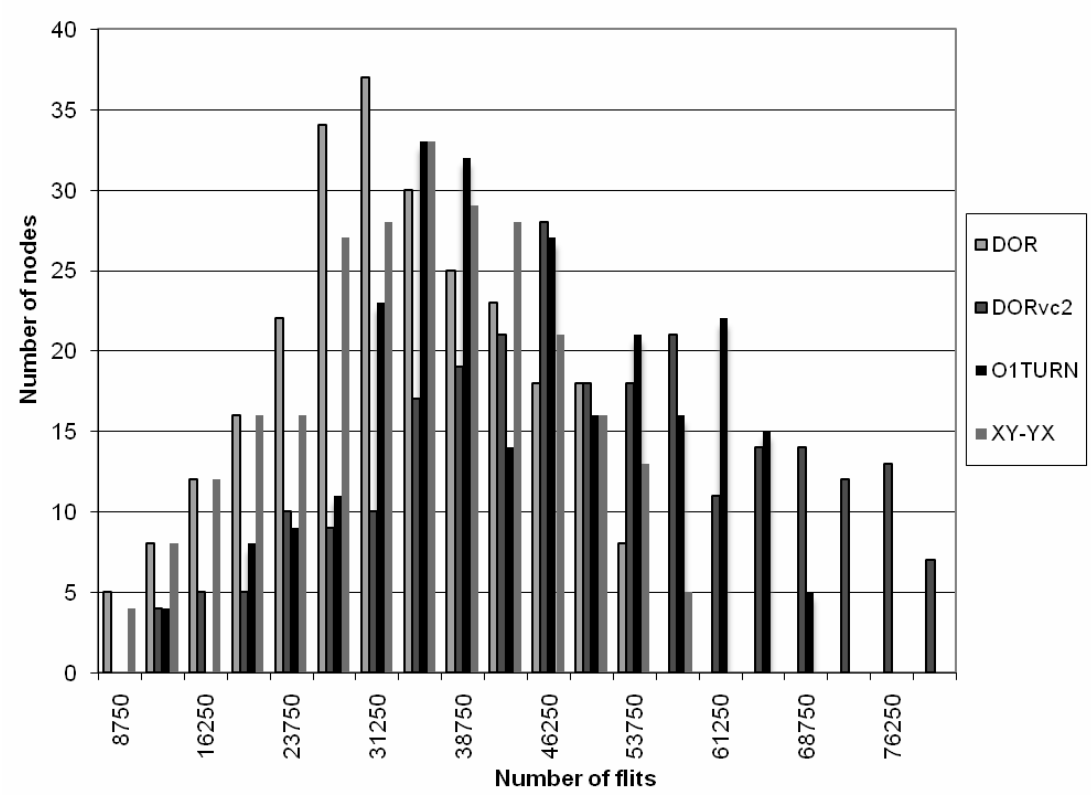

Figure 58 Distribution of load

Figure 59 illustrates combined representation of load distribution in the networks. We put it for clearer interpretation of Figures 54-57. In each part of this figure we can see 16 hillocks, each of them corresponds to a row in Figures 54-57 begining from one and ending with sixteen. The top of each hillock represents the node in the center of a certain row. The sharper the top of a hillock the less uniformly distributed the load among the nodes of that row. Here we can see that from load distribution point of view O1TURN is the most efficient algorithm. We can notice that XY-YX - routing more efficiently uses two opposite sides of central rows in comparison with DOR (XY).

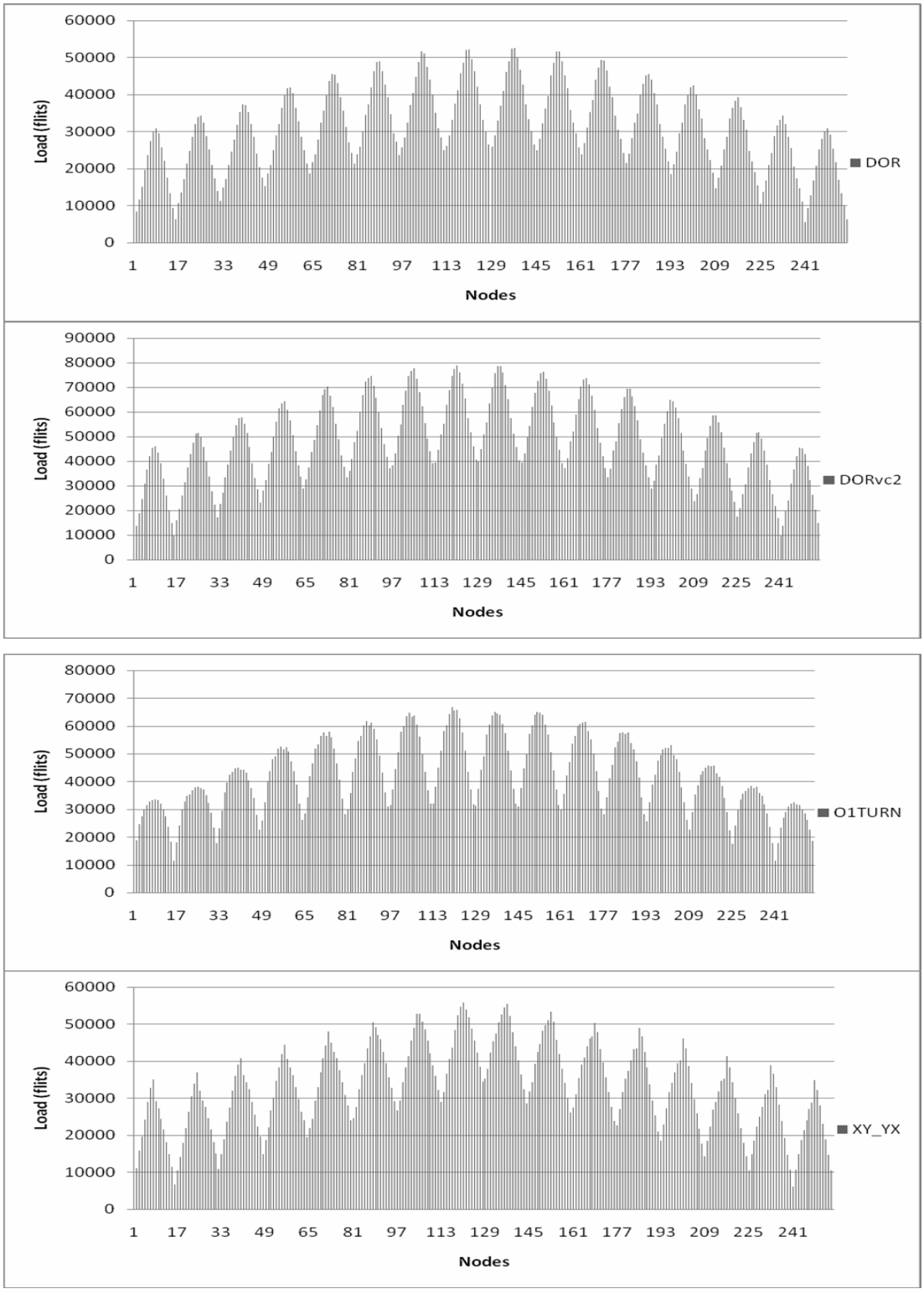

Figure 59 Combined representation of load distribution in networks

CHAPTER V

CONCLUSION AND FUTURE WORK

In this thesis we examined DOR and O1TURN routing algorithms for pipelined router model. Also a modified combination of these two routing algorithms was proposed. We estimated the effect of parameters such as buffer, packet, and network sizes on total performance of the NoC architecture.

Our simulations showed us that buffer size does not affect much the overall performance, but packet length and network size impact delay and throughput significantly. Also we observe that XY routing overloads the center of our network. If that area is bypassed better performance results can be achieved. We proposed deterministic XY-YX routing algorithm but realized that it cannot keep up with problem of high level load in the center but slightly changes the load distribution. If we increase the number of quadrants (rectangles) it is possible that the load distribution of the network takes different shape distributing burden of heavy loaded nodes more uniformly, thus capturing light loaded ones. In spite of usage of separate virtual networks it still behaves like a single network. Another disadvantage of our algorithm is that it is more complex than original XY routing algorithm because of using additional virtual channel. DOR algorithm in spite of its simplicity is still one of the best algorithms for uniformly distributed traffic pattern.

Our future work includes implementation of our algorithm splitting the network into greater amount of quadrants where some adaptivity will be added by allowing

switching of flits from one virtual network to another. However, the number of switchings will be limited to exactly one; otherwise it may lead to a deadlock.

As we mentioned in Chapter II no realistic traffic model has been proposed yet. It is very interesting and attractive research area. If a realistic traffic model is introduced, it can significantly relieve prediction of the real behavior of NoC in early stages of its design.

REFERENCES

[1] **A**. **Jantsch and H. Tenhunen.** *Networks on Chip.* s.l. : Kluwer Academic Publishers, 2003.

[2] **P. P. Pandeet al.** Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE Transactions on Computers.* August 2005, Vol. 54, 8, pp. 1025-1040.

[3] **J**. **Henkel, W. Wolf, S. Chakradhar.** On-chip networks: a scalable, communication-centric embedded system design paradigm. *Proceedings of 17th International Conference on VLSI Design.* 2004, pp. 845-851.

[4] **L**. **Benini, and G. De Michelli.** Networks on chips: a new SoC paradigm. *IEEE Computer.* January 2002, Vol. 35, 1, pp. 70-78.

[5] **M**. **Horowitz, W. Dally.** How scaling will change processor architecture. *IEEE International Solid-State Circuits Conference (ISSCC).* February 2004, pp. 132-133.

[6] **L. Benini, D. Bertozzi.** Network-on-chip architectures and design methods. *IEE Proceedings - Computers and Digital Techniques.* March 2005, Vol. 152, 2, pp. 261-272.

[7] **W. J. Dally and B. Towles.** Route packets, not wires: On-chip interconnection. *Proceedings of the 38th Design Automation Conference.* June 2001.

[8] **S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani.** A network on chip architecture and design methodology. *Proceedings of IEEE Computer Society Annual Symposium on VLSI, 2002.* April 25-26, 2002, pp. 105-112.

[9] **P. P. Pande, C. Grecu, A. Ivanov, R. Saleh.** Design of a switch for network on chip applications. *Proceedings of the 2003 International Symposium on Circuits and Systems. ISCAS '03.* May 2003, Vol. 5, pp. 217-220.

[10] **J. Hu, R. Marculescu.** Application-Specific Buffer Space Allocation for NoCs Router Design. *IEEE/ACM International Conference on Computer Aided Design, ICCAD-2004.* Nov. 2004, pp. 354 - 361.

[11] **T. T. Ye, L. Benini, G. De Micheli.** Packetization and routing analysis of on-chip multiprocessor networks. *Journal of Systems Architecture.* Feb 2004, Vol. 50, 2-3, pp. 81-104.

[12] **U. Y. Ogras, J. Hu, R. Marculescu.** Key research problems in NoC design: a holistic perspective. *Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS '05.* . Sept. 2005, pp. 69-74.

[13] **J. Duato, S. Yalamanchili, L. Ni.** *Interconnection Networks: an Engineering Approach.* s.l. : Morgan Kauffman, 2002.

[14] **C. J. Glass, L. M. Ni.** The Turn Model for Adaptive Routing. *The 19th Annual International Symposium on Computer Architecture Proceedings.* May 19-21, 1992, pp. 278-287.

[15] **J. Hu and R. Marculescu.** DyAD - Smart Routing for Networks-on-Chip. *Proceedings of the 41'st ACM/IEEE DAC.* 2004.

[16] **L. Shang, L. S. Peh, N. K. Jha.** PowerHerd: dynamic satisfaction of peak power constraints in interconnection networks. *Proceedings of International Symposium on Supercomputing.* June 2003.

[17] **P. Guerrier, A. Greiner.** A generic architecture for on-chip packet-switched interconnections. *Design, Automation and Test in Europe Conference and Exhibition 2000.* March 2000, pp. 250-256.

[18] **D. Marconett.** A Survey of Architectural Design and Implementation Tradeoffs in Network on Chip Systems. Davis : University of California.

[19] **F. Karim, A. Nguyen, S. Dey.** An interconnect architecture for networking systems on chips. *IEEE Micro.* Sept-Oct 2002, Vol. 22, 5, pp. 36-45.

[20] **R. I. Greenberg, G. Lee.** An improved analytical model for wormhole routed networks with application to butterfly fat-trees. *Proceedings of the 1997 International Conference on Parallel Processing.* 1997, pp. 44-48.

[21] **D. H. Linder, J. C. Harden.** An adaptive and fault tolerant wormhole routing strategy for k -ary n-cubes. *IEEE Transactions on Computers.* January 1991, Vol. 40, 1, pp. 2-12.

[22] **L. M. Ni and P. K. McKinley.** A survey of wormhole routing techniques in direct networks. February 1993, Vol. 26, 2, pp. 62-76.

[23] **W. J. Dally and H. Aoki.** *Adaptive routing Using Virtual Channels.* MIT Laboratory for Computer Science. 1990. Technical report.

[24] **A. A. Chien.** A cost and speed model for k-ary n-cube wormhole routers. *IEEE Transactions on Parallel and Distributed Systems.* February 1998, Vol. 9, 2, pp. 150-162.

[25] **J. Duato, P. Lopez.** Performance Evaluation of Adaptive Routing Algorithms for k-ary-n-cubes. *Proceedings of the First International Workshop on Parallel Computer Routing and Communication.* 1994, pp. 45-59.

[26] **L. S. Peh, W. J. Dally.** A delay model for router microarchitectures. *IEEE MICRO.* Jan.-Feb, 2001, Vol. 21, 1, pp. 26-34.

[27] **L. S. Peh, W. J. Dally.** A Delay Model and Speculative Architecture for Pipelined Routers. *Proceedings of the 7th International Symposium on High-Performance Computer Architecture.* Jan. 2001, pp. 255-266.

[28] **W. J. Dally and C. L. Seitz.** Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers.* May 1987, Vols. C-36, 5, pp. 547-553.

[29] **L. S. Peh and W. J. Dally.** Flit-reservation flow control. *Proceedings of High Performance Computer Architecture.* January 2000, pp. 73–84.

[30] **A. Ivanov and G. De Micheli.** Guest Editors Introduction: The Network-on-Chip Paradigm in Practice and Research. *IEEE Design and Test of Computers.* October 2005, Vol. 22, 5, pp. 399-403.

[31] **W. J. Dally.** Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems.* March 1992, Vol. 3, 2, pp. 194-204.

[32] **R. Kumar, V. Zyuban, and D. M. Tullsen.** Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. *Proceedings of the 32nd Annual International Symposium on Computer Architecture.* 2005.

[33] **R. Mullins, A. West, and S. Moore.** Low-latency virtual-channel routers for on-chip networks. *Proc. of the 31st ISCA.* 2004, p. 188.

[34] **L. G. Valiant, G. J. Brebner.** Universal schemes for parallel communication. *Proceedings of the 13th annual ACM symposium on Theory of computing.* 1981, pp. 263 - 277.

[35] **T. Nesson, S. L. Johnsson.** ROMM routing on mesh and torus networks. *Proceedings of the 7th annual ACM symposium on Parallel algorithms and architectures.* 1995, pp. 275 - 287.

[36] **D. Seo, A. Ali, W.-T. Lim, N. Rafique.** Near-optimal worst-case throughput routing for two-dimensional mesh networks. *Proceedings of 32nd International Symposium on Computer Architecture, 2005. ISCA '05.* pp. 432 - 443.

[37] **L. Benini, G. De Michelli.** *Networks on Chips: Technology and Tools.* : Morgan Kaufmann Publishers, 2006.

[38] **E. B. Jung, H. W. Cho, N. Park, and Y. H. Song.** SONA: An on-chip network for scalable interconnection of AMBA-based IPs. *International Conference on Computational Science.* May 2006, Vol. 3994, pp. 244-251.

[39] **G. Varatkar, R. Marculescu.** Traffic analysis for on-chip networks design of multimedia applications. *Proc. Design Automation Conference. (DAC).* June 2002, pp. 510-517.

[40] **G. Varatkar, R. Marculescu.** On-chip traffic modeling and synthesis for MPEG-2 video applications. *IEEE Transactions of Very Large Scale Integration (VLSI) Systems*. Jan. 2004, Vol. 12, pp. 108-119.

[41] **V. Soteriou, H. Wang, L. S. Peh.** A statistical traffic model for on-chip interconnection networks. *Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'06).* Sept. 2006, pp. 104-116.

[42] **P. P. Pande.** *Networks on chip: Emerging interconnect infrastructures for MP-SoC platforms.* PhD thesis, Dept. of Electrical and Computer Engineering, University of British Columbia. July 2005.

[43] **P. P. Pande, G. De Micheli, C. Grecu, A. Ivanov, and R. Saleh.** Design, synthesis, and test of networks on chips. *IEEE Design and Test of Computers.* Sept.-Oct. 2005, Vol. 22, pp. 404-413.

[44] **T. Bjerregaard, S. Mahadevan.** A survey of research and practices of network on chip. *ACM Computing Surveys.* March 2006, Vol. 38, pp. 1-51.

[45] **J. Hu, R. Marculesu.** Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures. *Proceedings of the Desin Automation and Test in Europe Conference and Exhibition.* 2003, pp. 688-693.

[46] **E. Nilsson, M. Millberg, J. Öberg, A. Jantsch.** Load distribution with the proximity congestion awareness in a network on chip. *Proceedings of the Desin Automation and Test in Europe Conference and Exhibition.* 2003, pp. 1126-1127.