PERFORMANCE MODELLING AND EVALUATION OF NETWORK ON CHIP UNDER BURSTY TRAFFIC

H. M. IBRAHIM

PhD

2014

Performance Modelling and Evaluation of Network On Chip Under Bursty Traffic

Performance evaluation of communication networks using analytical and simulation models in NOCs with Fat tree topology under Bursty Traffic with virtual channels

Hatem Musbah IBRAHIM

Submitted for the degree of Doctor of Philosophy

Faculty of Engineering and Informatics

2014

ABSTRACT

HATEM MUSBAH IBRAHIM

PERFORMANCE MODELLING AND EVALUATION OF NETWORK ON CHIP UNDER BURSTY TRAFFIC

PERFORMANCE EVALUATION OF COMMUNICATION NETWORKS USING ANALYTICAL AND SIMULATION MODELS IN NOCS WITH FAT TREE TOPOLOGY UNDER BURSTY TRAFFIC WITH VIRTUAL CHANNELS

Keywords: Network On Chip, Bursty traffic, Virtual Channels, Latency

Physical constrains of integrated circuits (commonly called chip) in regards to size and finite number of wires, has made the design of System-on-Chip (SoC) more interesting to study in terms of finding better solutions for the complexity of the chip-interconnections. The SoC has hundreds of Processing Elements (PEs), and a single shared bus can no longer be acceptable due to poor scalability with the system size. Networks on Chip (NoC) have been proposed as a solution to mitigate complex on-chip communication problems for complex SoCs. They consists of computational resources in the form of PE cores and switching nodes which allow PEs to communicate with each other.

In the design and development of Networks on Chip, performance modelling and analysis has great theoretical and practical importance. This research is devoted to developing efficient and cost-

effective analytical tools for the performance analysis and enhancement of NoCs with m-port n-tree topology under bursty traffic.

Recent measurement studies have strongly verified that the traffic generated by many real-world applications in communication networks exhibits bursty and self-similar properties in nature and the message destinations are uniformly distributed. NoC's performance is generally affected by different traffic patterns generated by the processing elements. As the first step in the research, a new analytical model is developed to capture the burstiness and self-similarity characteristics of the traffic within NoCs through the use of Markov Modulated Poisson Process. The performance results of the developed model highlight the importance of accurate traffic modelling in the study and performance evaluation of NoCs.

Having developed an efficient analytical tool to capture the traffic behaviour with a higher accuracy, in the next step, the research focuses on the effect of topology on the performance of NoCs. Many important challenges still remain as vulnerabilities within the design of NoCs with topology being the most important. Therefore a new analytical model is developed to investigate the performance of NoCs with the m-port n-tree topology under bursty traffic. Even though it is broadly proved in practice that fat-tree topology and its varieties result in lower latency, higher throughput and bandwidth, still most studies on NoCs adopt Mesh, Torus and Spidergon topologies. The results

gained from the developed model and advanced simulation experiments significantly show the effect of fat-tree topology in reducing latency and increasing the throughput of NoCs.

In order to obtain deeper understanding of NoCs performance attributes and for further improvement, in the final stage of the research, the developed analytical model was extended to consider the use of virtual channels within the architecture of NoCs. Extensive simulation experiments were carried out which show satisfactory improvements in the throughput of NoCs with fat-tree topology and VCs under bursty traffic. The analytical results and those obtained from extensive simulation experiments have shown a good degree of accuracy for predicting the network performance under different design alternatives and various traffic conditions.

ACKNOWLEDGEMENTS

The work described in this thesis was carried out in the School of Electrical Engineering and Computer Science at the University of Bradford.

First of all, my strongest thanks to ALLAH the most merciful, without his help and blessing, this thesis would not have progressed of have seen the light.

I would like to acknowledge the people who have helped and inspired me along the way. I would like to express sincere appreciation to my supervisors Prof. Irfan Awan for all of his motivation, guidance, advice, support, patience and thoughts that made this work possible.

Many sincerest thanks to my parent for their love and support throughout my life and their effort to make everything perfect for my studies. I also wish to thank my sisters, my brothers and my friends.

Finally, I wish to express my sincere thanks to Libyan Ministry of higher Education for awarded me scholarship and supported me during the period of my study.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SoC	System on chip
NoC	Network on Chip
HPC	High-performance computing
IBA	InfiniBand architecture
IN	Interconnection network
MMPP	Markov-modulated Poisson process
OMNeT++	Objective Modular Network Testbed in C++
PE	Processing element
MPSoCs	Multiprocessor System-on-Chip

Chapter 1.

Introduction

1.1 Introduction

The developments in semiconductor technologies make it possible to integrate many processing elements (PEs) on a single chip, understood as the System on Chip (SoC), consisting of large number of gates and many processing elements operating at different clock frequencies. As the limitation of a bus based system is that it is non-scalable, and given that the size and the complexity will not allow one to start design SoC from scratch, the need to replace the bus based system seems to be inevitable. A solution for the interconnect problem in large SoC design has been proposed by adopting Network on Chip (NoC) [1].

Networks on chip (NoCs) draw on concepts inherited from distributed systems and computer networks subject areas to interconnect PE cores in a structured and scalable way. NoC emerged as a promising alternative to bus-based interconnect networks to handle the increasing communication requirements of the large systems on chip. Congestion in NoCs reduces the overall system performance. This effect is particularly strong in networks where a single buffer is associated with each input channel, which simplifies router design but prevents packets from sharing a physical channel at any given instant of time.

There are many important aspect to be consider to study of network on chip, for instance topologies, traffic pattern, switches and routing algorithms. Furthermore, these elements affect the NoC performance in terms of latency, power consumption, and throughput.

Employing an appropriate topology for a NoC is of high importance, mainly because it typically offers trade-offs between cross-cutting concerns, such as performance and cost. There is similarity between the NoC and Interconnection Network (IN) architecture in high-performance computing systems (HPC) [2], therefore the topology of IN may be adopted in NoC design. Fat-tree has been chosen as a NoC topology because of the required low latency communication, performance scalability and flexible. Moreover, many research efforts have been made on tree-based topologies in the NoC community, proving their superior performance over 2D meshes under different types of traffic patterns [3, 4]. Thus, the fat-tree topology and its variants are widely adopted in practice. The *m*-port *n*-tree is a typical example of fat-tree topologies [5]

The increasing diversity of NoC applications' performance demands usually prevents adopting a fixed architecture for a wide range of applications [6]. There are a number of methods and techniques that are widely employed to improve the performance of topology. A traditional approach is by employing the virtual channel without increasing the node degree to improve the performance.

Routing algorithm policy is among the most important considerations in network on chip design. The routing strategy determines the path that each message or packet follows between a source-destination pair. There are two type of routing algorithms: deterministic and adaptive. Deterministic routing algorithms are usually adopted by NoCs because they have simpler logic compared to their adaptive counterparts [7]. Implementing the simpler logic in turn leads to smaller routers, using less resources [8] by using wormhole switching

The data communication between PE components in NoC systems is made by using packet switching, wormhole or virtual cut-through switching methods. The network on chip router uses a wormhole switching method with a synchronous parallel pipeline technique [9]. The deterministic and wormhole switching are common routing and switching techniques for NoCs. Using Wormhole switching with deterministic routing made the routing very popular [10]. In addition the wormhole switching is an efficient switch for NoCs.

Reducing the network latency and increases the throughput can be done by dividing the buffer storage associated with each network channel into several virtual channels. Adding virtual channels to the network can be described as adding lanes to a street network; without virtual channels a network is composed of simply a one lane street. In such a network, when a blocked packet occurs in the channel it blocks all subsequent packets.

The traffic pattern has important effects on the performance of an on-chip network. To gain an accurate and deep understanding of the network's performance, it is necessary to employ accurate models in order to capture the realistic network traffic patterns. There are two important characteristics used to define the network traffic patterns, which are message arrival process and destination distribution [10].

1.2 **Motivation**

Although there are several advantages of NoC, it has been proposed as a promising solution to reduce the overheads of buses and Multiprocessor System-on-Chip (MPSoCs) by means of general purpose communication architectures, there are still some challenges which have not yet been comprehensively discussed. Using short communication links instead of using a shared bus can also reduce the power consumption and enhance reliability [11]. Additionally the wire length and delay of the physical links can be better controlled while at the same time the wires are used more efficiently, demanding fewer wires. Thus, NoCs have come to be regarded as the favoured on-chip communication paradigm by presenting an integrated solution to a wide range of challenges in the development of the large MPSoC [1].

Performance modelling and evaluation of NoC has received significant research attention yet there remains need to design and develop advanced performance tools and methods to keep up with the rapid evolution and ever

increasing complexity of the network on chip. Most of existing studies of NoC are based on the unrealistic assumption of non bursty Poisson arrival with uniformly distributed message destination, which are not appropriate for evaluation and optimisation of network performance, since Poisson is Synthetics traffic.

The network topology plays an important role in enhancing the network performance in network on chip such as low latency communication, performance scalability and flexible routing. A NoC has many topologies that have been proposed for interconnection networks. More or less the architectures introduced or adopted for NoC domain are mesh, tours, and variations of ring, butterfly, fat tree, and spidergon topology.

1.3 **Research Aims**

The main aims of this thesis are outlined as follows:

- Propose analytical model validated with simulation to evaluate the performance of Network on Chip with respect to traffic patterns which exhibited by real-world applications.
- Applying both virtual and physical channels to investigate Network on chip behaviour in regard avoiding the deadlock free.
- To develop performance models to investigate the Message latency in network on chip.

- To propose scheme to investigate different network conditions to improve delay.

1.4 Thesis Content

This thesis is organized as follows:

The background research of interconnection network and development of NoC are presented in Chapter 2. Next, the advantages of adding virtual channel in the network and implementing the realistic traffic pattern are elaborated.

Chapter 3 deals with the simulation model we used to estimate the performance. The topology, the routing experiments setting and the important parameters of traffic are discussed.

To examine the performance of NoC in the presence of bursty traffic with uniformly distributed message destinations, an analytical model is designed in Chapter 4. The significant results are elaborated through simulation study.

An analytical model is also developed in Chapter 5 to investigate the performance of NoC in the presence of bursty traffic with virtual channel and uniformly distributed message destinations. The proposed model is then employed to evaluate the impact of this traffic pattern on network performance**.**

In Chapter 6, a brief review of the results of the previous chapters along with some general comments is given and proposed future work is also presented.

Chapter 2.

BACKGROUND

This chapter presents an overview of how Network on chip (NoCs) has evolved during recent years and what their major building blocks are. It also highlights some related aspects, such as topology issues, routing algorithms, switching and flow control techniques. Finally a brief review of the works in the field of NoCs relating to this thesis is presented.

2.1 Interconnection Networks

An interconnection network (IN) is the hardware fabrics which support communications between single processors in high performance computer systems (HPC) [10, 12, 13]. In these systems, a task is divided into smaller sub-tasks, which use multiple collaborating processors to process in a parallel scheme. An IN is a programmable system that transports data between terminals, and it performs a main role in determining the overall performance of multi-computer systems that make the study of interconnection networks very important. The performance of the interconnection networks in a communication switch largely determines the capacity of the switch because the demand for interconnection network performance has grown rapidly in comparison to the capacity of the underlying wires networks.

IN can be simply classified according to the topology into two broad categories: direct and indirect connection.

In direct networks, each node has a direct connection to other nodes allowing for direct communications among processors. The typical examples of direct INs include hypercube, mesh, torus and star graph, as shown in Figure 2-1 below. These have been widely employed in recent practical multi-computer systems (e.g., Intel iPSC, Intel Cavallino, Intel Delta, and nCUBE [12]. The main advantage of such networks is that the resources of a node are available to each switch.

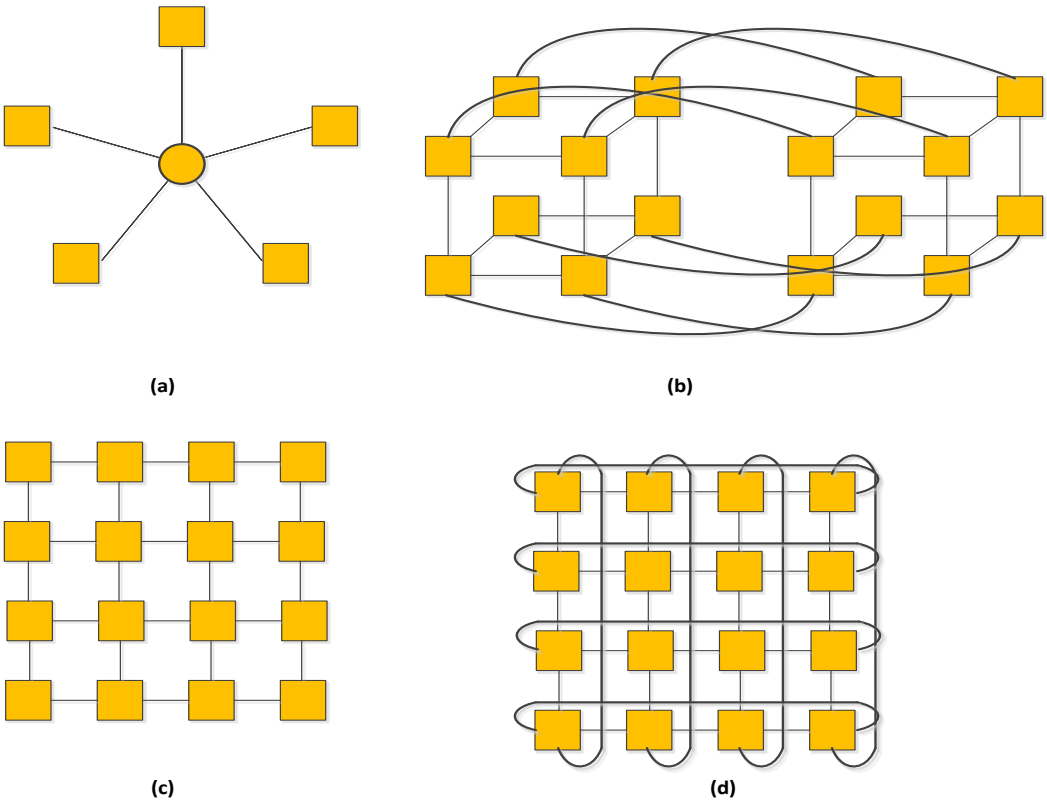

Figure 2-1: Direct interconnection network: (a) star, (b) 3D Hypercube, (c) 2D Mesh, (d) 2D Torus.

Unlike direct networks which have a direct connection, nodes under indirect networks are connected to other nodes through multiple intermediate stages of switches. Figure 2-2 below shows the common examples of indirect Ins, which include k-ary n-fly and fat-tree, which are adopted by many experimental and commercial parallel machines, such as DEC GIGA switch, Cray X/Y-MP, Myrinet, Cenju-3, NEC, IBM SP, IBM RP3, Meiko CS-2, Thinking Machine, and Hitachi SR2201[12]. Fat-tree topology is a popular type of the interconnection network in cluster-based systems in particular [10, 14]. The fat-tree comes from the fact that, the number of links going down to its siblings is equal to the number of links going up to its parent in the upper level. Therefore, the links get fatter towards the root [15]. In fact this type has been the most popular network topology of all over the past fifty years. Commercial machines and many research prototypes have adopted different kinds of fat-tree.

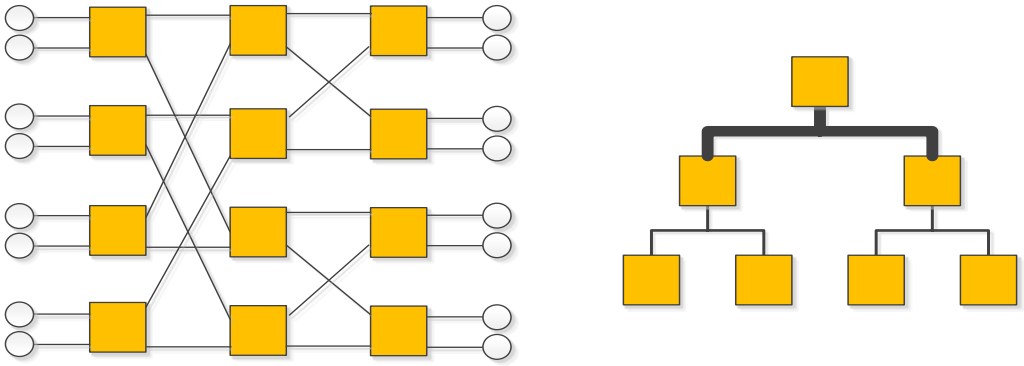

Figure 2-2 Indirect interconnection network: (a) k-ary n-fly, (b) Fat Tree

2.2 The Network on Chip Development

System on chip networks (SoCs) are composed of more than one processor. The interconnection within the SoC has the job of providing the communication infra-structure necessary for the communication of the processors. The first solution for interconnecting the processors is to use dedicated wires. Dedicated wires mainly connect every processor to every other required resource available on the chip. This solution is applicable only when each resource has to communicate with a limited number of other resources. In cases like SoCs where the processors should have the opportunity to communicate with every other available resource, the number of dedicated wires would increase rapidly [16]. This solution brings two major problems. First, when the number of resources placed on a single chip increases, the number of dedicated wires needed should increase as well; in order to provide a multicast platform (scalable and support fault tolerance) for the processors they would increase enormously. Secondly, the flexibility

needed for communication within the hardware platforms would not be possible when using dedicated wires.

Another solution for the implementation of communication infra-structure within the SoCs is the use of Shared Data Buses. Shared Data Busses are composed of wires that have the responsibility of routing messages to numerous resources. Figure 2-3 shows a Shared Data Bus, which is responsible for providing the communication between four resources [17].

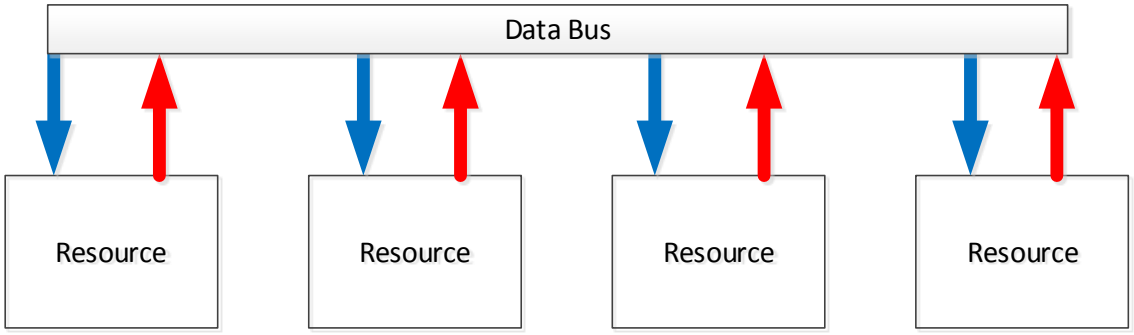

Figure 2-3: Shared Data Bus.

Shared data buses do not allow parallel communication between processors and other blocks, so only one resource can read the data available on the bus or send a message to another resource on to the bus at each clock cycle. The signalling delay of silicon technologies is continuously increasing, thus in cases like SoCs, where many pair of resources might need to communicate with each other, and the shared data bus would become a communication bottle neck.

In the past communication was not cheap compared to now computation; but this has changed today with the appearance of microchip technologies. With the advent of highly computational processors the expense of computation has decreased enormously, whereas with the existence of physical limitations like propagation delay of electrical signals, the power used for guiding the signals through wires or cables, the cost of communication has gradually increased significantly.

Therefore it is evident that the traditional bus based inter-connection technologies are not suitable for the communication of newly presented SoCs; so NoCs were presented to cover the deficiencies of bus based interconnections using networking techniques.

2.3 The Network on Chip

The advancement of semiconductor technologies miniaturisation has enabled the integration of hundreds of processing units to operate on a relatively small chip under different clock frequencies. Buses have been adopted as the communication tool in Systems-on-Chip (SoC) in the past [18, 19]. However, the growth in size and complexity of SoCs requires a new, modular type of architecture to supersede the bus-based approach since a bus is inherently non-scalable. Because of the interconnection problem in large SoC designs, communication-centric architectures or “Networks-on-Chip” (NoC) have been proposed as a solution recently [20].

To achieve a modular view on the communication fabric is the main driving factor behind the introduction of NoCs, which help resolve the electrical problems in new deep sub-micron technologies when they are used to structure and manage global wires. Meanwhile, the wires are reducing in total wire length and being used more efficiently. A NoC architecture also achieves lower power consumption and better reliability through implementing the “globally asynchronous, locally synchronous” (GALS) paradigm [21]. GALS is a structural design that maintains the benefits of synchronous systems, while by passing the problems caused by global clock net. The architecture of GALS is made of a number of synchronous blocks that communicate asynchronously. However, their clocks a synchronous locally. Hence, through offering a unified solution to a wide range of challenges in the development of very large MPSoCs, NoCs have become the preferred on-chip communication paradigm [22].

The Network On Chip (NoC) has developed as an alternative ad-hoc wiring or bus-based global interconnection network [23], as a consequence of technology scaling, which allowed the integration of a higher number of processing elements, computational cores and memories. The complexity of communication between these cores is also increased [24]. Network on chip share many similarities with the traditional interconnection networks for parallel computers with multiple processors. For instance, in interconnection networks nodes are physically near enough to each other and have high link reliability. In Addition, in order to achieve effective parallelization, its design is made under latency and bandwidth constraints.

This constraints will drive network on chip design. Nevertheless, NoCs can be distinguished from the parallel computers by their characteristics. The main differences between them are: energy and area constraints; design specialization and degree of heterogeneity of the employed components. Low power consumption is a factor of great importance in the SoC domain. Besides, the NoCs may be designed for a particular application, which, unlike networks for parallel computers, are expected to deploy for a wide range of unknown applications. A variety of components including digital signal processing (DSP) cores and field-programmable gate array (FPGA) fabric may be distributed over a single SoC, as well as the memories and processors.

2.4 **Routing Algorithms in NOC**

NoCs are mainly described based on topology, routing strategies, switching techniques, flow control, arbitration and buffering. The topology deals with the way the nodes and the channels of the network are arranged within a graph. NoC Routing is slightly similar to any network routing; the routing techniques determine how the messages are to travel within the network; starting from their source, which paths they should take to reach their destinations in the shortest and possible way [16, 25]. The arbitration is responsible for determining the order of the buffers and channels to be used by the messages based on their priorities. The control of message transmission rate and the traffic forced on buffers and channels is the job of flow controlling techniques. Switching is a mechanism implemented within

the structure of the router, which deals with the removing of data from the input channel and its placement on the appropriate output channel ports. Buffering defines the approach used in storing messages in case where the router arbitration circuits are not able to schedule the storage for all incoming and outgoing messages.

Routing algorithms can be classified into two categories: oblivious and adaptive algorithms. Oblivious algorithms route packets without any information related to the network conditions and network traffic. On the other hand, adaptive algorithms use information about the network state, typically queue occupancies, to route packets over the network. The first type, oblivious algorithm, is also divided into two subcategories, which are deterministic and stochastic algorithms. Deterministic algorithms route packets without any information about the traffic conditions of the network; therefore, they always route packets along the same path. Stochastic routing is based on randomness, where the route of the message is determined at the beginning of submission and cannot change during routing, low performance.

2.4.1 Deterministic Routing

In deterministic routing the algorithm's path is established as a function of the destination address; therefore, the output is always the same path for the same pair of nodes. To be more precise, we could say that deterministic routing is distinguishable from oblivious routing. In oblivious, a

routing table containing several output channels is composed (based on the destination address) and the packet chooses between these predefined paths. The available paths are chosen based on some selection algorithms. In the case of a deterministic routing algorithm, only one determined path is available for the packet at each step. So we come to the conclusion that deterministic algorithms are oblivious, whereas the opposite is not always true, and the deterministic routing algorithms generally follow the shortest path. The most widely used type of deterministic routing is XY routing, a variation of dimensional routing.

2.4.2 Adaptive Routing

The other routing type is Adaptive routing. The difference between deterministic routing and adaptive routing, is in Adaptive routing a message can have more than one path to travel towards its destination node, and this is could lead to a less short path. The traffic information at each possible node is also taken into account in adaptive routing when the decision is made for moving the message to the next link at each intermediate node. Adaptive routing outperforms deterministic routing in irregular traffic [26].

2.5 Network on chip Topologies

The study of NoCs requires the study of two aspects: Routing and Topology. In this section we are going to consider different existing

topologies for NoCs. Topology deals with the way the network components are connected with each other.

A number of different topologies have been proposed for MPSoC platforms are used in parallel computing systems. High throughput and low latency are two of the most important characteristics of MPSoC platforms. So in the design of these platforms, the designers should consider power consumption factors prior to high speed designs [27]. Here we will introduce some of the most important and widely used NoC topologies.

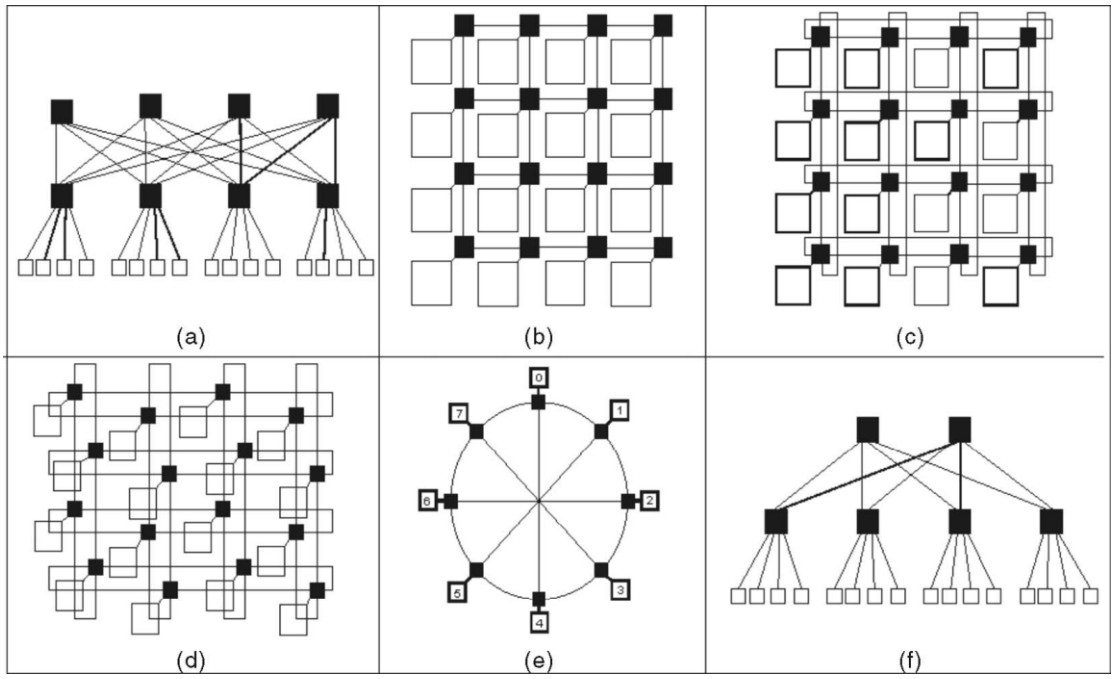

Figure 2-4: NOC architectures.(a)SPIN, (b)CLICHÉ, (c)Torus, (d)Folded torus, (e)Octagon, (f)BFT [27]

The first topology for the NoCs was introduced by [28]. Guerrier and Greiner proposed a generic interconnect platform named SPIN (Scalable, Programmable, and Integrated Network). This platform was based

on the structure of fat trees in order to interconnect the IP blocks. As shown in Figure 2-4(a), in this topology every node has four children and the parent is replicated four times for each level. From the Figure 2-4 we can see SPIN of size 16 links (number of Nodes=16). This means that the number of PE cores implemented in this network is equal to 16. The problem with this topology is that the size of the network increases with the speed of (N log N)/8 with each added node. In this topology the IP blocks reside at the leaf nodes, whereas the switches are placed at the vertices.

In [29], Kumar et al. have proposed a topology which became well known as the CLICHÉ (Chip-Level Integration of Communicating Heterogeneous Elements), shown in Figure 2-4(b). Today this topology is known as the Regular Mesh topology. Nodes topology as such are arranged in an M×N mesh, where the routers are placed at the intersection of two wires, and are composed of five input and output ports. Four of the ports are to connect each router to its neighboring routers and two are to connect the router to its relative PE. But for the routers placed at the edges, the story is a little different since they are connected to fewer neighbours. In this topology, the number of routers is equal to the number of PEs. Most current multiprocessors and multi-computers use low dimensional meshes. The reason for such interest is the simplicity and efficiency of this topology.

The Torus, in Figure 2-4(c), is an improved version of the basic mesh network. Dally and Towels [20] proposed the 2D Torus as a novel NoC topology. The architecture of the torus is basically the same as mesh; the

only difference is that the routers at the edges are connected to the routers at the opposite edge with wrap around channels. Unlike the mesh topology, where the edge routers have fewer ports, in the torus topology all the routers have exactly five ports. In this topology the number of routers is the same as the number of PE cores. The torus network has better path diversity than the mesh network, and it also has more minimal routes. It is obvious that the wrap around channels result in long delays, but to solve this problem a new topology named Folded Torus was presented (shown in Figure 2-4(d)).

Other very common topologies for NoCs are Tree and Fat Tree [30-33]. In the tree topology the nodes of the tree are the routers and the leaves are the IP cores. The nodes above the leaves are called ancestors and so the leaves connected to the ancestor nodes are called the children. The Fat Tree has the same structure except that each node has replicated ancestors, so there would be many alternative routes between the nodes [34]. A butterfly network is a unidirectional or bidirectional topology which is usually used with deterministic routing.

Figure 2-4(f) shows a new topology named Butterfly Fat Tree presented by [35]. In this architecture, as in fat trees, the PE cores are placed at the leaves and the routers are placed at the vertices. Each node is labelled with a pair of coordinates, one resembling the node's level and the other denoting its position in that level. The intermediate routers have four children ports and two parent ports. The PEs are connected to N/4 routers at the first level. The proposed models in this thesis are using Fat Tree topology

due to its simple physical implementation, deliver low latency and high performance, and fully bisectional bandwidth. More details about the advantages of using this topology is given in chapter four.

2.6 Switching and Flow Control Techniques

Network flow control is related to the existing limitations in data acceptance between routers in the links and also the buffering limitations of the routers. In circuit switching techniques there is no need for flow control. This is because the connections are established and the resources are reserved before the transmission of messages. But in packet switching techniques, since the data is buffered at every router before being transmitted to the next node, flow control is inevitable. This is due to the fact that the buffers of the routers have limited sizes and can only manage data up to a specific amount. Network flow control is managed with different switching techniques [36].

NoCs are mainly implemented based on the packet switching mechanism. It is also important to note that circuit switching techniques have also been proposed for the use within NoCs. For implementing packet switching within the NoCs, different switching modes should be considered. Switching mode deals with the way packets are forwarded through the switches. The switching mode is not directly related to the routing techniques even though some routing algorithms are designed based on specific switching methods.

The requirement to minimise buffer sizes in NoC routers has been addressed by employing wormhole switching. In wormhole switching, a packet is divided into elementary units, called flits. Each flit is composed of a few bytes for transmission and flow control. The header flit controls the route and the remaining data flits follow it in a pipelined way. Hence, if the header flit blocks, the remaining flits are blocked in situ. Routers may be implemented with buffers for a single flit per channel so that the packets are not required to exist in each intermediate router as a whole. Wormhole switching can approach low message latency at a low cost.

Different switching methods have been proposed in literature. Some of them are: store-and-forward switching, virtual cut-through switching and wormhole switching [36].

2.6.1 Store and forward switching

This switching technique is the simplest method in which packets move throughout the network as a whole. The entire packet is to be received and stored in the buffers of the router before being forwarded to the next node. This means that the buffers should be as big as the max size of a packet.

2.6.2 Virtual cut through

This methodology is an improvement on the store and forward method. As soon as the next router gives permission, the router can begin to

send packets. The packet is stored in the buffers until the forwarding begins. In this method forwarding can begin even before the whole packet is received and stored. This method uses buffers as much as the store and forward technique does, but the latency here is lower.

2.6.3 Wormhole switching

In wormhole switching the packets are divided into smaller units called flits (flow control digit or flow control unit). The header of the flits contains required routing information. The first flit of a packet is routed similar to the packets routed in virtual cut-through switching. After transmitting the first flit, the established route is reserved for the purpose of routing the remaining flits of the packet. This route is called wormhole. Switches implemented based on this method do not require large buffer spaces due to the small size of the flits; therefore, wormhole switching requires less memory than other switching modes because only one flit has to be stored at once. Routers implemented based on wormhole switching are small and fast. Usually routing algorithms based on wormhole switching techniques are hardware implemented. For this reason, mostly simple routing algorithms are used for developing wormhole switching routers. Moreover, wormhole switching provides high data transmission rates due to its pipeline nature [37].

Wormhole switching is used in many NoCs with different routing algorithms. For example, in a typical wormhole switching with virtual

channels (VC) the buffer size is chosen proportional to the number of virtual channels as well as packet length. The simplicity and adequate performance of wormhole switching makes it the switching technique of choice for most NoC applications, especially for low traffic load as too many VCs per physical channel can degrade the performance of NoCs. Increase in the number of VCs results in the increase of buffers used, since a buffer must be dedicated to each VC. Therefore, a practical number of VCs used per physical channel is two for moderate packet sizes [37]. The only problem with wormhole switching is its high sensitivity to packet blocking. This causes high performance networks to operate at low utilization and also results in using virtual channels [38].

2.7 **Messages, Packets and Flits**

The messages are used to describe the data sent or received in the network. There are still some differences of the data that is routed. A complete data that needs to be transmitted is called message, although a packet can be defined as part of message. Message is divided into packets in packet switching, the packet contain a header that contains the dentation address, body and tail flits that indicate the end of the packet, figure below . when the header flit arrives at node A, it must obtain three resources before it can be forwarded to next node B. virtual channel, one flit buffer, and one flit for channel bandwidth. The flit size in a NoC is often equal to the physical channel width. The channel width could for example be 8-bit or 32-bit wide. A flit would then have the same size [39]. To send the packet from the source

to destination, the header flit adjust the path through the network, and the remaining packet follow it in pipelined fashion, if the channel transmits the header. In case if the channel busy the header stop advancing and remain spread through the channels that have already acquired.

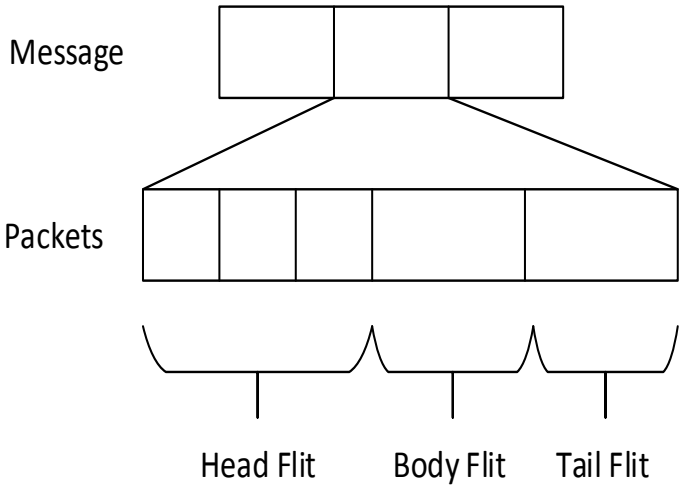

Figure 2-5: Message Composition

2.8 **Virtual Channels**

Network latency can be reduced and throughput can be increased by dividing the buffer storage associated with each network channel into several virtual channels. Typically, a virtual channel consists of buffers that can hold one or more flits of the packet and associated state information. Providing multiple buffers for each channel in the network to decouple allocation of buffers from allocation of channels [40].

Adding virtual channels to the network can be described as adding lanes to a street network; without virtual channels a network is composed of a one lane street. In such a network, when a blocked packet occurs it blocks

all subsequent packets. On the other hand, having virtual channels adds lanes. allowing blocked packets to be passed [40]. Figure 2-6 below shows two cases, one without and one with virtual channels.

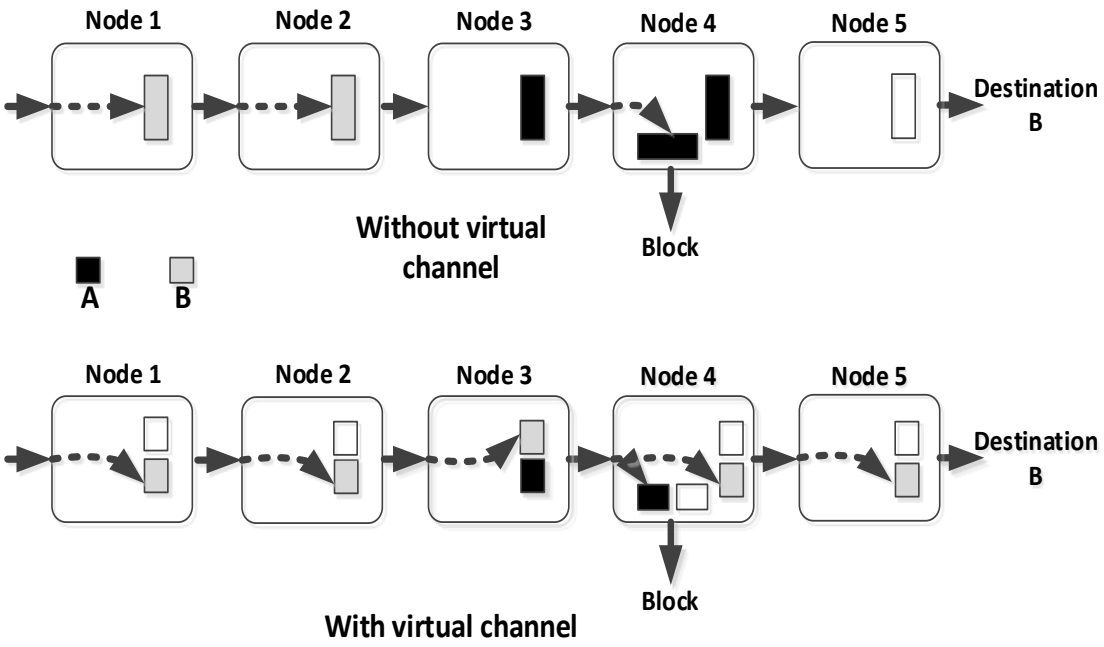

Figure 2-6: Virtual Channels [40]

2.9 **Network Patterns**

The traffic pattern has a significant impact on network performance, there are some parameters that are important to notice at the time of applying different traffic on NoC (message arrival time, message length and message distribution). Hence, the performance study of wired or wireless networks has to combine with accurate models of the network traffic. Due to the implementation of real applications being costly and time consuming, mathematical modelling of network traffic can be employed instead to evaluate the network performance at an early stage of the design

process [41]. Two key parameters of message are arrival process and destination distribution which are used to define the network traffic patterns.

2.9.1 Markov-modulated Poisson Process (MMPP)

Traffic information is an important key to evaluate the network performance at each channel. This section presents detailed information regarding the traffic on each channel in the m-port n-tree NoC. The MMPP is the doubly stochastic Poisson process with arrival rate λ_i that varies according to an irreducible continuous time Markov chain, it has been widely applied to the model arrival process of bursty traffic. MMPP has the ability to capture the time-varying arrival rate and the important correlations among inter-arrival time while keeping the analytical solution of related queueing performance tractable [42-44] . in addition, MMPP is closed under the splitting and superposition [44] and it can be used to model the splitting and superposition in the wireless or wired networks. The success of the aforementioned features make the MMPP more attractive for traffic generated by multimedia application [45, 46].

We studied the message arrival process of the network traffic generated by using two-state MMPP, as shown in Figure 2-7 below, which has been generally used to model the bursty nature of the message arrival [47-49].

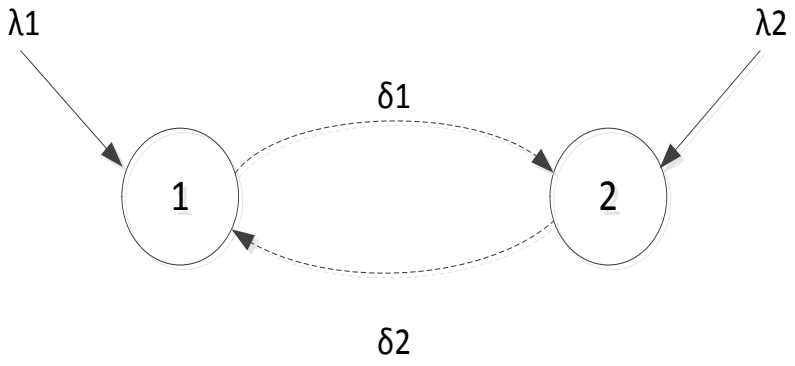

Figure 2-7: 2-state MMPP

The arrival traffic follows Poisson process with rate λ_i in state i where i = (1, 2). The transition rate out of state 1 to 2 is ∂_1, and the rate out of state 2 to 1 is ∂_2. The period of state i is in accordance with an exponential distribution with mean$^{1}/_{\partial_i}$, is shown in Figure 2-8 below:

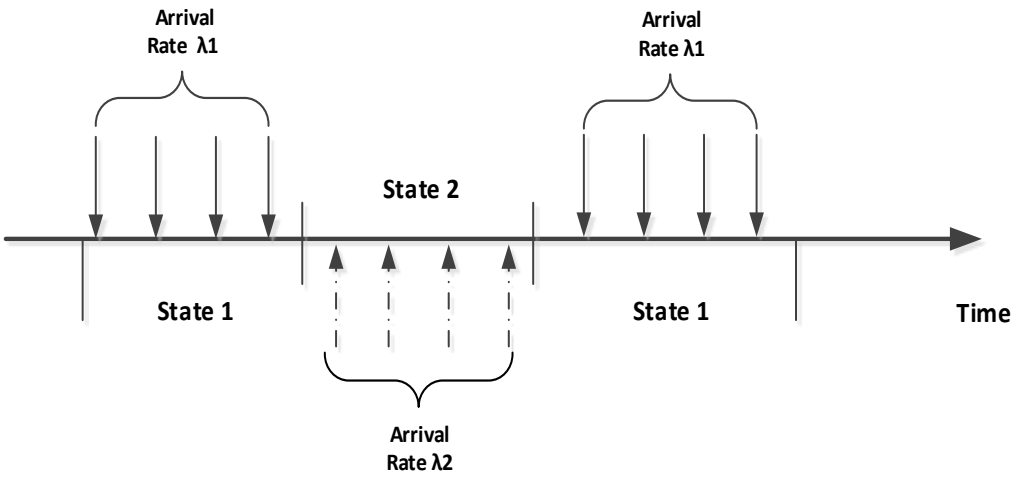

Figure 2-8: : Traffic Arrival Process of the MMPP

The characteristics of MMPP are infinitesimal generator Q_s of the underlying Markov chain and rate matrix Λ_s as:

$$Q_s = \begin{bmatrix} -\partial_1 & \partial_1 \\ \partial_2 & -\partial_2 \end{bmatrix} \quad \text{and} \quad \Lambda_s = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{2.1}$$

The traffic rate of the MMPP and its covariance function are related to how dependent the rate at one period of time is to the rate at another period of time. This is a key point in the method which will described for calculating the traffic arrival process at network channels.

Let we denote the steady-state vector of Markov chain is [50]:

$$\pi = [\pi_1 , \pi_2] = \frac{1}{\partial_2 + \partial_2} [\partial_2 , \partial_1] \tag{2.2}$$

The mean arrival rate is:

$$E\,[N\,(t)] \frac{\lambda_1 \partial_2 + \lambda_2 \partial_1}{\partial_1 + \partial_2} t \tag{2.3}$$

The variance is:

$$Var[N(t)] = \frac{\lambda_1 \partial_2 + \lambda_2 \partial_1}{\partial_1 + \partial_2} t + \frac{2\,(\lambda_1 - \lambda_2)^2\, \partial_1 \partial_2}{(\partial_1 + \partial_2)^3} t - \frac{2 \partial_1 \partial_2\, (\lambda_1 - \lambda_2)^2}{(\partial_1 + \partial_2)^4} (1 - e^{-(\partial_1 + \partial_2) t}) \tag{2.4}$$

The third centralised moment is:

$$\mu^{(3)} = E[N(t)] - E[N(t)]^3 = \frac{\lambda_1^3 \partial_2 + \lambda_2^3 \partial_1}{\partial_1 + \partial_2} \tag{2.5}$$

Distributed deterministic routing mostly used in commercial available multicomputer. Uniform distributed destination employed with deterministic routing algorithms usually perform well. On the other hand, by applying no uniform distribution the performance will be poor, especially for some pairs of non-neighbour nodes exchange information very often [10].

2.10 **Related work**

Many studies have been done on the area network on chip performance. Most of the architectures adopted for NoC domain mesh, torus, butterfly and spidergon [51-54]. An analytical model for network on chip has been porposed in [55] Kiasari, the model based on the queuing theory to analysis the delay in wormhole switch. The result predicted for the average message latency more than the result from the simulation by more than four orders of magnitude. In [56] Jin Liu has presented an analytical model for hypercube network on chip with wormhole switching and fully adaptive routing for predicting average message latency. Uniform distributed traffic has been employed and simulation approach has been used to validate the analytical model. In [57] Liu Weichen has been implemented realistic traffic benchmark suit called MCSL for eight real applications using different NoC topologies to investagate the impact of the traffic in the network. The study by [51] Moadli M has been proposed an analytical model for Spidergon topology with virtual channel using Poisson traffic. The work curried out that adding virtual channel can enhance the network performance. The simulation for NoC topologies has been presented by [58] Takabatake T, the study is

conducted to compare the Hierarchical Completely-Connected Networks (HCC) with Mesh, Ring, and Spidergon topologies based on simulation appeoach. In [59] Anam Z, presented the generic methodology for the formal verification of circuit-switched NoC using the SPIN model. This model has provided guidelines for generic model, and deadlock free. In [60] Shaoteng Liu, comparison between the Circuit and packet switches has been analysed and evaluated. The work suggest that the circuit switch operate at high clock more than packet switch.

In [4] Bononi has carry out comprehensive evaluations and compared different NoC architectures in terms of latency, throughput, and energy dissipation, based on simulator using flit-level event-driven wormhole switching. Multicast scheme in wormhole switch presented by [61] Zhonghai Lu, shows that the multicasting starts after a multicast group is established, and beneficial in throughput. A tiny NoC has been presented in [62] Marcon. The work has developed the architecture of 3D mesh to enhance the latency and the area of NoC comparing with basic mesh using synthetic and mapping independent traffic.

Most of the proposed models on the performance study have been done in the area of network on chip carried out in a mesh or tours topologies. There are many other topologies need more investigation in order to get better performance. Analytical models in network on chip performance reported in the literature are based on synthetic traffic not real traffic. In

addition few analytical models has been presented in the literature using Spidegon topology with virtual channel under Poisson traffic.

Chapter 3.

SIMULATION DESIGN MODELLING

With the rapid increase in complexities in communication systems, due to their containing several components performance evaluation of the communication systems is considered one of the most cost effect and reliable tools to guarantee the success of the systems before deployment. The success of system deployment can be evaluated based on computer simulation tools, which is targeted by performance evaluation. The communication system is represented as computer programs that emulate the system's behaviour in a realistic environment. In this imulation model is built using the OMNeT++ simulation package, which is very well designed, modular and widely used for education purposes [63].

3.1 Network systems modelling and evaluation

Designing a large scale and complex model for network systems is an essential matter that must be addressed in a very accurate environment to observe the particular behaviour of the employed system in realistic environments. Many tools and techniques have been broadly used for modelling the network system to be evaluated. In designing new network technologies, efficient and accurate modelling is needed, assisting optimization. Such modelling will save time, effort and resources throughout the development process.

3.2 Approaches for modelling and evaluation

In order to measure a networking system's performance and its behaviour, there are three methods required. These include: (i) Mathematic analysis, (ii) computer simulation, (iii) experimental measurements, for representing the system and evaluating its feasibility. The above three methods are known to be practical and efficient, saving both time and resources. [63]. Further explanations of methods insights will be offered in the following sections.

i. **Mathematic analysis:**

This method is a mathematical model that uses theories and equations to represent and analyse the whole system's functions and behaviour. These equations are used to represent the system performance measures as numerical factors, where they are analysed and evaluated according to the system being designed. This method may not be easy to use for complicated models; moreover it is not always possible to find solutions for real systems by reason of imprecision and the approximation required, and so this is less frequently used for large complex systems.

ii. **Computer simulation:**

Computer simulation has been widely demonstrated to be valuable in increasing our understanding of a wide range of network systems and in evaluating the efficiency, in terms of time and resources, of a network system. Given that the system can be analysed and evaluated without the

need to build a real functional system, the optimization techniques can be applied simply during the development process. The method here is representing the whole networking system based on writing a piece of computer program to evaluate the system's behaviour and performance, using the programming languages or simulation packages designed for it. The written and built of the intended network system to state the changes in the system as a function of time. By running the simulation with different simulation scenario can evaluate the system's performance, and then evaluate the system responses to such changes in the system environment.

iii. **Experimental measurements:**

This is directed method to evaluating system performance, although it is necessary to build a full size, real, and functional system. When the system has been built, it is run and its actions and events observed for performance metrics. This approach is not often used, since it is time consuming and costly to implement particularly for large and complex systems.

3.3 Computer Simulation Methods

Computer simulation is sometimes referred to as computer experiments. Modelling, simulating and evaluating have been well known tools, widely used to develop more efficient network systems. To evaluate a networking system under different environments with many scenarios applied, the system is closely monitored and improvements sought. The

results are collected from the simulation and the performance requirements assessed by comparing them together to get the system performance evaluations in order to judge the system's feasibility in terms of complexity, efficiency and general QoS [64].

In order to study a system using a computer simulation, we first abstract those features from the system that we think are significant in determining its performance. An abstraction is named as a system model. Next a computer program is written whose execution emulates the model's behaviour [65]. Figure 3-1 (below) illustrates the general role of simulation in design, beginning from the real-world system at the start until a valid simulation system is ready to be implemented.

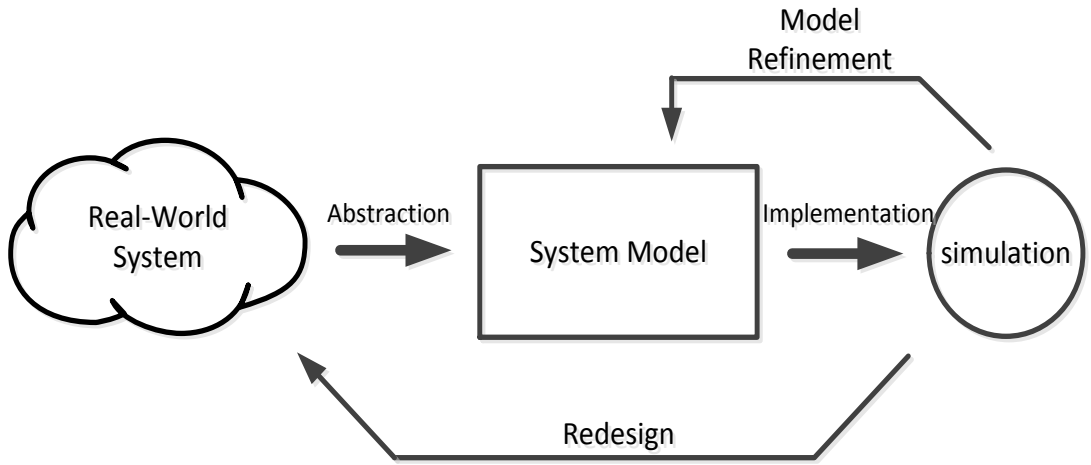

Figure 3-1: Simulation design process.

3.3.1 OMNeT++ Simulation Platform

Computer simulation tools are important for the study of a communication networks' performance. Designing a good network model can help in the process of developing and improving the network. The activities of

the network model consists of events which occur at points in time which may change the state of the model. Discrete and continuous events time are two types of computer simulation based on the system events time. In the first type, the discrete event, the change occurs in the state variable at distinct points of time “events”. When the number of events are finite between two events, we call the simulation a discrete event; this type can serve as an approximation. However, in some systems the state variable changes continuously, and in such cases continuous simulation events is more suitable. OMNeT++ is one of the discrete event simulation tools used in the field of communication network design and analysis [63].

3.3.2 OMNeT++ Simulation Package

It is an object-oriented modular package and one of the discrete event simulation tools designed to simulate computer networks, multi-processors and other distributed systems. OMNeT++ stands for Objective Modular Network Testbed in C++. It is an open source simulator and its applications can be extended for modelling other systems as well. It has become a popular network simulation tool. OMNeT++ model consists of simple or compound modules, which communicate by passing messages along connections that represent links in real networks, as shown below. Each object in the network is labelled by a module, these simple modules being written in C++ language. Many object libraries are included to be used to make simple modules, and the best simple modules to be used are the ones that are implemented [66].

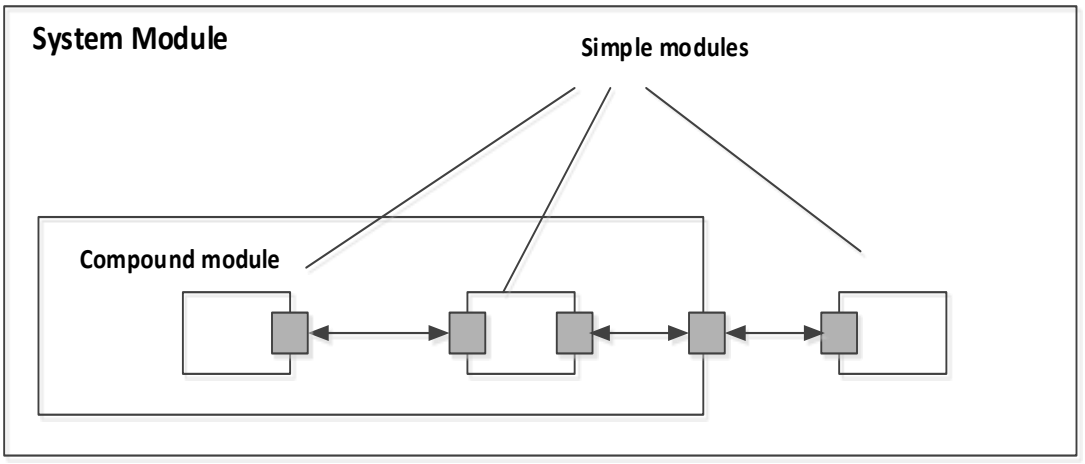

Figure 3-2: Simple and Compound Modules [66]

OMNeT++ provides efficient tools for the user to describe the structure of the actual system. The model in OMNet++ consists of hierarchically nested modules that communicate by passing messages to each other. The system module is the top level of the module, which consists of sub-modules and these can also contain further sub-modules themselves. The nesting module is unlimited to help the developer to map the logical structure of the real system, as in Figure 3-2 which describes the model structure in OMNet++ by NED language [66].

OMNeT++ consists of many components that interact with each other to perform simulation tasks:

1. Simulation kernel library
2. Compiler for the NED topology description language (nedc)
3. Graphical network editor for NED files (GNED)
4. GUI for simulation execution, links into simulation executable (Tkenv)
5. Command-line user interface for simulation execution (Cmdenv)

6. Graphical output vector plotting tool (Plove)
7. Utilities (random number seed generation tool, makefile creation tool, etc.)
8. Documentation, sample simulations, contributed material, etc.

3.4 Simulation Modelling in OMNeT++

Using OMNeT++ to model and simulate network systems can be divided into levels. The top level of the network model is comprised of different network nodes connected by links. These nodes generally consist of simple or compound modules. The basic building blocks are named as simple modules for other modules, which can be grouped together or nested to form a compound module. The purpose of designing modules is to perform a particular system's functions. [66]

There are three kinds of model levels: node, network and process models, which will be described in the next sections.

- **Node model**

Nodes are connected with each other with specific functions to create a network. A node contains a collection of fundamental connected modules representing the node's function. These modules are able to handle messages passing through the network according to the predefined procedures in the process model. Also a module can be any network objects, for instance a router or processor.

- **Network model**

As mentioned before, a system module is divided into levels. The top level is represented by a network model, which can be defined as a network topology, which includes a number of nodes connected by links. The system behaviour needs to be controlled during the simulation by the network model throughout its parameters. For a large system, a network module can be nested in a hierarchal approach to simplify the complexity of the system.

- **Process model**

This process model is the simplest model written in C++ languages, which is represented as a finite-state machine defining the basic behaviour of the system modules along with the current event. Once the event has taken place the simulation kernel passes control to the specific process proposed to perform the required action. After that the process replies to that by moving from one state to another and executing the selected procedures and functions. The process model will be in idle state when there is no action to take.

3.5 **Simulation Design**

The final step and most important in the design is the simulation for validate the proposed model. Simulate the network with different configurations and scenario can be used to evaluate the analytical model. When the simulation finished, the results collected used to compare the

results obtained from the model. The methodology and the steps followed in the simulation design will be discussed in this section.

3.5.1 Simulation structure

Simulation built as modules hierarchically that interact with each other made up to closely be similar to the build-up of real network on chip using OMNET++, Figure 3-3 show the simulation structure. The compound models consist of simple modules defined using NED file and C++.

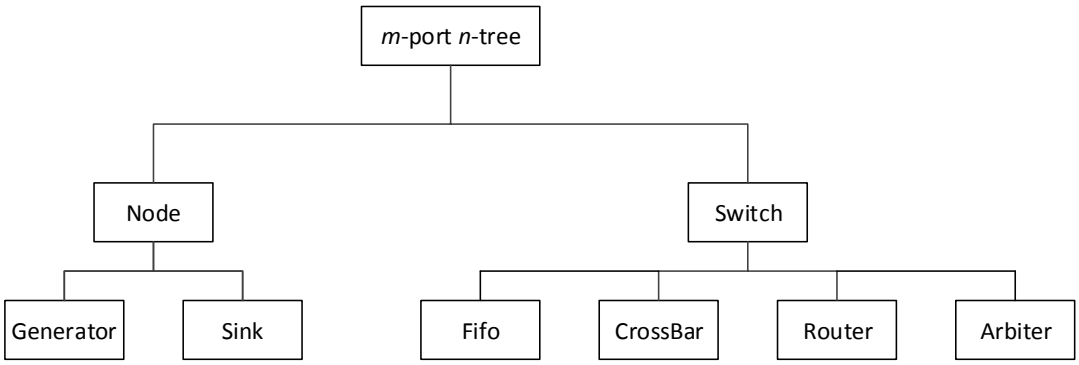

Figure 3-3: Network simulator module hierarchy

3.5.1.1 m-port *n*-tree

In the simulation model initialization phases are required to start the simulation, for instance topology. The topology of an on-chip network specifies the structure in which the processing nodes are connected. In the model presented the fat tree topology was adopted. The m-port n-tree (4-port 3-tree) is one fat tree topology; every network has m communication ports that are connected with other processing nodes or switches. In addition, the processing nodes and the switches descending use half of the ports to

connect to switches and the other half are used to connect to switches' ancestors. The root switch uses all the communication ports to connect with their descending or processing nodes. The topology consists of a number of processing nodes N and switches N_{sw} are given by:

$$N = 2\left(m/2\right)^{n} \tag{3.1}$$

$$N_{sw} = (2n-1)\left(m/2\right)^{n-1} \tag{3.2}$$

3.5.1.2 The Node

Each node has traffic generator according to MMPP distribution and sink modules receives the packets from network interface through its ejection channel and simply drops them after collecting some statistics as shown in Figure 3-4.

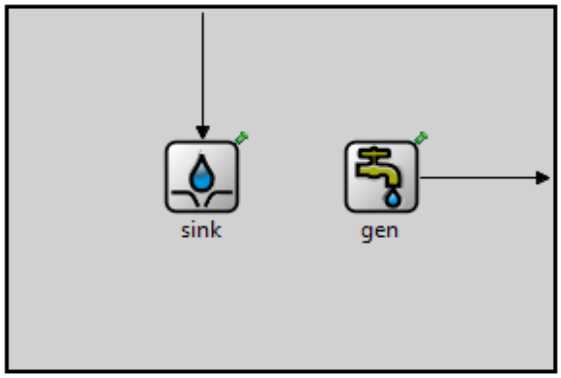

Figure 3-4: Node Structure

3.5.1.3 The switch

The switch has its own compounded that defined the structure and connectivity of Fifo, Router, Arbiter, and CrossBar Figure 3-5 show the switch structure. Fifo is a flit buffer where all the packets are stored before entire the network. Router is responsible for route the flit and request the outport for it, the routing algorithm used was presented in [67], the routing algorithm is deterministic based on advancing tables stored in the switches. Arbiter is to determine which lane is granted the outport CrossBar is a fully connected and it responsible to route the flit to outport.

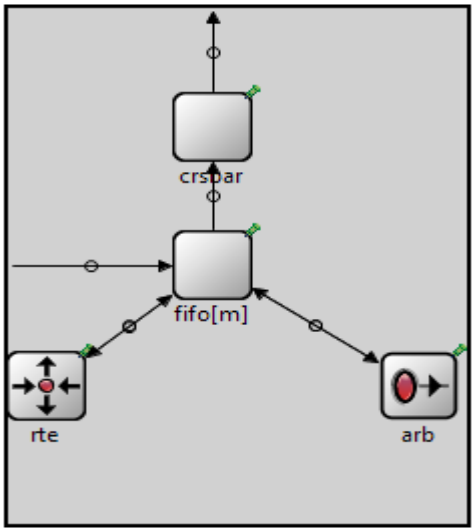

Figure 3-5: Switch Structure

3.5.2 Simulation Cycle

In this section will discuss how simulation works, Figure 3-6 illustrate the simulation flow diagram for each node. Initialization and build

the topology required to star the simulation such as topology structure and links. From the NED file read the topology parameters followed by the link. Traffic generated arrival according the MMPP distribution. It is represented as a sequence of scheduled number of flow requests. Once the connection request arrives the traffic generator provides the connection request with uniform distribution destination node. Schedule events consists of all jobs need to done during the simulation. After setup the events start the first job by handle the message and star routing. Finally collect the statistic results.

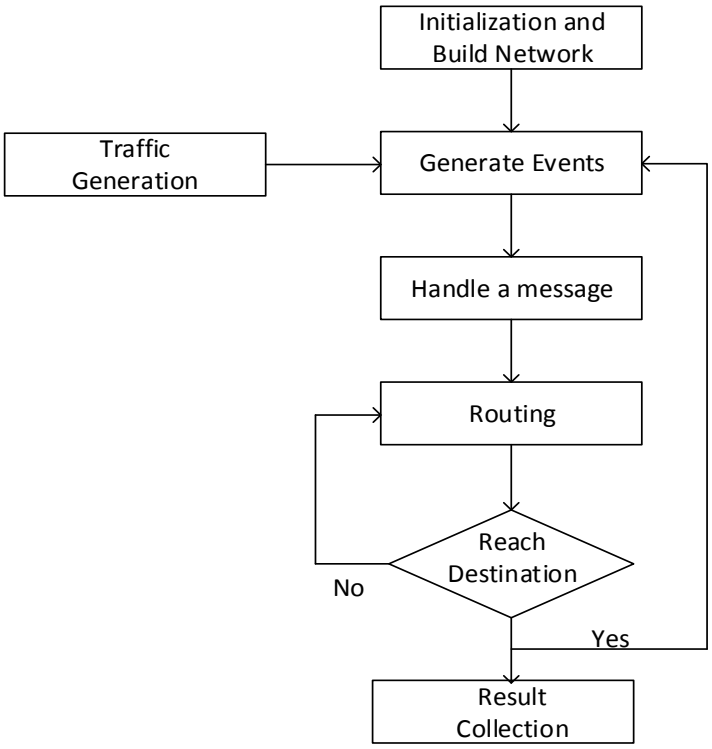

Figure 3-6: Simulation Model

3.5.3 Simulation Environment and setup

Network on Chip (NoC) has developed as a new on-chip communication method. It offers better scalability, latency and area

compared to bus-based on-chip interconnection and throughput. Yet, the NoC design space is very large and high dimensional. It includes the optimization of topology, traffic switching technique, routing mechanism and virtual channels per physical channel.

To validate the analytical model by means of a discrete event simulator, operating at the flit level. The simulation experiment was run until the network reached its steady state; that means any increase in the simulation time will not change the statistical result by any significant amount. The message latency can be known as the mean time from the message being generated until the last flit reaches the processing element (PE) at the destination node. The network cycle time in the simulator is the transmission time of a single flit to cross the network from one node to another. Every message generated from the source node is modelled by Markov Modulated Poisson Process (MMPP) with the infinitesimal generator Q_s , and rate matrix Δ_s.

In the experiments the first 8,000 messages were discarded allowing for a warm up stage to guarantee the simulation has reached the steady state. The messages are split into 10 batches, and each batch size has 8000 messages, and thus message latency is collected for a total number of 80,000 messages. The batch means method [68] is used to collect the statistics of performance metrics, and, as mentioned above, the messages are divided into many batches and the statistics are gathered for these batches.

To calculate the batch means, let ***a*** and ***b*** denote the batch number and batch size respectively. Taking a set from a single simulation run $\{X_1, X_2, \ldots . X_{ab-1}\}$, where X_i $(0 \leq i \leq ab-1)$ is the message latency of i-th message, the message latency S_j gathered for the j-th batch, can be calculated by

$$S_j = \frac{1}{b}\sum_{s=0}^{b-1} X_{bj} + s \tag{3.3}$$

where $0 \leq j \leq a-1$. Thus the mean message latency can be determined as the mean of batch means:

$$S = \frac{1}{a}\sum_{j=0}^{a-1} S_j \tag{3.4}$$

As each batch means sample is an average over many of the original samples, the variance between batches means is greatly reduced, which in turn decreases the standard deviation of the measurement performance metrics. This leads to better confidence in the performance results. Extensive simulation experiments have been achieved to validate the model for various combinations of message lengths, different MMPPs traffic inputs and number of virtual channels per physical channel.

Chapter 4.

PERFORMANCE MODELLING AND ANALYSIS OF NETWORK ON CHIP UNDER M-PORT N-TREE BURSTY TRAFFIC

4.1 Introduction

Networks on Chip has been proposed as a solution to mitigate complex on-chip communication problems. NoCs are composed of PE cores (or processing elements), which are interconnected by on-chip switching fabrics. A step in the design process of NoCs is hardware virtualization, which is mapping the PE cores on to the tiles of chips [69-71]. The communication among the PE cores greatly affects the performance and power consumption of NoCs, which itself is closely related to the placement of PEs on to the tiles of the network.

To overcome the problems of scalability, complexity and performance, Networks-On-Chips (NoCs) have been proposed as a promising solution to reduce the overheads of buses and MPSoCs connected by means of general-purpose communication architectures [2, 72-74]. This technology provides a solution to overcome the limitations that exist in the traditional bus-based interconnection technologies. The scalability of NoC has made it ideal for larger designs. Since the NoC technology builds on top of the latest evolutions of bus architectures, and uses packet-based communication paradigms for the means of communication, it can overcome

many of the issues related to the interconnect fabric design better than shared buses architectures.

Most of the work has been done in analytical models on the performance study in the area of NoCs have been focus on specific topology and Poisson traffic only [51, 55, 75]. The well-known Markov Modulated Poisson Process has been used for the purpose of its ability to model the time varying arrival rate and correlation between inter arrival time. In this chapter the analytical model has been implemented investigate the performance of NoC by employed *m*-port *n*-tree topology under bursty traffic.

4.2 The m-port n-tree NOC

Network on Chip topology identifies the physical organization of the interconnection network. It describes how nodes, switches and links are connected to each other. NoC topologies are classified into two classes based on the type of connectivity. On one hand, there is the direct connect network, where each node is connected to at least one core (PE), and on the other hand is the indirect connect network, where the node is not connected to any core (PE). Most of NoC employs regular topology, since it is easy to be laid out on a chip surface. Fat-tree has been the most popular networks topology for the past fifty years, and has been adopted for NOC domain. Commercial machines and many research prototypes have adopted the variations of different kinds of fat-tree. Bisection bandwidth is one of the important properties for fat-tree, which scales linearly with the network size[76]. In this work we focus on fat tree formed by m-port n-tree topology.

Comparison between a different network topologies has been proposed by Pandel [27], considering throughput, latency, and energy consumption.

An m-port n-tree topology contains N processing nodes and N_{sw} switches. The processing node is known as n-tuple $\mathrm{PN}(\mathrm{P}_0, \mathrm{P}_1, \ldots \ldots \mathrm{P}_{n-1})$, where P $\in \{0,1, \ldots., m-1\}$ $\{0,1, \ldots., (m/2)-1\}^{n-1}$ and it can be calculated by

$$N = 2 \left(\frac{\mathrm{m}}{2}\right)^{\mathrm{n}} \tag{4.1}$$

The communication switch known as n-tuple $SW\ (w_0, w_1, w_2, \ldots\ldots, w_{n-2}, l)$ L is the level of the switch where l $\in \{0,1,2, \ldots., n-1\}$, and w $\in \{0,1, \ldots., (m/2)-1\}^{n-1}$ and can be calculated by

$$N_{sw} = (2n-1) \left(\frac{\mathrm{m}}{2}\right)^{\mathrm{n-1}} \tag{4.2}$$

Figure 4-1 shows the 4-port 3-tree:

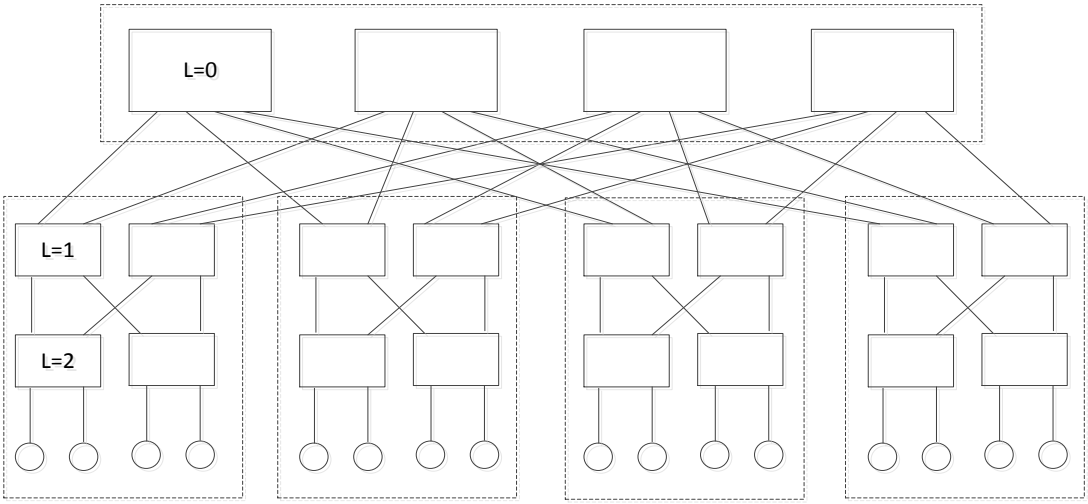

Figure 4-1: 4-port 3-tree topology

4.3 Traffic Analysis

Traffic information is an important key to evaluate the network performance at each channel. This section presents detailed information regarding the traffic on each channel in the m-port n-tree NoC. The MMPP is a doubly stochastic process with an arrival rate governed by m-stat irreducible continuous time Markov chain [44]. The 2-state MMPP has been broadly used for its simplicity, ability to capture the time-varying arrival rate and correlation between intra-arrival times. Additionally the operations of superposition and splitting are highly recommended to be used by the MMPP traffic. These two operations of MMPPs give rise to a new MMPP [44]. The success of the aforementioned features make the MMPP more attractive for traffic generated by multimedia application [45, 46].

4.4 Assumptions of the model

The model uses some assumptions that are widely used in the literature [44, 77-80]:

1. Nodes generate correlated traffic, which follows an independent MMPP [81] whose infinitesimal generator Λ_s , and Υ_s is the rate matrix as given in the previous section.
2. Message length (Lm flits).
3. Messages generated by the source nodes are sent with probability ϕ.
4. The network switches are input buffered and each channel is associated with a single buffer.
5. Deterministic routing is employed, with the shortest path routing algorithm.

Table 4-1: Key notations used in the derivation of the model.

Λ_s, Υ_s	Parameter matrices of MMPPs to model the traffic generated by the source node
L	Mean Message Latency
T	Mean Network Latency
Lm	Message length in flits
R	Average time for tail flit to reach its destination
P_k	The probability of a message traversing $2k$ links
d	mean message distance
T_s	The service time experienced by *2j* link at $S-1$ stage
$T_{s,j}$	The service time at internal stages *s*

$W_{I,j}$	Blocking time that message acquires a channel at stage *s*
$P1_{s,j}$	Probability that the message is blocked at this stage *s*
$Wp_{I,j}$	Waiting time by the message to acquire a channel
m	Number of port
n	Number of tree
W	The waiting time at the source node
N	Number of node
Nsw	Number of switch
α_{net}	Network latency
β_{net}	Transmission time
j	Number of link (ascending or descending link)
S	Network stage to calculate service time

4.5 **The Analysis Method**

The mean message distance mainly is affected by the traffic pattern, the average number of links that message needs to reach its destination. The probability of a message traversing $2k$ links (one k link in ascending phase and another k link in descending phase) to reach its destination is P_k. Different values of P_k could produce different distributions of message destinations and, thus, different mean message distances. In [5] the number of nodes at distance $2n$ is $(m/2)^{n-1}(m-1)$ in *m*-port *n*-tree topology, thus P_k is used:

$$P_k = \begin{cases} \dfrac{\left(\frac{m}{2}-1\right)\left(\frac{m}{2}\right)^{k-1}}{N-1} & for\ k = 1,2 \ldots, n-1 \\ \dfrac{(m-1)\left(\frac{m}{2}\right)^{k-1}}{N-1} & for\ k = \mathrm{n} \end{cases} \quad (4.1)$$

Message latency can be defined as the amount of time that message takes to traverse through the network to reach its destination including any time spent buffered at the source node, to compute mean message latency L in this network, it comprises of three factors: the mean network latency, T, the mean waiting time seen by messages at the source node , W, and the average time for tail flit to reach its destination, R. Therefore, the mean message latency, L, can be written as:

$$L = T + W + R \quad (4.2)$$

Moreover, each message generated through any of the source node to be sent with mean message distance, can be given by $d = \sum_{k=1}^{n} 2jP_k$. The traffic arriving at the network channel is t_{sa} , the number of nodes is N and the messages generated by these nodes are sent to $4nN$ channels in the network, thus t_{sa} is:

$$t_{sa} = d/4n \quad (4.3)$$

4.5.1 Mean Network Latency

Each message may travel to a different number of nodes to reach its destination by taking into account the transmission delay of *2j* link. The location of the switches between the source and destination node labelling as the network stages. The first stage starts numbering from the stage next to the source node (stage 0), and goes up till it is closer to the destination node. In our topology (m-port n-tree) the number of stages to travel *2j* link is $S = 2j - 1$. Determine the service time experienced by the message at the final stage first and then carried out backward to the first stage. Thus, the service time experienced by *2j* link at $S - 1$ stage is:

$$T_s = Lm\, t_s \tag{4.4}$$

Where t_s is a type of connection in this topology node to switch or switch to node, and switch to switch. The t_s can be calculated as $t_s = 0.5\alpha_{\text{net}} + F_m\beta_{\text{net}}$, where α_{net} is network latency, and β_{net} is the transmission time of one byte, and Lm is the message length in flits. At internal stages $0 \le s \le S - 2$ the service time might be more based on channel and would be idle when the channels of subsequent stages are busy. The service time at internal stages can be found as:

$$T_{s,j} = \sum_{l=k+1}^{K-1} W_{l,j} + Mt_{s1} \quad 0 \le k \le K - 2 \tag{4.5}$$

where t_{s1} is a switch to switch connection, and can be calculated as $t_{s1} = \alpha_{sw} + L_m\beta_{net}$, where α_{sw} is switch latency. Also $W_{I,j}$ blocking time that message acquires a channel at stage *s,* can be calculated by the waiting time of the message to acquire a channel when blocking occurs $Wp_{I,j}$, and the probability that the message is blocked at this stage $P1_{s,j}$. We can get:

$$Wp_{I,j} = W_{I,j}\, P1_{s,j} \tag{4.6}$$

To calculate the $Pb_{s,j}$ first it is necessary to compute the joint probability $P_{a,b}$ using a bivariate Markov chain shown in Figure 4-2: Bivariate Markov Chain for Determining Blocking Probability , state $P_{a,b}$where *a* donates that channel busy or idle and $MMPP(2)_A$ is in state *b* .The transition rate out of state $P_{a,b}$ to $P_{a+1,b}$ is λ_A, where λ_A is the traffic arrival rate on the network channel when the $MMPP(2)_A$ is in state *b,* the rate from $P_{a+1,b}$ to $P_{a,b}$ is $1/T_{s,j} - \lambda_A$. The reduction of λ_A is used to account for the arrival of message while a channel in this state [12].

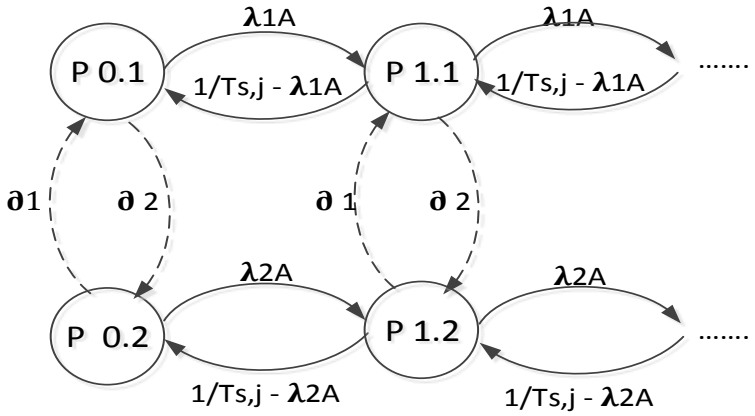

Figure 4-2: Bivariate Markov Chain for Determining Blocking Probability

The arrival rate λ_A can be found from rate matrix $\Lambda_{\boldsymbol{A}}$ while the ∂_A is a transition rate given by infinitesimal generator $\boldsymbol{Q}_{\boldsymbol{A}}$. In the steady state the model yields the following:

$$\begin{cases} (\lambda_{1A} + \partial_{2A}) = \partial_{1A} \mathrm{P}_{1,2} + \left(\frac{1}{T_{s,j}} - \lambda_{1A}\right) \mathrm{P}_{1,1} \\ (\lambda_{1A} + \partial_{1A}) = \partial_{2A} \mathrm{P}_{0,1} + \left(\frac{1}{T_{s,j}} - \lambda_{1A}\right) \mathrm{P}_{1,2} \\ \left(\frac{1}{T_{s,j}} - \lambda_A + \partial_{2A}\right) \mathrm{P}_{1,2} = \lambda_{1A} \mathrm{P}_{0,1} + \partial_{1A} \mathrm{P}_{1,1} \\ \left(\frac{1}{T_{s,j}} - \lambda_{1A} + \partial_{1A}\right) \mathrm{P}_{1,2} = \lambda_{1A} \mathrm{P}_{0,2} + \partial_{2A} \mathrm{P}_{1,1} \\ \sum_{a=0}^{1} \sum_{b=1}^{2} \mathrm{P}_{1,b} = 1 \end{cases} \tag{4.7}$$

Solving the above system of equation yields the probability $P1_{s,j}$.

To calculate the waiting time $W_{I,j}$, the channel at source node is modeled as an MMPP/G/1 queuing system, according to [10]. The waiting time can be calculated as:

$$W_a = \frac{1}{2(1-\rho)} [2\rho + \lambda_{tot} h_2 - 2h_1((1-\rho)g + h_1 \pi \Lambda_A)(Q_A + e\pi)^{-1} \lambda \tag{4.8}$$

$$W_{I,j} = \frac{1}{\rho}\left(W_a - \frac{1}{2}\lambda_{tot} h_2\right) \tag{4.9}$$

where the $h_1 \, and \, h_2$ are the first two moments of the service time on the network channel, which can be determined from Laplace-Stieltjes transform

[82]. $\boldsymbol{\rho}$ is the traffic intensity, and can be calculated by $\rho = h_1\,\lambda_{tot}$ where λ_{tot} is the mean arrival rate at network channels, and it is equal to $\lambda_{tot} = \pi\,\lambda$, where π is the steady-state vector of MMPP_A , and $\lambda = \Upsilon_s e$the column unit vector of length 2 $e = \begin{bmatrix}1\\1\end{bmatrix}$. The algorithm for calculating the matrix, *g* has been described in [10].

The average of all possible destinations of a message in the network is the network latency

$$T = \sum_{j=1}^{n} P_{k,\mathrm{n}}\; T_j \tag{4.10}$$

where D_i is equal to the service time of the message at stage $s = 0$ ($D_j = D_{0,j}$)

4.5.2 The mean waiting time

At the source node the message is injected into the network with equal probability, therefore the traffic arriving at an injection channel at source node represented by MMPP_{s1} is the fraction of that generated by a source node F_{s1}. Based on cookbook [44], it is assumed that the resulting process from the splitting of $MMPP$ has the same underlying Markov chain, such as the original $MMPP$. The infinitesimal generator Λ_s and the rate matrix Υ_s are given by:

$$\Lambda_{s1} = \Lambda_s = \begin{bmatrix} -\partial_{s1} & \partial_{s1} \\ \partial_{s2} & -\partial_{s2} \end{bmatrix} \text{ and } \quad \Upsilon_{s1} = \mathrm{F}_{s1}\Upsilon_s \begin{bmatrix} \lambda_{s1} & 0 \\ 0 & \lambda_{s2} \end{bmatrix} \tag{4.11}$$

The message experiences before entering the network modelled as a $MMPP/G/1$ queuing system, where the arrival process modeled by $MMPP_{s1}$ and the service time at transmission delay is given by $T = \sum_{j=1}^{n} P_{k,\mathrm{n}}\ T_j$. Waiting time W at the source node can be calculated using the method we used in the previous section for network latency T. Thus, the wating time can be processed as:

$$\mathrm{W}_1 = \frac{1}{2(1-\rho)}[2\rho + \lambda_t h_2 - 2h_1((1-\rho)g + h_1\pi\Upsilon_{s1})(\Lambda_{s1} + e\pi)^{-1}\lambda_{s1} \tag{4.12}$$

$$\mathrm{W} = \frac{1}{\rho}(\mathrm{W}_1 - 0.5\lambda_t h_2) \tag{4.13}$$

where $h1$ and $h2$ denote the first and second moments of the service time seen by the message respectively, and can be calculated by differentiating $L_t(s)$ Laplace-Stieltjes Transform, as mentioned above. Also ρ is the traffic intensity, which can be calculated by $\rho = h_1\ \lambda_t$ where λ_{tv} is the mean arrival rate at network channels. It is equal to $\lambda_{tv} = \Pi\ \lambda_c$ where π is the steady-state vector of MMPP_c, and can be written as $\pi = [\pi_1, \pi_2] = \frac{1}{\partial_{1\mathrm{s}} + \partial_{2\mathrm{s}}}\ [\partial_{2\mathrm{s}}, \partial_{1\mathrm{s}}]$, $\lambda_s = \begin{bmatrix} \lambda_{1s} \\ \lambda_{2s} \end{bmatrix}$, and e is the column unit vector of length 2 $e = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and the algorithm to compute g can be found in [44].

The average time for the tail flit to reach its destination R can be given as:

$$R = \sum_{i=1}^{n} [\, P_{i,\mathrm{n}} \, (\sum_{k=1}^{K-1} t_{cs} + t_{cn}) \,] \tag{4.14}$$

4.6 **Validation and Analysis**

In this work the simulation is carried out using OMNeT ++ [63] in order to validate the accuracy of the above analytical model. Various simulation experiments have been performed to validate the model. However, further increase in the simulation time will not change the results by any significant amount. The message latency can be defined as the mean time from the message being generated at the source node until the last data flit reaches the destination node. Moreover, in the simulation the network cycle time is defined as the transmission time of a single flit to cross from one node to another. As mentioned the messages generated by the source node are modelled by $MMPP$ with its parameters being the infinitesimal generator Λ_s and the rate matrix Υ_s .

Extensive simulation experiments have been achieved to validate the model for several message lengths, different $MMPP$ traffic inputs and switch size. Yet, for the sake of specific illustration, latency results are presented for the following scenarios: for network topology: 4-port 3-tree the system parameters are set at the following:

1. Message length: Lm=64 and 128 flits;

2. Flits length: $Lf = 256, 512 \text{ and } 1024$ bytes
3. Network and switch latency are 0.005 and 0.01 time unit respectively.
4. Λ_s is the infinitesimal generator of $MMPP$ and is set as follows:

$$\Lambda_s = \begin{bmatrix} -0.04 & 0.04 \\ 0.08 & -0.08 \end{bmatrix}, \Lambda_s = \begin{bmatrix} -0.04 & 0.04 \\ 0.09 & -0.09 \end{bmatrix},$$

$$\Lambda_s = \begin{bmatrix} -0.05 & 0.05 \\ 0.09 & -0.09 \end{bmatrix}, \Lambda_s = \begin{bmatrix} -0.04 & 0.04 \\ 0.06 & -0.06 \end{bmatrix},$$

$$\Lambda_s = \begin{bmatrix} -0.06 & 0.06 \\ 0.09 & -0.09 \end{bmatrix}, \Lambda_s = \begin{bmatrix} -0.03 & 0.03 \\ 0.09 & -0.09 \end{bmatrix}$$

$$\Lambda_s = \begin{bmatrix} -0.1 & 0.1 \\ 0.09 & -0.09 \end{bmatrix}.$$

We have decided to use a range of message lengths (64 to 128 flits) to evaluate the model, this is because state of the art work [39, 83, 84] show that these message lengths exhibit different saturation points, it is therefore necessary to evaluate the latencies at varying saturations. The flit size is affect the performance within a network on chip router, unfortunately most of the literature do not justify the choice of a particular size. Although a recent work by Junghee Lee [85] has shown that it is difficult to select an optimum flit size, it has also been reported that large flit size add overhead cost these by reducing performance. Since no much work has been done to prove Junghee Lee's observation we have decided to choose a range of flit size (256,512, and 1024) with a view to having optimum performance, and better answer to the question of flit size selection. The proposed network and switch latency (i.e. 0.005 and 0.01 time unit) have been used to get better performance.

The figures below, describe the performance results for the message latency predicted by the analytical model plotted beside those by simulator as a function of traffic rate.

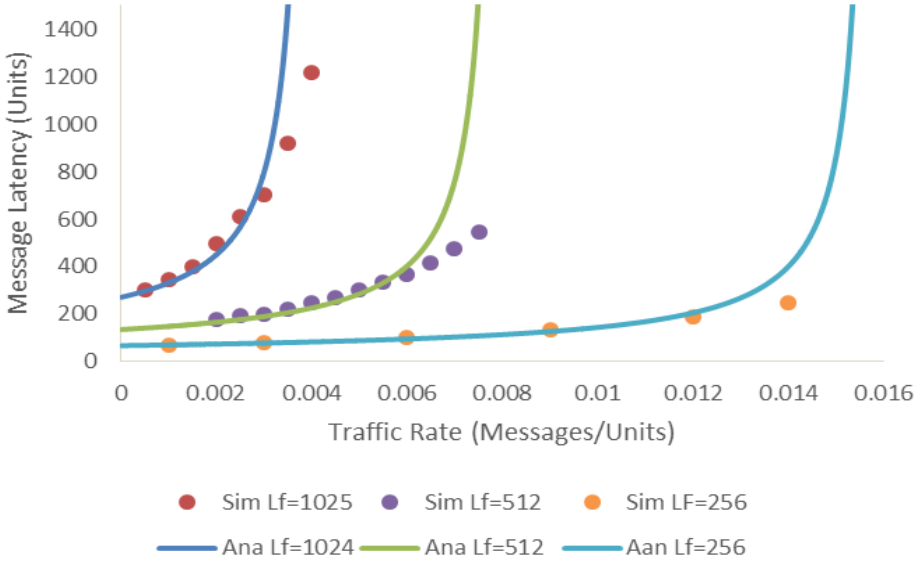

Figure 4-3: Latency predicted by the model and simulation: Lm=128 ∂_{s1}=0.04, ∂_{s2}=0.08,

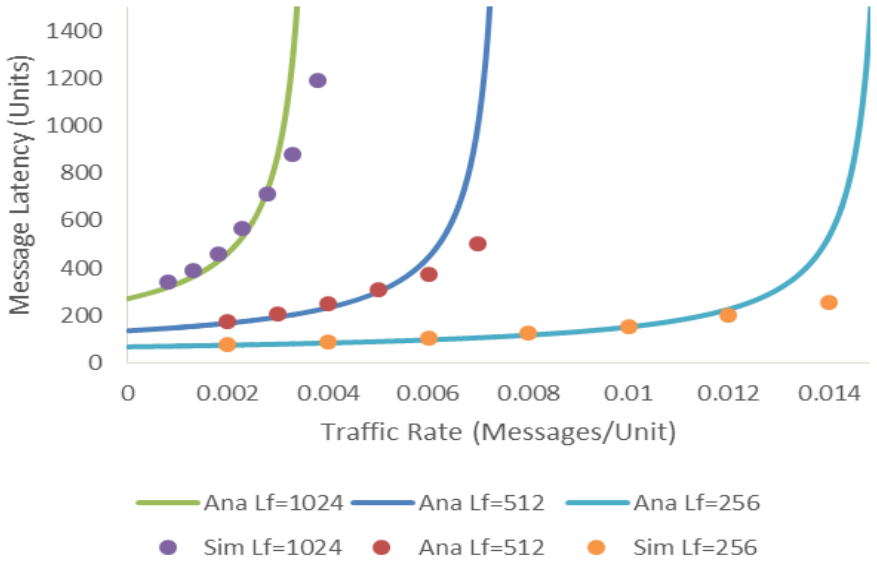

Figure 4-4: Latency predicted by the model and simulation: Lm=128 ∂_{s1}=0.04, ∂_{s2}=0.09,

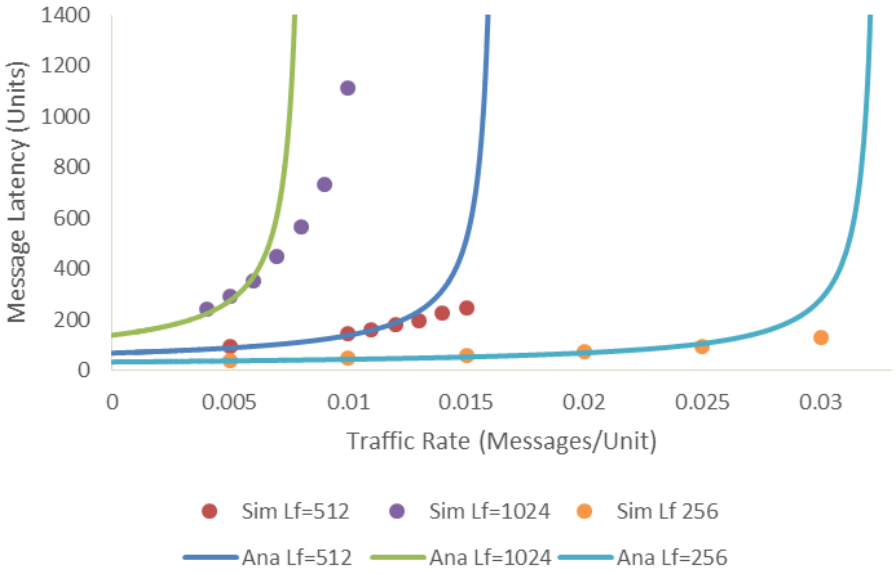

Figure 4-5: Latency predicted by the model and simulation: Lm=64 ∂_{s1}=0.05, ∂_{s2}=0.09,

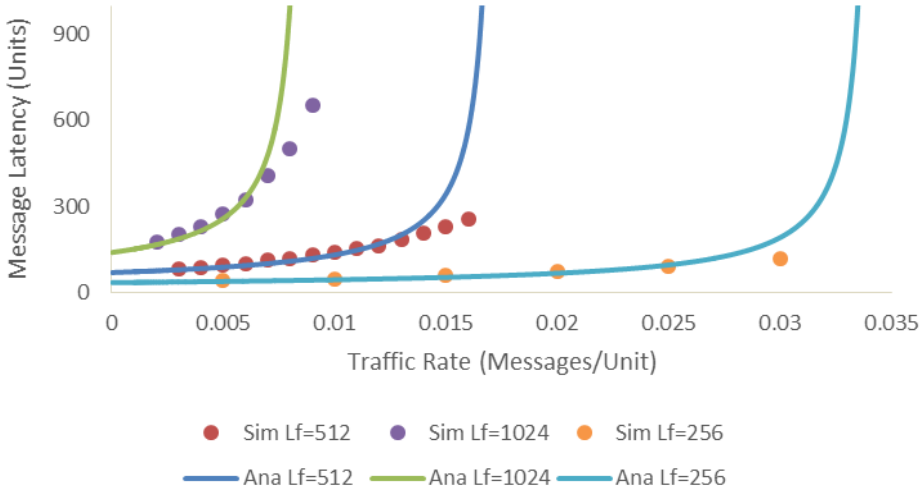

Figure 4-6: Latency predicted by the model and simulation: Lm=64 ∂_{s1}=0.04, ∂_{s2}=0.06,

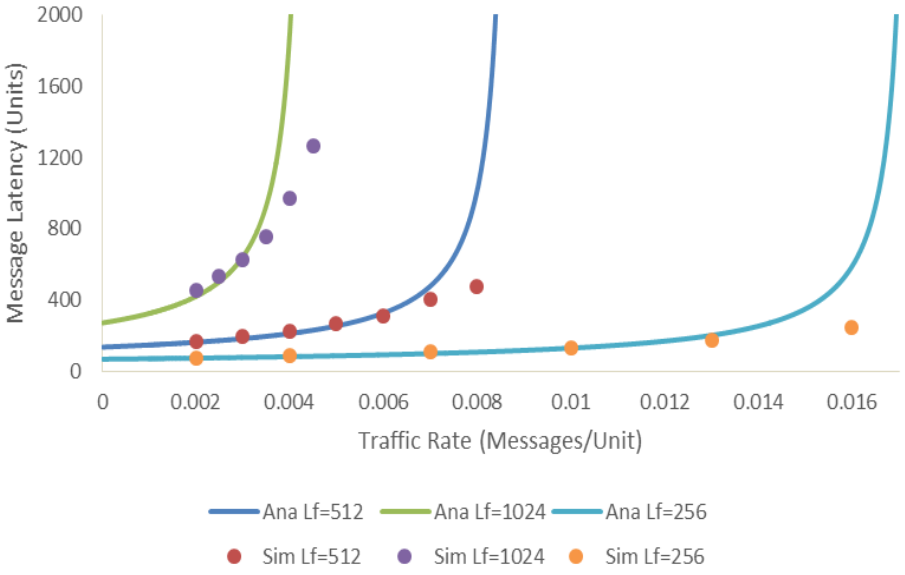

Figure 4-7: Latency predicted by the model and simulation: Lm=128 ∂_{s1}=0.06, ∂_{s2}=0.09,

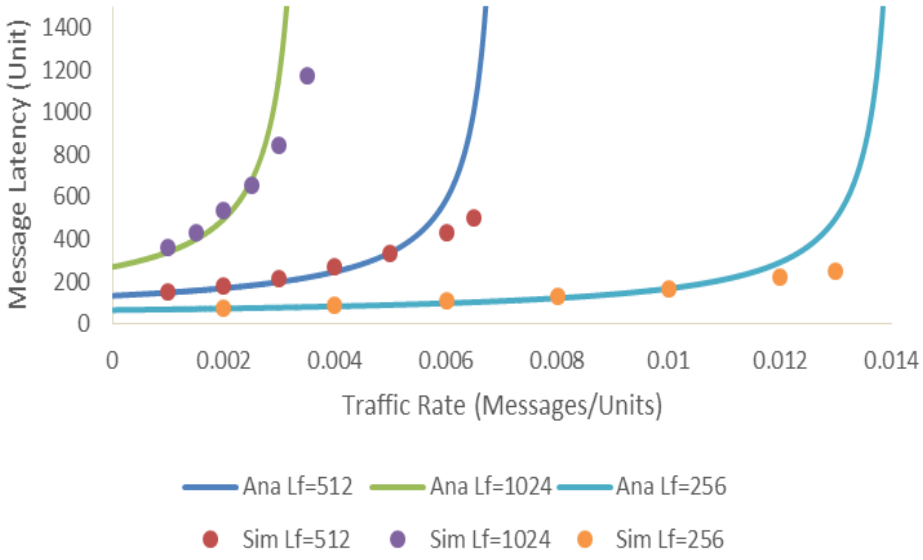

Figure 4-8: Latency predicted by the model and simulation: Lm=128 ∂_{s1}=0.03, ∂_{s2}=0.09,

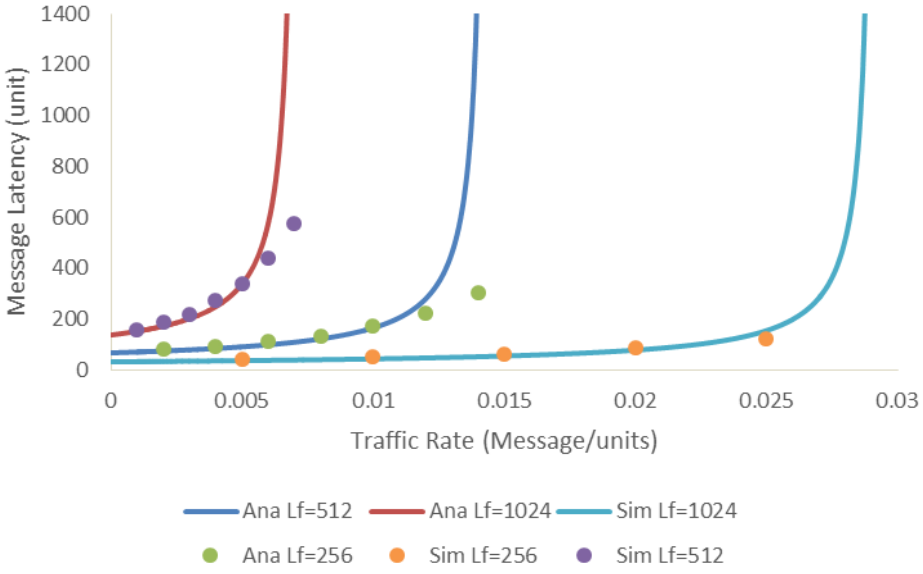

Figure 4-9: Latency predicted by the model and simulation: Lm=64 ∂_{s1}=0.03, ∂_{s2}=0.09,

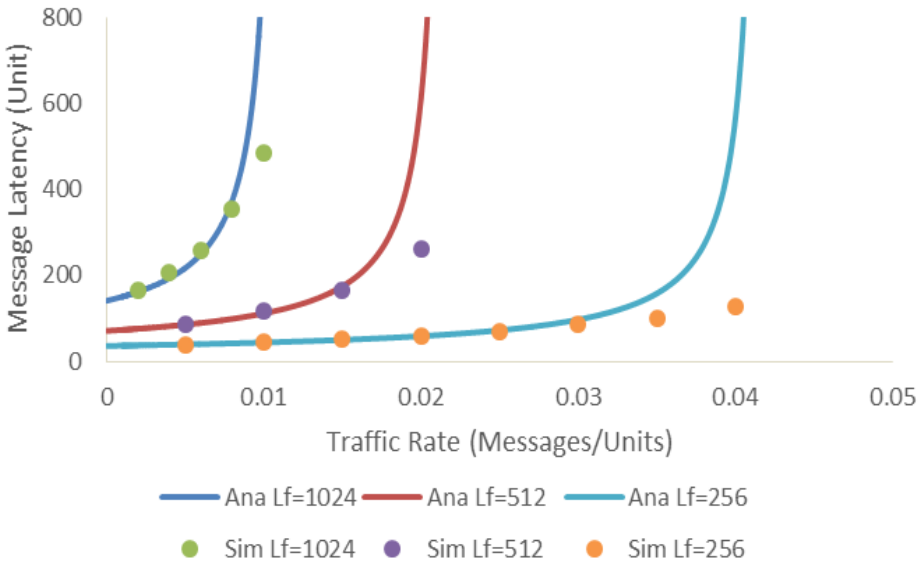

Figure 4-10: Latency predicted by the model and simulation: Lm=64 ∂_{s1}=0.1, ∂_{s2}=0.09,

The horizontal and the vertical axis in those figures represent the traffic rate λ_s at which the node message is injected into the network when the $MMPP$ is in state 1, and the mean message latency obtained from the above model. To make figures clear we have purposely set the arrival rate λ_{s2} at state 2 equal to zero, because we need to use three dimensional graphs to demonstrate the results. As can be seen, results represented in those figures show that the obtained message latency for the analytical model closely match those obtained from the simulation. It can be concluded that the results produced by the proposed model are accurate in the steady state. Furthermore, the result from the analytical model are shown as a curve for all possible values of traffic generation rates until reach the saturated point. When the utilization of the system becomes one or greater

the network enters the saturation region. Additionally, the latency increases as the traffic rate increases. As the network approaches the saturation point, some divergence appears from the simulation result with respect to the analytical result, however this is not a region of concern when performing network performance test. Rather the steady state regions are the determinants of the success of the performance[86].

After validation of the accuracy of the analytical model, we need to use this to investigate the effects of the bursty traffic with different degrees of traffic burstiness and correlations imposed by $MMPP$ input parameters on the network on chip. We can find that the bursty traffic reduces the network performance significantly, since latency increases, especially under moderate and heavy traffic loads. Moreover, the maximum throughput that the network is able to support decreases when subject to the bursty traffic. To clarify the observations from the results developing and using the realistic model is important for the study and optimisation of network on chip, and to show that using the bursty traffic can lead to the network performance suffering significantly from degradation. Detailed discussion on error analysis is represented in section below.

4.7 **Error Evaluation**

Divergence is observed between the analytical and simulation results due to network configurations and numerous parameters that had to be tweaked. Thus, performance of the simulation result can be evaluated by computation of absolute errors (AEs) as well as the relative errors (REs),

where AE is a measure of error magnitude for each simulation from its corresponding analytical value, and the relative error measures the percentage of the total error given by:

$$RE = \frac{AE}{Analytical\ value} \tag{4.15}$$

Where $$AE = |Simulation\ value - Analytical\ value| \tag{4.16}$$

Furthermore the mean RE (MRE) can be computed for each graph using

$$MRE = \sum \frac{RE}{N} \tag{4.17}$$

Where N is the number of samples.

Having computed the RE, as well as MRE for each experiment. The worst case and the best case scenarios for each massage lengths 64,128 bits are represents on Tables 4-2, 4-3, and 4-4 below:

Table 4-2: Summary of Errors for 64bit

Traffic Rate	Simulation Result	Analytical Result	AE	RE	RE %
0.0005	300.412	296.783	3.629	0.012228	1.22
0.001	345.265	331.426	13.839	0.041756	4.17
0.0015	397.699	379.244	18.455	0.048663	4.86
0.002	495.444	449.893	45.551	0.101249	10.12
0.0025	610.952	565.83	45.122	0.079745	7.97
0.003	703.234	794.614	91.38	0.114999	11.49
0.0035	920.369	1479.948	559.579	0.378107	37.81
Mean Relative Error					11.09

Table 4-3: Summary of Errors for worst case of 128bit

Traffic Rate	Simulation Result	Analytical Result	AE	RE	RE %
0.001	71.4654	70.158	1.3074	0.018635	1.86
0.003	80.654	77.894	2.76	0.035433	3.54
0.006	103.357	95.089	8.268	0.08695	8.7
0.009	136.279	126.65	9.629	0.076028	7.6
0.012	188.236	205.866	17.63	0.085638	8.56
0.014	245.427	402.707	157.28	0.390557	39.06
		Mean Relative Error			11.55

Table 4-4: Summary of Errors for best case of 128bit

Traffic Rate	Simulation Result	Analytical Result	AE	RE	RE %
0.002	177.037	165.643	11.394	0.068786	6.87
0.0025	192.602	176.696	15.906	0.090019	9.00
0.003	199.84	189.875	9.965	0.052482	5.24
0.0035	223.175	205.881	17.294	0.084	8.39
0.004	245.356	225.762	19.594	0.086791	8.67
0.0045	269.651	251.117	18.534	0.073806	7.38
0.005	301.755	284.895	16.86	0.05918	5.91
0.0055	333.041	331.944	1.097	0.003305	0.33
0.006	367.924	402.495	34.571	0.085892	8.58
0.0065	413.242	520.796	107.554	0.206518	20.65
		Mean Relative Error			8.10

Observing the results of Tables 4-2, 4-3, and 4-4, it is obvious that all the sample points which lie on the steady state region have RE less than 10%; which is an acceptable relative error. Additionally, considering the MREs of 11.09%, 11.55%, and 8.10% reveal that the effect of divergence due to points outside the steady state region (i.e REs 37.81%, 39.06%, and 20.65% for the three tables above respectively) do not undermine the performance of the system. Moreover, regions outside the steady state are not considered when evaluating network performance [86].

As presented above the errors are within the acceptable range, however these can be further improve by a careful selection of the network parameters with a view to achieving optimum results. It is worth noting that this will be an interesting topic for further research.

4.8 **Conclusion**

An analytical model was developed and implemented to evaluate the performance of network on-chip under bursty traffic. The bursty traffic is modelled by the well-known MMPP. The topology constructed in network on-chip architecture is the popular fat-tree m-port n-tree. For this work extensive simulation experiments have been conducted to validate the accuracy of the model. The accuracy and tractability of the model make it a practical and cost-effective tool to gain insight into the performance of network on chip in the presence of realistic network traffic. The analytical results have shown that the network performance degrades considerably under such traffic patterns.

Chapter 5.

PERFORMANCE MODELLING AND ANALYSIS OF NETWORK ON CHIP UNDER M-PORT N-TREE BURSTY TRAFFIC WITH VIRTUAL CHANNELS

5.1 Introduction

The principle of NoC and the interconnection networks are similar for parallel computers with multiple processors, NoC has some features that differentiate it from parallel computers. These features include, design specialization, energy constraints and a degree of heterogeneity. Devices that utilize NoC applications are operated using batteries; therefore energy saving and customization have major importance in the field. Networks for parallel computers are expected to be deployed for a wide range of unknown applications, while NoCs may be designed for a specific application.

Although there are a number of methods and techniques widely used to employ or enhance the performance of a topology where high performance demand is required, NoC prevents adopting a fixed architecture for a wide range of applications. However, virtual channels can be used. Virtual channels are traditional approaches to improve the performance of the direct interconnecting networks without increasing the node degree. The analytical model derived from numerous researches [14, 40, 87] detected that a reasonable number of virtual channels can significantly improve the network performance. Improve message latency can be done by adding

virtual channels, via allowing messages to share a physical channel when blocking occur [10].

The analytical model has been proposed in this chapter to investigate the impact of the virtual channel under bursty Poisson process for NoC. The model has been validated by comparing the prediction of performance from the model with results obtained from the simulation

5.2 Assumptions and notations

The model uses some assumptions that are widely used in the literature [44, 51, 77-80]:

1. Nodes generate correlated traffic, which follows an independent MMP[81]P whose infinitesimal generator Ω_{s1} , and Δ_s in the rate matrix are given in the previous section.
2. Message length (Lm flits).
3. Messages generated by the source nodes are sent with probability β.
4. The network switches are input buffered and each channel is associated with a single buffer.
5. V, ($V \geq 2$), virtual channels share each physical channel.
6. Deterministic routing is employed; shortest path routing algorithm used.

Table 5-1: Key notations used in the derivation of the model

Ω_s, Δ_s	Parameter matrices of MMPPs to model the traffic generated by the source node

S	Mean Message Latency
D	Mean Network Latency
Lm	Message length in flits
R	Average time for tail flit to reach its destination
$P_{i,n}$	The probability of a message traversing $2i$ links
d	mean message distance
$D_{\mathfrak{U},i}$	The service time experienced by 2i link at $S-1$ stage
$D_{\mathfrak{U}-1,i}$	The service time at internal stages
$W_{I,i}$	Blocking time that message acquires a channel at stage s
$P1_{s,i}$	Probability that the message is blocked at this stage
$Wp_{I,i}$	Waiting time by the message to acquire a channel
m	Number of port
n	Number of tree
W	The waiting time at the source node
N	Number of node
Nsw	Number of switch
α_{net}	Network latency
β_{net}	Transmission time
V	Mean degrees of virtual channel
i	Number of link

5.3 **Traffic Pattern**

The traffic pattern mainly affects the mean message distance, the average number of links that message needs to reach its destination. The probability of a message traversing $2i$ links (one i link in ascending phase

and another i link in descending phase) to reach its destination is P_i. Different values of P_i could produce different distributions of message destinations and, thus, different mean message distances. In [5] the number of nodes at distance $2n$ is $(m/2)^{n-1}(m-1)$ in *m*-port *n*-tree topology, thus $P_i$is given by:

$$P_{i,\mathrm{n}} = \begin{cases} \dfrac{\left(\frac{m}{2}-1\right)\left(\frac{m}{2}\right)^{i-1}}{N-1} & for\ i = 1,2\ldots, n-1 \\ \dfrac{(m-1)\left(\frac{m}{2}\right)^{i-1}}{N-1} & for\ i = \mathrm{n} \end{cases} \tag{18}$$

In addition, messages generated by each source node to be sent with the mean message distance, can be given by $d = \sum_{i=1}^{n} 2jP_i$. The traffic arriving at the network channel is t_s, the number of nodes is *N* and the messages generated by these source nodes are sent to $4nN$ channels in the network, therefore t_{sa} times is :

$$t_s = {}^{d}/_{4n} \tag{19}$$

Meanwhile the superposition and splitting of multiple $MMPP_s$ leads to getting new $MMPP$ [44] let denote $MMPP_{s1}$. let denote $MMPP_{s1}$ We need to derive the new matrix parameters of the new $MMPP_{s1}$, in general d is not an integer value and is dependent on traffic loads and the network size. Let's divide the value of d into two parts: integer F_i and fraction. The result of the traffic from splitting of $MMPP_s$ leads to new $MMPP_f$, on the basis of the

principle of splitting $MMPP$ [88]. The new infinitesimal generator and rate matrix Ω_{s1} and Δ_{s1}respectively are:

$$\Omega_{s1} = \ \Omega_{s1} = \begin{bmatrix} -\vartheta_{s1} & \vartheta_{s1} \\ \vartheta_{s2} & -\vartheta_{s2} \end{bmatrix} \text{ and } \Delta_{s1} = F\,\Delta_s = \begin{bmatrix} F\lambda_{s1} & 0 \\ 0 & F\lambda_{s2} \end{bmatrix} \qquad (20)$$

The new parameters of $MMPP_{s1}$ are selected to match the statistical characteristics [50], which are mean arrival rate, variance of the arrival rate, third central moment of the arrival rate and integral of the covariance function of the arrival rate.

5.4 The Analytical Model

The mean message latency **S** consists of the mean network latency **D**, (that is the mean time to cross the network, the mean waiting time seen by messages at the source node **W**, and average time for tail flit to reach its destination **R)**. In addition to that the mean message latency has to be scaled by factor **V̄** which determines the effect of the virtual channel multiplexing. Thus, we can write the mean message latency as:

$$\mathrm{S} = (\mathrm{W} + \ \mathrm{D} + \ \mathrm{R})\ \bar{\mathrm{V}} \qquad (21)$$

The derivation of these will be in the following sections.

5.6.1. Mean Network latency

For illustration purposes, the switches between the source and destination are numbered, and these switches are labeled as stages. Numbering begins with the stage that is located next to the source node and is labeled stage 0 and increments as it reaches towards the destination node. The numbering for the stages can be calculated using $\mathfrak{U} - 1$, where $\mathfrak{U}$ is the stage. In order to calculate the stages that will be crossed by a $2i$links in an m-port n-tree, the formula $\mathfrak{U} = 2n - 1$ is used.

A message may use a variety of numbers of channels to reach its destination from its source, taking into consideration that it has a transmission delay of$2i$ link. When messages arrive at their destination, they are transferred to their local nodes. To calculate the service time that has occurred, a backwards direction calculation is employed; that is calculating from the final stage to the first stage. To obtain the service time for a $2j$links at the stage $\mathfrak{U} - 1$the following is used:

$$D_{\mathfrak{U}-1,i} = L_{\mathrm{m}} t_{\mathrm{cn}} \tag{5.5}$$

$L_m$is the length of the message calculated in flits; each flit consists of $L_f$bytes. t_{cn} is the transmission time for a single flit on either a node-to-switch or switch-to-node connection. $t_{\mathrm{cn}(I)}$ can be calculated using:

$$t_{\mathrm{cn}} = 0.5\varrho_{\mathrm{net}} + L_f\varpi_{\mathrm{net}} \tag{22}$$

where $\varpi_{\mathrm{net}}$and $\varrho_{\mathrm{net}}$are the transmission time and the network latency is of one byte.

The service time can be determined by actual message transmission time and delay due to message blocking at the stages $\mathfrak{U}(0 \leq \mathfrak{U} \leq \mathfrak{U}-2)$, therefore the service time can be expressed by:

$$D_{\mathfrak{U},i} = \sum_{l=\mathfrak{U}+1}^{\mathfrak{U}-1} Wb_{l,i} + L_{\mathrm{m}}t_{\mathrm{cs}} \quad 0 \leq \mathfrak{U} \geq \mathfrak{U}-2 \tag{23}$$

where $Wb_{l,i}$ and t_{cs} are the blocking time that the message acquires a link at stage $\mathfrak{U}$, and the time for a flit to transmit on switch to switch connection respectively. $Wb_{\mathfrak{U},i}$can be calculated by the probability that V virtual channels are busy, and the waiting time experienced by the message to acquire a link when blocking occurs $Pb_{\mathfrak{U},i}$ and $Wc_{\mathfrak{U},i}$, respectively.

$$Wb_{\mathfrak{U},i} = Wc_{\mathfrak{U},i}Pb_{\mathfrak{U},i} \tag{24}$$

To determine $Pb_{\mathfrak{U},i}$, we need to compute the joint probability $P_{\omega,\tau}(0 \leq \omega \leq \mathrm{V})$ where $(\tau = 1,2)$ that virtual channels are busy and the underlying MMPP_w . To model the arrival process at the network channel is in state can be calculated using a bivariate Markov chain. Let us consider $\mathfrak{Q}_{\omega,\tau}$ in the case where the channel is idle $(\omega = 0)$ or busy $(\omega = 1)$ and the

MMPP_w is in state ω as mentioned above. The figure below shows that the transition rate out of state $\mathrm{Q}_{\omega,\tau}$ to $\mathrm{Q}_{\omega+1,\tau}$ is λ_1 where it is a traffic rate on the network channels, while MMPP_w is in state $\boldsymbol{\tau}$. However the rate from $\mathrm{Q}_{\omega+1,\tau}$ to $\mathrm{Q}_{\omega,\tau}$ is $1/D_{\mathrm{U},i} - \lambda_1$, the decrease of λ_1 to account for messages arrival while in this state [12].

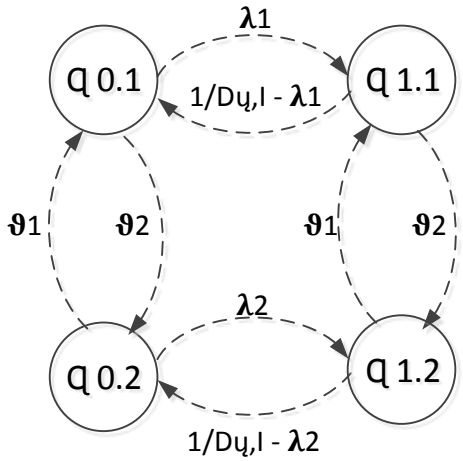

Figure 5-1: Transition Diagram to Calculate virtual channel Occupancy Probabilities

However the transition rates out of state $\mathrm{Q}_{\omega,1}$ to $\mathrm{Q}_{\omega,2}$ is ϑ_1, while the rate from $\mathrm{Q}_{\omega,2}$ to $\mathrm{Q}_{\omega 2}$ is ϑ_2, in the steady state the model yields the following equations:

$$\begin{cases} (\lambda_{\,1} + \vartheta_2)\mathrm{P}_{0,1} = \vartheta_{\,1}\,\mathrm{P}_{0,2} + 1/D_{\mathrm{U},i} - \lambda_1\mathrm{P}_{1,1} \\ (\lambda_{\,2} + \vartheta_{\,1})\mathrm{P}_{0,2} = \vartheta_{\,2}\,\mathrm{P}_{0,1} + 1/D_{\mathrm{U},i} - \lambda_2\mathrm{P}_{1,2} \\ \left(1/D_{\mathrm{U},i} - \lambda_1 + \vartheta_{\,2}\right)\mathrm{P}_{1,1} = \lambda_{\,1}\mathrm{P}_{0,1} + \vartheta_{\,1}\mathrm{P}_{1,2} \\ \left(1/D_{\mathrm{U},i} - \lambda_2 + \vartheta_{\,1}\right)\mathrm{P}_{1,2} = \lambda_{\,2}\mathrm{P}_{0,2} + \vartheta_{\,2}\mathrm{P}_{1,1} \\ \sum_{\omega=0}^{1}\sum_{\tau=1}^{2}\mathrm{P}_{1,\tau} = 1 \end{cases} \tag{25}$$

Solving the above equations to calculate the probability $P_{\omega,\tau}$. in virtual channel flow control, several virtual channels share the bandwidth of a physical channel in a time-multiplexed manner.

After calculating the $Pb_{\mathfrak{U},i}$ we need to calculate the waiting time experienced by the message to acquire a link when blocking occurs $Wc_{\mathfrak{U},i}$.

Our network expressed is as a network of queues, where each channel in the network is modelled as MMPP/G/1 [44], and the arrival process is modelled by MMPP_w and the service time is $D_{\mathfrak{U},i}$, thus the $Wc_{\mathfrak{U},i}$ will be [44]:

$$Wc_{1\mathfrak{U},i}=$$

$$\frac{1}{2(1-\rho)}[2\rho+\lambda_{tot}h_2-2h_1((1-\rho)g+h_1\Pi\Delta_{s1})(\Omega_{s1}+e\Pi)^{-1}\lambda_{s1} \tag{26}$$

$$Wc_{\mathfrak{U},i}=\frac{1}{\rho}\left(Wc_{1\mathfrak{U},i}-\frac{1}{2}\lambda_{tot}h_2\right) \tag{27}$$

In the above equations $h1$ and $h2$ denote the first and second moments of the service time seen by the message respectively, and can be calculated by differentiating $L_t(s)$ Laplace-Stieltjes Transform where

$$L_t(s)=\begin{cases} e^{-sMt_{cn}} & k=K-1 \\ Wb_{\mathfrak{U},i}(s)e^{-sMt_{cn}} & 0\le k\le K \end{cases} \tag{28}$$

where the $Wb_{\mathfrak{U},i}(s)$ is the Laplace Transform for the blocking time $Wb_{\mathfrak{U},i}$.

The $Wb_{\mathfrak{U},i}(s) = \frac{\varrho}{\varrho+s}$, where the ϱ is selected to match the mean blocking time experienced by the message at network channels:

$$\varrho = \frac{1}{\sum_{\mathfrak{U}=k+1}^{K+1} Wb_{\mathfrak{U},i}} \tag{29}$$

Also ρ is the traffic intensity, which can be calculated by $\rho = h_1\, \lambda_{tot}$ where λ_{tot} is the mean arrival rate at network channels. it is equal to $\lambda_{tot} = \Pi\, \lambda_{s1}$ where π is the steady-state vector of MMPP_c, and can be written as $\Pi = [\Pi_1, \Pi_2] = \frac{1}{\vartheta_2+\vartheta_2}\,[\vartheta_2, \vartheta_1]$, $\lambda_{s1} = \Delta_{s1} e$, where e is the column unit vector of length 2 $e = \begin{bmatrix}1\\1\end{bmatrix}$, and the algorithm to compute g can be found in [44].

The average of all possible destinations of a message in the network, is the network latency

$$\mathrm{L} = \sum_{i=1}^{n} P_{i,\mathrm{n}}\; D_i \tag{30}$$

where D_i is equal to the service time of the message at stage $\mathfrak{U} = 0$ ($D_i = D_{0,i}$)

5.6.2. Mean waiting time at the source node

A source node generated message enters the network over one of the V injection with equal probability $t_s = 1/\mathrm{V}$. In fact the process of splitting

the result of $MMPP$ can be done as mentioned in an earlier section, the traffic arriving at the source node denoted by $MMPP_c$. The infinitesimal generator and the rate matrix are Ω_c, and Δ_c respectively, of the resulting $MMPP_c$ characterising the traffic arriving at an injection virtual channel in the source node are given by [88]:

$$\Omega_c = \Omega_s = \begin{bmatrix} -\vartheta_{1c} & \vartheta_{1c} \\ \vartheta_{2c} & -\vartheta_{2c} \end{bmatrix} \text{ and } \Delta_c = \Delta_s/\mathrm{V} = \begin{bmatrix} \lambda_{1c}/\mathrm{V} & 0 \\ 0 & \lambda_{2c}/\mathrm{V} \end{bmatrix} \quad (31)$$

To compute the mean waiting time W , that a message experiences before entering the network, MMPP/G/1 queueing system has been modeled by the injection virtual channel, where the arrival process modelled by $MMPP_c$ can be obtained in a similar way to that used for the calculation of the waiting time $Wc_{\mathfrak{U},i}$.

$$\mathrm{W}_1 = \frac{1}{2(1-\rho)}[2\rho + \lambda_{tv}h_2 - 2h_1((1-\rho)g + h_1\Pi\Delta_c)(\Omega_c + e\Pi)^{-1}\lambda_c \quad (32)$$

$$\mathrm{W} = \frac{1}{\rho}\left(\mathrm{W}_1 - \frac{1}{2}\lambda_{tv}h_2\right) \quad (33)$$

where $h1$ and $h2$ denote the first and second moments of the service time seen by the message respectively, and can be calculated by differentiating $L_t(s)$ Laplace-Stieltjes Transform as mentioned above. Also ρ is the traffic intensity, which can be calculated by $\rho = h_1\,\lambda_{tv}$ where λ_{tv} is the

mean arrival rate at network channels, it is equal to $\lambda_{tv} = \Pi\, \lambda_c$ where π is the steady-state vector of MMPP_c, can be written as $\Pi = [\Pi_1, \Pi_2] = \frac{1}{\vartheta_{1c} + \vartheta_{2c}} [\vartheta_{2c}, \vartheta_{1c}]$, $\lambda_c = 1/\mathrm{V} \begin{bmatrix} \lambda_{1c} \\ \lambda_{2c} \end{bmatrix}$, and e is the column unit vector of length 2 $e = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and the algorithm to compute g can be found in [44].

The mean time for the tail flit to reach the destination R in the network can be found as:

$$R = \sum_{i=1}^{n} [\, P_{i,\mathrm{n}} \, (\sum_{k=1}^{K-1} t_{cs} + t_{cn}) \,] \tag{34}$$

5.5 Validation and Analysis

A discrete-event simulator operating at the flit level to validate the above analytical model was developed in C++ using OMNet++ simulation environment [63]. The aspect of message latency is the amount of time since the message generated at the source node until the flit reaches the destination. Each simulation experiment was run until the network reaches its steady state. The arrival of generated messages at each source node according to an independent $MMPP$ with infinitesimal generatorΩ_s and matrix of rate $\boldsymbol{\Delta_s}$. The message destination nodes are determined using a uniformly distributed. Many experiments have been performed for several combinations of network sizes, message lengths, numbers of virtual

channels and different MMPP input traffic. However, for the sake of specific illustration, latency results are presented for the following cases only.

Simulation experiments have been extensively applied to validate the model for several message lengths, different $MMPP$ traffic inputs, and number of virtual channels. Yet, for the sake of specific illustration, latency results are presented for the following scenarios: for network topology: 4-port 3-tree the system parameters are set at the following:

1. Message length Lm=32, 48, and 64 flit.
2. Flits length: 128, 256, 512 bytes.
3. Network and switch latency are 0.005 and 0.01 time unit respectively.
4. Virtual channel V=2,3,5,6.
5. Q_s is the infinitesimal generator of $MMPP$ are set as follows:

$$Q_s = \begin{bmatrix} -0.001 & 0.001 \\ 0.001 & -0.001 \end{bmatrix}, Q_s = \begin{bmatrix} -0.07 & 0.07 \\ 0.05 & -0.05 \end{bmatrix}$$

$$Q_s = \begin{bmatrix} -0.6 & 0.6 \\ 0.6 & -0.6 \end{bmatrix}, Q_s = \begin{bmatrix} -0.009 & 0.009 \\ 0.009 & -0.009 \end{bmatrix}$$

$$Q_s = \begin{bmatrix} -0.008 & 0.008 \\ 0.008 & -0.008 \end{bmatrix}, Q_s = \begin{bmatrix} -0.002 & 0.002 \\ 0.002 & -0.002 \end{bmatrix}$$

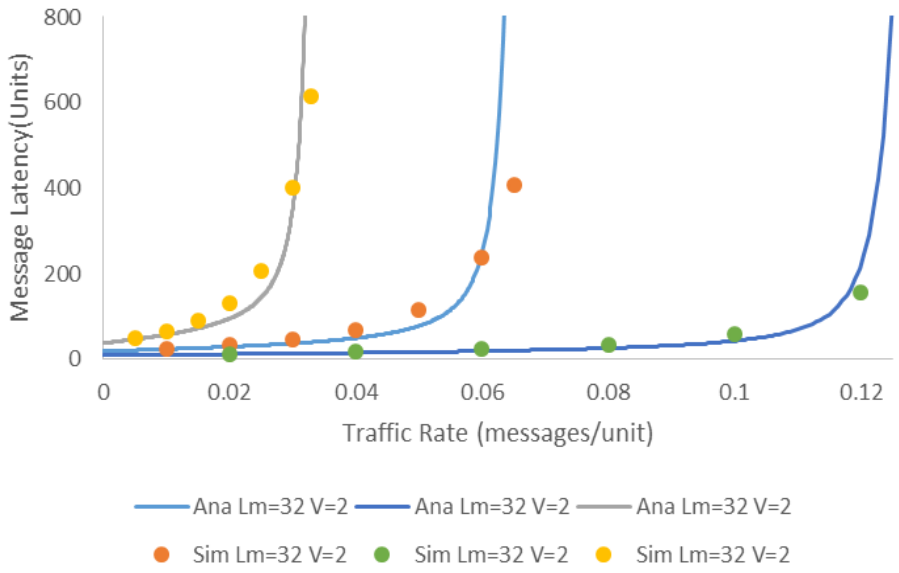

Figure 5-2: Latency predicted by the model and simulation: Lm=32 Fl=128,256,512, V=2 φ_{s1}=0.001, φ_{s2}=0.001,

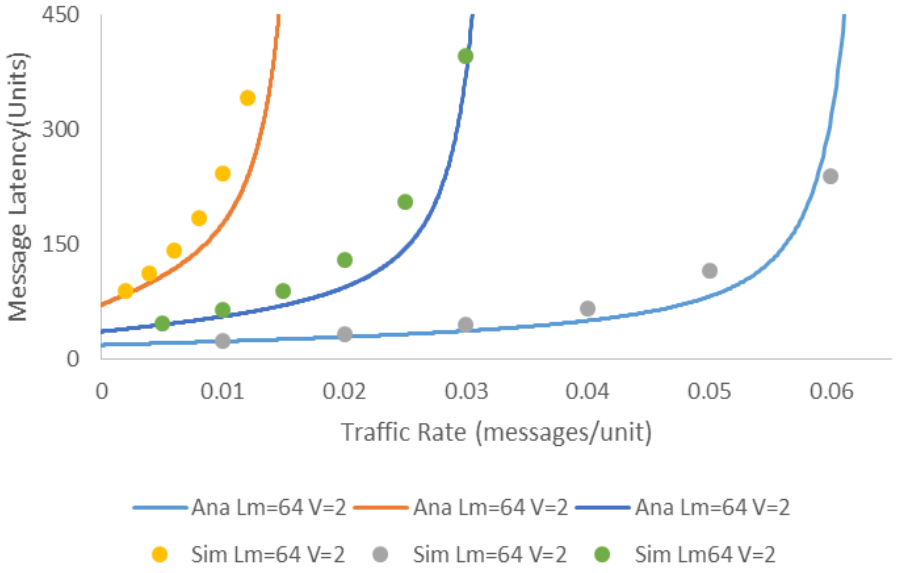

Figure 5-3: Latency predicted by the model and simulation: Lm=64 Fl=128,256,512, V=2 φ_{s1}=0.001, φ_{s2}=0.001,

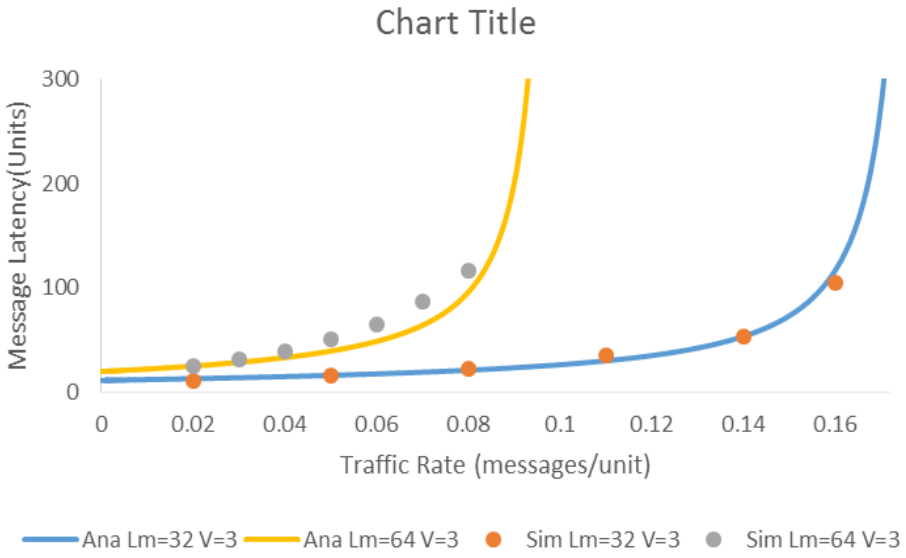

Figure 5-4: Latency predicted by the model and simulation: Lm=32,64 Fl=128 , V=3 ϱ_{s1}=0.07, ϱ_{s2}=0.05,

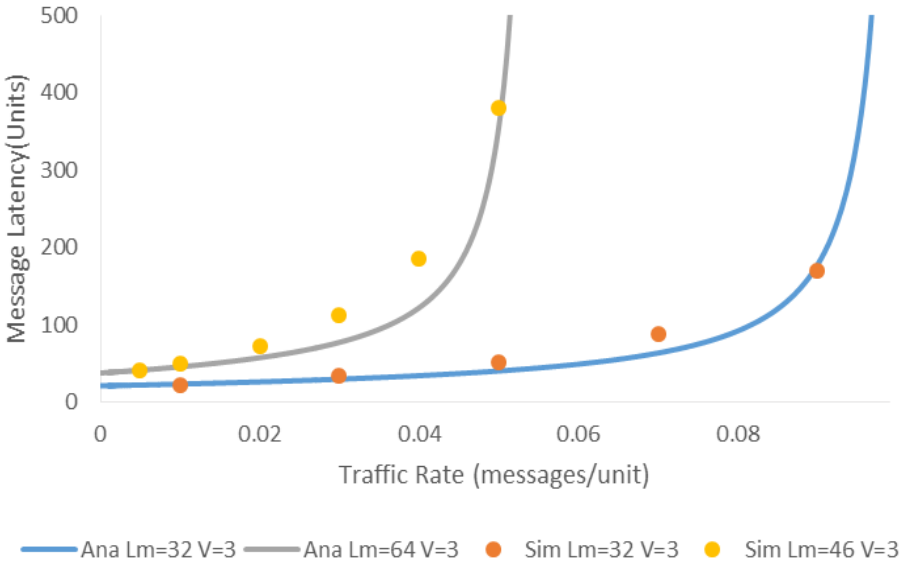

Figure 5-5: Latency predicted by the model and simulation: Lm=32,64 Fl=256 ,V=3 ϱ_{s1}=0.07, ϱ_{s2}=0.05,

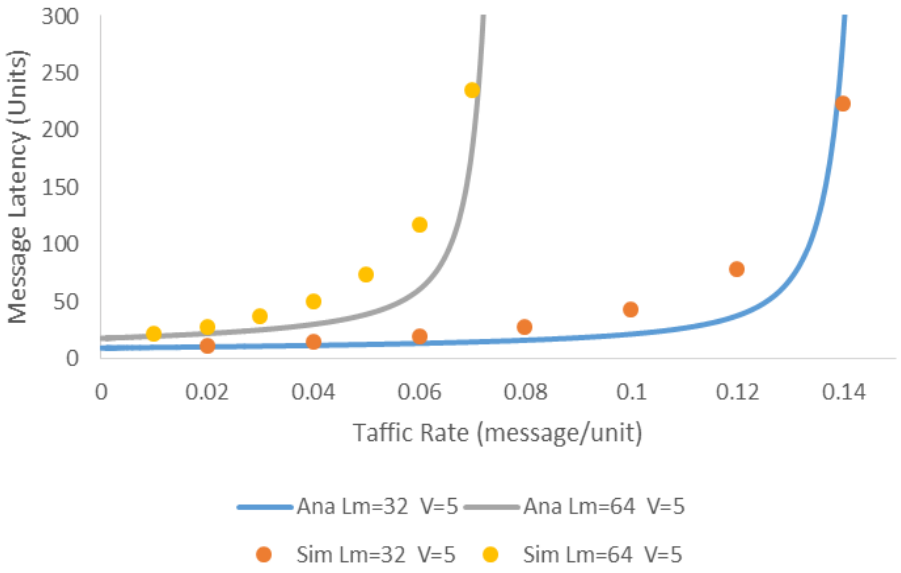

Figure 5-6: Latency predicted by the model and simulation: Lm=32,64 Fl=256,V=5 ϱ_{s1}=0.002, ϱ_{s1}=0.002,

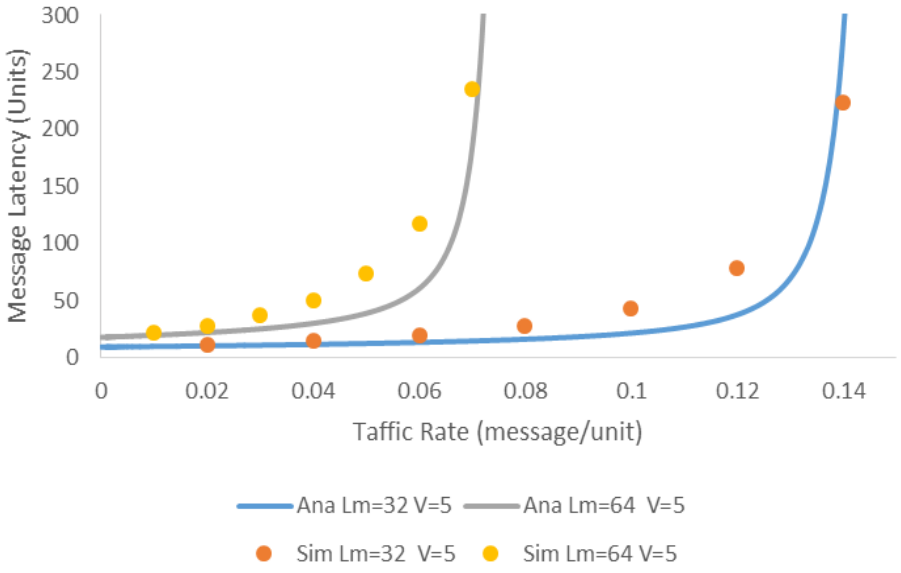

Figure 5-7: Latency predicted by the model and simulation: Lm=32,64 Fl=128,V=5 ϱ_{s1}=0.008, ϱ_{s2}=0.008,

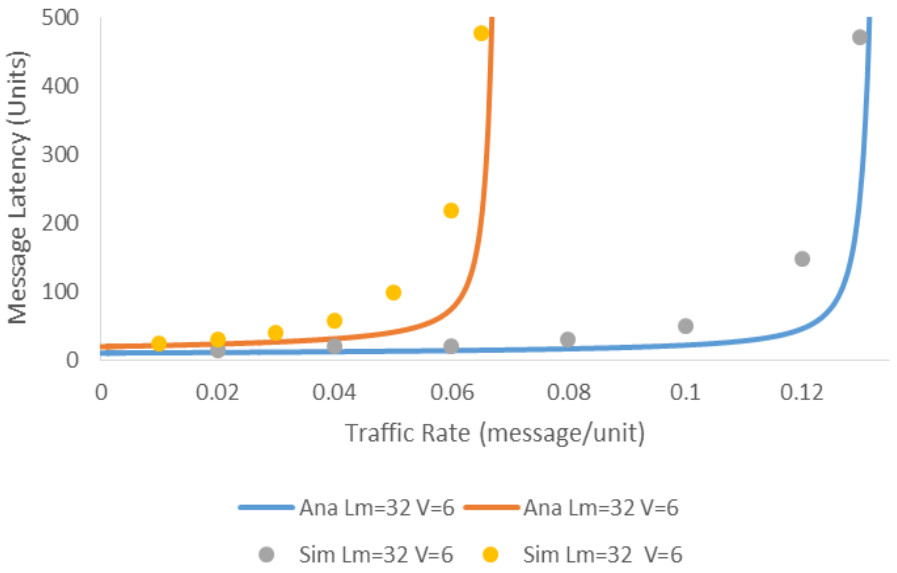

Figure 5-8: Latency predicted by the model and simulation: Lm=32 Fl=128,256,V=6 ϱ_{s1}=0.001, ϱ_{s2}=0.001,

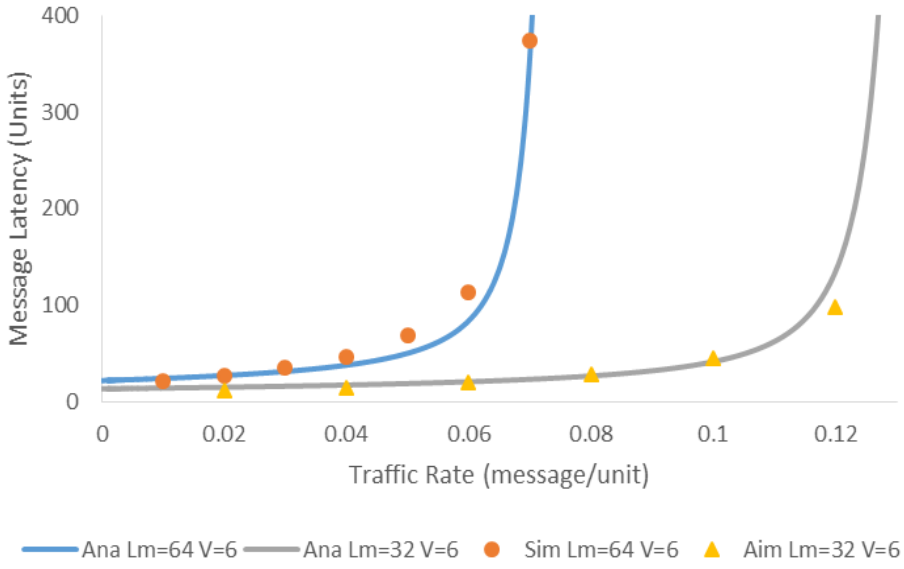

Figure 5-9: Latency predicted by the model and simulation: Lm=64 Fl=128,256,V=6 ϱ_{s1}=0.009, ϱ_{s1}=0.009,

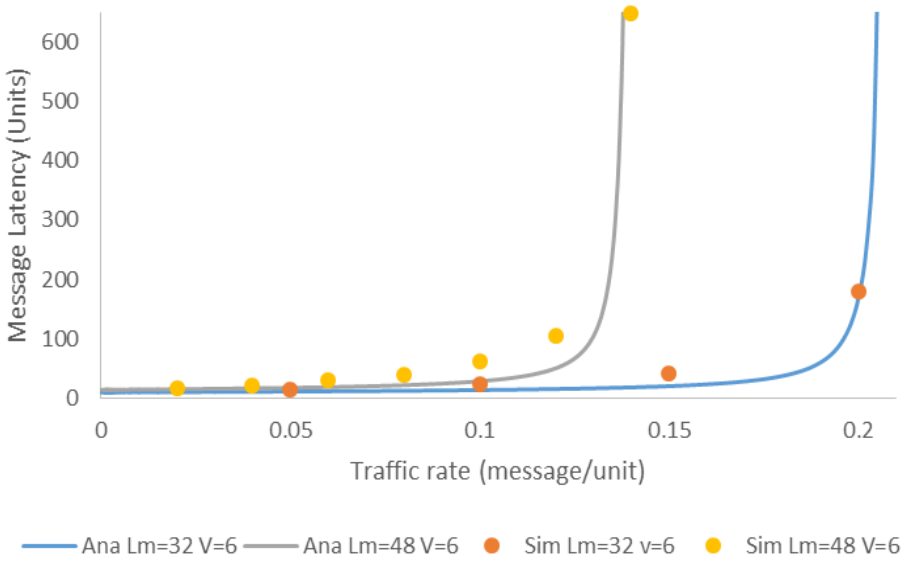

Figure 5-10: Latency predicted by the model and simulation: Lm=32,48 Fl=128,V=6 ϱ_{s1}=0.6, ϱ_{s2}=0.6,

In the figures above the horizontal and the vertical axis show the message latency and the traffic rate respectively. As can be seen from the figures in the analytical model, it presents a good approximation of the network latency with different network configurations. The figures show that the model predicts accurately the network saturation points. The exhaustive experiments showed that implementing virtual channels can increase the saturation point. In addition, the overall configuration experiments revealed that increasing the virtual channels to 3,5,6 will improve the average of saturation points.

5.6 The impact of virtual channels under bursty traffic

The performance effect of virtual channels on the network was investigated using the analytical model, after subjecting the network to bursty traffic. This investigation has uncovered the significance of using virtual channel flow control in trying to reduce the degrading effects of bursty traffic on network performance. It became evidently clear that when the network has a moderate number of virtual channels (e.g., $2 \leq V \leq 6$), the performance improves with increase in the number of virtual channels per physical channel. It was also observed that adding more virtual channels does not give considerable performance gains, when the number of virtual channels approaches a threshold (e.g., $V = 12$). This is as a result of the network approaching the actual limit imposed by the bandwidth of its physical channels. The figure below show the impact of adding virtual channels:

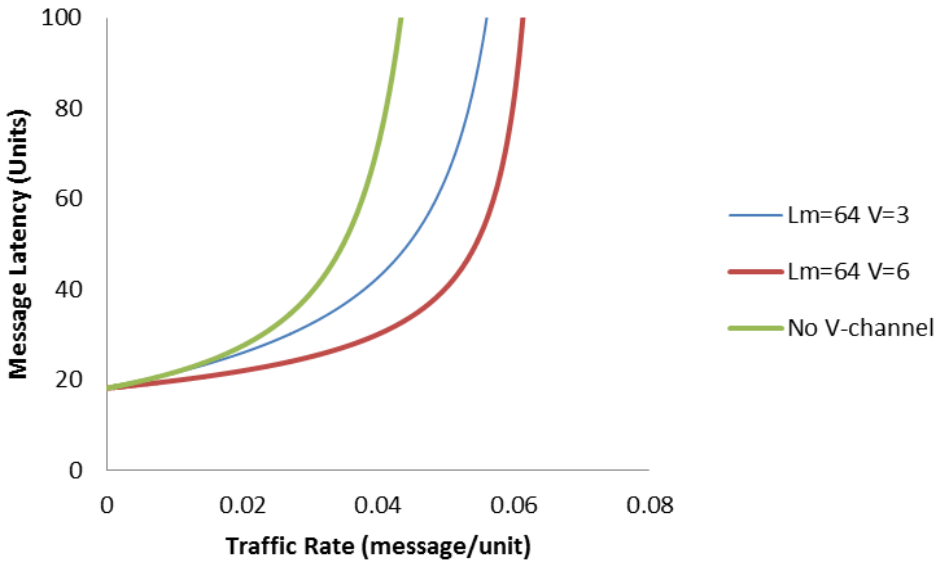

Figure 5-11: Comparison of message latency under different virtual channels

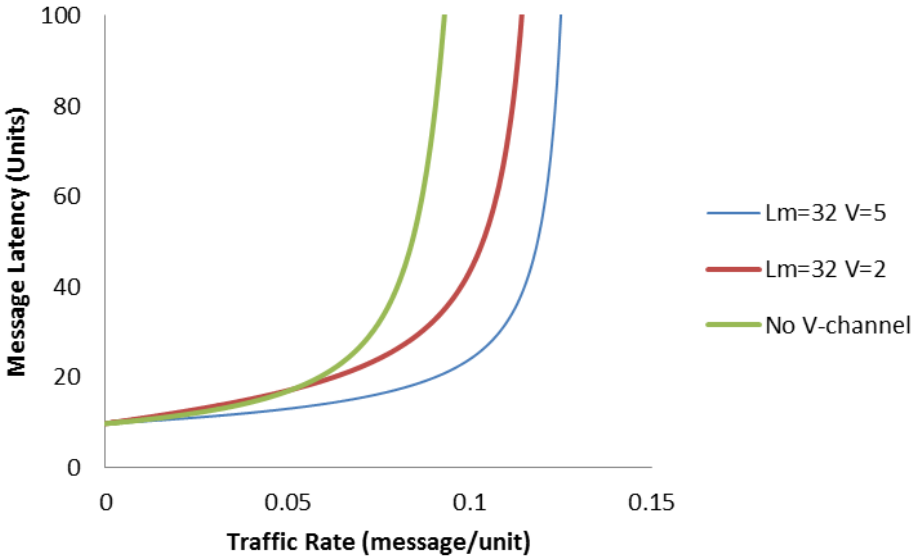

Figure 5-12: Comparison of message latency under different virtual channels

5.7 **Conclusion**

The model presented in this chapter was designed and employed to evaluate the network performance of network on chip in term of message latency under bursty traffic with virtual channels. The fat tree topology was adopted for the architectural design. Extensive simulation results have shown that the analytical model predicts the message latency with a good degree of accuracy in the network with different configurations regarding traffic patterns, number of virtual channels, and message length. The results showed that adopting a virtual channel could improve the performance of network on chip.

Chapter 6.

CONCLUSION AND FUTURE WORK

6.1 Conclusion

This chapter draws a conclusion of the thesis and provides recommendations for future related work in the area of network on chip performance modelling.

As a result of buses often becoming a bottleneck, NoCs have been designed and developed on SoCs with the aim of increasing the number of IP cores per chip. The increase in the number of IP cores has a significant effect on the reduction of message latency.

The thesis has presented new analytical tools for performance analysis and enhancement of NoC's topology with subject to bursty traffic. The accuracy of the proposed analytical models developed has been validated through comparison to results gained from extensive simulations.

As the first step, the thesis has investigated the performance of network on Chips by means of analytical modelling in the presence of bursty traffic. A thorough investigation into the impact of traffic generated through the application of Poisson Process, Markov Modulated Poisson Process and self-similar Process are carried out. The results exhibit higher accuracy in latency and throughput of NoCs under bursty traffic generated using MMPP

and highlight the importance of taking into account the bursty nature of network traffic for accurate evaluation of NoC's performance measures. As a result, under bursty traffic the throughput of the network is lower compared to when traffic is modelled using Poisson Process. With Pes generating realistic bursty traffics the network reaches saturation point quicker. This increases the message latency considerably and rapidly degrades network's performance. These results highly implicate the importance of using realistic traffic models in the study of NoCs.

In chapter 4, an analytical model has been proposed to investigate the performance of network on chip under m-port n-tree in the presence of bursty traffic through the employment of wormhole switching. The developed analytical model accurately predicts the saturation points of the network under different configurations. The results show that the performance measures predicted by the model closely match those obtained from the simulation. Compared to the Mesh topology, message latency decreases significantly when m-port n-tree topology is employed which in turn results in higher network throughput. As the network approaches the saturation point, some divergence appears from the simulation result with respect to the analytical result, however this is not a region of concern when performing network performance test. Rather the steady state regions are the determinants of the success of the performance.

Another analytical model has been developed in chapter 5 to examine the network on chip performance under m-port n-tree topology

using bursty traffic with virtual channels. Extensive simulation experiments have shown that the analytical model predicts the target network performance measures with a good degree of accuracy. Different configurations have been considered in terms of message length, traffic and number of virtual channels. The results show that under the same simulation settings, use of virtual channels in the architecture of NoCs can greatly increase the network throughput while decreasing the latency. The results show consistency under variant message lengths and traffic loads. Use of virtual channels in the architecture of NoC's makes the network more robust and reliable however, the number of virtual channels employed in the architecture of each node within the network can only increase up to a maximum value after which the network shows no response towards increase in efficiency and performance.

Overall the comparison between the analytical results and those obtained from extensive simulation experiments have shown a good degree of accuracy for predicting the network performance under different design alternatives and various traffic conditions. The results have revealed the importance of accurate traffic modelling in performance evaluation of NoCs as well as the importance of balance between the number of virtual channels and message size employed within the architecture of NoCs to maximise the system performance. Based on this fact it is safe to conclude that traffic patterns and virtual channels have a serious impact on the performance of NoCs.

6.2 **Future Work**

The models of this work have been implemented in a way that it can be used improve the NoC performance through investigate the system under different aspects like routing mechanisms, topology scenarios, switch methods and traffic patterns. In addition considered secondary performance metrics *i.e*. energy efficiency.

Furth more, investigate the effect of different topologies with different sizes on network message latency may lead to more research with the aim of enhance the system efficiency mainly in terms of message latency and other metrics like power consumption.

In a later stage we plan to examine the integration of adaptive routing mechanism into NoC along with deterministic routing. We expect that the mechanism of choosing the shortest path adopted by adaptive routing will play significant roll on the performance of the message latency

Hotspot is defined as a hotspot node and all other nodes send a specific portion of their messages to this node. In wormhole based NoCs, hotspot modules or nodes dramatically reduce network efficiency and unfairly allocate the system’s resources. For example, nodes near the hotspot module receive larger portions of their capacity. A single hotspot module within a NoC topology can greatly decrease the performance of the entire system; therefore, some works have been done in order to solve this problem [38].

REFERENCES

1. Dally, W.J. and B. Towles. *Route packets, not wires: On-chip interconnection networks*. in *Design Automation Conference, 2001. Proceedings*. 2001. IEEE.

2. Benini, L. and G. De Micheli, *Networks on chips: a new SoC paradigm.* Computer, 2002. **35**(1): p. 70-78.

3. Grecu, C., et al. *Structured interconnect architecture: a solution for the non-scalability of bus-based SoCs*. in *Proceedings of the 14th ACM Great Lakes symposium on VLSI*. 2004. ACM.

4. Bononi, L. and N. Concer. *Simulation and analysis of network on chip architectures: ring, spidergon and 2D mesh*. in *Proceedings of the conference on Design, automation and test in Europe: Designers' forum*. 2006. European Design and Automation Association.

5. Lin, X.-Y., Y.-C. Chung, and T.-Y. Huang. *A multiple LID routing scheme for fat-tree-based InfiniBand networks*. in *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*. 2004. IEEE.

6. Patooghy, A., H. Tabkhi, and S.G. Miremadi. *RMAP: a reliability-aware application mapping for Network-on-Chips*. in *Dependability (DEPEND), 2010 Third International Conference on*. 2010. IEEE.

7. Gomez, C., et al. *Deterministic versus adaptive routing in fat-trees*. in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. 2007. IEEE.

8. Moadeli, M., et al., *An analytical performance model for the Spidergon NoC with virtual channels.* Journal of Systems Architecture, 2010. **56**(1): p. 16-26.

9. Samman, F.A., T. Hollstein, and M. Glesner, *Networks-on-chip based on dynamic wormhole packet identity mapping management.* VLSI Design, 2009. **2009**: p. 2.

10. Duato, J., S. Yalamanchili, and L.M. Ni, *Interconnection networks: An engineering approach*. 2003: Morgan Kaufmann.

11. Benini, L. and G. De Micheli. *Powering networks on chips: energy-efficient and reliable interconnect design for SoCs*. in *Proceedings of the 14th international symposium on Systems synthesis*. 2001. ACM.

12. Dally, W.J. and B. Towles, *Principles and practices of interconnection networks*. 2004: Morgan Kaufmann.

13. Miguel-Alonso, J., C. Izu, and J. Gregorio, *Improving the performance of large interconnection networks using congestion-control mechanisms.* Performance Evaluation, 2008. **65**(3): p. 203-211.

14. Javadi, B., M.K. Akbari, and J.H. Abawajy, *A performance model for analysis of heterogeneous multi-cluster systems.* Parallel computing, 2006. **32**(11-12): p. 831-851.

15. Solnushkin, K.S., *Automated Design of Two-Layer Fat-Tree Networks.* arXiv preprint arXiv:1301.6179, 2013.

16. Bjerregaard, T. and S. Mahadevan, *A survey of research and practices of network-on-chip.* ACM Computing Surveys (CSUR), 2006. **38**(1): p. 1.

17. Abawajy, J., *An efficient adaptive scheduling policy for high-performance computing.* Future generation computer systems, 2009. **25**(3): p. 364-370.

18. Agarwal, A., C. Iskander, and R. Shankar, *Survey of network on chip (noc) architectures & contributions.* Journal of engineering, Computing and Architecture, 2009. **3**(1): p. 21-27.

19. Dall'Osso, M., et al. *Xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs*. in *Computer Design (ICCD), 2012 IEEE 30th International Conference on*. 2012. IEEE.

20. Dally, W.J. and B. Towles. *Route packets, not wires: On-chip interconnection networks*. 2001. IEEE.

21. Benini, L. and G. De Micheli. *Powering networks on chips: energy-efficient and reliable interconnect design for SoCs*. 2001. ACM.

22. Pande, P.P., C. Grecu, and M. Jones, *Performance evaluation and design trade-offs for network-on-chip interconnect architectures.* IEEE Transactions on Computers, 2005: p. 1025-1040.

23. Benini, L. and D. Bertozzi. *Network-on-chip architectures and design methods*. in *Computers and Digital Techniques, IEE Proceedings-*. 2005. IET.

24. Bouhraoua, A., O. Diraneyya, and M.E. Elrabaa. *A simplified router architecture for the modified Fat Tree Network-on-Chip topology*. in *NORCHIP, 2009*. 2009.

25. Ni, L.M. and P.K. McKinley, *A survey of wormhole routing techniques in direct networks.* Computer, 1993. **26**(2): p. 62-76.

26. Coppola, M., et al., *Design of cost-efficient interconnect processing units: Spidergon STNoC*. 2008: CRC press.

27. Pande, P.P., et al., *Performance evaluation and design trade-offs for network-on-chip interconnect architectures.* Computers, IEEE Transactions on, 2005. **54**(8): p. 1025-1040.

28. Guerrier, P. and A. Greiner. *A generic architecture for on-chip packet-switched interconnections*. in *Proceedings of the conference on Design, automation and test in Europe*. 2000. ACM.

29. Kumar, S., et al. *A network on chip architecture and design methodology*. in *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*. 2002. IEEE.

30. Kumar, S. and L. Kale. *Scaling collective multicast on fat-tree networks*. in *International Conference on Parallel and Distributed Systems*. 2004.

31. Ludovici, D., et al. *Assessing fat-tree topologies for regular network-on-chip design under nanoscale technology constraints*. in *Proceedings of the Conference on Design, Automation and Test in Europe*. 2009. European Design and Automation Association.

32. Rahmani, A.-M., et al. *Research and practices on 3D networks-on-chip architectures*. in *NORCHIP, 2010*. 2010. IEEE.

33. Wachter, E., et al. *Topology-agnostic fault-tolerant NoC routing method*. in *Proceedings of the Conference on Design, Automation and Test in Europe*. 2013. EDA Consortium.

34. Karim, F., A. Nguyen, and S. Dey, *An interconnect architecture for networking systems on chips.* IEEE micro, 2002. **22**(5): p. 36-45.

35. Pande, P.P., et al. *Design of a switch for network on chip applications*. in *Circuits and Systems, 2003. ISCAS'03. Proceedings of the 2003 International Symposium on*. 2003. IEEE.

36. Rijpkema, E., et al., *Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip.* IEE Proceedings-Computers and Digital Techniques, 2003. **150**(5): p. 294-302.

37. Nousias, I. and T. Arslan. *Wormhole routing with virtual channels using adaptive rate control for network-on-chip (NoC)*. in *Adaptive Hardware and Systems, 2006. AHS 2006. First NASA/ESA Conference on*. 2006. IEEE.

38. Isask'har Walter, A.C., R. Ginosar, and A. Kolodny, *Curing Hotspots in Wormhole NoCs.* DATE, ElectricalEngineering Department, 2006.

39. Jerger, N.E. and L.-S. Peh, *On-chip networks.* Synthesis Lectures on Computer Architecture, 2009. **4**(1): p. 1-141.

40. Dally, W.J., *Virtual-channel flow control.* Parallel and Distributed Systems, IEEE Transactions on, 1992. **3**(2): p. 194-205.

41. Marculescu, R., et al., *Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives.* Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2009. **28**(1): p. 3-21.

42. Heffes, H., *A class of data traffic processes-covariance function characterization and related queueing results.* Bell Syst. Tech. J, 1980. **59**(6): p. 897–929.

43. Shah Heydari, S. and T. Le Ngoc, *MMPP models for multimedia traffic.* Telecommunication Systems, 2000. **15**(3): p. 273-293.

44. Kathleen, F., *The Markov-modulated Poisson process (MMPP) cookbook.* Performance Evaluation, 1993. **18**(2): p. 149-171.

45. Ghandali, S. and S.M. Safavi. *Modeling multimedia traffic in IMS network using MMPP*. in *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*. 2011. IEEE.

46. Kang, S.H. and D.K. Sung, *A CAC scheme based on real-time cell loss estimation for ATM multiplexers.* Communications, IEEE Transactions on, 2000. **48**(2): p. 252-258.

47. Liu, K.H., et al., *Performance analysis of prioritized MAC in UWB WPAN with bursty multimedia traffic.* Vehicular Technology, IEEE Transactions on, 2008. **57**(4): p. 2462-2473.

48. Min, G. and M. Ould-Khaoua, *Performance modelling and evaluation of virtual channels in multicomputer networks with bursty traffic.* Performance Evaluation, 2004. **58**(2-3): p. 143-162.

49. Yulei, W., et al. *A Performance Model for Integrated Wireless Mesh Networks and WLANs with Heterogeneous Stations*. in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. 2009.

50. Heffes, H., *A class of data traffic processes—Covariance function characterization and related queuing results.* Bell System Technical Journal, 1980. **59**(6): p. 897-929.

51. Moadeli, M., et al. *Communication modelling of the Spidergon NoC with virtual channels*. in *Parallel Processing, 2007. ICPP 2007. International Conference on*. 2007. IEEE.

52. Jabbar, M.H., D. Houzet, and O. Hammami. *Impact of 3D IC on NoC Topologies: A Wire Delay Consideration*. in *Digital System Design (DSD), 2013 Euromicro Conference on*. 2013. IEEE.

53. Mello, A., et al. *Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC, Integrated Circuits and Systems Design*. in *18th Symposium on Volume, Issue, Date*. 2005.

54. Hammami, O., A. M'zah, and K. Hamwi. *Design of 3D-IC for butterfly NOC based 64 PE-multicore: Analysis and design space exploration*. in *3D Systems Integration Conference (3DIC), 2011 IEEE International*. 2012. IEEE.

55. Kiasari, A.E., Z. Lu, and A. Jantsch, *An analytical latency model for networks-on-chip.* Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 2013. **21**(1): p. 113-123.

56. Liu, J., et al., *An Analytical Model for Hypercube Network-On-Chip Systems with Wormhole Switching and Fully Adaptive Routing.* 2012.

57. Liu, W., et al. *A noc traffic suite based on real applications*. in *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*. 2011. IEEE.

58. Takabatake, T. *Simulations of NoC topologies for generalized hierarchical completely-connected networks*. in *Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on*. 2011. IEEE.

59. Zaman, A. and O. Hasan. *Formal verification of circuit-switched Network on chip (NoC) architectures using SPIN*. in *System-on-Chip (SoC), 2014 International Symposium on*. 2014. IEEE.

60. Liu, S., A. Jantsch, and Z. Lu. *Analysis and evaluation of circuit switched NoC and packet switched NoC*. in *Digital System Design (DSD), 2013 Euromicro Conference on*. 2013. IEEE.

61. Lu, Z., B. Yin, and A. Jantsch. *Connection-oriented multicasting in wormhole-switched networks on chip*. in *Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on*. 2006. IEEE.

62. Marcon, C., et al. *Tiny NoC: A 3D Mesh Topology with Router Channel Optimization for Area and Latency Minimization*. in *VLSI Design and 2014 13th International Conference on Embedded Systems, 2014 27th International Conference on*. 2014. IEEE.

63. Varga, A. *The OMNeT++ discrete event simulation system*. in *Proceedings of the European Simulation Multiconference (ESM'2001)*. 2001. sn.

64. Weingartner, E., H. Vom Lehn, and K. Wehrle. *A performance comparison of recent network simulators*. in *Communications, 2009. ICC'09. IEEE International Conference on*. 2009. IEEE.

65. Sinclair, J., *Simulation of Computer Systems and Computer Networks: A Process-Oriented Approach.* Feb, 2004. **15**: p. 1-261.

66. Varga, A., *OMNeT++ Version 3.0 user manual*. 2004.

67. Lin, X.Y., Y.C. Chung, and T.Y. Huang. *A multiple LID routing scheme for fat-tree-based InfiniBand networks*. 2004. IEEE.

68. Dally, W.J. and B.P. Towles, *Principles and practices of interconnection networks*. 2004: Elsevier.

69. Sahu, P.K. and S. Chattopadhyay, *A survey on application mapping strategies for network-on-chip design.* Journal of Systems Architecture, 2013. **59**(1): p. 60-76.

70. Singh, A.K., et al. *Mapping on multi/many-core systems: survey of current and emerging trends*. in *Proceedings of the 50th Annual Design Automation Conference*. 2013. ACM.

71. Yang, B., et al. *Tree-model based mapping for energy-efficient and low-latency network-on-chip*. in *Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on*. 2010. IEEE.

72. Hemani, A., et al. *Network on chip: An architecture for billion transistor era*. in *Proceeding of the IEEE NorChip Conference*. 2000.

73. Goossens, K., et al. *Networks on silicon: Combining best-effort and guaranteed services*. in *Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings*. 2002. IEEE.

74. Lei, T. and S. Kumar. *A two-step genetic algorithm for mapping task graphs to a network on chip architecture*. in *Digital System Design, 2003. Proceedings. Euromicro Symposium on*. 2003. IEEE.

75. Moadeli, M., et al. *An analytical performance model for the Spidergon NoC*. in *Advanced Information Networking and Applications, 2007. AINA'07. 21st International Conference on*. 2007. IEEE.

76. Leiserson, C.E., et al. *The network architecture of the Connection Machine CM-5*. in *Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures*. 1992. ACM.

77. Boura, Y.M. and C.R. Das, *Performance analysis of buffering schemes in wormhole routers.* Computers, IEEE Transactions on, 1997. **46**(6): p. 687-694.

78. Po-chi, H. and L. Kleinrock. *A queueing model for wormhole routing with timeout*. in *Computer Communications and Networks, 1995. Proceedings., Fourth International Conference on*. 1995.

79. Greenberg, R.I. and L. Guan. *An improved analytical model for wormhole routed networks with application to butterfly fat-trees*. 1997. IEEE.

80. Javadi, B., J. Abawajy, and M. Akbari, *Performance Analysis of Interconnection Networks for Multi-cluster Systems*, in *Computational Science – ICCS 2005*, V. Sunderam, et al., Editors. 2005, Springer Berlin / Heidelberg. p. 205-212.

81. Gomez, C., et al. *Deterministic versus adaptive routing in fat-trees*. 2007. IEEE.

82. Heindl, A., *Decomposition of general queueing networks with MMPP inputs and customer losses.* Performance Evaluation, 2003. **51**(2-4): p. 117-136.

83. De Micheli, G. and L. Benini, *Networks on chips: technology and tools*. 2006: Academic Press.

84. Bouhraoua, A. and M. Elrabaa. *An efficient network-on-chip architecture based on the fat-tree (FT) topology*. in *Microelectronics, 2006. ICM'06. International Conference on*. 2006. IEEE.

85. Lee, J., et al. *Do we need wide flits in networks-on-chip?* in *VLSI (ISVLSI), 2013 IEEE Computer Society Annual Symposium on*. 2013. IEEE.

86. Min, G. and M. Ould-Khaoua, *Performance modelling and evaluation of virtual channels in multicomputer networks with bursty traffic.* Performance Evaluation, 2004. **58**(2): p. 143-162.

87. Wu, Y., et al., *Modeling and analysis of communication networks in multicluster systems under spatio-temporal bursty traffic.* Parallel and Distributed Systems, IEEE Transactions on, 2012. **23**(5): p. 902-912.

88. Atov, I. and R.J. Harris, *Approximate model for merging Markovian arrival processes*.