DESIGN OF SWITCH ARCHITECTURE FOR THE GEOGRAPHICAL CELL TRANSPORT PROTOCOL

A thesis submitted to the college of
Graduate Studies and Research
In Partial Fulfillment of the Requirements
For the Degree of Master of Science
in the Department of Electrical and Computer Engineering
University of Saskatchewan
Saskatoon

By

UMESH GYAWALI

PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor, who supervised my thesis work or, in his absence by the Head of Department or the Dean of the college. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of Department of Electrical and Computer Engineering
University of Saskatchewan
57 Campus Drive
Saskatoon, Saskatchewan, Canada
S7N 5A9

ABSTRACT

The Internet is divided into multiple layers to reduce and manage complexity. The International Organization for Standardization (ISO) developed a 7 layer network model and had been revised to a 5 layer TCP/IP based Internet Model. The layers of the Internet can also be divided into top layer TCP/IP protocol suite layers and the underlying transport network layers. SONET/SDH, a dominant transport network, was designed initially for circuit based telephony services. Advancement in the internet world with voice and video services had pushed SONET/SDH to operate with reduced efficiencies and increased costs [1]. Hence, redesign and redeployment of the transport network has been and continues to be a subject of research and development. Several projects are underway to explore new transport network ideas such as G.709 [2] and GMPLS [3].

This dissertation presents the Geographical Cell Transport (GCT) protocol as a candidate for a next generation transport network. The GCT transport protocol and its cell format are described. The benefits provided by the proposed GCT transport protocol as compared to the existing transport networks are investigated. Existing switch architectures are explored and a best architecture to be implemented in VLSI for the proposed transport network input queued virtual output queuing is obtained. The objectives of this switch are high performance, guaranteed fairness among all inputs and outputs, robust behavior under different traffic patterns, and support for Quality of Service (QoS) provisioning. An implementation of this switch architecture is carried out using HDL.

A novel pseudo random number generation unit is designed to nullify the bias present in an arbitration unit. The validity of the designed is checked by developing a traffic load model. The speedup factor required in the switch to maintain desired throughput is explored and is presented in detail. Various simulation results are shown to study the behavior of the designed switch under uniform and hotspot traffic. The simulation results show that QoS behavior and the crossing traffic through the switch has not been affected by hotspots.

ACKNOWLEDGEMENTS

I would like to thank Dr. Carl McCrosky for being a wonderful Supervisor. I could write an entire chapter of thanks to Dr. McCrosky, but will choose to simply say that his support and guidance throughout my graduate career were invaluable. He provided constant support and encouragement, made himself readily available whenever and wherever, and always had a patient ear, even for all the nonsense I had to say. I am both incredibly proud and immensely lucky to be able to refer to myself as one of his students. Everyone needs a supervisor like you.

I appreciate Dr. J. Eric Salt for providing financial support throughout the course of the project. Also, thanks to Dr. Daniel Teng. I have learned a lot from him. I must thank Dr. Dwight Makaroff for his valuable suggestions. I must thank my family for their support and encouragement during my graduate career.

Finally, I would like to thank the countless developers, documentation writers, bloggers and the administrative staffs at the department of Electrical and Computer Engineering through whom I've benefited during the course of this thesis.

DEDICATION

To my family and my supervisor

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ALUT	Adaptive Look-up Table
ATM	Asynchronous Transfer Mode
BER	Bit Error Rate
CID	Consecutive Identical Digits
CMOS	Complementary Metal Oxide Semiconductor
CRC	Cyclic Redundancy Check
DWDM	Dense Wavelength Division Multiplexing
FEC	Forward Error Correction
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GCT	Geographical Cell Transport
GE	Gigabit Ethernet
GFP	Generic Framing Protocol
GMPLS	Generalized Multiprotocol Label Switching
HDL	Hardware Descriptive Language
HDLC	High-level Data Link Control
HoL	Head-of-Line
IP	Internet Protocol
ISO	International Organization for Standardization
LCAS	Link Capacity Adjustment Scheme
LTL	Logical Transport layer
MPLS	Multiprotocol Label Switching
MTU	Maximum Transmission Unit
OAM&P	Operations Administration Maintenance and Provisioning
OTL	Optical Transport Layer
PIM	Parallel Iterative Matching
POS	Packet Over SONET
PPP	Point-to-Point Protocol
QoS	Quality of Service

RPA	Reservation with Preemption and Acknowledgment
RPR	Resilient Packet Ring
SDH	Synchronous Digital Hierarchy
SONET	Synchronous Optical Networking
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TTL	Time to Leave
UDP	User Datagram Protocol
VLSI	Very Large Scale Integration
VoIP	Voice over Internet Protocol
VOQ	Virtual Output Queuing
WDM	Wavelength Division Multiplexing
WFA	Wavefront Arbiter
WWFA	Wrapped Wavefront Arbiter

Chapter 1
INTRODUCTION

1.1 Introduction

This chapter motivates and introduces this dissertation. The chapter begins with a brief introduction to transport networks and then presents the motivation for doing this work. The goal of this thesis is then described. Finally the organization of the dissertation is described at the end of the chapter.

1.2 Motivation

Throughout history, communications had a major impact on all societies. With the introduction of digital computers and the Internet, communication technologies have become even more important. Demands for communication technology have grown explosively due to the exponential growth of the Internet. At the same time, numerous new applications have been proposed that drive the demand for communication bandwidth at an even faster pace.

TCP/IP has been one of the most important foundations of the internet along with the continuing growth of information age. TCP assumes responsibility for end-to-end flow and IP makes a best effort attempt to deliver IP frames to intended endpoints. It is the standard interface by which users of the Internet make use of the facilities of transport networks. The TCP/IP suite has been the dominant set of telecommunications protocol used by wide range of applications at the endpoint devices. Thus, TCP/IP protocol suite has a major impact on the underlying transport network [1].

Transport networks are unseen infrastructures that provide local, regional and international connections for nearly all digital communication applications. The transport networks serve as the bearers of services like voice, data, and video signals. Due to the impact of the continuing growth of traffic and the introduction of numerous new services to the end users, demand has increased for transport network capacity. More efficient solutions

of handling traffic growth and managing new applications are required [1]. As transport networks grow in size and capacity, the focus lies in extending the capabilities of today's networks. The standard is based on advanced technological solutions provided today by VLSI and optical communications.

Transport networks underwent a revolution with the introduction of SONET/SDH in the 1980s. SONET/SDH optical transport systems replaced copper and microwave radio transmission media. With the development of fiber and optical component technologies and also with the severe internet growth, transport network itself saw an explosive growth. Reducing Operation Administration Management and Provisioning (OAM&P) costs is important, as the operating costs of transport network are typically much larger than the establishing costs.

The efficiency of data networks in bandwidth use had dominated the use of voice traffic. SONET/SDH was designed initially for circuit based voice services. The development of data networks and other new application protocols has pushed SONET/SDH to operate with reduced efficiencies and increased costs. Hence, redesign and redeployment of the transport network has been and continues to be the subject of research and development. Several projects are underway to explore new transport network ideas such as GMPLS [3] and G.709 [2]. This dissertation motivates a new transport network by recognizing some of the weaknesses present in today's transport network.

The communication network consists of transmission links and switching nodes. Transmission links constitute the optical fiber and supporting optical regeneration equipment. Advances in fiber-optic transmission technologies such as wave length division multiplexing (WDM) have greatly pushed the envelope of bandwidth available in fibers. OC 768 (40 Gb/s) is currently possible with the fiber and CMOS technology. Despite the advancement in the VLSI technology, the gap between the data rate, that the optical transmission technology can deliver and those that electronic switches can process is broadening [1]. Thus it is important that the electronic switch used in system design be as efficient as possible for transferring

data from inputs to outputs by giving higher throughput. This dissertation looks into the prototype for a switch that will support proposed protocols and provide high throughput.

1.3 Design Task

The goal of this thesis is to design and prototype a switch using HDL that supports the proposed cell based protocol. First, the detail descriptions of the Geographical Cell Transport (GCT) protocol are explored. The proposed GCT transport protocol is then compared with the existing standards like SONET and some of the benefits for the proposed protocol are obtained. Several existing switch architectures are investigated and the one which offers close to ideal performance, guarantees fairness among all its inputs and outputs, behaves robustly under different traffic patterns, supports QoS provisioning and can be implemented in VLSI for high throughputs at a reasonable cost is chosen. The class of input queued, virtual output queuing (VOQ) architectures to find a candidate architecture that fulfills these requirements for the design of GCT switch is designed. Wrapped wavefront arbiter [20] is used as an arbitration unit. A novel pseudo random number generation unit to nullify the bias present in the arbitration unit is modeled using HDL.

A traffic load model is developed to evaluate the performance of designed switch. The speedup factor required in the switch to maintain desired throughput is obtained. Various simulations are performed to obtain the results and to study the behavior of the designed switch under uniform and hotspot traffic.

1.4 Dissertation Organization

This dissertation is structured as follows: Chapter 1 is a general introduction. It contains a brief introduction to transport networks and introduces the problem domain and the goal of the thesis. Chapter 2 gives an overview of the problem associated with the current transport network. Some legacy solutions and some proposed solutions are also briefly described in Chapter 2. The description of the proposed GCT protocol, its header information, its overhead cost comparison and the advantages over existing transport network

solution are also given in Chapter 2. Chapter 3 describes the various classes of conventional switch architectures. The advantages and disadvantages of different crossbars based switch architectures are also presented. The design principles and the operation of the switch architecture used in the design are then described in detail in Chapter 3. Chapter 4 describes about the scheduling problem and solution in detail. The random number generation procedure for achieving fairness is also described in Chapter 4. Chapter 5 describes the load generation model used in verifying the performance of the designed switch. The speedup required in the switch and the performance characteristics of the switch under different traffic load conditions are also described in Chapter 5. Finally, Chapter 6 presents the conclusions arrived at in the course of the work presented in this dissertation, and indicate useful directions for future research.

Chapter 2
GEOGRAPHICAL CELL TRANSPORT

2.1 Background

TCP/IP is the dominant set of telecommunication protocols used by a wide range of endpoint devices. TCP/IP is organized into layers as shown in Figure 2.1 [1]. It uses a simplified structure of four layers viz. application, transport, network and link layers. Each layer has its own roles and responsibilities. However, there exist transport problems which are inherent to the TCP/IP protocol suite due to the mismatch between the strengths of TCP/IP suite and the underlying network. The back off model during congestion, single shortest possible path due to IP routing algorithm, lack of QoS, and costs of routing are some of the difficulties associated with TCP/IP protocol suite.

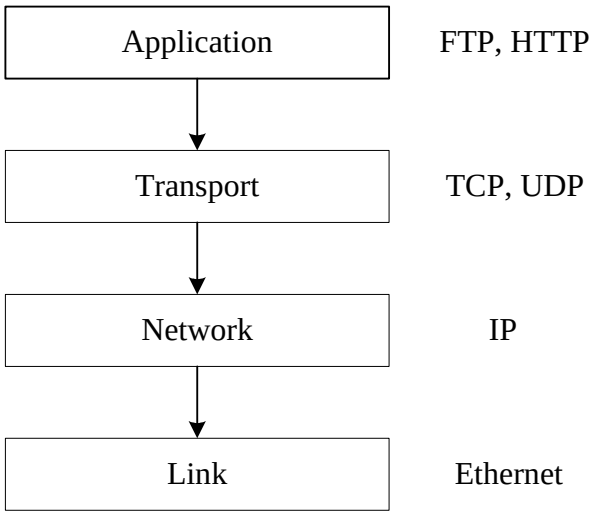

Figure 2.1 Four layer protocol stack of TCP/IP

The goal of this chapter is to motivate the need for a next generation transport network and to propose a particular next generation transport network solution. Section 2.2 gives an introduction to the transport problems inherent to the TCP/IP protocol suite. The legacy solution that has been used for transporting under the SONET is discussed in Section

2.3. Various proposed solutions to overcome the legacy problem are then given in Section 2.4. Section 2.5 presents an overview of the GCT protocol. An overhead cost comparison is given in Section 2.6. Finally, a summary of this chapter is presented in Section 2.7.

2.2 Problems with the TCP/IP Suite

The principal responsibility of TCP/IP is to establish and manage end to end communications flows and to provide reliable services. But with the nature and established use of the TCP/IP protocol suite, a set of systematic challenges to the transport networks exists. The back off model during congestion, one shortest possible path due to IP routing algorithm, lack of QoS, and costs of routing are some of the challenges provided by TCP/IP protocol suite to the transport networks [1]. It is not possible to support differentiated QoS with TCP/IP when all classes of traffic must be routed over the same path. Also, it is not possible to make use of secondary routes, slightly longer than the shortest route, when IP routing algorithms send all traffic over the shortest route. New IP routes cannot be determined quickly enough in the face of link or node failures, as distributed IP routing algorithms are slow. Also, it is desirable to find an improved solution to fill the gap between the underlying fiber and the TCP/IP protocol suite. The presence of these difficulties prevents TCP/IP alone from satisfying the needs of modern networks. There are several solutions proposed in the literature. Any solution must guarantee QoS, provide efficient use of network resources, must have provision for network survivability in presence of hardware failures, and support OAM&P.

A common approach to the problems presented by the TCP/IP protocol suite is to add a new protocol between the TCP/IP protocol suite and the underlying fiber. Some of the protocols used are Asynchronous Transfer Mode (ATM) [5], Generic Framing Procedure (GFP) [6], Multiprotocol Label Switching (MPLS) [5], Gigabit Ethernet (GE) [7], Resilient Packet Ring (RPR) [8] and G.709 [2].

2.3 Legacy Solutions

Synchronous Optical Network (SONET) [4] is a time-division multiplexing (TDM) protocol. It was originally designed to carry voice traffic and has been used for transporting data since 1990's. Voice and data networking has been constantly evolving as the technology evolves. Data networks have evolved to the point such that both voice and data networks are converged into the same data infrastructure. SONET rings were designed and deployed to transport voice but could transport data by breaking down the data into manageable pieces. TDM protocols are not an ideal solution for transporting data due to its inability to share all of the circuit resources in response to the varying demands.

Several solutions were provided to carry data through the SONET infrastructure. IP frames were embedded over SONET byte streams as either packet over SONET (POS) or point-to-point protocol (PPP). This protocol stack for taking an IP frame over SONET is shown in Figure 2.29 (a). The drawback to the POS is its non-deterministic bandwidth requirement [9]. The addition of control information in the byte stream require twice as much space in POS as they required at native IP layer. Also, it is impossible to provide differential QoS for different TCP/IP traffic classes. In addition to this, provision to interrupt a lower priority frame in favor of a higher priority QoS frame, such as voice over internet protocol (VOIP), is not available.

The shortcomings present in PPP or POS can be eliminated by the use of ATM. The protocol stack with TCP/IP over ATM [1] over SONET is shown in Figure 2.2 (b). The IP frames are chopped up into segments and are carried as ATM payload. ATM provides differential QoS for different services and different service level agreements. Also, with the independent scheduling of ATM cells, there is the possibility of interrupting lower class traffic by higher class traffic. ATM has been used extensively in some service providers' networks. The cell tax overhead and the ATM hardware cost and complexity were the hindrances to the success of ATM, otherwise ATM might have provided a network-wide solution with benefits comparable to MPLS based solution [5].

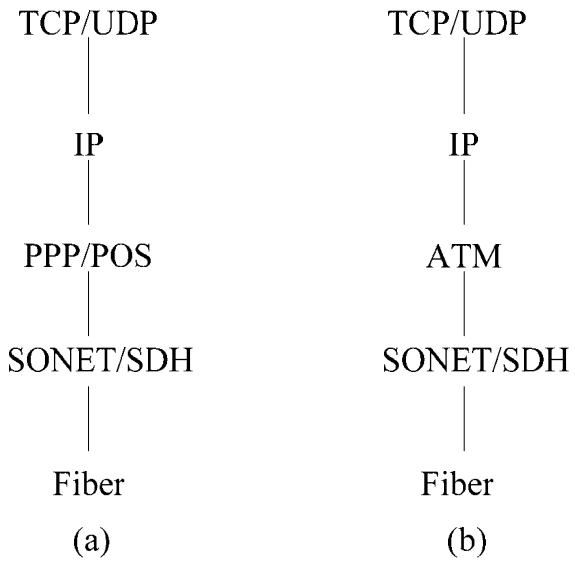

Figure 2.2 A protocol stack for IP over SONET (a) PPP/POS (b) ATM

2.4 Proposed Solutions

A proposed protocol stack for the next generation Internet is shown in Figure 2.3 (a). The TCP/IP protocol suite and other application protocols are compatible with the proposed network. TCP and UDP over IP are preserved at the network layer. Any other protocols are also supported. The bottom layer is also preserved. The traffic is still carried in SONET over fiber. The two new intermediate layers provide a bridge from the TCP/IP protocol suite to the SONET fiber layers by overcoming the shortcomings of TCP/IP protocol suite. The first layer MPLS [5] provides most of the solutions to the problems stated before. Management of traffic over multiple paths, differential QoS, proper use of SONET pipes by dividing into multiple MPLS channels are the advantages provided by MPLS.

The next layer is GFP [6]. Although the GFP has very few functions, it allows the simple, efficient transport of a variety of data signals over the same SONET Fiber. GFP distinguish the starts of frames within the carrying byte streams in SONET. It distinguishes the different type of frames within the byte stream. In addition to these, Cyclic Redundancy

Check (CRC) based protection for all GFP headers and payload is also provided by GFP protocol.

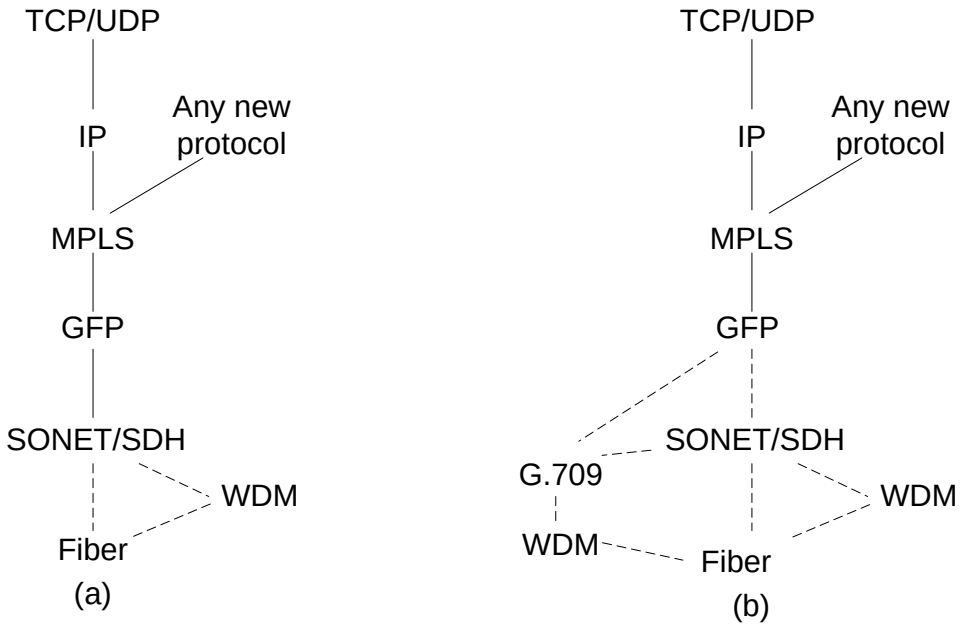

Figure 2.3 Protocol stacks using (a) MPLS and (b) MPLS with G.709

Digital wrapper technology (G.709) [2] is a layered structure like SONET that includes protection, performance, monitoring and other management services. There exists some weaknesses in SONET like overhead ratio, expensive protection, difficult provisioning and expensive unused legacy features (DSO clocking). G.709 remedies these weaknesses of the SONET and acknowledged the diversity of the protocols used in real networks. The protocol stack using G.709 is shown in Figure 2.3 (b). G.709 has the ability to carry any type of data like SONET, Ethernet, ATM, IP or Fiber Channel. Forward error correction (FEC) is mandatory and has been standardized in G.709. Another advantage provided by G.709 is the prompt provisioning of the optical channels. With prompt provisioning, carriers offer service level agreements on any protocol or service. End to end optical channel monitoring and management is possible with G.709.

Wavelength division multiplexing (WDM) take optical signals, each carries information at certain bit rate, gives them a specific wavelength, and then sends them down

the same fiber. Each input optical signal has the illusion of possessing its own fiber. Dense wavelength division multiplexing (DWDM) uses eight or more wavelength to be sent on a single fiber. SONET/SDH and G.709 are carried through the fiber using WDM. Optical fiber technology has enabled extreme growth in networks by providing lower costs, higher bandwidth and high fidelity. SONET has grown from 155 Mb/s to 40 Gb/s in a very short period of time. Significant amount of work is in progress to provide more colors in DWDM, more bandwidth and distance per color and lower cost amplifiers.

However, there are certain limitations to achieve all optical networks. Light is extremely good for communications due to low loss and interference. But light is poor for computation and storage. Electric signals are poor for long distance communications but have strong computation and storage capability. Telecommunications system requires information to be used for routing and also requires storage in the form of queues for the waiting egress link. Light is difficult to sense due to its weak computation and storage, thus making the routing process difficult. Also, light has no simple and economical storage technologies. The practical solution for the light storage is to loop the light paths. Also, optical power management is a severe problem. These are the reasons that preclude all optical networks. The feasible solution is hybrid optical/electrical networking. Foreseeable future, communication will continue to be done in light and computation and storage will continue to be done in electricity.

A major disadvantage of the above proposed solution, which adds protocol layers in between TCP/IP protocol suite and the underlying SONET over fiber, is the requirement of additional hardware. Different sets of hardware are required to overcome the performance mismatch of the TCP/IP protocol suite at different layers like in routing layer, logical switching layer and physical switching layer. This is unnecessarily expensive. With the increase in number of hardware, the management cost will also be increased and the overall network efficiency will be decreased.

Automatic protection switching is used in SONET/SDH to restore service in the case of an optical fiber failure or a network equipment failure. SONET standards require

restoration of service within 50 ms. The most common SONET topology is the ring. An additional fiber is used for the protection. This requirement of double resources is expensive. An improved solution should be less expensive protection and at the same time protection must be more powerful. In addition to this, to achieve optimal resource utilization, maximize throughput, and minimize response time, load balancing is essential. SONET offers little dynamic load balancing. Only the link capacity adjustment scheme (LCAS) provides limited load balancing. However, the full mesh of any large network has the ability for enormous load sharing, balancing and protection. Multipath routing is generally used for obtaining load balancing and protection. The next section describes the new Geographical Cell Transport (GCT) protocol, which we believe will provide a simplified solution to the existing transport network.

2.5 Geographical Cell Transport (GCT) Protocol

A protocol stack using GCT is shown in Figure 2.4. GCT transport network has two layers: a logical transport layer (LTL) and the optical transport layer (OTL). GCT transport networks as a whole are hybrid optical/electrical networks. The network addressing scheme utilizes latitude and longitude, which motivates the name of the protocol as Geographical Cell Transport protocol. The latitude and longitude addressing scheme is used directly in routing decisions. Besides that, FEC is used to help reduce the bit error rate. The use of FEC reduces the burden in the optical layer to improve the bit error rate.

The TCP/IP protocol suite and the underlying fiber technology remain unchanged. As shown in Figure 2.4, the legacy TCP and UDP over IP are preserved and any other new application protocols are also supported. GCT is inserted in between the underlying fiber technology and the TCP/IP protocol suite or any other protocol suites. GCT will provide efficient protection, load balancing, and strong QoS. Routing is done as multipath routing. With the use of multipath routing, network protection, load balancing and QoS are assured.

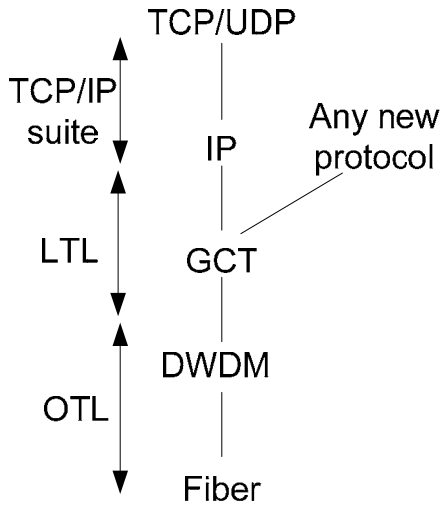

Figure 2.4 Protocol stack using Geographical Cell Transport

Figure 2.5 shows an abstract view of GCT making connection to the TCP/IP world. In the figure, SONET legacy connections are also shown. This SONET connection is independent of the GCT transport network. TCP/IP connections from one IP island to another IP island are made through GCT and similarly any other new application protocols can also be transported from any island to any other island using GCT transport network.

GCT uses a fixed single length cell for all communications. There are several advantages of using fixed length cells. Using fixed length cells, the scheduling decision in the switch becomes easier compared to the case with variable length packets. Scheduling can be done at a regular interval of the fixed cell time. Fixed length cells improve switch efficiency and scalability.

However, with the cell based system, additional overhead is introduced and thus internal speedup is required. There is also a moderately complex segmentation and reassembling process. The required speedup factor will be discussed in Chapter 5. However, with GCT switching, the segmentation and reassembling process is done at the end points, while the whole routing process only involves the fixed length cells. Another advantage of a cell-based system is that all data is self addressing which is not the case in TDM systems.

Cell scheduling enables flexibility in allocating bandwidth to separate logical flows. Also, FEC coders encourage the use of fixed length cells than the variable length cells [10].

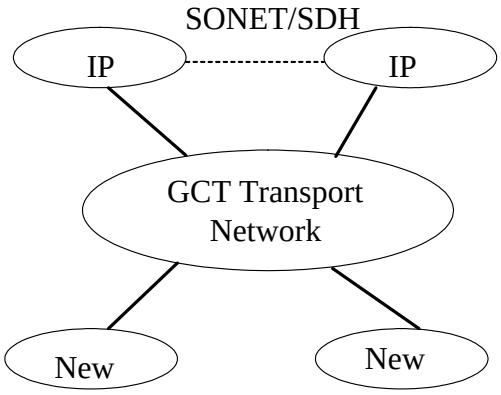

Figure 2.5 GCT transport network connections

The GCT cell size is determined by several factors. The cell size chosen should be large enough to make sure to reduce the relative header overhead costs. It should be appropriate for the error correction code. Another important factor for choosing the cell size is that the cell arrival rate must be practical for reasonable receiver logic. Also, it should be practical for transferring GCT cells to and from queues.

The chosen logical GCT cell length is 256 bytes. The chosen cell size offers low header overhead costs with a header size of 12 bytes. For a link of 40 Gb/s, a 256 bytes cell arrives in 51.2 ns. For VLSI hardware with 400 MHz clock, 20 clock ticks are required to process each arriving GCT cell. Similarly, for the 40 Gb/s flow, a GCT cell of 256 bytes arrives at a rate of 19.5 megacells per second. Hence, requires 2 x 19.5 megacells per second to be transferred to and from RAM. For the RAM with 400 MHz, this can be achieved with 11 reads/writes per cell with 200 bits in parallel.

2.5.1 GCT Cell Format

The GCT cell format consists of 12-byte header, 228-byte payloads and 16-byte for FEC. FEC covers all the bytes except the Time to Live (TTL) byte present in the header. Hence FEC is used to protect 11 bytes header and 228 bytes of payload. The TTL value changes from hop to hop. The header and the payload of the GCT cell are shown in Figure 2.6. The header of the GCT cell consists of:

Destination (Word 0, bits 0 to 29)

This field contains an integer in the range from 0 to $2^{30} - 1$. Out of 30 bits, 22 bits are used for GCT switches and the remaining 8 bits are used for GCT ports which lead to GCT/IP or any other terminations.

<table>
<tr><td colspan="3">Destination (30b)</td><td>Class (2b)</td></tr>
<tr><td colspan="3">Source (30b)</td><td>Spare (2b)</td></tr>
<tr><td>Cell Sequence Number (16b)</td><td>Cell Start (8b)</td><td colspan="2">TTL (8b)</td></tr>
<tr><td colspan="4">Payload (1824b)</td></tr>
</table>

Figure 2.6 GCT cell format

Class (Word 0, bits 30 to 31)

In order to maintain QoS, the GCT transport protocol supports four classes represented in two bits. Class field is used to distinguish different GCT priorities. Class 0 is the highest priority and class 3 is the lowest priority. Class 0 or priority 0 is used for GCT control cells. A GCT control cell includes link control cells and directed path control cells. Class 1 is used for real time services. Real time services include telephony, video and security. Call acceptance control (CAC) is used for this class. Class 2 is used to represent

other prioritized data which also requires CAC. Class 3 is used for best effort data. TCP/IP falls into this category. CAC is not used for this class.

Source (Word 1, bits 0 to 29)

This field contains an integer in the range from 0 to $2^{30} - 1$. Out of 30 bits, 22 bits are used for GCT switches and the remaining 8 bits are used for GCT ports which act as a source for a GCT/IP or any other terminations.

Spare (Word 1, bits 30 to 31)

The two bits are unused and can be used for any other purposes required in the future. They are presently defined to be zeros.

Cell sequence number (Word 2, bits 0 to 15)

This field contains an integer in the range from 0 to $2^{16} - 1$. This field contains a sequence number as a label for each cells. The sole aim is to identify the original transmitted order of the cells.

Cell start (Word 2, bits 16 to 23)

This field contains an integer in the range from 0 to $2^{8} - 1$. This field identifies the starting position of a frame in cell. A default value of all 1s indicates the existence of no new frame in a cell.

Time to live (TTL, Word 2, bits 24 to 31)

The purpose of the TTL field is to avoid infinitely looping cells that would be capable of congesting the network. The field TTL, as the name suggests is used to identify the remaining life time of a cell. This field contains an integer in the range from 0 to $2^{8} - 1$. Out

of eight bits, one bit is used to protect the TTL as an odd parity and other seven bits give the value for the remaining life time of the cell. At first a maximum number is assigned to the cell. When the cell takes a switch-to-switch hop, the TTL value is decremented by one. When the TTL value goes to zero, the cell is discarded.

2.5.2 Other Issues in GCT

The GCT Payload is 228 bytes. The whole payload and the header except the TTL are then protected by FEC. FEC is normally checked on each hop. The FEC code is to be the Reed-Solomon code, RS (255, 239) [10]. With this code, for every 239 bytes of data, 16 bytes are added for error correction. In an error detection mode, RS (255, 239) can detect up to sixteen bit errors in a code word. In the error correction mode, RS (255, 239) can correct up to eight bit errors in a code word.

The combination of header, payload and FEC makes the chosen logical GCT cell of 256 bytes length. For consecutive identical digit (CID) protection, one bit is added for each 32 bits. This CID protection is not the process of encrypting the data, but is the process to give the transmitted data a useful engineering property. This means that there will not be too many 1s or 0s in a row. Also, the addition of one bit is to achieve direct current balance and to provide enough state changes or transitions to allow reasonable clock recovery. This is an important attribute in a signal that needs to be sent at high rates as it helps in reducing inter-symbol interference. Hence, the total length of GCT cell after the addition of one bit for each 32 bit is 264 bytes. With the addition of one bit, 32 bits of data are transmitted as a 33 bit entity. The addition of the bit is done at the end of 32 bits.

A sufficient number of transitions are assured by an addition of this single bit. The idea is to first find the number of transitions in the 32 bit entity. The number of transitions is found by performing exclusive OR for every pair of bit adjacent to each other, then taking the sum of these results to get the total number of transitions. If the total number of transitions is greater than or equal to 15, the extra bit will be a zero and the original 32 bits remains unchanged. However, if the total number of transitions is less than 15, then the extra

bit will be the value 1. In addition to this, every other bit in the 32 bit word will be inverted. The word with alternate bits inverted will have more than 15 transitions and the sum of transitions in the original word and the partially inverted word will be 31. This ensures a relatively even distribution of 1s and 0s in the transmitted data.

For the time being, we will not concern ourselves with how the entire route through the network is determined and established. The issue of routing is beyond the scope of this work. However, the routing used in a GCT transport network is multipath routing. Multipath routing is used to reduce the congestion in a network as well as to increase overall network utilization and protection efficiency. Multipath routing alleviates the congestion by routing data from highly utilized links to the links which are less highly utilized. Load balancing is achieved by spreading the traffic along multiple routes, thus avoiding congestions and bottlenecks.

Protection is also insured by multipath routing. Multipath routing assigns multiple paths to route the data from the source to the destination. Multipath routing can provide high fault tolerance in the presence of route failures. As long as there is presence of at least one path due to the failure of all other paths, the data does not fail to reach to the destination.

The other advantage of multipath routing is the utilization of bandwidth. Routing along a single path may not provide enough bandwidth for a connection from one hop to another hop. However, if multiple paths are used simultaneously to route data, the aggregate bandwidth of the paths may satisfy the bandwidth requirement of the application.

2.6 Overhead Comparison

The comparisons of overhead ratio on transporting 512 bytes of data through SONET and through GCT are shown in Table 2.1 and Table 2.2 respectively. The TCP header of 20 bytes is first added to the 512 bytes of data. The IP header of 20 bytes is then added. MPLS and GFP add 4 bytes and 8 bytes of header, respectively. The resulting packet is then transported within SONET OC-192 with a date rate of 9.953 Gb/s. The compound

throughput is found to be 0.8746 as shown in Table 2.1. If FEC is used in the SONET, the overall throughput is 0.8094.

Table 2.1 Overhead ratio for SONET OC-192

512 B over STS - OC 192	Overhead Ratio
TCP	0.9624
IP	0.9624
MPLS	0.9922
GFP	0.9846
SONET	0.9666
Compound Throughput	0.8746
With SONET 1 + 1 Protection	0.4373
G.709	0.9255
Compound Throughput (FEC)	0.8094
With SONET 1 + 1 Protection (FEC)	0.4047

Similarly, on transporting 512 bytes over GCT, the header is the same for TCP and IP. GFP* is used instead of GFP. The header of GFP* is of 4 bytes as compared to 8 bytes of GFP. The length and header protection, each of 2 bytes, is discarded to obtain GFP*. The reason for this is the use of FEC in GCT which is mandatory. So, no added protection is necessary at the GFP layer.

Table 2.2 Overhead ratio for GCT

512 B over GCT	Overhead Ratio
TCP	0.9624
IP	0.9624
GFP*	0.9922
GCT (No FEC)	0.9500
Compound Throughput	0.8730
GCT with FEC	0.8906
Compound Throughput	0.8184
GCT with Scrambling	0.8636
Compound Throughput	0.7936
With GCT 1:5 Protection	0.6349

From Table 2.1 and Table 2.2, it was found that the overheads for GCT without FEC and the SONET are almost the same. The overhead for GCT with FEC is slightly higher with SONET. However, using FEC in the SONET, the overheads are almost same. The great advantage of GCT over SONET is the improvement in its protection costs.

As shown in Table 2.1, for the SONET with 1:1 protection, one working line and other protection line [1], throughput becomes 0.4047. With the use of multipath routing, various paths are provided in GCT. Considering GCT with 1:5 protections, the throughput is 0.6349 as shown in Table 2.2. Hence, GCT provides large gain in protection, load balancing and also provides lower network design costs.

2.7 Summary

This chapter describes GCT as a candidate for the next generation transport network. First the disadvantages of TCP/IP protocol suite for the transport services were described. The legacy solutions that have been used to transport TCP/IP through SONET were then presented. A brief description on the proposed solutions by adding different protocol stacks in between TCP/IP protocol suite and the underlying fiber were then discussed. The disadvantages to the proposed solutions and the advantages that will be provided by the proposed candidate transport network using GCT protocol were then examined. The header format of the GCT protocol was then described. The comparison of the overhead was then discussed. From the comparison table, it was found that GCT provides large gain in protection, load balancing and lowers network design costs than to the existing transport network.

Chapter 3
SWITCH ARCHITECTURE

Growth in the requirements for communication network capacity has led to a move from conventional switch architectures to architectures that can be scaled to multi-gigabit per second fabrics. The major components of any switch architecture are ingress ports, egress ports, and a switch fabric. The switch fabric is the core component of the switch that provides and controls a physical path between ingress line cards and egress line cards. The line cards receive the traffic from the network interfaces and perform route lookups to determine the destination egress port. The line card then forwards the incoming traffic to switch fabric, which is responsible for transporting them from ingress line card to egress line card.

The goal of this chapter is to develop a switch architecture that can be scaled to achieve high throughput. This chapter looks into the ingress queuing and VOQ architecture to find an appropriate architecture. Section 3.1 gives an introduction to switch architecture. Different conventional architectures and its limitations are discussed in Section 3.2. The advantages of the crossbar as a switched backplane are then given in Section 3.3. Section 3.4 gives the overview of the different architectures that employ crossbar switched backplane. The ingress queuing, VOQ that has been used in our switch is described in Section 3.5. Finally, a summary is given in Section 3.6.

3.1 Introduction

The important functions performed by any switch can be categorized as datapath functions and control functions. Figure 3.1 shows the common architectural components of any switch. Functions like forwarding decisions, backplane and output scheduling, are performed in hardware, whereas management and maintenance functions are performed in software. The hardware functions are referred to as fast path operations and software functions are referred to as slow path operations. The datapath functions are performed in the hardware. The performance of any switch is limited by the datapath function.

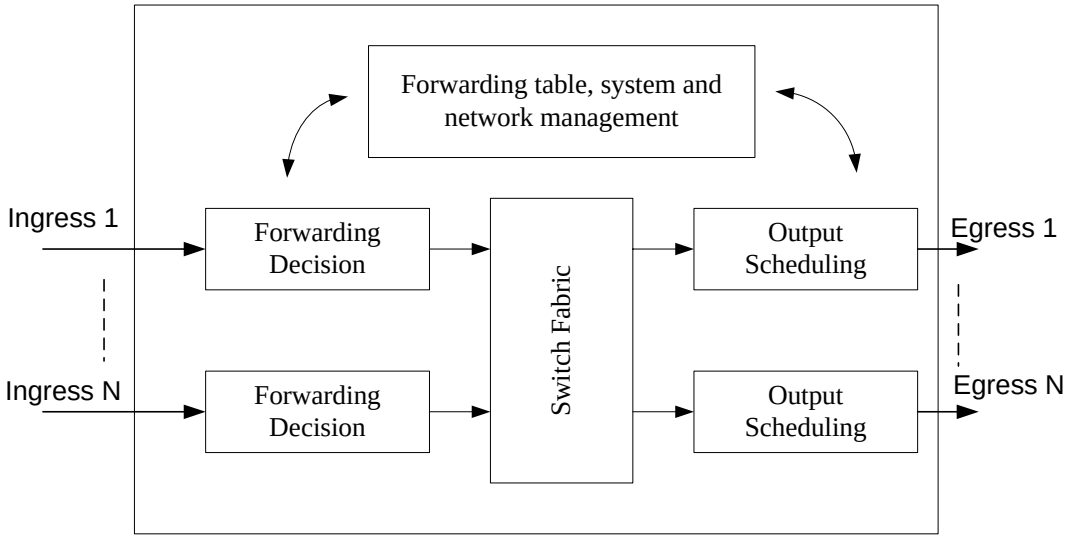

Figure 3.1 Common architectural components of a switch

Almost all switches perform the same basic functions. When a cell or packet arrives at the ingress side, first its header is examined. The destination address is looked up in forwarding table. If the address is found, the egress port through which it should pass is determined. The cell or packet is then forwarded across the switch core to its outgoing port. Cells or packets may need to be queued before they are actually transferred across the backplane. If the queues fill, the incoming cells may be dropped. Different scheduling mechanisms may also be required at the output side before they are transferred to the output link. The next section describes the different architectures that performed these basic functions.

3.2 Architecture Overview

The choice of switch architecture depends on the required number of ports, the required performance and the available technology. Different switch architectures have evolved over time [11] and this section summarizes their strengths and weaknesses. The earlier ones were built around conventional computer architectures. Figure 3.2 shows an architecture consisting of shared bus, central CPU, memory and peripheral line cards.

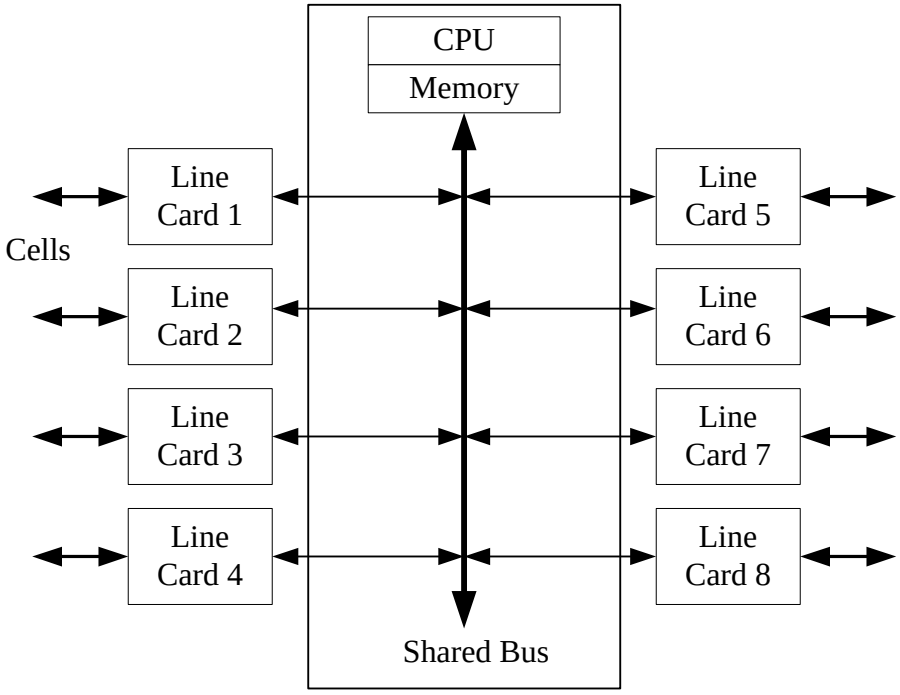

Figure 3.2 Shared CPU switch architecture

A CPU with memory and multiple line cards are connected by a shared bus. Each line card provides connectivity to the external links. The frames arriving from the link are transferred across the shared bus to the CPU. All the forwarding decisions and queue management are performed in the CPU. The cells are then transferred across the bus again from the CPU to their outgoing line card and then onto the external link. Since all the line cards share the same CPU for the forwarding functions, this architecture is known as a shared CPU switch architecture. The advantages of this architecture are the simplicity and the flexibility of its implementation. However several limitations to this architecture exist. The central CPU must process every cell and the cell has to traverse the shared backplane twice, thus limiting the throughput of the system. These limitations gave rise to the architecture where multiple CPUs process cells in parallel. This is shown in Figure 3.3.

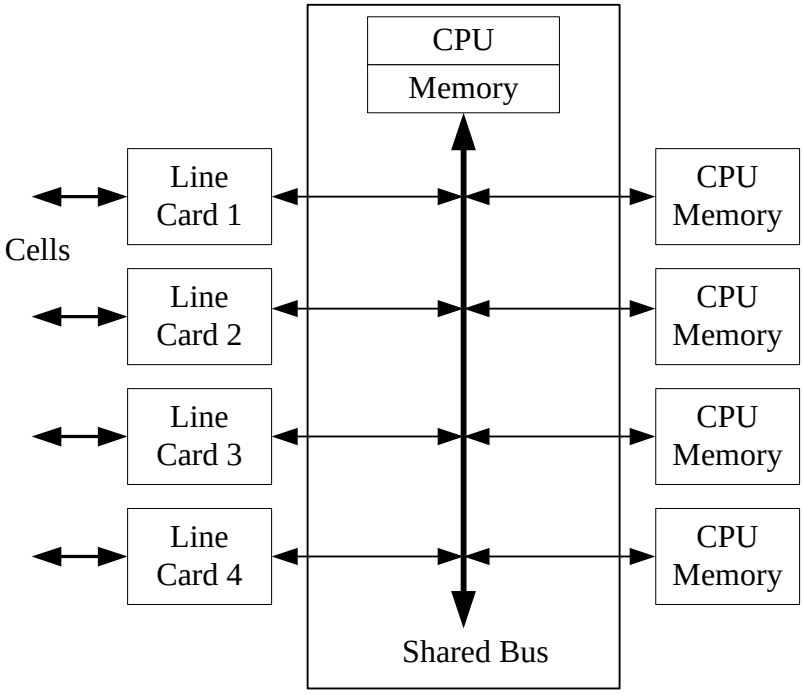

Figure 3.3 Parallel shared CPU switch architecture

In this multiple CPU architecture, incoming cells are transferred to a CPU as before; however, the cells are sent to the first available CPU, or the cells having the same destination are transferred to the same CPU. The advantage is the available parallelism and thus, the increase in system throughput as compared to shared CPU switch architecture. The disadvantage of this architecture is that the central CPU must provide routing information to every CPU and the cell has to traverse the shared backplane twice. The next architecture is shown in Figure 3.4. In this architecture, a CPU is provided at each port. A forwarding decision is made at the dedicated CPU, and the cells are transferred immediately to the outgoing interface. The central CPU is used to maintain forwarding tables to provide the forwarding decision to each separate CPU. The advantage is that the cells need to traverse through the bus only once, hence increasing the system throughput.

As separate CPUs are provided for forwarding decisions, this architecture is also known as the forwarding engine architecture. A disadvantage of this architecture is that the forwarding decisions are performed in software and are thus limited by the speed of the

CPUs. Also, the sharing of a bus limits its performance. Only one cell has the ability to transfer through the bus at a time between two line cards. Performance can be increased if multiple cells are allowed to move across the bus simultaneously. A crossbar switch has the ability to transfer multiple cells at once. This is the main reason for the crossbar switch to be used instead of the shared bus. The next section describes the crossbar as a switched backplane.

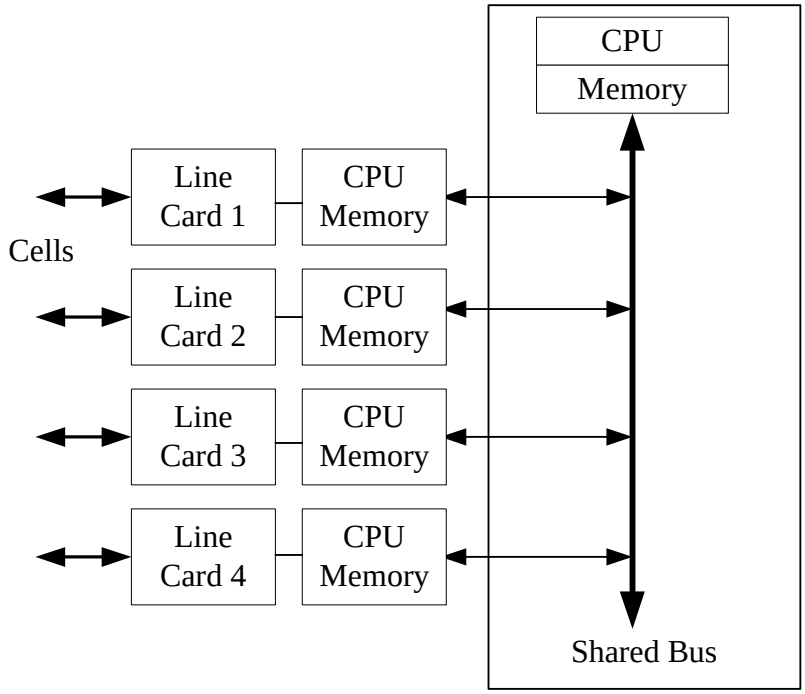

Figure 3.4 Forwarding engine switch architecture

3.3 Crossbar Switch as a Switched Backplane

In a crossbar switch, multiple line cards can communicate with each other simultaneously, thus increasing the system throughput. In the shared backplane, if the arrival rate of cells exceeds the bus bandwidth, buffers will overflow and data will be lost. Due to electrical loading limits, it is difficult to design high-speed shared busses. Today, crossbar switches are widely used for switched backplanes because of their simplicity and their power.

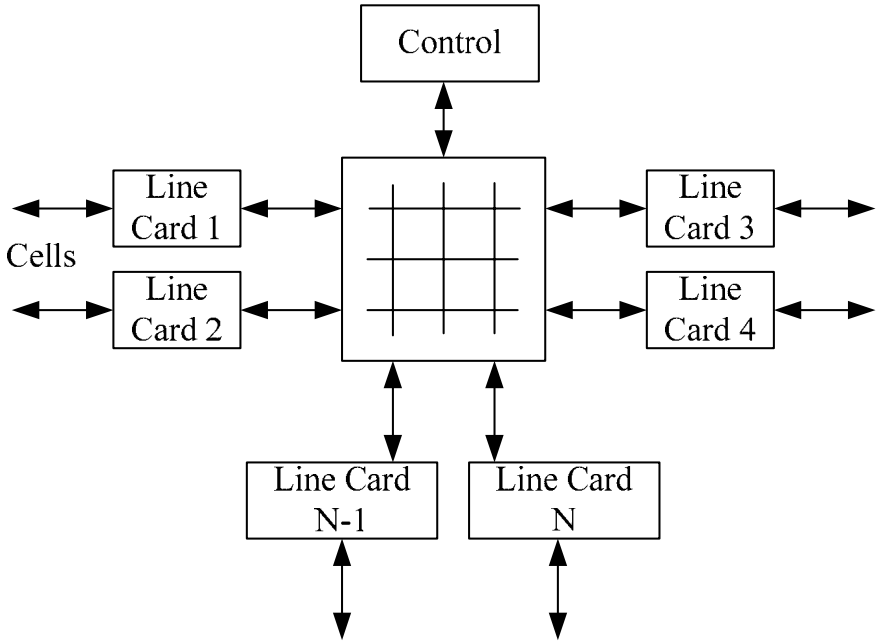

Figure 3.5 Switched backplane

Figure 3.5 shows the switched backplane comprised of multiple line cards arranged around a central crossbar switch. There are several advantages of using crossbar switch. The connections from line cards to the switch are simple point-to-point links. With the advancement of CMOS technology and serial link design, point to point serial links operate at very high speed beyond 10 Gbps. Also, the crossbar switch supports multiple bus transaction simultaneously, thus increasing the aggregate bandwidth of the system. By closing several crosspoints at the same time, the switch has the ability to transfer frames between multiple ports at once. Crossbar switches are internally non-blocking as they allow all inputs and outputs to transfer frames simultaneously. Different architectures which use crossbars are described below.

3.4 Crossbar Switch Architectures

There are different ways to classify crossbar switch architectures. Some of the criteria are blocking vs. non-blocking, buffering strategy like input-buffered, output-buffered or combined. Some other criteria are lossy vs. lossless, single-stage vs. multi-stage, buffer

implementation like partitioned, grouped and shared and also time or space divided crossbar switch. For the purpose of this thesis queuing discipline, being the determining factor for the switch performance will be discussed in detail. The blocking and non-blocking switch architecture is described briefly and then queuing architectures are described.

3.4.1 Blocking and Non-Blocking Switch Architectures

Two forms of blocking can be distinguished in cell switch architectures. When two or more than two cells arrive for the same destination at the ingress port, only one can be forwarded to the egress port and all other cells must wait. The switch has buffers to accommodate the cells that are not forwarded immediately. This form of blocking is called output contention. The second form of blocking is associated with the internal structure of the switch (e.g. in Banyan networks, multi-stage switches [12]). In a multi-stage switch, even if the cells are destined to different output ports, contention for fabric internal links occurs. This blocking property has an undesirable effect on the performance of the switch. Non-blocking switch architectures fulfill the requirements that an arriving cell destined for a particular egress port to which no other cells are destined can be forwarded immediately without considering the destination of all other arriving cells.

3.4.2 Buffer Placement

The queue or buffer is used to resolve output contention by delaying some frames. The placement of these buffers is of great importance for the overall switch performance. Two queuing solutions, output queuing and input queuing, are described:

A switch belongs to the class of output queued switches if the buffering function is performed after the routing function. Figure 3.6 shows an example of output queuing with FIFO queues. The buffers are placed at the switch fabric outputs. Theoretically, output queuing offers maximum, ideal performance as it does not suffer from head of line blocking and input contention. Head of line blocking is the presence of blocked cells at the head of an input queue which prevents any other cell behind it to be forwarded to the idle egress port.

Output queued switches, having queues of infinite size, offer the best performance in terms of both throughput and delay [13]. However, output queuing comes at a significant cost.

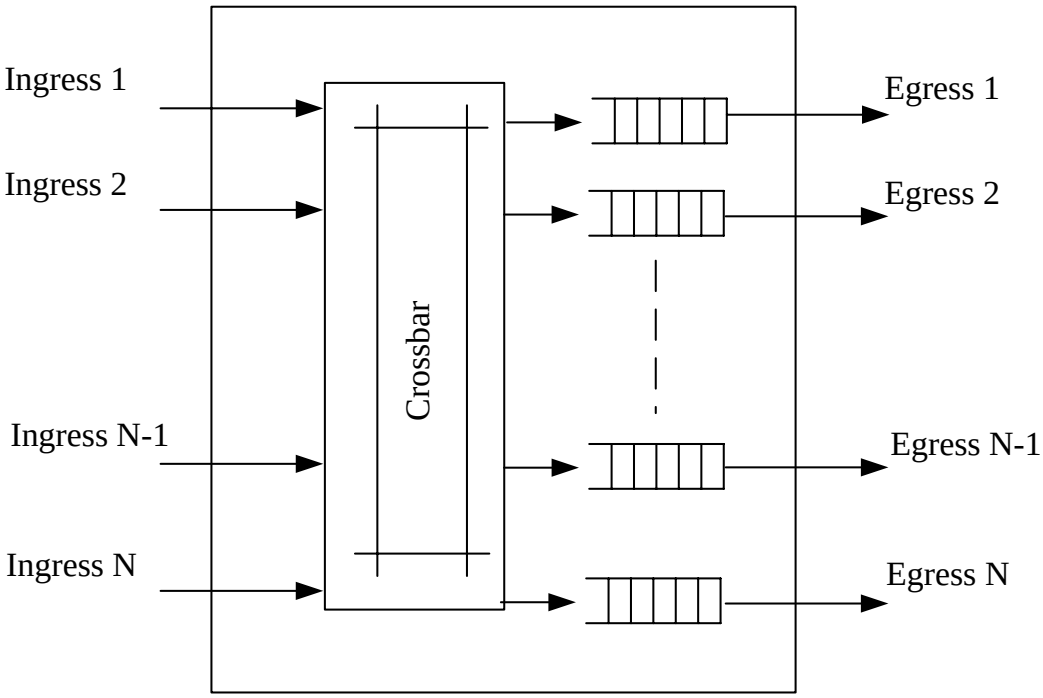

Figure 3.6 Output queuing architecture

The bandwidth requirement on a single buffer is proportional to both the speed of the port and to the number of input ports. In one cell cycle, *N* cells may arrive destined for the same output, hence *N* writes and one read are required. The bandwidth requirement equals *(N+1) R* per output queue, where *N* is the number of input ports and *R* is the port speed. The aggregate bandwidth equals *N(N+1)R,* which is quadratic in the number of ports. Output queuing is less scalable to more and faster ports than input queuing. Examples of output queued switch architectures include the Knockout [14], the Gauss [15] and the ATOM switch [16].

A switch belongs to the class of input queued switches if the buffering function is performed before the routing function [13]. Figure 3.7 shows input queuing with FIFO queues. The buffers are placed at the switch fabric inputs and the incoming cells which cannot be forwarded immediately are stored in these buffers. Only one read and one write

operation are required per cell cycle on each buffer. Thus, the bandwidth requirement in input queued switches on each buffer is only proportional to the port speed and not to the number of switch ports. If *N* is the number of input ports and *R* is the port speed, the bandwidth required is *2R* per buffer. The aggregate bandwidth equals *2NR* for a switch of size having *N* ingress and *N* egress ports.

The output contention present in the input queuing architecture is controlled by an arbiter unit as shown in Figure 3.7. In each cell cycle, a selection is made from the head-of-queue cells to be forwarded to the outputs. Only one (or zero) cell is forwarded to any single output and only one (or zero) cell is forwarded from any input. The selection policy in the arbiter must provide fairness among inputs and outputs. The arbiter is discussed in detail in the next chapter.

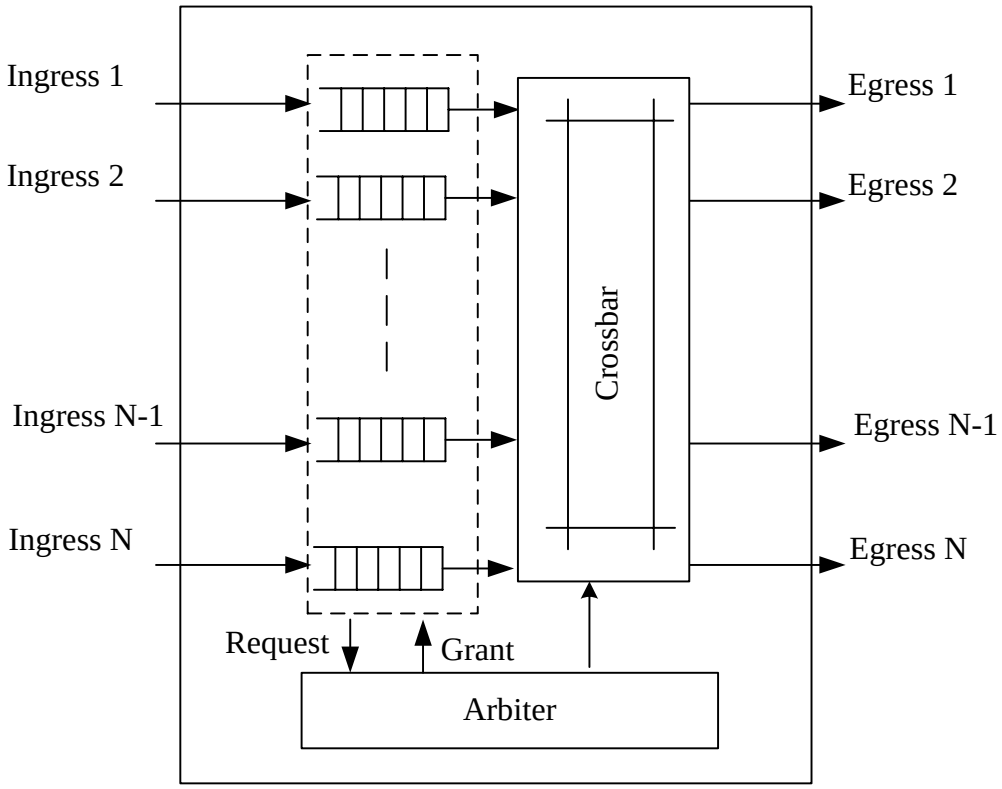

Figure 3.7 Input queuing architecture

Once the selection is made, the selected cells are removed from their buffers and are forwarded to the output through the crossbar. In an input queued switch with FIFO queues, the cell which is ready to be forwarded is the Head-of-Line (HoL) cell. Several approaches like random selection, round-robin selection, longest queue selection, least recently used selection have been proposed [17]. However, the throughput of these input queuing switches is low. This is due to the presence of blocked cells at the head of an input queue which prevents any other cell behind it to be forwarded to the idle egress port. HoL blocking is shown in Figure 3.8, where the HoL cell from ingress port *N* is sent to egress port 3, as shown by a dashed line. However egress port 2 being idle cannot be serviced by ingress port 1 because of presence of HoL cell destined to egress port 3, as shown by dotted line.

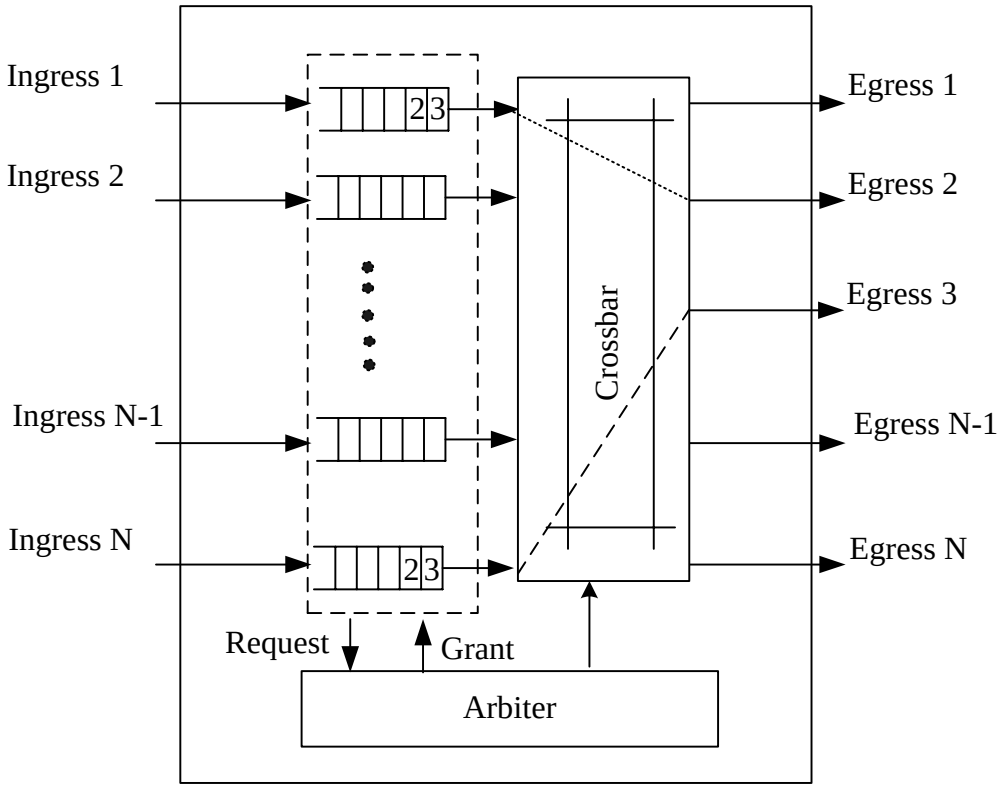

Figure 3.8 Input queuing showing head of line blocking

In the input and output queued switch architecture, each input port and output port has been allocated a fixed amount of buffer space. This may lead to the case where some ports are heavily loaded where others are idle. Using a shared queue, heavy loaded ports

benefit from getting more resources. With a shared queue, buffer resources in a switch fabric are grouped together. The sharing concept is effective in reducing cell loss rates. The other advantage of shared output queuing over dedicated output queuing architecture is that the aggregate bandwidth equals *2NR* which is the same as input queued architecture, instead of *N(N+1)R*. However, it can lead to unfairness where cells from one particular input or destined to one particular output can dominate the shared resources and hence cause performance degradation [18] [19].

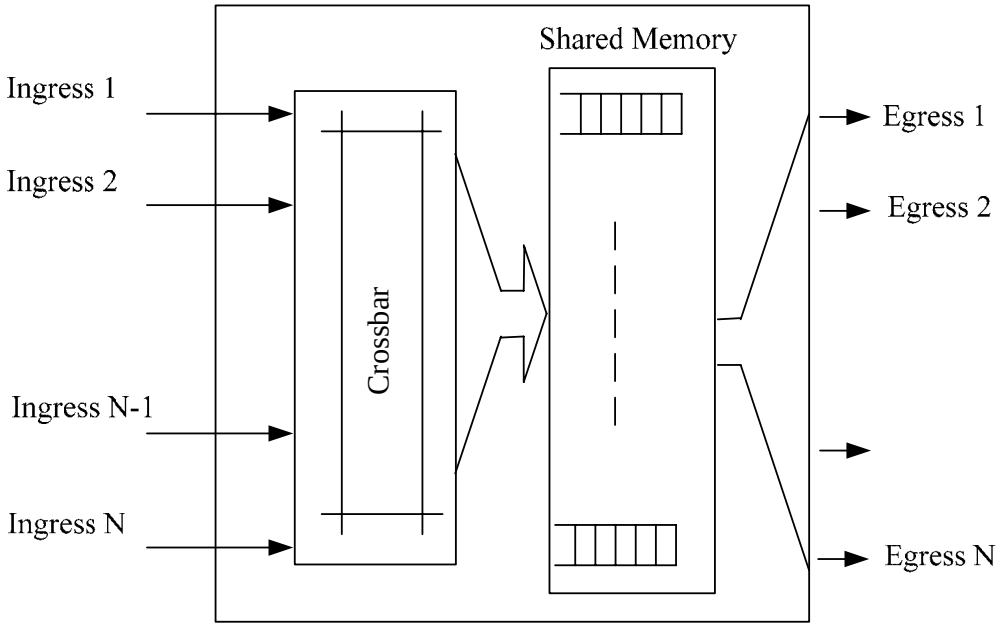

Figure 3.9 Shared memory output queuing architecture

The shared memory output queuing architecture is shown in Figure 3.9. The implementation of shared output queuing is a great challenge. In input queued switches, the memory is distributed over *N* inputs and is also physically separated. However, in the shared-memory switch, the entire bandwidth is funneled through a single memory. This is the main drawback of shared memory architecture. Several approaches using wide memory, interleaved memory banks, pipelining memory or combinations of these have been employed to obtain the desired bandwidth [20].

3.5 Virtual Output Queuing

The primary limitation of output queued and shared memory switches are an inadequate scalability due to the memory bandwidth constraints. The limitation present in the input queued switch is HoL blocking. Virtual output queuing (VOQ) [21] is used for the elimination of HoL blocking. In VOQ, at each input, a separate queue is maintained for each output. Figure 3.10 shows the input queued VOQ switch architecture.

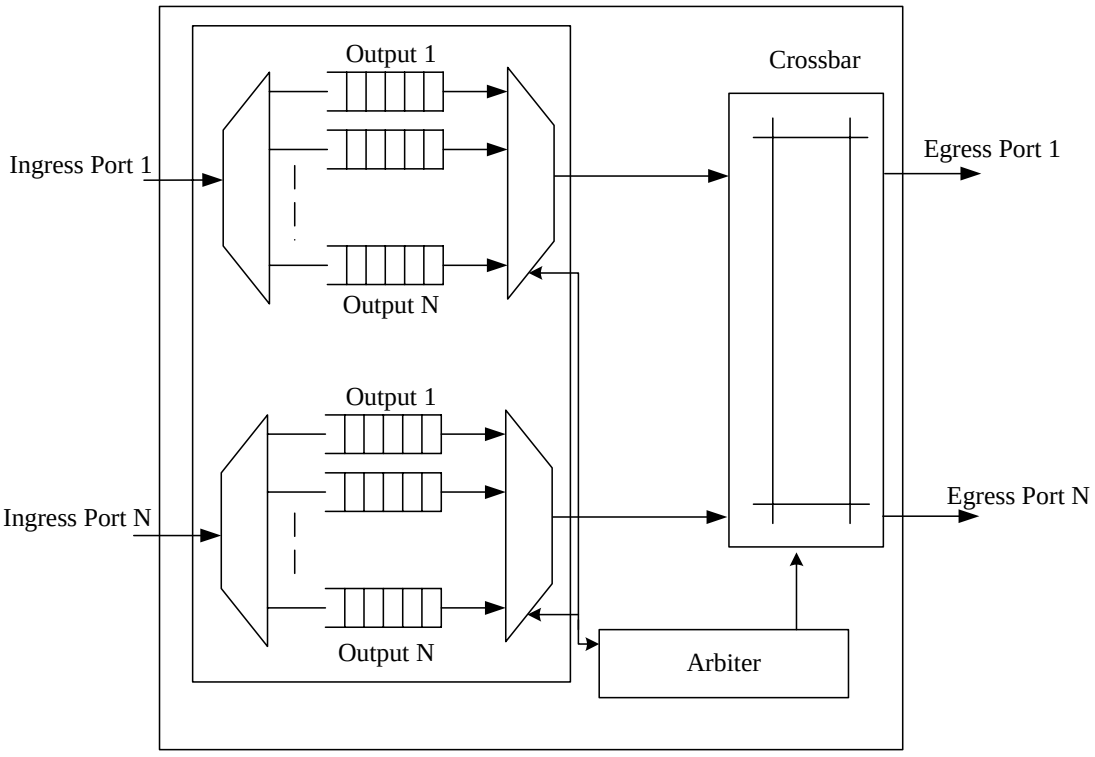

Figure 3.10 Virtual output queuing (VOQ) architecture

For N ingress ports, N egress ports, and for C different classes, the total number of VOQs required at the input side is N^2C, with NCs queues at each ingress port. Due to the presence of N^2C HoL cells instead of just N cells, the HoL blocking problem is eliminated. However, the problem of selecting which cells are to be transmitted becomes more complex. The required arbitration algorithm actually determines the performance of the VOQ architecture. The arbitration algorithm is described in detail in the next chapter.

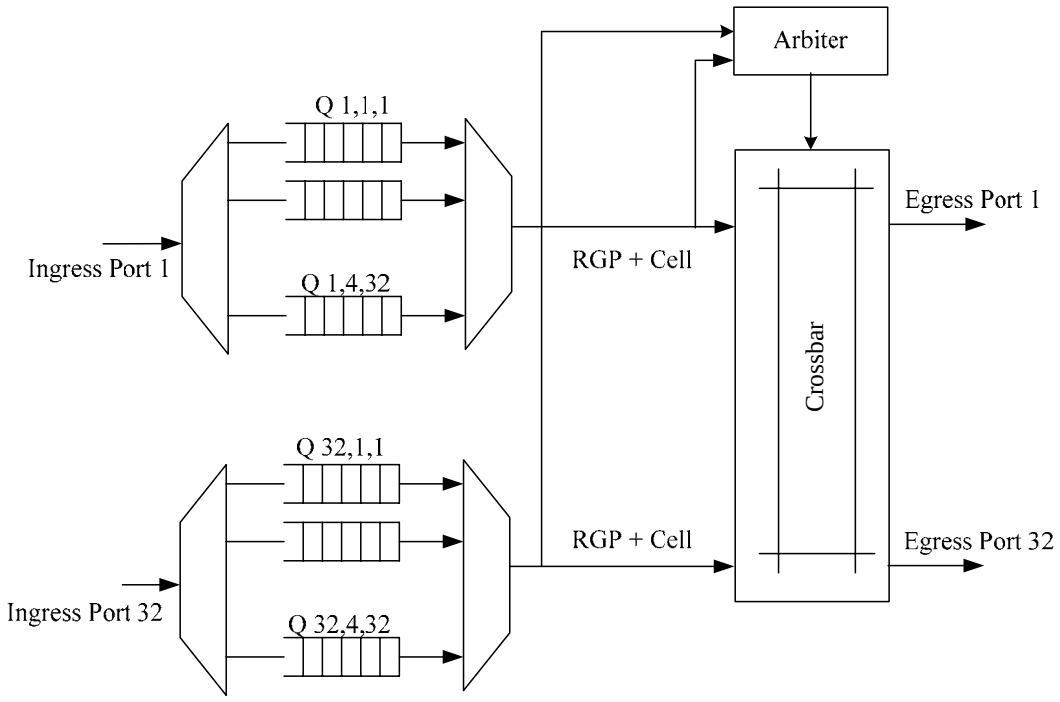

Figure 3.11 GCT Switch architecture using request grant protocol

The designed GCT switch uses ingress queuing with virtual output queues. It has the advantage that the queues are distributed over the port devices. This avoids placing of queues in the switch core and thus the problem associated with scaling due to the limitation in VLSI area cost is eliminated. The GCT protocol has four classes. The designed GCT switch has 32 ingress and 32 egress ports. With the VOQ, the queues are partitioned according to the number of classes and the number of egress port pair, hence, the total number of queues required for each port is 128. All non-empty ingress queues communicate with the switch core to make a bid for the cells to transfer through the crossbar to the destined egress port. The switch core consists of the arbiter and the crossbar. The communication of the queues residing in the ports to the switch core is done by a control protocol known as the request grant protocol. The switch architecture using request grant switch protocol is shown in Figure 3.11.

In order for the arbiter to make sensible scheduling decisions, the arbiter must be provided with the status of the queues at the ingress ports. The simplest solution is to provide the arbiter with the count of the number of cells present in each queue rather than the queue itself. The request grant protocol maintains the counts on the switch core. In each cell cycle, the ingress port may receive a cell. With the information from the forwarding engine, the cell is placed into a queue as per class and egress port. At the same time, a request is sent to the arbiter. The request contains information about the queue which indicates the presence of a cell waiting to go to the destined egress port with the particular class. On receiving the request message, a counter associated with that queue is incremented. For each cell transfer time, the arbiter examines the set of all queue counters and selects a set of ingress/egress connections to schedule for the next cell transfer time. The winning queue counters in the switch core are decremented to reflect the true depth of the cells present in the ingress port. The winning ingress port and the queue in that particular port are informed. This is accomplished by the grant message from the switch core to the ingress ports. Once the ingress port receives the grant, the winning HoL cell is immediately forwarded to the switch core. The switch core, with the crossbar settings information obtained from the arbiter, sets the crossbar to the correct connection to cause each incoming cell to be forwarded to the appropriate egress port. The request and grant messages are transferred as a part of the header in the cell.

3.6 Summary

This chapter describes the variety of single-stage switch architectures. First the disadvantages of shared backplane were described and the advantages of the crossbar switch as a backplane were discussed. The various switch architectures based on the queuing discipline with the crossbar as a backplane are then discussed. The ingress queuing, VOQ architecture was chosen for the GCT switch. The main advantage of this architecture was the queues reside on the port cards and the switch core resides on a separate chip consisting of the crossbar and the arbiter. The request grant protocol that is used to make communications between switch core and the port card is also discussed.

Chapter 4
ARBITER

4.1 Background

A typical communication switch includes N input ports, N output ports, an N X N crossbar switching mechanism and buffer memory. As discussed in the previous chapter, the VOQ switch receives cells arriving at its ingress ports and places them in a FIFO queue associated with each ingress port. A forwarding decision is made to place the arriving cell in the appropriate queue, depending on the cells outgoing port and class. Each such arrival event causes a corresponding request message to be sent to the arbiter. These request messages communicate the ingress and egress port number and the class of the newly arrived cell. For each cell time, each ingress port and each egress port can only be used for one cell. A switch must have an arbitration system for determining the order in which requests are granted in order to resolve conflicting resource demands and to provide efficient and fair scheduling of the available resources.

The goal of this chapter is to develop an arbiter architecture that provides QoS provisioning, fairness among all ingress and egress ports and behaves robustly under different traffic patterns. Firstly, a brief introduction to the arbiter is given in Section 4.2. Then in Section 4.3, some of the scheduling algorithms are briefly discussed. Section 4.4 describes about our wavefront arbiter strategy. Next in Section 4.5, a wrapped wavefront arbiter is described. Arbitration bias and a solution to the bias are presented in Section 4.6. Section 4.7 describes the random number generation unit and Section 4.8 briefly describes the arbiter solution for multiple classes. Finally, a summary of the chapter is given in Section 4.9.

4.2 Introduction

An arbitration system is key to the performance, scalability and the hardware complexity for the switch. As shown in Figure 4.1 in the Request-Grant switch protocol, the arbitration system includes an arbiter that receives connection request from the ingress ports. It monitors the status of the request counts and determines the order in which pending requests are granted. Arbiters that resolve the conflicts and provide efficient and fair scheduling of the available resources are critical for providing the maximum possible performance for a given switch. Once the requests are granted by the arbiter, acknowledgements are transmitted to the ingress ports in the form of grants and, at the same time control data are transmitted to the switching core. The request count is also decreased according to the grant value obtained from the arbiter. The switch uses the control data to make the desired connection between the ingress and the egress ports for the next cell transfer time. After receiving the acknowledgement, the ingress port transmits the data to the egress port through the switch core.

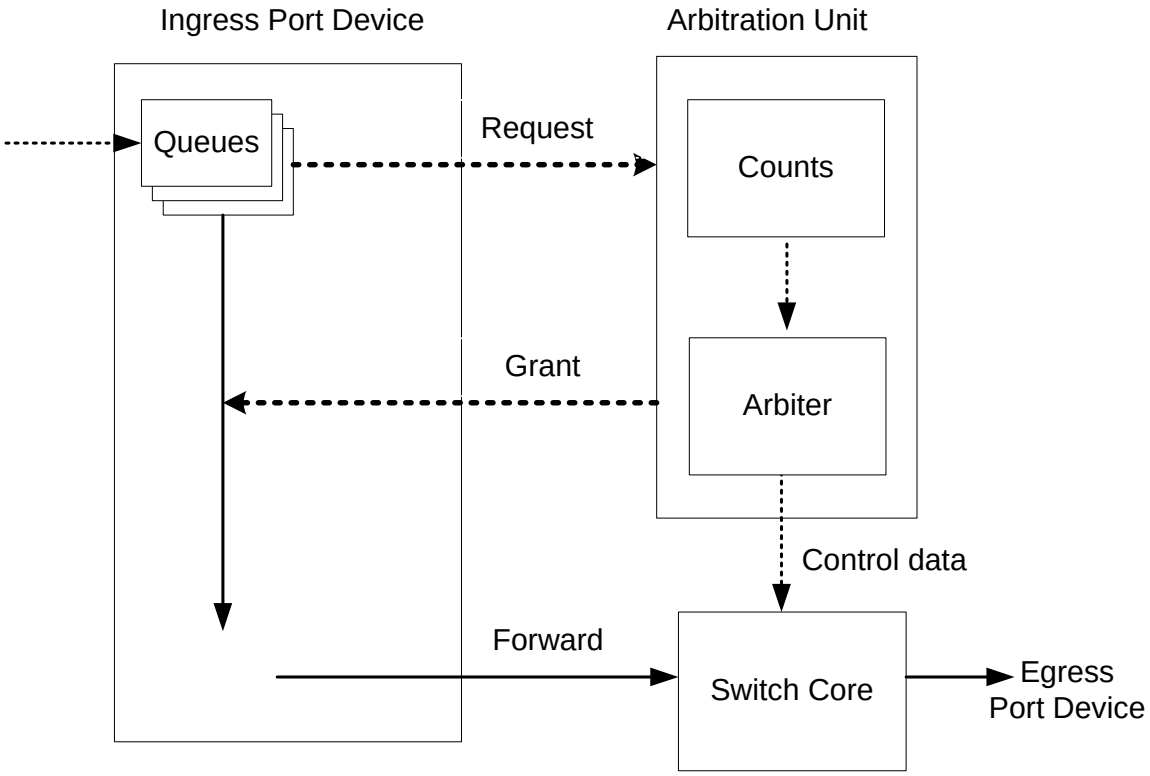

Figure 4.1 Request-Grant Switch Protocol

4.3 Literature Review

Numerous arbitration algorithms have been proposed for VOQ switches including PIM, iSLIP, RPA, FIRM, and DPRM [22], [23], [24], [25] and [26]. These arbitration algorithms iterate the following steps until no more requests can be accepted for a given number of iterations [27]:

1. *Request*: Each unmatched input sends a request to every output for which it has a queued cell.
2. *Grant (outputs)*: If an unmatched output receives any request, it grants one by selecting (in some fashion) a request over all requests.
3. *Accept (inputs)*: If an unmatched input receives grants, it accepts one by selecting (in some fashion) an output randomly among those grants to this input.

The major differences among the various algorithms regard the way in which the outputs choose which input to grant, and in the way in which inputs choose which grant to accept. These differences play an important role in the performance of the switch. A wavefront arbiter (WFA) [28] which uses a parallel arbitration algorithm has been used in the design and is described in detail in the following section.

4.4 Wavefront Arbiter

The WFA consists of arrays of small processing elements for the implementation of the parallel arbitration algorithm. Figure 4.2 shows an abstract view of a simple wavefront array for a symmetric four-port switch. The array has one row for each ingress port and one column for each egress port. The computation process begins from the top left of the array. The arbitration configuration is like a wavefront that moves diagonally from top left to the bottom right corner across the array. The arrows in the left side indicate the ingress port corresponding to each row, which is hunting for the idle egress to form a connection for the next cell transfer time.

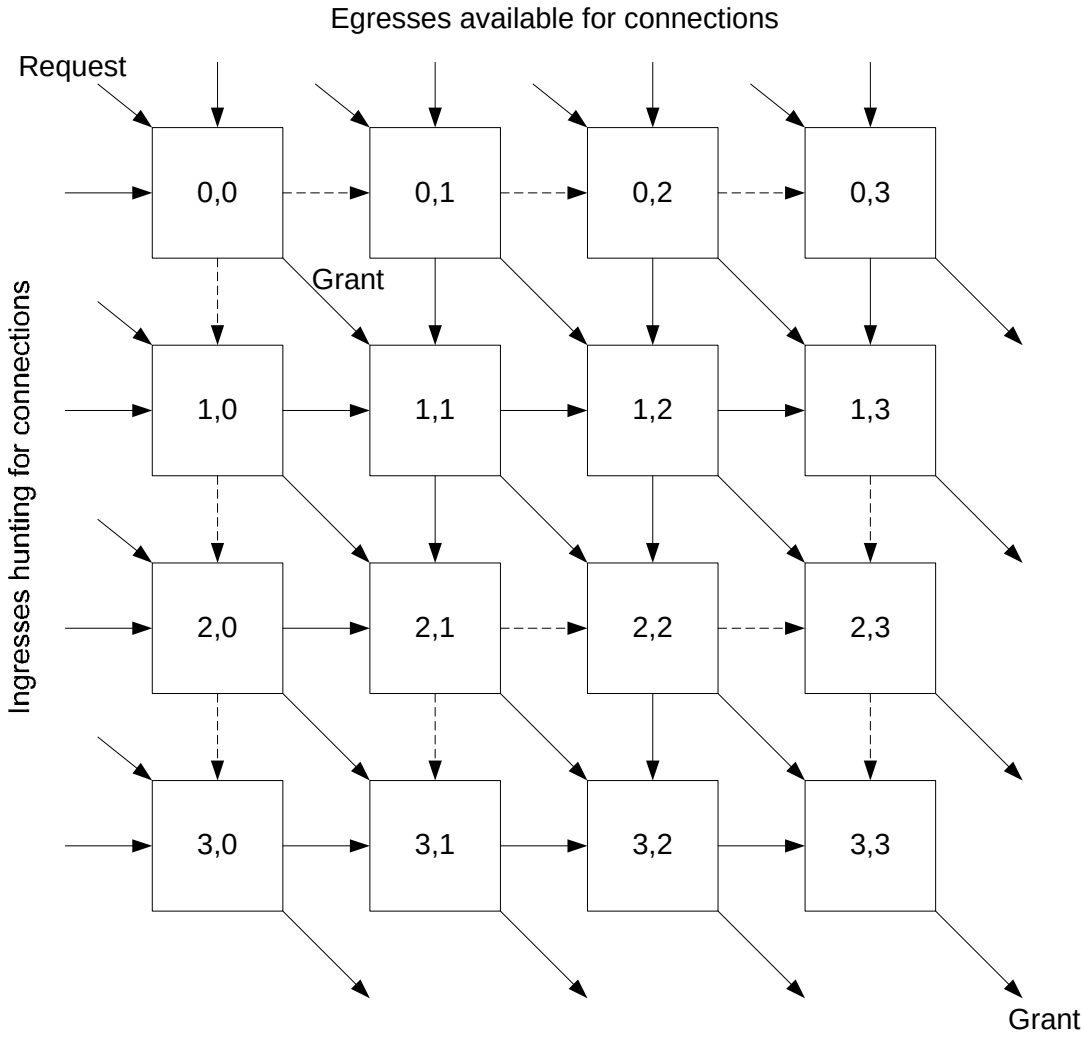

Figure 4.2 Conceptual wavefront arbiter [1]

Similarly, the arrows in the top side indicate the egress port corresponding to each column, which is available for any ingress port that is still hunting for the egress port. Each row-column arbitration cell in the wavefront array attempts to match a hunting ingress with an available egress, provided that there is request for that particular cell. When such a match is made, the cell passes the hunting signal with a value 0 further to the right and the available signal with the value 0 further down the column. This reflects that the ingress corresponding to that row is no longer hunting and the egress corresponding to the column is no longer available. However, if the conditions for the successful match are not met, any received

hunting and available signals are passed with their initial value to the next right cell and to the next bottom cell respectively.

Figure 4.3 shows the logic for each cell in the wavefront array. Each cell of the wavefront array computes its result only when all request, hunting and available signal are present. In absence of a request signal, both hunting and available signals are passed unchanged to the right and to the bottom of the cell.

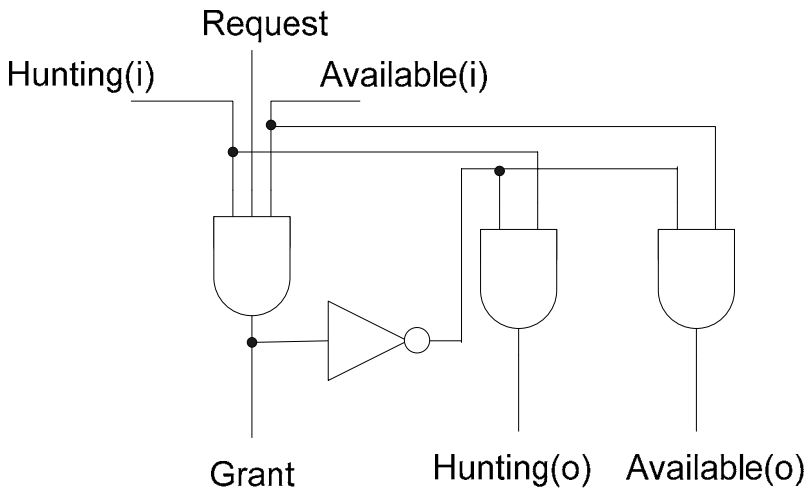

Figure 4.3 Wavefront arbiter cell logic

In the Figure 4.2, the incoming diagonal arrow shows the request and the diagonal arrow that is leaving the cell represents the grant. Let the request coming be given by the matrix as below:

		Egress			
		0	1	2	3
Ingress	0	1	0	0	0
	1	0	0	0	1
	2	0	1	0	0
	3	0	0	0	1

This represents the ingress ports 0, 1, 2 and 3 requesting the egress ports 0, 3, 1 and 3 respectively. Since cell (0, 0) receives hunting, available, and request, it produces a true grant result. This indicates that ingress 0 and egress 0 are matched and for the next cell time a connection will be made between ingress 0 and egress 0. This cell output available and hunting signals with the value 0. This is shown by broken arrows in Figure 4.2. Next, cells (0, 1) and (1, 0) are processed. Since there is no request coming in, they pass the available and the hunting signal unchanged to the right and to the bottom cell. Next cells (0, 2), (1, 1) and (2, 0) are processed. Cell (1, 1) has both a hunting signal and the available signal; however it does not have the request signal. Cells (0, 2) and (2, 0) do not have high request signals. These cells pass their received hunting and available signals unchanged to the right and to the bottom cells respectively. When (0, 3), (1, 2), (2, 1) and (3, 0) are processed next, the same phenomenon happened for cells (0, 3), (1, 2) and (3, 0). They again pass their hunting and available signals unchanged to the next available right and bottom cells. But there is a request for cell (2, 1). Due to the availability of true request, hunting and available signals, cell (2, 1) also issues a high grant signal. The output hunting signal and available signal to the right and down from the cell (2, 1) are 0.

Next cells (1, 3), (2, 2) and (3, 1) are processed. Since (2, 2) and (3, 1) do not have high requests coming in, they pass their hunting and the available signals unchanged to the next available right and bottom cells. However, cell (1, 3) has request, hunting and available signals all high, hence it also issues a high grant signal. The available signal to cell (2, 3) is then 0. These processes continue till we reach cell (3, 3). Cell (3, 3) has both the request signal and the hunting signal high. However, available signal is 0, hence cell (3, 3) does not issue a grant value. Hence we have a grant for cells (0, 0), (2, 1) and (1, 3). The grant matrix is seen as below:

		Egress			
		0	1	2	3
Ingress	0	1	0	0	0
	1	0	0	0	1
	2	0	1	0	0
	3	0	0	0	0

This represents ingress ports 0, 1 and 2 granting connections to the corresponding egress ports 0, 3 and 1. Thus a connection between these ingress and egress pair will be made for the next cell transfer time.

If T is the time taken by each cell to compute, then a total of $(2N - 1)T$ time is required for the computation for the whole $N \; X \; N$ wavefront array. There exists a variant of wavefront array that computes the whole wavefront array in NT time. The following section describes the wrapped wavefront arbiter (WWFA) [28], which is capable of computing the whole wavefront array in NT time.

4.5 Wrapped Wavefront Arbiter

In the first stage of the arbitration, only one cell is processed with the wavefront array. In our previous example, only cell (0, 0) is processed at first, and a maximum of only one grant can be issued at that stage. However, it is possible to compute N cells which are guaranteed not to conflict. The N cells of any wrapped diagonal of the wavefront array are guaranteed not to conflict as they are all on different rows and different columns. Hence a maximum of N grants can be obtained in the first stage. For our example, cells (0, 0), (3, 1), (2, 2) and (1, 3) are processed at first time stage. The wrapped wavefront arbiter is shown in Figure 4.4. In a total of four time stages, all the cells will be processed. The process of passing the hunting and the available signals is similar to the wavefront array. At the end of first time stage, the hunting signals from the cells (0, 0), (3, 1), (2, 2) and (1, 3) are passed to cells (0, 1), (3, 2), (2, 3) and (1, 0) respectively. Similarly, the available signals from these cells are passed to cells (1, 0), (0, 1), (3, 2) and (2, 3) respectively. The problem associated with this arbiter is the arbitration bias it possesses. In the next section we describe the arbitration bias problem, and the solution that has been used in the design of this switch arbiter.

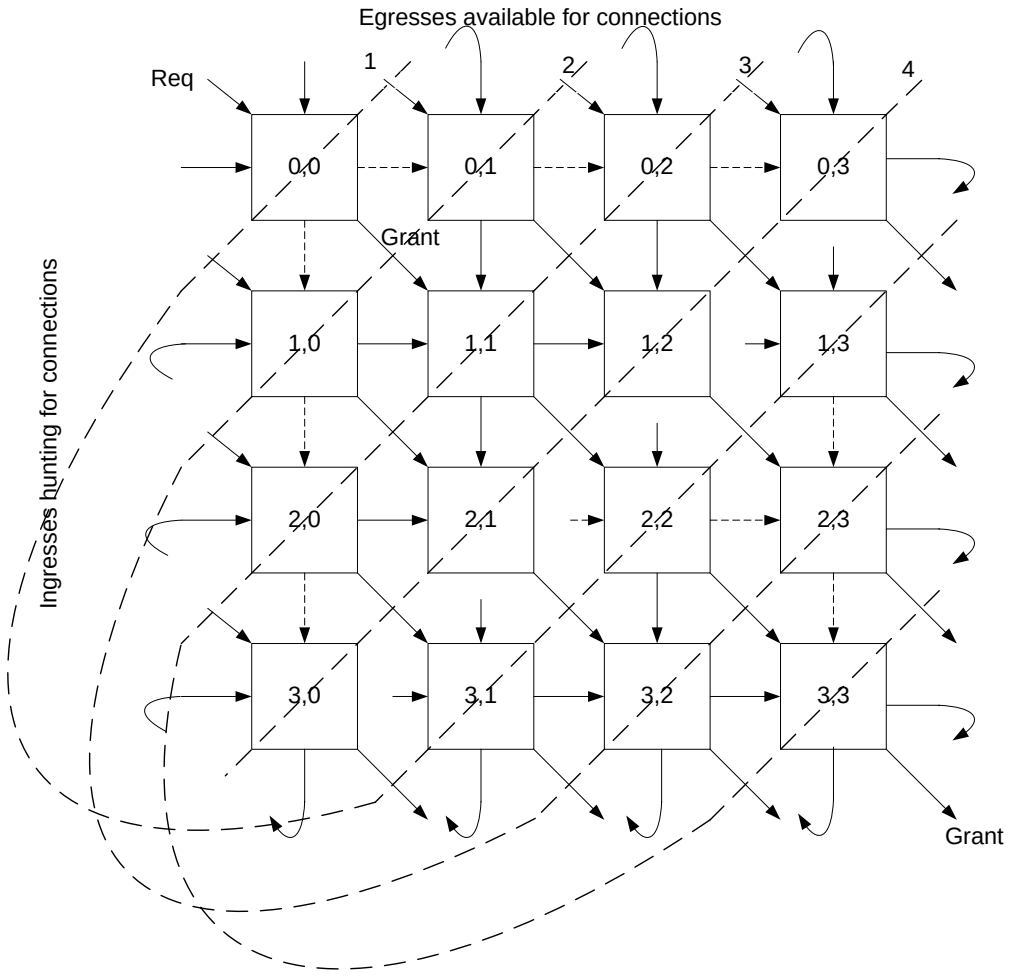

Figure 4.4 Wrapped Wavefront Arbiter

4.6 Arbitration Bias

In order to provide high efficiency and throughput, an arbiter must be fair in the sense that it should not favor one source or destination over another source or destination. In the wavefront array, each ingress and each egress are locked in some particular position in the rows and columns, which determines its relative priority. Ingress zero always has the first chance to get any egress. Similarly egress zero is the first path that will be found by any ingress that is hunting. A serious bias is thus present that always gives the best service to port zero.

The solution to this problem is to use additional hardware that provides a shuffling scheme. The shuffling scheme permutes the row and the column ordering of the wavefront array from one cell transfer time to another cell transfer time. The shuffling scheme discussed here is used for both row and column respectively before the arbiter. The de-shuffling scheme is then applied to column and row respectively just after the arbiter. The input to the shufflers can be considered as a two-dimensional matrix. Similar to the WFA, each row is the request from the particular ingress and each column being the request for the particular egress. Taking the example for the symmetrical four port switches, the request matrix might be

Ingress \ Egress	0	1	2	3
0	1	0	1	1
1	1	1	0	1
2	1	0	1	0
3	0	1	0	1

This represents ingress port 0 requesting egress ports 0, 2 and 3; ingress port 1 requesting egress ports 0, 1 and 3; ingress port 2 requesting egress ports 0 and 2; and ingress port 3 requesting egress ports 1 and 3. Without the use of shuffling scheme, the wrapped wavefront arbiter generates the following grant matrix:

Ingress \ Egress	0	1	2	3
0	1	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0

This shows that ingress ports 0, 1, 2 and 3 have the fixed bias to make connections to egress port 0, 3, 2 and 1 irrespective to the other requests. However, with the use of the shuffling scheme, the fixed bias no longer holds.

The shufflers work first by rearranging the rows. For the shuffler with control values [1 0 3 2], the requests after row permutation look like this

Ingress \ Egress	0	1	2	3
0	1	1	0	1
1	1	0	1	1
2	0	1	0	1
3	1	0	1	0

After permutation of the columns with the same control values, the permuted requests look like this

Ingress \ Egress	0	1	2	3
0	1	1	1	0
1	0	1	1	1
2	1	0	1	0
3	0	1	0	1

This is the shuffled ingress requests to the arbiter which represents ingress port 0 requesting egress ports 0, 1 and 2; ingress port 1 requesting egress ports 1, 2 and 3; ingress port 2 requesting egress ports 0 and 2; and ingress port 3 requesting egress ports 1 and 3.

The wrapped wavefront arbiter then uses these shuffled requests and generates the shuffled grants as follows.

Ingress \ Egress	0	1	2	3
0	1	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0

The shuffled grants represent ingress ports 0, 1, 2 and 3 matched to the egress ports 0, 3, 2 and 1. The grants are de-shuffled in the same way as the requests are shuffled but with an inverse arrangement. First the inverse permutations of the columns with the de-shuffling control value [1 0 3 2] is done and the inverse permuted column grants become:

		Egress 0	1	2	3
Ingress	0	0	1	0	0
	1	0	0	1	0
	2	0	0	0	1
	3	1	0	0	0

And by inverse permutation of the rows, the inverse permuted row grants become:

		Egress 0	1	2	3
Ingress	0	0	0	1	0
	1	0	1	0	0
	2	1	0	0	0
	3	0	0	0	1

The grants obtained are the final valid grants. Hence the final grants represent ingress ports 0, 1, 2 and 3 granted for connection to the egress ports 2, 1, 0 and 3 respectively for the next cell transfer time. This demonstrates that the fixed bias of ingress ports 0, 1, 2 and 3 acquiring connections to egress ports 0, 3, 2 and 1 no longer holds.

The control values to the shuffler determine the priority position for the requested ingress port and granted egress port. With different control values to the shuffler, different ingress/egress pair connections are made. The following section describes the way to generate control values to the shuffler and de- shuffler.

4.7 Random Number Generation

The control value to the shuffler is a random number generated by a random number generation unit. Different numbers are generated from one cell time to another cell time to smooth out the fixed bias present in the wrapped wavefront arbiter. The random number generation unit is divided into two units. One unit generates a sequence $(N-1)!/2$ and other unit generates a sequence $2N$. Overall the random number generation unit generates all the permutation of *N* elements which is $N! = \frac{(N-1)!}{2} * 2N$ long.

A random number sequence generator for $N = 6$ is described. This is shown in Figure 4.5. The same procedure is used to generate random number sequence for $N = 32$ as required for *32 X 32* port switches. Initially a register holds a value $[1\ 2\ 3]$. Then for $N = 4$, $N-1 = 3$ different values are generated

$$[1\ 4\ 2\ 3],\ [1\ 2\ 4\ 3],\ [1\ 2\ 3\ 4]$$

This is done by inserting 4 at each position from second position to last position. This can be seen in the Figure 4.5.

Then for $N = 5$, the same procedure is again used and a total of $N-1 = 4$ different values are generated for each of the values. Taking $[1\ 2\ 4\ 3]$ as one of the new generated value, the final resulting values may be any one of the following.

$$[1\ 5\ 2\ 4\ 3],\ [1\ 2\ 5\ 4\ 3],\ [1\ 2\ 4\ 5\ 3],\ [1\ 2\ 4\ 3\ 5]$$

Similarly for $N = 6$, the same procedure is applied again. Here a total of $N-1 = 5$ different values are obtained.

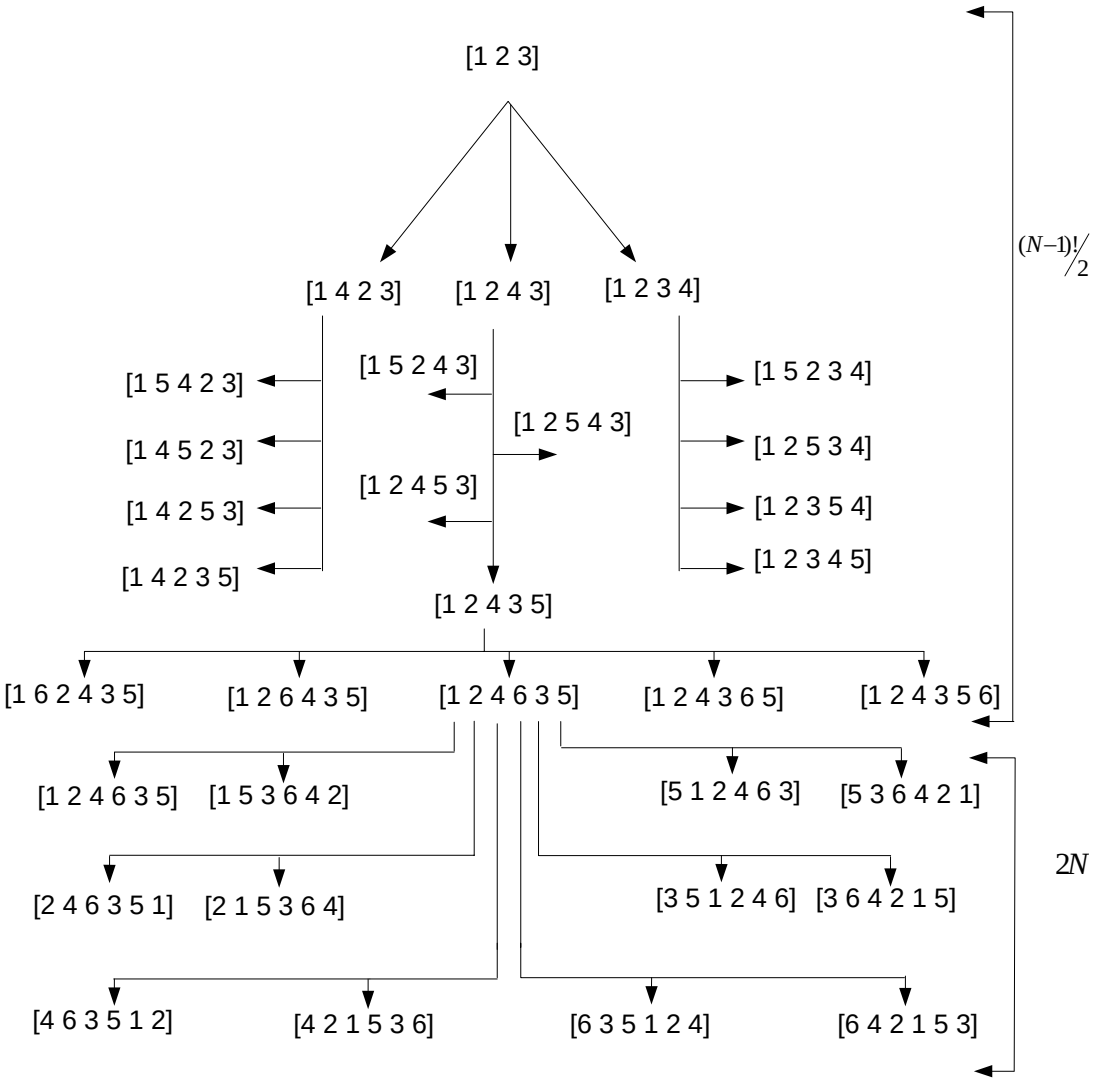

Figure 4.5 Random number generated sequence for $N = 6$.

Taking $[1\ 2\ 4\ 3\ 5]$ as the new generated value, the final resulting values may be any one of the following.

$$[1\ 6\ 2\ 4\ 3\ 5],[1\ 2\ 6\ 4\ 3\ 5],[1\ 2\ 4\ 6\ 3\ 5],[1\ 2\ 4\ 3\ 6\ 5],[1\ 2\ 4\ 3\ 5\ 6].$$

Since each one is capable of generating five different values, from twelve different possible values obtained at $N = 5$, a total of 60 different values can be generated at $N = 6$. This is equal to $(N-1)!/2$ for $N = 6$.

For generating the $2N$ sequence, two new parameters are chosen. First a position of the number is chosen according to Bruijn counter, and then from that particular position clockwise and anticlockwise direction is chosen to generate the $2N$ sequence. The 5-bit Bruijin counter is shown in Figure 4.6. The Bruijn counter can generate all zero sequence in addition to the sequence generated by maximal linear feedback shift register. The Bruijn counter virtually selects the position of each number generated from the $(N-1)!/2$ sequence. The chosen position is determined by the output of the Bruijn counter and the generated $(N-1)!/2$ sequence number. Then for each cell transfer time, first a clockwise direction is chosen and for the next immediate cell transfer time an anticlockwise direction is chosen.

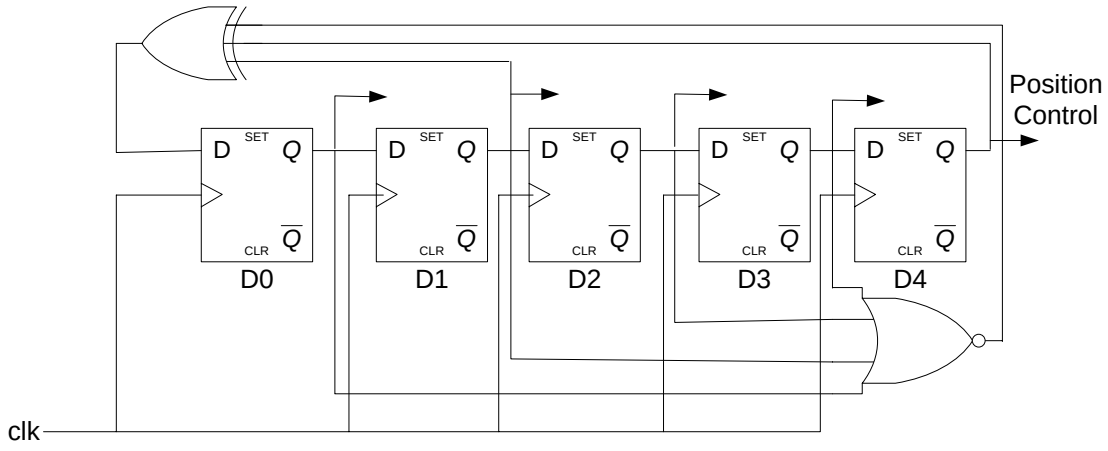

Figure 4.6 5-bit Bruijn Counter

The logic diagram for the implementation of $2N$ sequence unit is shown in Figure 4.7. The following example illustrates this phenomenon.

As shown in Figure 4.5, taking the values $[1\ 2\ 4\ 6\ 3\ 5]$, one of the positions 1, 2, 4, 6, 3, 5 is picked according to the output from the Bruijn counter. Let us suppose, position 4 is picked. For the clockwise direction, the control output to the shuffler will be $[4\ 6\ 3\ 5\ 1\ 2]$ and the next immediate control output by taking anticlockwise direction will

be $[4\ 2\ 1\ 5\ 3\ 6]$. Thus for each particular position two control values to the shuffler are obtained. For N positions, a set of $2N$ control values to the shuffler are generated.

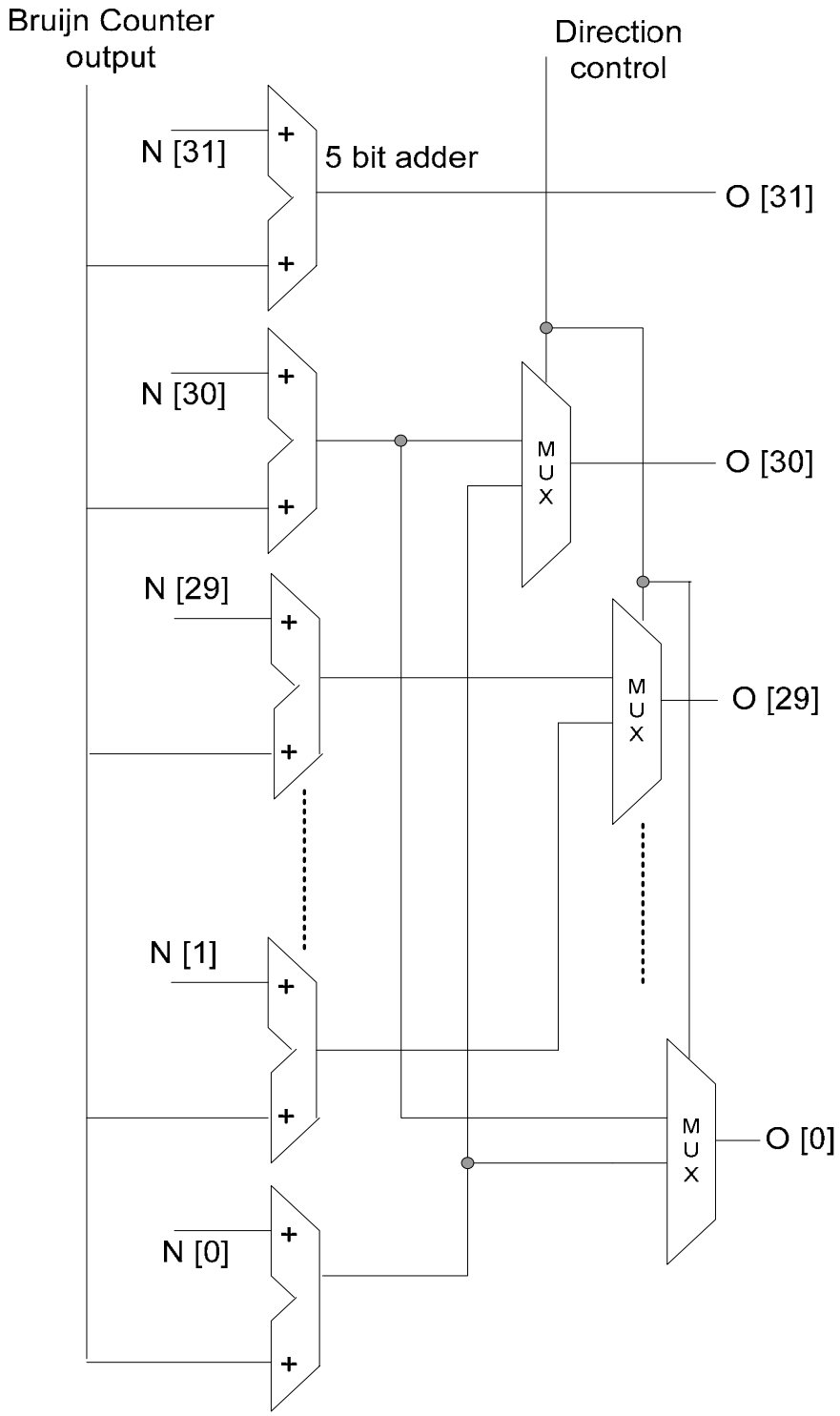

Figure 4.7 Logic for $2N$ sequence generator

Many shuffling procedures have been used in the literature. A round-robin shuffling sequence has built-in unfairness between any two pair of ports at a time. This is because in a round-robin sequence, Port zero may have higher priority than Port one, $N-1$ times, where N is the number of ports.

The advantage of this scheme is that, with in a subset of permutations, each number in a pair occurs half the time with one number first and half the time with the other number first. Taking an example for $[1\ 2\ 4\ 6\ 3\ 5]$ as shown in Figure 4.5, the number of times position 1 precedes position 2 is six. Similarly the number of times position 2 precedes the position 1 is also six. Thus, the number of times the position of any number preceding the position of another number in a pair is always equal to N within a $2N$ sequence unit. This ensures a first order of fairness.

However, taking the same example, $[1\ 2\ 4\ 6\ 3\ 5]$, in a sample of numbers with position 1 preceding the position 2, the number of times position of 3 precedes the position of 5 is four. The number of times position of 5 precedes the position of 3 is two. This does not give us the property that the other numbers, except the chosen pair, are equivalently treated fairly when one number is already preceding the other number in a chosen pair.

Hence, our random number unit uses a inner loop to generate a sequence of $2N$ and outer loop to generate a sequence of $(N-1)!/2$. A total of $N!$ sequence is generated with short term fairness.

4.8 Multiple Classes

The GCT protocol has four classes. In the above sections, arbitration has been described only for a single priority class. For multiple classes, multiple $N\ X\ N$ wrapped wavefront arrays are required, one for each priority class. The highest priority class is initialized with all ingress ports hunting and all egress ports available. The possible assignments for the top priority request are made. After completion of the arbitration for the

top priority class, some hunting signals and some available signals are still present which indicates the ingress port is still hunting for an available egress port. These hunting and available signals are then passed to the next wrapped wavefront arbiter for the next priority class. The arbiter then uses these residual hunting and available signals and possible assignments for the ingress/egress pair for the next cell transfer time are made. The same procedure is performed until all of the classes are processed.

The major disadvantages of this phenomenon are that the area of the circuit goes on increasing as the numbers of classes is increased. The time required for the arbitration process to complete also increases with increasing class depth.

In the GCT protocol, the higher class services are always given higher priority than lower class services. Also, the cell size of the GCT is large. One of the advantages of cell size being large is that it allows enough time for running the arbiter. The reduction in hardware is accomplished by using the same wrapped wavefront arbiter iteratively to cover all four classes. Additional registers and gates are used to hold the requests and grants and to pass the available and hunting signal to the next class arbitration. The same shuffled and de-shuffled control values are used for all four classes in each cell transfer time.

Synthesis was carried out by considering Altera Stratic III FPGA. The result of synthesis for random number generation has found to consist of 5085 adaptive look-up table (ALUT) and 3479 dedicated register logic. The worst case delay was found to be 5.234 ns. This delay provides sufficient time to generate a pseudo random number for each cell time. This is just a rough estimate as several fitting algorithms and timing analyzers like register packing, megafunction, netlist optimization were not considered. Similarly, on running a symmetrical four-port arbiter, 33 ALUT was found, and the worst case delay was found to be 5.261 ns.

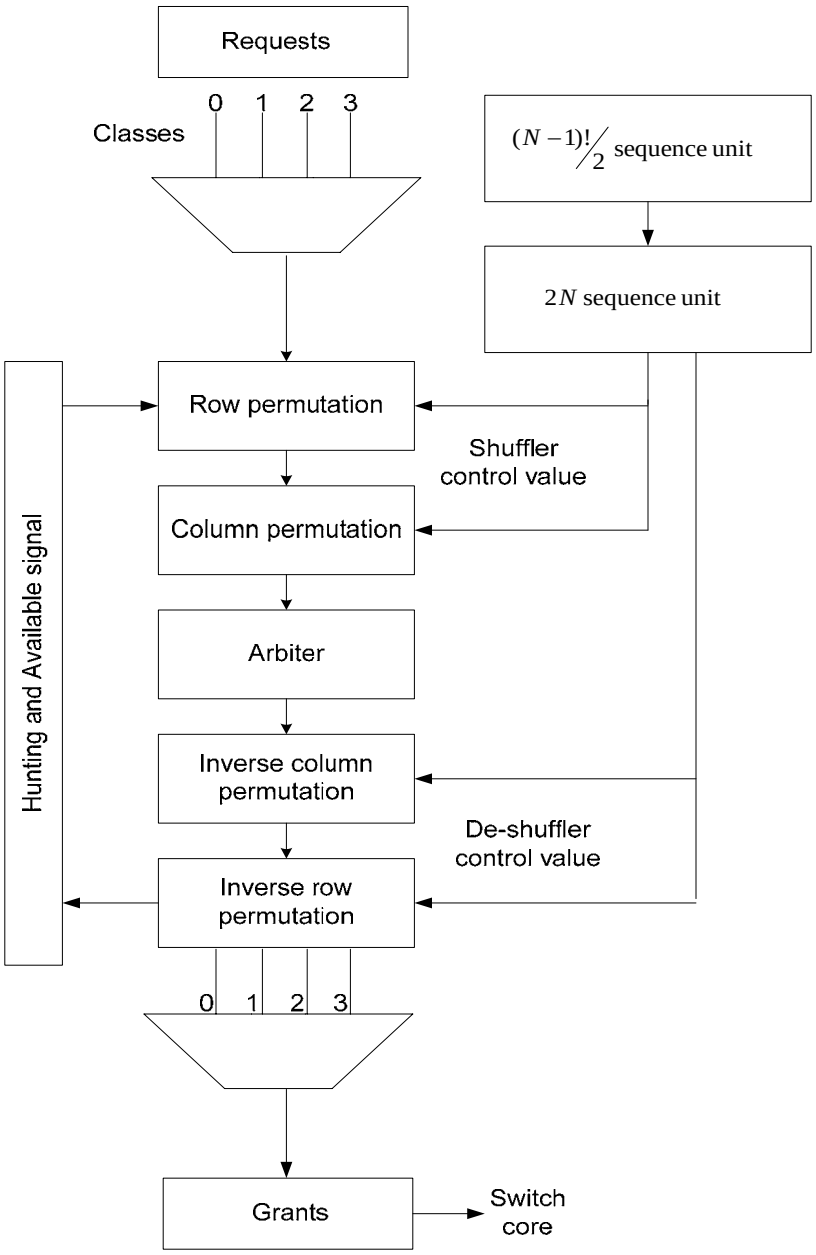

Figure 4.8 Arbitration architecture

4.9 Summary

This chapter described the arbitration method that has been used in the GCT switch. The arbiter takes the request from each destination-class pair ingress queues and makes a count of these requests. As shown in Figure 4.8, the highest priority class requests are then

feed to the wrapped wavefront arbiter. The described random number generation unit issues a control value to the shuffler and the de-shuffler. The shuffler and the de-shuffler eliminate the arbitration bias present in the wrapped wavefront arbiter. The final grants obtained from the unbiased wrapped wavefront arbiter are passed as an acknowledgement to the ingress ports through the switch core in the form of grant protocol. The switch core also receives these grants. The arbitration system uses these grants to decrease the requests count, and the switch core uses the grants to make the desired connection between the ingress and the egress ports for the next cell transfer time.

Chapter 5
PERFORMANCE EVALUATION

5.1 Introduction

Performance has different meanings for different applications. Throughput, cell/packet loss rate, latency and resource utilizations are some of the elements that determine the performance of a switch. Performance of the switch depends on the nature of the traffic used in a network. Different traffic models have been used in the literature [29]. The aim of all the models is to study the fairness, response times, queue behavior, loss probabilities and the resource utilizations under different operating conditions.

The goal of this chapter is to evaluate the performance of the switch by applying a traffic load model and to study the results. Section 5.2 will describe the traffic load generation unit. The overhead ratio of the IP traffic on moving through different stages in the GCT network is discussed in Section 5.3. Speedup factor required in the switch is also given in Section 5.3. The performance of the switch under uniform and hotspot traffic load conditions is then discussed in Section 5.4. Finally, a summary of this chapter is given in Section 5.5.

5.2 Traffic Load Generation Unit

IP is the standard network layer protocol that has been used from the desktop to the global internet. Hence, the IP traffic model is used for the simulation. The IP packet length chosen were 40 bytes, 256 bytes, 576 bytes and 1500 bytes. These limited sizes of packets approximately represent the large portion of internet backbone traffic [29]. 40 bytes represents TCP acknowledgement and control packets [30]. 256 bytes is the choice to represent shorter transmissions such as web page interaction. 576 bytes is used as maximum segment size by many applications such as IP and 1500 bytes is Maximum Transmission Unit (MTU), the characteristic packet size for Ethernet traffic [30][31].

The distributions of these packet lengths are chosen as 30% for 40 bytes, 30% for 256 bytes, 30% for 576 bytes and the remaining 10% for 1500 bytes [30]. The load generator model is designed using a built-in random number generator provided by the Verilog language in MODELSIM. This random number generator is used to generate the random number distribution. Destination, class, load and the packet length values are chosen from this random number generator. Different seeds are used to generate uncorrelated random numbers.

The distribution of the destination, where the next packet will go, is chosen as a uniform distribution. All the 32 ports of the switch used in the simulation are chosen uniformly. The idea of choosing uniform distribution was to keep both the ingress and the egress port equally busy. The distributions of the traffic by classes are chosen randomly as 5% for higher priority class 0 traffic. Class 1 is chosen as 10%, class 2 is chosen as 25% and class 3 traffic is chosen to be 60%.

In every packet cycle, a valid packet is generated with certain probability *p*, and invalid packet is generated with probability 1 - *p*. The valid packet represents the average offered input load. The simulation has been performed by choosing different values of *p*.

5.3 Overhead Ratio and Data Rate

SONET OC-192 with a data transmission rate of 9.953 Gb/s has emerged as a commonly used optical carrier signal [1]. OC-192 connections are most common for use on backbones of large Internet Service Providers. It provides low costs and additional bandwidth to accommodate broadband services such as video and multimedia. Thus SONET OC-192 has been taken into consideration to generate the traffic load for evaluating the performance of the switch designed in this thesis.

The SONET OC-192 rate includes the rate after an addition of GFP and SONET overhead to the IP packet. The equivalent IP packet length at each stage in GFP, SONET without FEC and SONET with FEC is shown in Table 5.1.

Table 5.1 Equivalent packet length at different stages in SONET network

Portion of Traffic	Original IP length (B)	GFP addition (B)	SONET addition (B)	FEC addition (B)
0.3	40	48	50	53
0.3	256	264	273	291
0.3	576	584	604	645
0.1	1500	1508	1560	1665
Wgt. avg. length	411.6	419.6	434.1	463.2
Ratio	1.00	1.02	1.05	1.12

From Table 5.1, it is clearly seen that due to the overhead possessed by the GFP and the SONET, the size of the weighted average length at different stages in the GFP and in the SONET has gone up. The original IP length is found to be increased to a factor of 1.12 in SONET after the addition of FEC. For the case of GCT we stripped off both the SONET overhead and the GFP overhead.

Table 5.2 shows the rate of the data traffic at different stages in SONET network. The data rate per second is obtained using SONET OC-192 rate of 9.953 Gb/s and the equivalent packet length at SONET OC-192 stage shown in Table 5.1. The obtained rate is then used to calculate the rate of the different length packets at different protocol level. The data rate possessed by each packet length is specified at the corresponding row of each original packet length in Table 5.2. The overall rate is also shown in the Table. From the table, it is found that due to the addition and subtraction of the overhead, the overall required data rate at each level is changing. The equivalent IP data rate after the removal of GFP overhead and SONET overhead is also shown and is found to be 9.437 Gb/s. Addition of SONET and GFP header had caused the data rate required in the switch to increase to 9.953 Gb/s. The addition of FEC to the SONET increased the required data rate to a factor of 1.067 is also shown in table.

Table 5.2 Data rate at different stages in SONET network

Original IP length	Original IP (Gbps)	GFP (Gbps)	SONET (Gbps)	SONET + FEC (Gbps)
40 B	0.275	0.330	0.341	0.364
256 B	1.761	1.816	1.879	2.004
576 B	3.962	4.017	4.156	4.434
1500 B	3.439	3.458	3.577	3.817
Overall rate (Gbps)	9.437	9.621	9.953	10.62
Ratio	0.948	0.967	1.0	1.067

The original IP packet can also be obtained after eliminating both the GFP and the SONET headers. This is the IP packet length generated by the load generator model.

For the GCT, an extra four bytes of GFP* is added to the IP packet length generated by the load generator unit. Addition of extra GFP* bytes is equivalent of stripping four bytes of header from the GFP. The length and header protection each of two bytes present in the GFP is discarded to obtain GFP*. The reason for this is the use of FEC in GCT. Thus, additional protection is not necessary at the GFP layer.

The equivalent packet length after the addition of GFP* overhead and the GCT overhead is shown in Table 5.3. As predicted, the weighted average ratio goes on increasing as we move from IP packet to scrambling GCT. This is due to the addition of overhead present in each successive stage. The ratio is found to be increased by a factor of 1.17 while moving from IP to scrambling GCT.

Table 5.3 Equivalent packet length at different stages for GCT network

Portion of Traffic	GFP removal (B)	GFP* addition (B)	GCT addition (B)	Scrambling GCT addition (B)
0.3	40	44	49	51
0.3	256	260	292	301
0.3	576	580	651	672
0.1	1500	1504	1689	1741
Wgt. avg. length	411.6	415.6	466.5	481.3
Ratio	1.00	1.01	1.13	1.17

Table 5.4 shows the rate of traffic at each stage in GCT network. After the addition of GFP* the overall data rate is found to be 9.529 Gb/s. Also shown in the table is the date rate possessed by each of the IP length traffic at different stages in the GCT network. Addition of GCT overhead with embedded FEC caused the overall data rate to increase to 10.70 Gb/s. The addition of the scrambling bit increased the overall data rate internally in the switch to 11.03 Gb/s.

Table 5.4 Data rate at different stages in GCT network

Original IP/ GFP removal	GFP* (Gbps)	GCT (No FEC) (Gbps)	GCT (FEC) (Gbps)	Scrambling GCT (Gbps)
40 B	0.302	0.319	0.339	0.350
256 B	1.788	1.883	2.008	2.071
576 B	3.990	4.200	4.480	4.620
1500 B	3.448	3.630	3.872	3.993
Overall rate (Gbps)	9.529	10.03	10.70	11.03

Segmentation is the process of breaking a packet into smaller units before transmission. Reassembling is done at the receiving end of the communication. Packets are made smaller into manageable sized ones by segmentation process. Since segmentation and reassembly is not the focus of this thesis, however, a brief description how it has been used is discussed below.

As mentioned earlier this dissertation is focused on incoming IP traffic. This incoming IP traffic from the load generation unit is segmented into manageable size. The manageable size in this dissertation is the length of GCT Cells. The segmented process used here is also modeled using the non-synthesizable Verilog code using the MODELSIM. During the segmentation process, 12 bytes of header is added for each 228 bytes of arriving data, which accounts as a payload for the GCT. The GCT cell is generated only when the total incoming bytes from the load generation unit is equal to or greater than the payload of the GCT.

Considering the traffic generated for only a single port, if 44 bytes of GFP* packet arrives from the load generation unit, this will be kept in its unique destination class pair queues unless the total bytes in this queue exceed or equal to the payload of the GCT cells, (228 bytes). In the mean time, the process will wait for the time equivalent to the time required to transmit 44 bytes GFP* packets. Thus, speedup required in the segmentation process is avoided. If 580 bytes of the GFP* packets then arrives from the load generation unit, with the same destination and class pair to that of the earlier 44-byte packet, the segmented process will generate back-to-back two GCT cells having the same destination and class. The generated GCT cell is then placed in a separate GCT queue. From that GCT queue, the GCT cells will be delivered to the ingress port of the GCT switch. The remaining payload of 168 bytes is kept in the same destination class pair queue, until its size will be equal to or exceed the payload size of 228 bytes. The segmentation process again waits for the time, equivalent to the time required to transmit 580 bytes from the input load model. However, the GCT ingress port will receive GCT cells continuously from the GCT queue.

5.4 Performance Results

The simulation has been carried out in MODELSIM using HDL. For the simulation, different values of *p* are chosen for the IP packet load. This provides the IP traffic of load *p* from the load generation unit. As described in chapter 3, the switch is based on request grant protocol architecture, so the request grant protocol overhead is also added. Due to the addition of GFP*, GCT and request grant overhead, some speedup is required in the switch. Speedup is needed in the switch to maintain the linear flow of data through the networks. Larger the speedup value required in the switch, lesser is the switch efficiency.

Table 5.5 shows the speedup factor required in the switch. For this particular dissertation, FEC and scrambling bits are assumed to be subtracted and added, before and after the switch respectively. Hence, the speedups required due to these factors are not included during performance evaluation. The speedup requirement in the switch also includes the overhead contributed by the request grant header. Taking the data rate of IP as a reference, the speedup factor in the switch is found to be 1.0815. The valid IP traffic generated by the traffic load generator with packet validity probability of *p* will provide a load of 1.0815 *p* to the switch.

Table 5.5 Speedup factor

	Overall rate (Gbps)	Speedup factor
IP	9.437	1.0000
GFP*	9.529	1.0097
GCT (No FEC)	10.03	1.0629
Request Grant Protocol	10.21	1.0815
GCT	10.70	1.1337
Scrambling GCT	11.03	1.1691

First, the simulation is performed by taking a uniform IP load of 75%. The IP loads are increased to 85%, 87% and 89%. The actual loads in the switch are found to be 81.11%, 91.92%, 94.1% and 96.25% respectively.

Figure 5.1 shows the average queue depth per port against the cell time for different values of IP load. The figure verifies that the higher the input traffic rate, the higher is the queue depth. The average variation of queue depth, by taking a sample of 20 cell time for a load of 85% is found to be almost negligible as shown in figure. The figure shows that the switch is behaving efficiently under the traffic of IP load 85%, which is equivalent to the GCT traffic of 91.92 % load. On increasing the load further, the average queue depth goes on increasing. The queue will build until full, at which point discards will begin. The inability of the switch to process traffic at loads near to its full capacity is due to constraint present in our switch. This can be understood by the following examples.

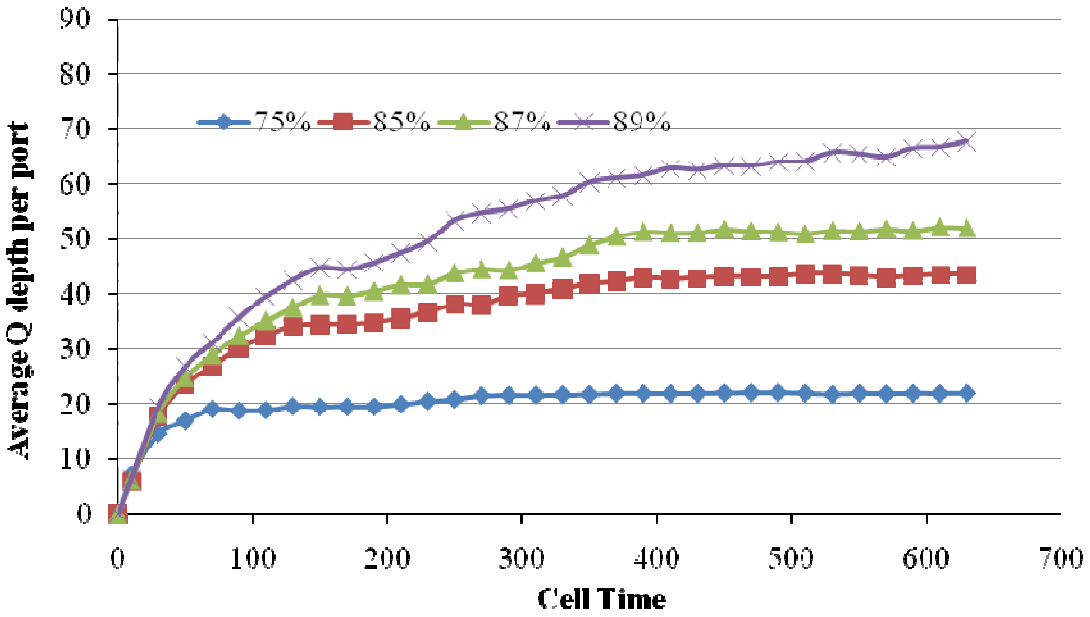

Figure 5.1 Average queue depths at different loads

Let us consider a four-input/four-output switch. Suppose ingress 0 has two requests for egress 0 and egress 3. Similarly, ingress 1 has only one request to egress 0, ingress 2 has two requests for egress 1 and egress 2 and ingress 3 has two requests for egress 2 and egress

3. This is shown in Figure 5.2. The small blank circle in the figure represents the egress requested while the small filled circle represents the connected ingress-egress pair. The figure shows that the ingress 0, 1, and 2 are serviced to egress 3, 0 and 2 respectively. We still have ingress 3 and egress 1 unused. However, the switch has the capability to make all four connections. If ingress 0, 1, 2 and 3 are serviced to egress 3, 0, 1 and 2 respectively, then all four connections will be made and the switch will perform to its full capacity. For these connections to be achieved, a backtracking algorithm is required. The backtracking algorithm has not been considered in our switch. The backtracking algorithm is complex and requires a large number of resources.

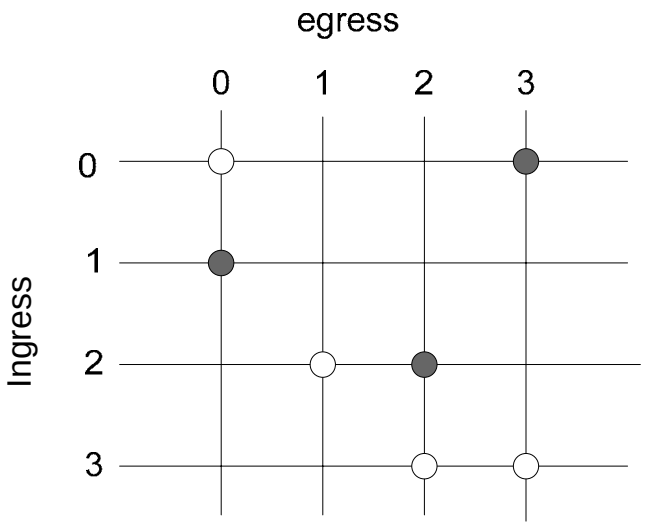

Figure 5.2 4 x 4 ingress-egress connections

In reality, the destination distributions of the traffic model are not always uniform. Traffic can be expected to exhibit strongly non-uniform characteristics. To enable simulation of non-uniform destination distributions, asymmetric destination distribution that overloads certain egress ports is used. This is a hotspot traffic model. The hotspot load has some bandwidth concentrated at some specific egress ports, such that the sum of the bandwidth directed at these outputs exceeds their capacity for some period of time. In this dissertation, 85% IP traffic load which behaves smoothly under the uniform traffic load is picked to simulate the behavior under the non-uniform distribution.

For that particular load of 85% IP traffic, hotspot behavior is applied. One egress port is chosen as a hotspot port, and the amount of traffic directed to that port is increased to 120%. The traffic at all other remaining ports is kept at 85%. Figure 5.3 shows the average queue depth against cell time for different classes of traffic targeting the hotspot port. The figure shows that the average queue depth for the class 0, class 1 and class 2 is attaining an idle behavior. The class 0, class 1 and class 2 traffic are serviced immediately, where as class 3 traffic queue builds up. The figure shows that the average queue depth for the class 3 traffic is building up and the trend will continue till the entire available queue for class 3 will be filled. After that a discard will start. From Figure 5.3, it can be concluded that QoS behavior has not been affected by hotspot.

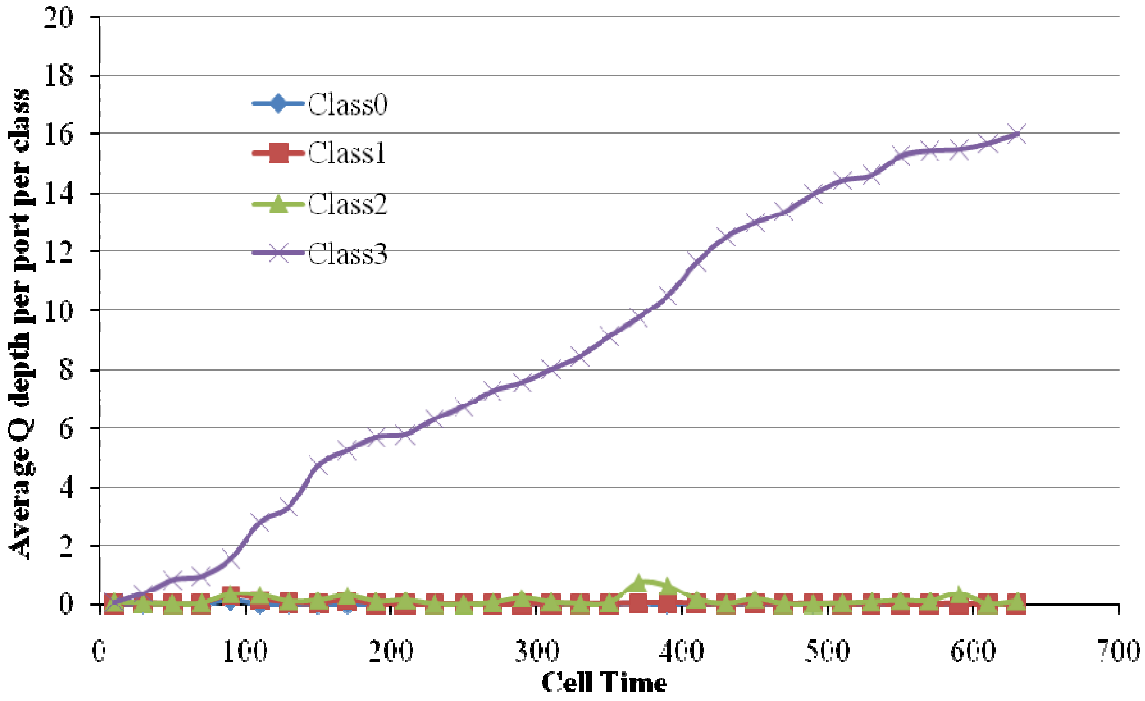

Figure 5.3 Average class queue depths normalized per port at hotspot

The behavior of the average queue depth normalized per port at the hotspot is shown in Figure 5.4. The figure shows that the queue for hotspot traffic is increasing. Despite the building up of the hotspot queue, the average queue depth for other destinations is stable. The

average queue depth behavior for all 31 ports, except the hotspot, is found to be similar to the behavior of average queue depth during uniform non-hotspot traffic. This shows that traffic going to non-hotspot ports proceeds normally, giving non-blocking nature for the switch. From the Figure 5.4, it can be concluded that the crossing traffic through the switch has not been affected due to the hotspot

.

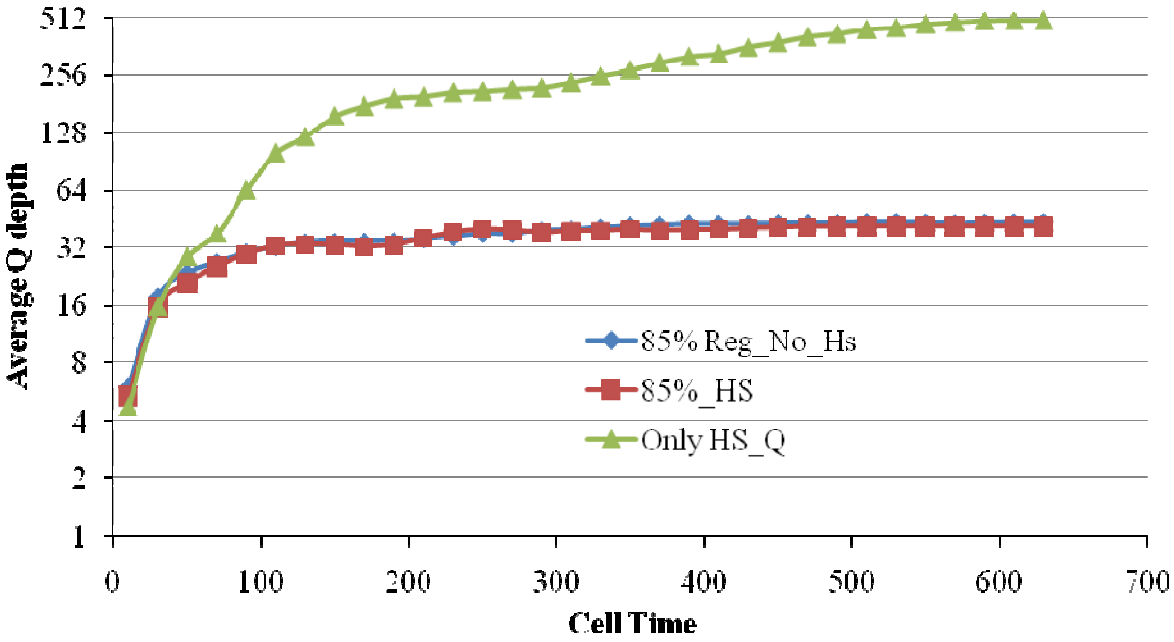

Figure 5.4 Average queue depths normalized per port at hotspot

5.5 Summary

This chapter describes the behavior of the switch under different traffic load conditions. First the traffic load generation unit was described. The equivalent IP packet length and the data rate at different stages in the network were then discussed. The performance of the switch by evaluating the queue depth against the cell time for the case of uniform distribution traffic load was then described. The 85% traffic load was then picked to verify the performance of the switch at the hotspot traffic. Various simulation results obtained were also discussed.

Chapter 6
CONCLUSIONS

The subject of this dissertation has been the design and prototyping of a switch that offers high performance, guarantees fairness among all its inputs and outputs, behaves robustly under different traffic patterns, supports QoS provisioning, and can be implemented in VLSI for higher throughputs at a reasonable cost. An input queued, virtual output queuing architecture was used to fulfill these requirements.

A large portion of the work has been invested in analyzing the GCT protocol as a transport network protocol. Similarly, much time has been spent studying different switch architectures. A significant period of time has been used to develop a pseudo random number generation unit to achieve fairness in a switch. In doing so, we have identified the problems associated with the existing transport network protocol, and the pros and cons of various existing switch architectures. With this knowledge, we have presented a switch architecture that supports the proposed new transport network protocol.

The organization of this chapter is as follows: Section 6.1 presents the summary of earlier chapters. Thesis contributions are summarized in Section 6.2. Future work to be done is presented in Section 6.3.

6.1 Summary

A brief introduction to the transport networks and the motivation for doing this thesis were described in Chapter 1. Since this thesis is based on a new transport network protocol, the problems related with the existing transport network were presented in Chapter 2. Some legacy solutions and some proposed solutions were also discussed in Chapter 2. Also, definition of the GCT protocol, its header information and its overhead cost comparison were given in Chapter 2.

Different conventional switch architectures were discussed in Chapter 3. The advantages and disadvantages of different switch architectures using crossbar were then discussed. The investigation of the switch architecture leads us to design an input queued, virtual output queuing switch architecture suitable for GCT switch. The detail designs of this switch architecture were described in Chapter 3.

The arbiter architecture, which is essential to the performance of the switch, was then described in Chapter 4. Some of the scheduling algorithms used in the literature were given first. The working principles of a wavefront arbiter and a wrapped wavefront arbiter were discussed in detail. The problems of biasing were then presented. The detail description of the proposed pseudo random number generation unit, which the thesis used to nullify the biasing problem, was then described in detail in Chapter 4.

In Chapter 5, the performance of the switch was presented. The load generation unit was modeled. The behavior of the traffic load at different stages in the existing network as well as in the proposed GCT transport network was discussed. The speedup factor required in the designed switch was also presented in Chapter 5. First, the generated uniform traffic load was used to verify the validity of the switch. Simulation results under different traffic load condition were presented. Then, hotspot traffic was generated and the behavior of the switch under the hotspot traffic was investigated. The simulation results under the hotspot traffic are also presented in Chapter 5.

6.2 Thesis Contributions

The dissertation has made the following contributions:

- Definition of the GCT protocol

  The problems present in the current existing transport network have been studied and a new candidate solution for the next generation transport network has been considered. The efficient protocol structure to the novel candidate solution has

been identified. The proposed scheme offers load balancing, QoS provisioning, high network protection and low hardware development costs.

- Prototype of a switch architecture

The constructive definitions of switch architecture are explored and a best architecture to be implemented in VLSI for the proposed transport network, input queued virtual output queuing, was obtained. The prototype of this switch has been designed in a Verilog model.

- Pseudo random number generator

The unfairness problem obtained in the arbitration unit using a wrapped wavefront arbiter [28] has been nullified to a certain extent by a proposed novel pseudo random number generation unit. The Verilog model of the pseudo random generation unit has been designed.

- Performance evaluation

The evaluation of the designed switch is carried out by modeling the load generation unit in Verilog. The validity of the switch behavior has been investigated by generating uniform traffic load from the load generation model. Different simulation results have been obtained under different traffic load condition. The hotspot traffic has been generated and the performance of the switch under the hotspot traffic has been investigated. The result showed that the QoS behavior and the behavior of the crossing traffics under hotspot have not been affected.

6.3 Future Work

The project of designing a new transport network protocol and producing a prototype in itself is a huge task. This work is the beginning for the realization of proposed new transport network protocol. There is still much work to do to integrate the best feature provided by this project.

Currently multipath routing has not been considered. In future work, hardware and software programs performing multipath routing to provide high network protection are to be designed.

The segmentation and reassembly process are to be designed. The process is transparent to the application and hence, is to be designed to support different independent existing and new application protocols.

The FEC is to be added. The exact size of the queue depth to be implemented in the switch is to be defined through simulation. Finally, the FPGA version of the switch is to be developed and verification process is to be done.

REFERENCES

[1] K. Iniewski, C. McCrosky, and D. Minoli, "*Network Infrastructure and Architecture, Designing High-Availability Networks*," Wiley, 2008.

[2] ITU Recommendation G.709/Y.1331, "*Interfaces for the Optical Transport Network (OTN),*" March 2003 (Amendment1 Dec 2003).

[3] L. Berger et al., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description," *RFC 3471*, Jan 2003.

[4] T1.105-2001 -"*American National Standard for Telecommunications – Synchronous Optical Network (SONET) – Basic Description including Multiplex Structure, Rates, and Formats*," 2001.

[5] H. G. Perros, "Connection-Oriented Networks: *SONET/SDH, ATM, MPLS and Optical Networks*," John Wiley & Sons, 2005.

[6] ITU-T Recommendation G.7041/Y.1303, "*Generic Framing Procedure*," (2001).

[7] Gigabit Ethernet Alliance, "Gigabit Ethernet Overview," 1997. http://www.gigabit-ethernet.org.

[8] IEEE Standard 802.17-2004, Information Technology - Telecommunications and Information Exchange between Systems - LAN/MAN – Specific Requirements - Part 17: Resilient packet ring (RPR) access method and physical layer specification.

[9] W. Simpson, "PPP over SONET/SDH," IETF RFC 2615, Jun 1999.

[10] I. S. Reed and X. Chen, "*Error-Control Coding for Data Networks*," Boston, Kluwer Academic Publishers, 1999.

[11] D. Mehdi and K. Ramasamy, "*Network Routing, Algorithms, Protocols and Architectures*," Morgan Kaufmann, 2007.

[12] A. Huang and S. Knauer, "Starlite: A wideband digital switch," *Proceedings of IEEE Globeconi*, pp. 121 - 125, 1984.

[13] I. Iliadis and W.E. Denzel, "Performance of Packet Switches with Input and Output Queueing," *in Proceedings of ICC '90*, pp. 747-753, Apr. 1990.

[14] Y. Yeh, M.G. Hluchyj, and A.S. Acampora, "The Knockout Switch: A Simple, Modular Architecture for High-Performance Packet Switching," *IEEE J. Sel. Areas Communications*, Vol. 5, no. 8, pp. 1274-1283, Oct. 1987.

[15] De Vries and R.J.F, "Gauss: A Single-Stage ATM Switch with Output Buffering," *Proceedings of International Conferences on Broadband Services and Networks*, pp. 248-252, 1990.

[16] H. Suzuki, H. Nagano, and T. Suzuki, "Output-buffer Switch Architecture for Asynchronous Transfer Mode," *Proceedings of ICC '89*, Boston, MA, pp. 99-103, Jun. 1989.

[17] R. Y. Awdeh and H.T. Mouftah, "Survey of ATM Switch Architectures," *Computer Networks and ISDN Systems*, Vol. 27, pp. 1567-1613, 1995.

[18] M. G. Hluchyj and M. J. Karol, "Queuing in High-Performance Packet Switching," *IEEE J. Sel. Areas Communications*, Vol. 6, no. 9, pp. 1587-1597, Dec. 1988.

[19] S. C. Liew, "Performance of Various Input-buffered and Output-buffered ATM Switch Design Principles under Bursty Traffic: Simulation Study," *IEEE Transactions on Communications.*, Vol. 42 no. 2/3/4, pp. 1371-1379, Feb/Mar/Apr 1994.

[20] M. Katevenis, P. Vatsolaki, and A. Efthymiou, "Pipelined Memory Shared Buffer for VLSI Switches," *Proceedings of ACM SIGCOMM '95*, pp.39-48, Aug. 1995.

[21] R. P. Luijten, T. Engbersen, and C. Minkenberg, "Shared Memory Switching + Virtual Output Queuing: A Robust and Scalable Switch," *Proceedings of ISCAS 2001*, Sydney, Australia, May 6-9, pp. 274-277, 2001.

[22] T. Anderson. S. Owicki, I. Saxe, and C. Thacker, "High Speed Switch Scheduling for Local Area Networks." *ACM Transactions on Computer Systems*, pp.319-52, Nov.1993.

[23] N. Mckeown, "Scheduling Cells in an Input-Queued Switch." *PhD thesis, University of California at Berkeley*, May 1995.

[24] D. N. Serpanos and P. L. Antoniadis, "FIRM: A Class of Distributed Scheduling Algorithms for High-speed ATM Switch with Multiple Input Queues," *Proceedings of IEEE ATM Workshop*, pp. 548-555, May 1998.

[25] N. McKeown , "iSLIP: A Scheduling Algorithm for Input-Queued Switches," *IEEE Transactions on Networking*, Vol I, No.2. pp. 188-201, April 1999.

[26] M. G. A. Marsan, A. Bianco, E. Fllippi. and E. Leonardi, "On the Behavior of Input Queueing Architectures." *Eur. Trans. Telecommunications*, Vol. 10, No. 2, pp. 111-124. Mar. 1999.

[27] C. K. Hung, M. Hamdi, and C. Tsui, “Design and Implementation of High-speed Arbiter for Large Scale VOQ Crossbar Switches,” *Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong.*

[28] Y. Tamir and H. Chi, “Symmetric Crossbar Arbiters for VLSI Communication Switches,” *IEEE Transactions on Parallel and Distributed System,* Vol. 4(1) pp. 13-27, Jan 1993.

[29] M. Fidler, V. Sander, and W. Klimala, “Traffic shaping in aggregate based networks: Implementation and analysis,” *Elsevier Computer Communications*, 28(3), pp. 274–286, Feb 2005.

[30] K. Thompson, G. J. Miller, and R. Wilder, “Wide-Area Internet Traffic Patterns and Characteristics,” *IEEE Network*, pp. 10–23, Nov 1997.

[31] J. Cong and B. E. Wolfinge, “A unified load generator based on formal load specification and load transformation,” *In Proceedings of the 1st international Conference on Performance Evaluation Methodologies and Tools* ,Pisa, Italy, Oct 11 - 13, 2006.