DESIGN, SIMULATION AND IMPLEMENTATION OF ENHANCED CROSSBAR COMBINED INPUT-OUTPUT QUEUED SWITCH ARCHITECTURE

ATIQ AWAN

NOTE TO USERS

Design, Simulation and Implementation of Enhanced Crossbar Combined Input-Output Queued Switch Architecture

by

© Atiq Awan

A thesis submitted to the School of Graduate

Studies in partial fulfillment of the

requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

April, 2004

St. John's Newfoundland

Your file Votre référence
ISBN: 0-494-02327-9
Our file Notre référence
ISBN: 0-494-02327-9

Abstract

The rapid growth in communication networks has led to an exponential increase in the traffic volumes thus requiring high-speed switches and routers, with high bandwidth, at the nodes. The networking research community has focused its effort on the development of high-bandwidth switches. However, advances in data transmission technology, particularly the development and use of the optical technology, have enabled reliable high transmission bandwidth at a relatively low cost. On the other hand, neither switches nor routers have kept pace with this development. Therefore, the switches and routers are increasingly becoming performance bottlenecks in high-bandwidth communication.

Until recently, most of the packet switches were based on output queueing due to its ability to provide high throughput. However due to the slow memory speeds compared to link bandwidth, output queued switches are no longer feasible for high-speed switching. In this work, we consider building a combined input-output queued packet-switch using multiple crossbars, which forms a step forward towards obtaining a better solution for high performance packet switching. In particular, we consider multiplane switch architecture with four crossbars in parallel. These crossbars transfer up to four packets to each output line to provide the high throughput however operating at the same speed as the line rate.

We investigate the performance of this architecture under both uniform and non-uniform traffic arrival patterns and show through simulation that this architecture

approximately emulates the pure output queued switch. This architecture employs pipelined scheduling, which eliminates the traffic scheduling overhead and provides a large time window for implementing fair but complex scheduling algorithms.

Finally, we describe the implementation of this architecture in VLSI using 0.18-micron CMOS standard cell technology. The distributed control leads to a high-speed implementation. We report on the design complexity and discuss implementation results.

Acknowledgements

I would like to thank my supervisor, Dr. Ramachandran Venkatesan for his support, guidance, and knowledge and for giving me the opportunity to be a member of the Computer Engineering Research Laboratory. I want to acknowledge all the members of this lab for their help and for being both great friends and colleagues.

My special thanks to Moya Crocker for helping me with all the documentation and Mr. Nolan White for helping me with problems related to CAD tools. I would also like to thank my friend Yassir Nawaz for being a wonderful friend.

Finally, I would like to thank my parents, my brother, my sister in law, and my wife for their great support and endless love during my study period, without whom it would be impossible.

Table of Contents

List of Figures

List of Tables

Notation and List of Abbreviations

m	number of queues in a MIQ switch or number of planes in a multiplane switch
λ	traffic arrival rate
n	number of outputs in a convolutional encoder
k	number of switching planes in a switching network
$\gamma_{\max}$	maximum throughput
ρ	link load
D_{mean}	Mean delay through the switch
D_{in}	delay though input buffers
D_{out}	delay through output buffers
L	parallelism factor in multiplane switches
N	number of ports in a packet switch
S	Speedup
ADSL	Asymmetric Digital Subscriber Loop
ASIC	Application Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
CIOQ	Combined Input-Output Queued
CMOS	Complementary Metal Oxide Semiconductor
FIFO	First-In First-Out
HOL	Head of Line
IP	Internet Protocol
IQ	Input Queued
MPLS	Multi-protocol Layer Switching
MIQ	Multiple input queueing
OQ	Output Queued
QoS	Quality of Service
RAM	Random Access Memory
URT	Uniform Random Traffic
VOQ	Virtual Output Queueing

VHDL	Very high-speed integrated circuit High Level Description Language
VLSI	Very Large Scale Integration

Chapter 1

Introduction

Communication networks underwent a dramatic change during the 1960s with the introduction of a new technology called packet switching. Before that all the interactive communication data networks were circuit-switched. In circuit switching, a path is established from the source to the destination at the connection set-up time. This path remains fully connected for the duration of the connection. Circuit switching is only cost effective when there is a continuous flow of data once the circuit has been established. This is the case for voice communication.

Communication among computers, however, happens in bursts. Data travels through the networks in the form of blocks. Users do not need the transmission link all the time. Therefore assigning a continuous bandwidth for such a connection is a waste of resources and results in low utilization. Packet switching divides the input flow of information into small blocks and only allocates bandwidth when a block of data is ready

to be sent. These blocks are referred to as packets. This reduces the wastage of available transmission bandwidth resources. To do this, packet systems require both processing power and buffer storage resources at each node in the network [1].

1.1 Packet Switched Network

In a packet switched network, the transfer of packets from one point to another involves two basic tasks: routing and switching. Routing is performed to determine a path, which the packets must take to reach the destination. Switching, which takes place at every node on the path, refers to placing the packets onto the path determined by a prior routing decision. Packet switching at each node in a network includes packet buffering and packet forwarding. Upon arrival of a packet at an input port, the switch determines from the header of the packet to which output port the packet should be forwarded. As illustrated in Figure 1.1, the packets can be switched through different routes to the same destination.

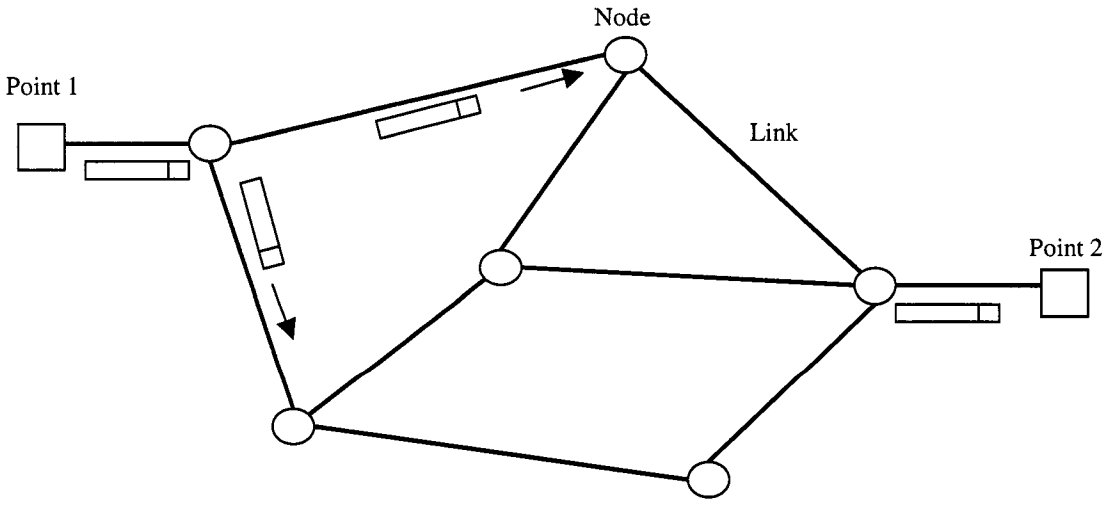


Figure 1.1 A packet switched network.

A router can take advantage of the high switching bandwidth offered by a packet switch since a packet switch implements the two tasks that a router also needs to perform. A router is a network device that passes traffic between two different IP networks which may be either LANs or WANs. This routing process is based on examining the destination IP address of the incoming packets and sending on the packets to an output port based upon a routing table. A packet switch is a network access device that is used to link physical segments of a network together.

High performance routers are increasingly using a switch as a backplane to handle packet buffering and packet forwarding [2]. Most high performance routers internally use small fixed size packets that are referred to as cells. These cells are not necessarily equal in length to a 53-byte ATM cell.

This simplifies the system design, allowing switches to run faster. Variable length packets are simply segmented into cells as soon as the packets arrive at the input ports of the routers, and reassembled back into the variable length packets at the outputs before the packets are sent out onto the transmission links.

1.2 ISDN and Broadband ISDN

Integrated Services Digital Network (ISDN) was intended to combine the existing telephone and data networks onto a single network. The overall goal of ISDN is to create the framework and standards necessary to permit public telecommunications networks to evolve to the point where worldwide fully digital end-to-end services can be provided [1]. ISDN, sometimes referred to as a narrowband ISDN, is based on the use of a 64–kbps

channel as the basic unit of switching and has circuit-switching orientation. The second generation, referred to as Broadband ISDN, supports very high data rates and has a packet switching orientation.

Initially Asynchronous Transfer Mode (ATM) was considered as the most suitable switching and multiplexing technique for Broadband ISDN due to its ability to provide Quality of Service (QoS) guarantees. However with the introduction of Multi-protocol Layer Switching (MPLS), the Internet Protocol (IP) can attain the same capabilities. MPLS can run over nearly any transport medium, including ATM and Ethernet.

Currently IP is the most popular networking protocol and IP routed MPLS is making inroads into the backbone and pushing ATM to the edges. However ATM is going to stay for some time and is being considered for the congressional ADSL network. Irrespective of the networking protocols used, the nodes in a network have the same performance requirements.

1.3 Queueing Techniques

Packet switches require some queueing mechanism to store the packets. This queueing can take place at the inputs, at the outputs, or at both inputs and outputs. For high bandwidth switches, the speed of memory, which is the basic building block of queues, can become the limiting factor in terms of size and speed of the switch [3]. Figure 1.2 illustrates a general packet switch with N inputs, N outputs, and a combination of input queueing and output queueing. Input port module (IPM) has an input port

controller and one or more queues for storing the packets arriving at the inputs. Similarly, output port module (OPM) has output port controller and one or more queues for storing packets coming from IPMs. In the following we compare three queueing techniques, namely: output queueing, input queueing, and combined input and output queueing.

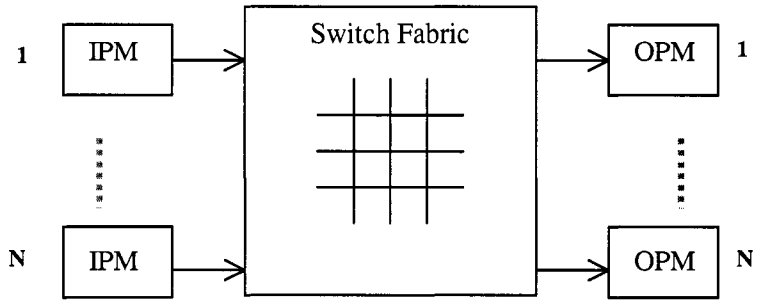


Figure 1.2 A general packet switch.

Output queueing is referred to as a queueing technique in which all queues are placed at the outputs. Switches employing this queueing technique are known as output queued (OQ) switches. Output queued switches have been popular due to their high throughput [3]. Throughput is the ratio of total number of cells transmitted on the output link to the total number of cells arrived at the inputs of a switch during a given period of time. However these switches are unable to meet the requirement of high-speed switches, due to memory bandwidth limitations.

In input queueing (IQ) all the queues are placed at the inputs. Input queued switches are the better choice with respect to memory speed, since memory is required to

operate only at the line rate. However input queued switches suffer from the head of line blocking (HOL) problem, which can degrade the performance of the switch considerably [4]. HOL blocking occurs when the packet at the head of a queue cannot be transferred to an output port due to contending packet from another input port. At the same time, a packet further back in the queue is blocked although its destination port is free to receive the packet. IQ switches also require scheduler/arbiters to configure the switch fabric before the transfer of packet from inputs to outputs.

Combined input-output queued (CIOQ) switches have queues both at inputs and outputs. CIOQ switches combine the advantages of input and output queueing and mitigate the drawbacks in both the techniques. Only one cell can arrive at each input in the CIOQ switches during each switching cycle, therefore the memory is not required to operate faster than the line rate. Switching cycle is the time taken to transmit or receive a fixed size packet. Switching cycle is also usually referred to as a time slot. In CIOQ switches, the memories at outputs are not required to operate at N times faster than the line rate. Various performance analyses of the CIOQ switches have shown that with a single First In First Out (FIFO) queue at each input port and a speed up of 4 to 5, a very high throughput can be achieved [3][5]. However switches involving input queuing require schedulers to configure the switch fabric. Use of scheduling algorithms for this purpose has shown very high throughput but no delay guarantees are provided and their runtime complexity is inadequate for implementation of large-scale switches.

1.4 Scheduling

Two different kinds of schedulers are required in switches with input queues. These are switching matrix schedulers and flow level schedulers. The switching matrix schedulers avoid internal and output blocking by making non-conflicting input-output pairs. On the other hand flow-level schedulers are used for providing Quality of Service (QoS) guarantees by controlling the delay that a packet may suffer, while traversing the switch [6]. In this thesis the term scheduler is used to refer to switch matrix scheduling. These are also sometimes referred to as arbiters in the literature.

In input queued switches, each output can receive at most one cell. The goal in cell scheduling is to find a one-to-one match between a nonempty input port and a free output port. In other words, the scheduler matches an unmatched input to an unmatched output. Every unmatched input makes a request to the scheduler telling it which outputs it wants to be matched.

Conceptually, the set of requests can be represented by a bipartite graph, called a request graph. The cell scheduling is equivalent to finding a bipartite graph matching. Given a request graph, the scheduling algorithm solves a bipartite graph-matching problem to find a match graph [7]. A match graph is a selection of edges such that no vertex has two connected edges.

1.5 Motivation for Research and Objective of the Work

Broadband ISDN is required to support different high bit rate services, such as videoconferencing, video on demand, medical imaging, high-definition television and

video, distance learning, and geophysical modeling. To provide these services network nodes should be capable of providing high bandwidth.

Output queued switches are capable of providing high throughput, however their size and speed are limited by the speed of memory. The other choice, input buffered switches, have limited throughput due to head of line blocking (HOL). Various studies on CIOQ switch architectures have shown the approximate emulation of output queued switches. Multiplane switches are an alternative to switches with speedup [3][8]. The number of planes in a multiplane switch architecture is called the parallelism factor (L). One example is that of the knockout switch in which up to L, $1 \leq L \leq N$, HOL cells can be transferred to each output in a switching cycle [9]. In [10], Chen *et al* showed that for $L = 4$ the throughput is in excess of 99%. This represents the throughput of the switch when the traffic load is 100%. Such high traffic load would require very large buffer space or it would result in a large amount of cell loss. However switches are usually not subjected to such high traffic loads. This number only represents the capability of that switch architecture. The average delay through input buffers decreases by increasing the parallelism factor and with $L = 4$, the switch approaches its optimal performance. The delay improvement for $L > 4$ is minimal.

Along with high speed, other issues that should be considered while designing packet switches are scalability and feasibility for VLSI implementation. Any switch architecture involving input buffering can be either self-routing or it can employ schedulers for configuring the switch fabric. In self-routing the control is distributed among the switching elements therefore no external scheduler is required for configuring

the switch fabric for packet transfer from input ports to the output ports. A crossbar switch is not self-routing therefore it needs schedulers. Almost all the existing algorithms either fail to maintain high throughput when the traffic becomes non-uniform or are too complex to operate at high speed, becoming a speed bottleneck in the switch [11]. Although there exist algorithms that can avoid the low throughput problem, they are unfortunately too complex and too slow for high-speed switches.

Speed and scalability are important measures in determining the overall performance of a high bandwidth packet switch. This thesis addresses both these parameters and attempts to provide a solution, using combined input-output queueing and multiple crossbars, which is both scalable and produces throughput close to that of an output queued switch.

1.6 Thesis Organization

The thesis is composed of four main chapters. Chapters 2 and 3 present the architectural options for packet switches and enhancement of throughput using various techniques. Chapter 4 presents the design and performance of enhanced crossbar CIOQ switch architecture. Chapter 5 presents the VLSI implementation. Chapter 6 concludes this thesis. In particular, the following issues are discussed in each chapter.

Chapter 2 briefly describes the various architectural options available for packet switches. Brief description of operation, advantages, disadvantages, and limitations in each of type of the switches is presented. Furthermore, it describes the scheduler

available for configuring the switch fabric before the packet transfer from the input ports to the desired output ports.

Chapter 3 discusses the throughput enhancing techniques for input queued switches, such as the use of multiple input queueing, speeding up the switch fabric or using multiplane switch fabrics. It also includes the description of popular traffic models used for simulation and analysis of switch architectures and the pipelining of the scheduling process that eliminates the scheduling overhead.

Chapter 4 presents the design of the switch architecture that employs combined input-output queueing, enhanced crossbar, and pipelined scheduling. This switch architecture is suitable for use in routers as a high throughput back plane or as an ATM switch. The switch architecture uses multiplane switch fabric for improving the throughput. The performance results of this switch architecture, obtained from simulations for both uniform random and bursty traffic arrival, are also presented.

Chapter 5 presents the design and VLSI implementation of the enhanced crossbar switch architecture to show that our work is easily implementable in readily available CMOS technology. The design complexity and implementation results are reported. The design is targeting 0.18-micron CMOS standard-cell ASIC technology. This chapter describes the design of each individual block and presents the total area and gate count for each block.

Chapter 2

An Overview of Packet Switches and Schedulers

This chapter provides an introduction to the existing packet switch architectures and scheduling algorithms. We briefly describe the various architectural options and schedulers available. However for detailed introduction to packet switch architectures one should refer to [1][12] and [13]. We also describe some of the approaches adapted for enhancing the performance of the packet switches.

The switch fabric is the interconnection network, which provides paths between the input and output ports. It switches between N links. A physical port is basically a bi-directional port that consists of an input and an output port. So there are N input ports and N output ports in each packet switch.

In the literature the switches have been mainly classified into two categories: time division switches and space division switches [1][12]. A simple classification of a switch fabric that includes most of the proposed approaches is illustrated in Figure 2.1. We

briefly describe the operation, advantages, disadvantages, and limitations in each of type of the switch.

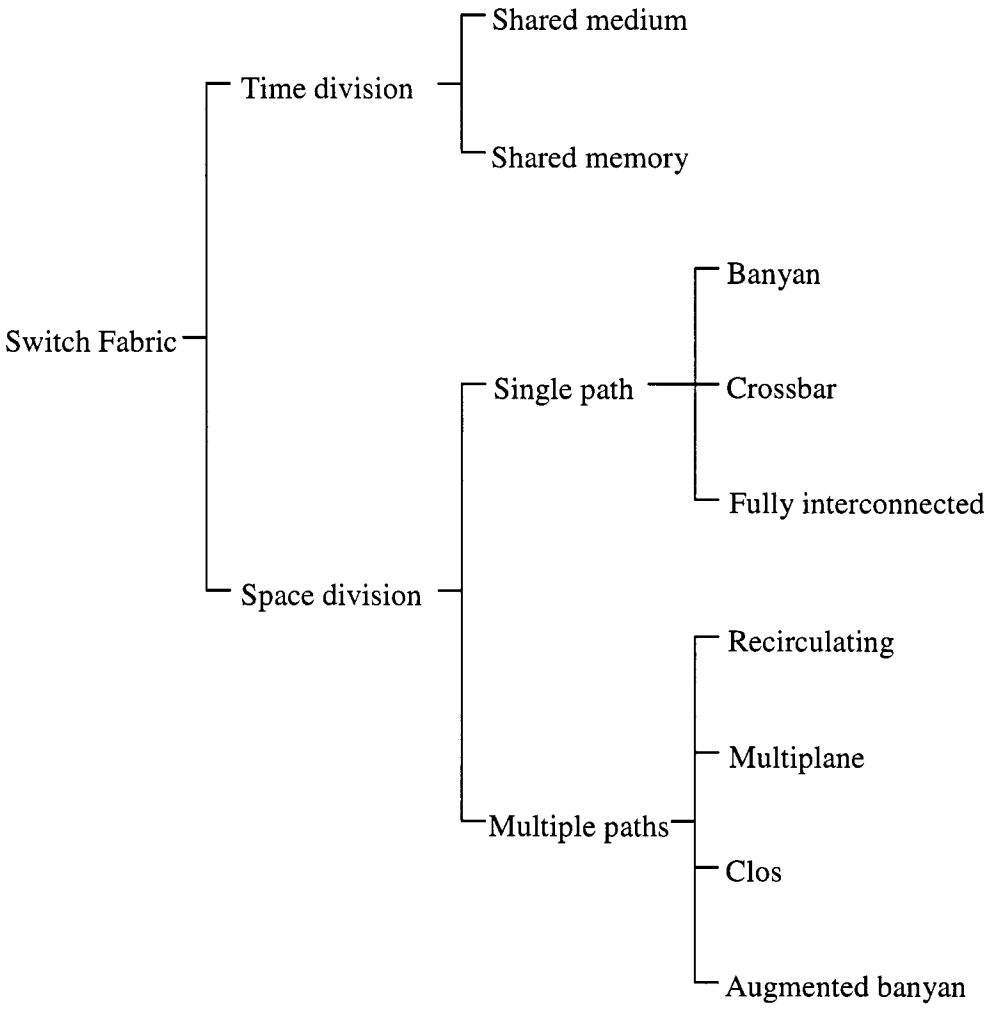


Figure 2.1 Classification of packet switch architectures based on switch fabric.

2.1 Time Division Switches

In these types of switches, all the cells traveling from input ports to the output ports share a single resource. This resource can be a bus, a ring or memory. The design and implementation of a large-scale switch based on time division switches is usually limited due to the capacity of the shared resource. As shown in Figure 2.1, there are two types of time division switches: shared medium and shared memory.

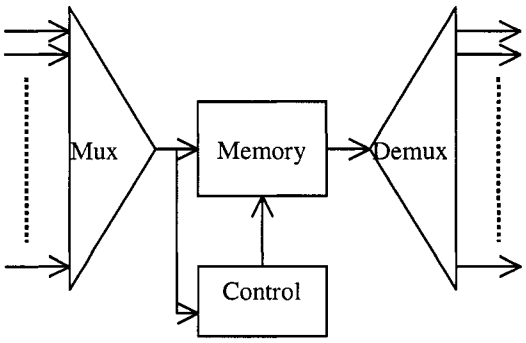


Figure 2.2 Shared memory switch

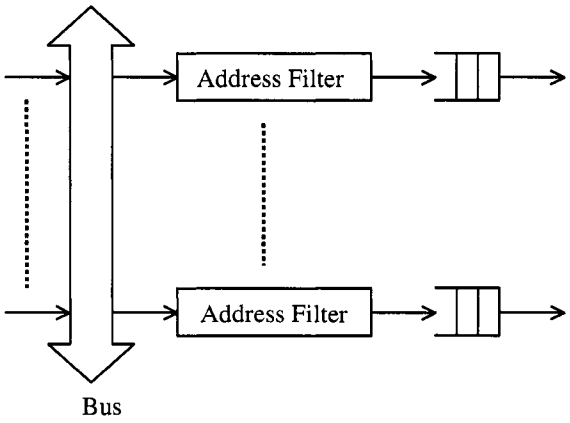


Figure 2.3 Shared medium switch

Shared memory switches use a common memory shared by all inputs and outputs. Incoming cells are multiplexed into a single data stream and written to the respective locations of the common memory with respect to their destination addresses. The multiplexing and demultiplexing of cells is shown in Figure 2.2.

Shared memory architectures are inherently free of internal blocking and provide the best memory utilization. Internal blocking occurs due to interconnect contention in the switch fabric. However they suffer from output blocking and the capacity of these

switches is limited by the bandwidth of the common memory. The centralized controller requirement is also a limiting factor in terms of size of a shared memory switch.

Shared medium switches use shared medium such as a bus or a ring for interconnection network. As shown in Figure 2.3, address filters filter the header of the incoming cells and accept only the cells destined for them. Shared medium switches are internally non-blocking and due to their broadcasting nature they can easily support the multicast operation. However they suffer from output blocking and the throughput of the shared medium, which determines the capacity of the switch, can become performance bottleneck in some cases. NEC's ATM output buffer modular (ATOM) is an example of shared medium switch [14].

2.2 Space Division Switches

In a space division switch multiple concurrent paths are established between inputs and outputs. These switches are classified based on the number of paths available from each input to each output. Those with single path are called single path space division switches and those with more than one path are called multipath space division switches. Unlike time division switches, in space division switches the control can be distributed which allows the design and implementation of high-speed large-scale space division switches.

2.2.1 Single Path Space Division Switch

Crossbar, banyan, and fully interconnected switches are examples of single path space division switches. Crossbar and fully interconnected are interconnect contention free, that is, each input-output pair has a dedicated path available. On the other hand Banyan-based switches suffer from interconnect contention.

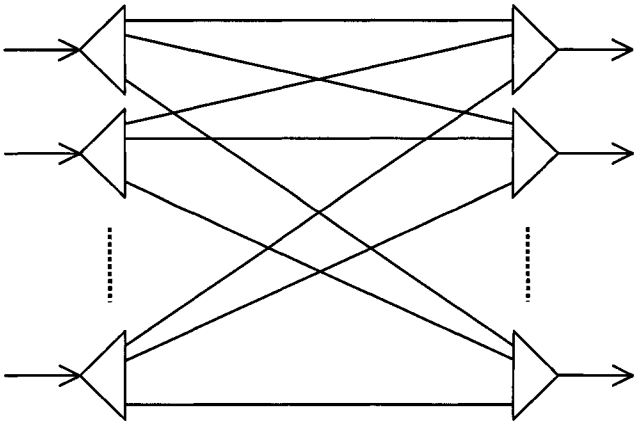


Figure 2.4 Fully interconnected switch fabric

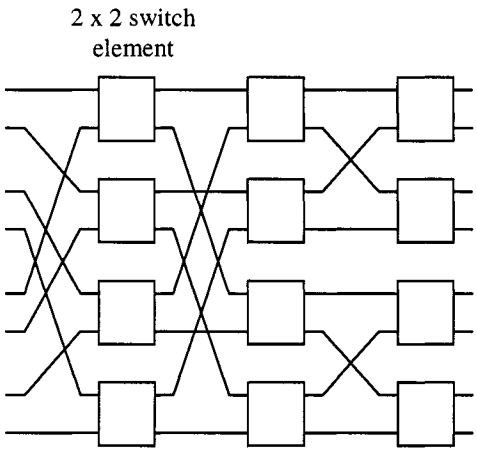


Figure 2.5 Banyan network

As illustrated in Figure 2.4, in a fully interconnected switch there are N broadcast buses from every input to every output port providing the connectivity. A cell from any input port is broadcast to every output port. This allows the simultaneous transmission of cells to the same output. Its very simple architecture, however requires N^2 separate buses. The Knockout switch is an example of fully interconnected switch [9].

Banyan based switches are self-routing and are built from 2 x 2 switching elements. As shown in Figure 2.5, there is a single path available for every input-output

pair. Self-routing makes these switches very attractive, as no centralized cell routing is required. However the major drawback is that of internal blocking which can significantly degrade the performance of the switch.

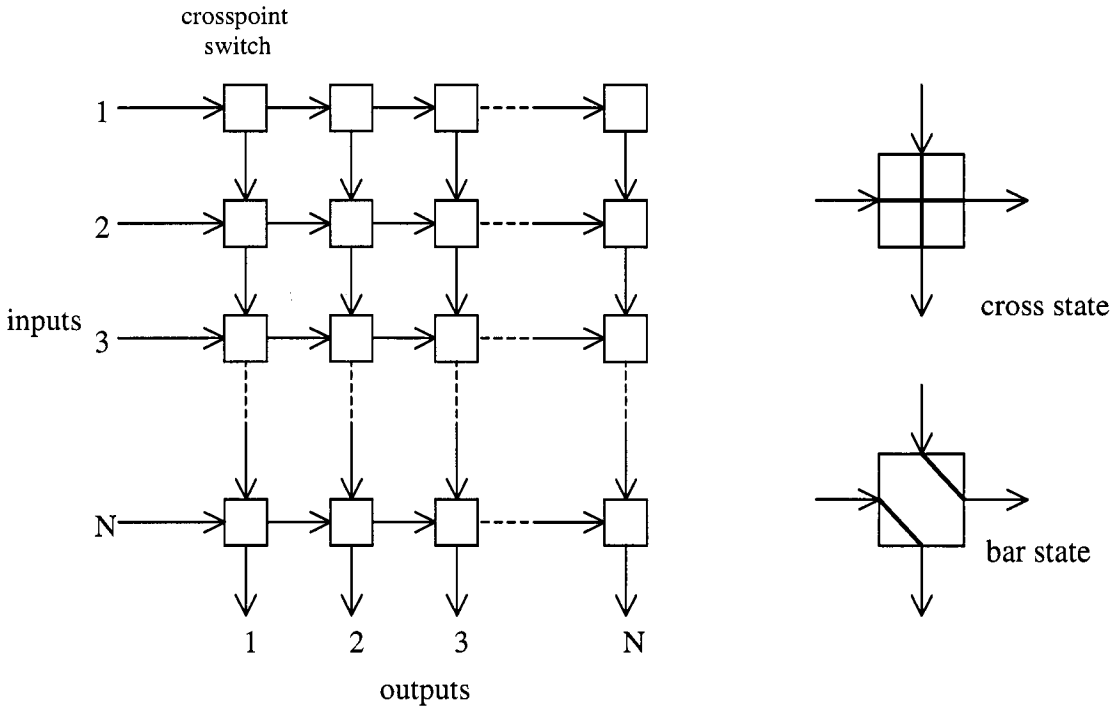


Figure 2.6 N x N crossbar switch

2.2.1.1 Crossbar switch

A crossbar is a single path interconnect contention free space division switch. It is the first electronic space division switch that came into existence [1]. It consists of N^2 crosspoints, one corresponding to each input-output pair. As illustrated in Figure 2.6 each crosspoint has two possible states, either cross or bar.

The main advantages of crossbar switches are their simple architecture, ability to facilitate all permutations between inputs and outputs, and modularity. Their main disadvantage is the number of crosspoints, which grows as N^2 . The throughput of input queued crossbar is limited due to head of line blocking.

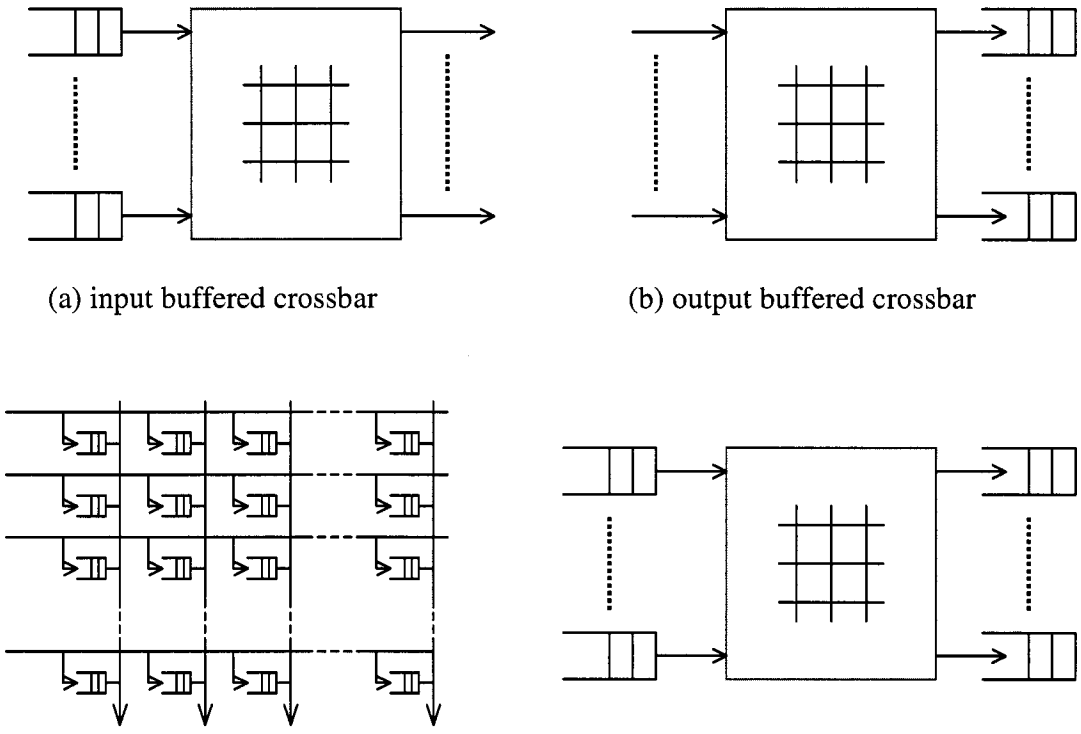
Different buffering strategies have been adopted for crossbar switches like input buffered, output buffered, crosspoint buffered, and input and output buffered crossbar switches. Figure 2.7(a) illustrates an input buffered crossbar. The main advantage in input buffered crossbar is the low memory speed requirement, the same as that of line rate. However these switches possess the head of line blocking which limits the throughput of the switch. The maximum throughput of the crossbar switches with a single FIFO queue at each input is known to be [15]

$$\gamma_{max} = 2 - \sqrt{2} . \qquad (2.1)$$

The output buffered crossbar switch, shown in Figure 2.7(b), is known for its high throughput. However, the output ports of this type of switches should be capable of accepting all the cells, during a switch cycle, destined to them to avoid output blocking. This requirement makes high-speed switches based on output buffered crossbar, impractical.

To avoid blocking in crossbars, buffers can be placed at the crosspoint as shown in Figure 2.7(c). If more than one cell is destined for the same output, one is routed to the output and others are stored in the crosspoint buffer and routed in the next switching cycles. This approach has neither head of line blocking nor output blocking. However this comes at the cost of more buffers, as there are N^2 crosspoints so N^2 buffers are

required compared to N buffers in both input and output buffering strategies. Fujitsu's BMX is an example of a buffered crossbar switch [16].



(a) input buffered crossbar

(b) output buffered crossbar

(c) crosspoint buffered crossbar

(d) input- output buffered crossbar

Figure 2.7 Buffers location in crossbar

Combined input-output queued crossbar switches combine the advantages of both input and output buffered crossbars, at the same time mitigating their drawbacks. A combined input-output buffered crossbar is shown in Figure 2.7(d). In this type of switch, the output buffer is not required to operate N times as fast as the line rate. Alternatively, some speed up or parallel planes can provide the required throughput. Combined input-

output crossbar switches are considered a better option for a high-speed large-scale switch architecture.

2.2.2 Multipath Space Division Switch

In multipath space division switches there is more than one path available from each input to each output. Parallel banyan, augmented Banyan, Clos, Balance Gamma and recirculating switch are some of the examples of multipath space division switches. The multipath switches have higher reliability due to the availability of more paths. However this comes at the cost of more hardware and thus increases the complexity of the design.

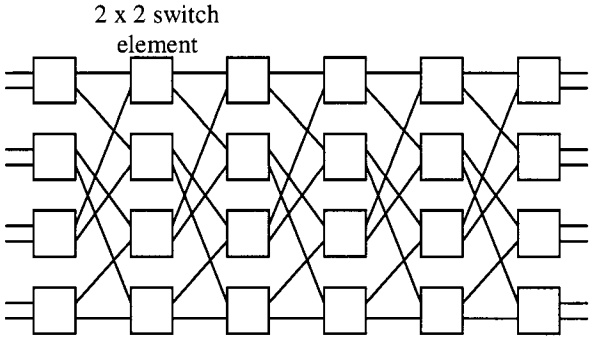


Figure 2.8 Augmented Banyan switch

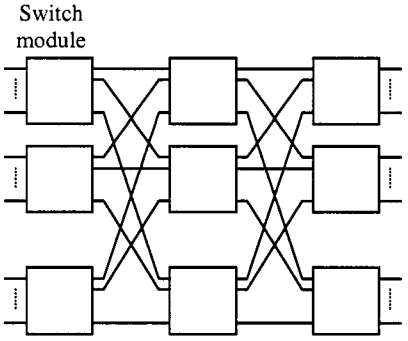


Figure 2.9 3-stage Clos switch

When multiple paths are introduced into a Banyan network by adding extra stages of switching elements as shown in Figure 2.8, it is called augmented Banyan switch. The number of paths between input and output ports is doubled for every extra stage that is added to the Banyan network. The extra stages are used to distribute the traffic evenly

across network and to improve the network's performance. However the improved performance comes at the cost of increased hardware complexity. Tandem Banyan switch is an example of augmented Banyan switch [17].

The Balanced Gamma (BG) switch is also a multipath switch [18]. The BG network has n –1 stages where n = $\log_2$ N. The first stage has 1x4 switch elements (SEs). Each of the following stages has 4 x 4 crossbar SEs. The last stage is a buffer and each stage has N SEs. Each output can receive up to four cells in a switching cycle. This provides a high throughput. However 4x4 switching elements are very complex.

Important parameters such as pin count, power consumption, and physical size of the chip limit the size of switches implementable on a single-chip. To overcome these limitations large switches can be constructed by 3-stage interconnection of small switch modules. The performance of these types of switches is dependent on the efficiency of the routing algorithm. A 3-stage Clos interconnection is depicted in Figure 2.9.

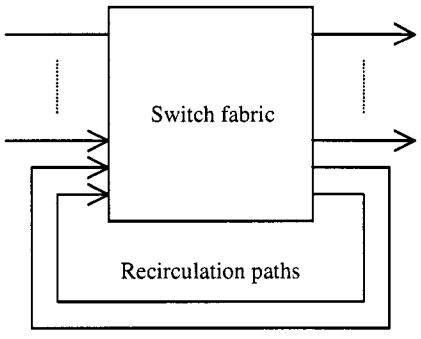


Figure 2.10 Recirculation switch

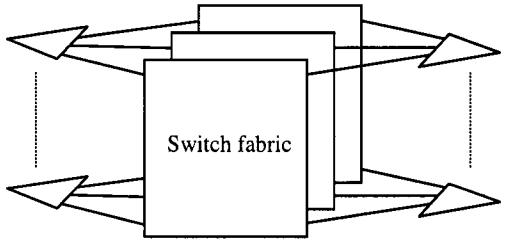


Figure 2.11 Multiplane switch

The recirculation switches are designed to address the output-blocking problem. As illustrated in Figure 2.10, cells which cannot reach their desired outputs during the current switching cycle are sent back to the inputs via recirculation paths. This reduces the cell loss and improves the throughput of the switch. However this increases the hardware complexity and also requires some mechanism to maintain the sequencing among the cells. The Sunshine switch is a well-known example of the recirculation switches [19].

In multiplane switches more than one switching plane is used to increase the throughput of the switch. A typical multiplane switch is illustrated in Figure 2.11. In case of interconnect contention free multiplane switch, the main purpose is to increase the throughput and reliability. On the other hand for switches suffering from interconnect contention, multiple planes are used to decrease the interconnect contention and increase the throughput of the switch. The parallel banyan switch is an example of multiplane switches.

2.3 Performance Enhancement of Crossbar Switches

Due to their modular nature and simplicity the crossbars are an attractive choice for high performance packet switches. However crossbars with single FIFO at input have limited throughput due to head of line blocking. Several modifications have been devised to enhance the performance of the crossbar switches like multiple input queueing (MIQ), virtual output queueing (VOQ), switch fabric speed up, and multiplane switch fabric. We

briefly discuss performance, merits, and drawbacks of these schemes in the following subsections.

2.3.1 Multiple Input Queueing

To overcome the throughput limitations of a crossbar with single FIFO queue at each input, m queues can be used at each input port, where $m \leq N$. This is referred to as multiple input queueing. The main idea is to increase the opportunity for cells arriving at the inputs. This significantly decreases the head of line blocking problem and provides significant increase to the throughput of the switch. However as the m increases, the complexity of the scheduling schemes also increases.

2.3.2 Virtual Output Queueing

Virtual input queueing is a special case of multiple input queueing. Instead of having m queues at each input, N FIFO queues are employed at each input, one for each output. It completely eliminates the head of line blocking. However it needs schedulers to resolve the internal and output blocking. VOQ switches have high throughput for uniform traffic however for non-uniform traffic their performance degrades considerably with the increase in the burstiness.

2.3.3 Speedup

In speeded up switches, the switch fabric is operated S times as fast as the line rate so that more than one packet can be transferred to the each output. The speed up of

the architectures necessitates the employments of output buffers. For a speeded up switch fabric the memory at output is also required to operate S times as fast as the line rate to enable it to store the cells routed to it during a switching cycle.

2.3.4 Multiplane Switch

In a multiplane switch, k planes are used in parallel to transfer up to k cells to each output during a switching cycle. A cell coming at an input can randomly select one of the planes, or the first cell is sent through the first plane and second through the second plane and so on. The former scheme is referred to as random loading and the latter as sequential loading. This arrangement significantly increases the throughput. However it comes at the cost of increased hardware complexity. As well, the speed of memory at output ports must be increased by k.

2.4 Scheduling Algorithms

The input queued switches, which are not self-routing, require schedulers to resolve the output port contention. Schedulers are also referred to as arbiters. The scheduling problem can be considered as finding a matching in bipartite graph. A graph is said to be bipartite if its nodes are divided into two sets and each edge has an end in one of the sets. In case of switches the inputs and outputs form the nodes and connections between them are the edges. The request and match graphs are shown in Figures 2.12 and 2.13, respectively.

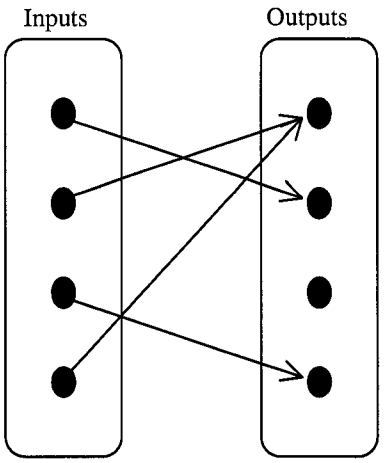


Figure 2.12 A request graph

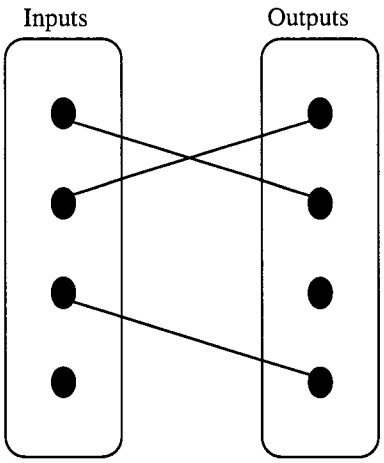


Figure 2.13 A match graph

Normally, a request is sent from the input ports for their head of line cells and the scheduler finds the best configuration of input-output pairs. This scheduling process is an overhead and increases with increase in the size of the switch. Therefore, these scheduling algorithms are required to be fast, fair and facilitate easy implementation in hardware. These scheduling algorithms can be classified as maximum size matching and maximum weight matching scheduling algorithms [20].

The maximum size matching algorithm tries to maximize the number of input-output pairs. These algorithms are stable for independent uniform traffic, however these are unstable for non-uniform traffic and some queues at the inputs can be starved along with the reduced throughput. These algorithms are also complex to be implemented in hardware. The most efficient maximum size matching algorithm currently known converges in $O(N^{2.5})$ running time.

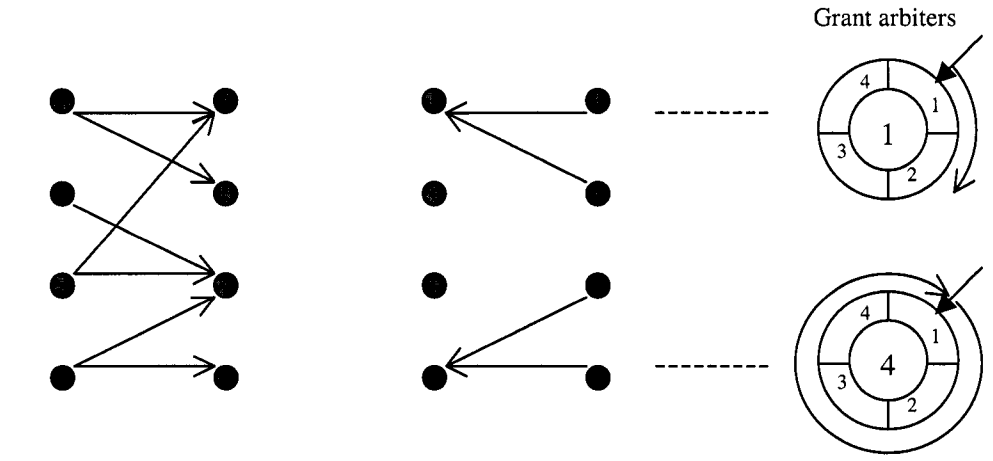
In maximum weight matching, the algorithm assigns a weight to each input queue. It finds the input-output match with the highest weight. The weight assigned to each queue is normally equal to the occupancy of the queue therefore the queue with the most cells has the highest size. These algorithms are stable for both uniform and non-uniform traffic and queues are not starved unlike in case of maximum size matching. However, the most efficient known maximum weight matching algorithm converges in $O(N^3 logN)$ running time. As well, maximum weight matching algorithms require multi-bit comparators to compare the weights of the queues.

Some of the early well-know scheduling algorithms are Longest Queue First (LQF) [21], Oldest Cell First (OCF) [22], Parallel Iterative Matching (PIM) [23], and iSLIP [24]. LQF is impractical, since it uses a maximum weight algorithm, which makes it too complex for high speed implementation in hardware. Similarly, OCF uses a maximum weight algorithm which makes it too complex to implement in fast and simple hardware, and hence unsuitable for use in high-bandwidth switches. PIM attempts to quickly converge on a conflict-free match in multiple iterations. On average, it converges in $O(\log_2 N)$ iterations, although in the worst case it may take up to N iterations. In the following we describe iSLIP in some detail.

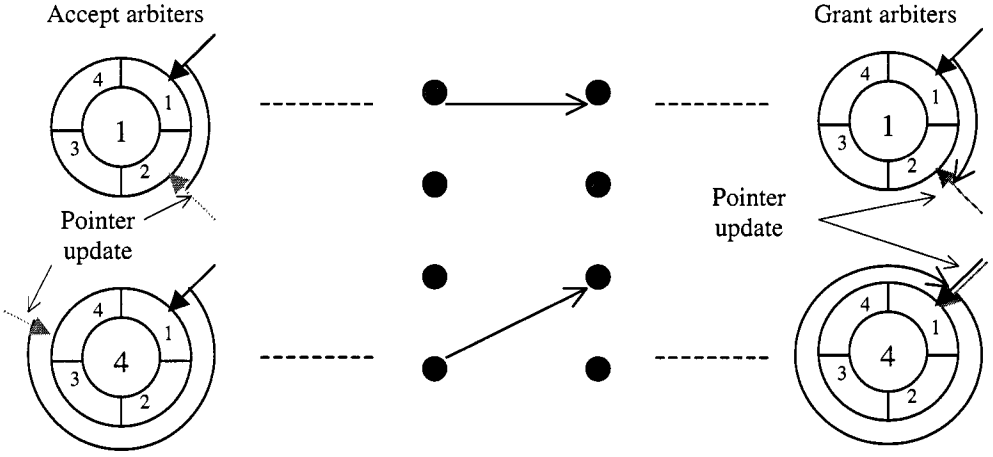
2.4.1 iSLIP

It is a maximum size matching algorithm which tries to find a maximal size match iteratively for multiple input queued switches. Each input and each output has one round

robin arbiter, and each arbiter uses a pointer to point to the highest priority output or input. In each iteration, the arbitration is carried out in three steps:



(a) Request phase (b) Grant phase



(c) Accept phase

Figure 2.14 iSLIP scheduling

A: Every unmatched input sends a request to every output for which it has a queued cell.

B: An unmatched output, starting from the highest priority input, searches in round-robin fashion and chooses the first requesting input. The output, then, notifies each input whether or not its request was granted.

C: An unmatched input, similarly, starting from the highest priority output, searches in round-robin fashion and accepts the first granting output. At this step, the pointer updating takes place. The pointers at both the accept and the grant arbiters are incremented to one point beyond the selected inputs and outputs, respectively.

This algorithm has been shown to achieve 100% throughput for uniform traffic with independent arrivals [24], and at the same time it can be implemented using very simple hardware. The algorithm converges in an average of $O(\log N)$, and a maximum of N, iterations.

In Figure 2.14, we present an example of iSLIP scheduling for a VOQ switch of size 4. In the request phase, as shown the in Figure 2.14(a), each input requests the output for which it has a cell. In the grant phase, as shown in Figure 2.14(b), the grant arbiter 1 selects input 1 and grant arbiter 4 selects input 4 in a round robin fashion. In the accept phase, as shown in Figure 2.14(c), the accept arbiters 1 and 4 select grants from outputs 1 and 3, respectively, and increment the pointers to the points beyond the accepted outputs. Similarly, in this phase, the grant arbiters increment the pointers to the points beyond the inputs that have accepted the grant signals.

2.5 Summary

In this chapter we presented an overview of packet switch architectures. We started with the packet switch classification based on switch fabric and described the advantages and disadvantages of both time and space division switches. We described crossbar switch in some detail with emphasis on buffer locations. Then we moved on to performance enhancing mechanisms usually adapted for crossbar switches. At the end we reviewed the scheduling and described the iSLIP scheduling algorithm is detail. In the next chapter, we will describe the performance enhancing schemes in detail. We will also describe pipelined scheduling adapted for eliminating the scheduling overhead.

Chapter 3

Throughput Enhancement Techniques

In the previous chapter, we reviewed the existing switch architectures and briefly described some of the techniques adapted for enhancing the throughput of input queued switches. In this chapter, we provide a detailed analysis of throughput enhancing techniques, such as the use of multiple input queueing, speeding up the switch fabric and using multiplane switch fabrics. The speeding up of the switch fabric or multiplane switch fabric necessitates the employment of queues at the output ports; therefore, the switch is no longer a pure input queued switch.

We begin this chapter with the description of popular traffic models used for simulation and analysis of switch architectures and move on to a detailed description of multiple input queueing (MIQ) followed by combined input-output queueing (CIOQ). We then describe the provision of speedup using multiplane switch fabrics. At the end we describe the pipelining of the scheduling process that eliminates the scheduling overhead

and provides a large time window for implementing slow and complex but fair scheduling algorithms.

3.1 Traffic Models

Analytical modeling, computer simulation, and experimentation are used to determine the behaviour of packet switches. These techniques require precise traffic models, which can emulate the characteristics of the actual traffic. On the other hand, if the traffic model fails to represent the actual traffic, the switch performance may be underestimated or overestimated.

Two processes, namely, the traffic arrival process and the distribution of the traffic across destination ports, describe the traffic models. The traffic arrival at the input ports can be modeled by different traffic models such as the uniform random traffic (URT) and bursty traffic models. The crossbar switch is nonblocking when the distribution of cells across outputs arriving in a switching cycle is a permutation of outputs, whereas it is blocking under URT or bursty traffic conditions that emulate the actual traffic much better than permutation traffic. Therefore we use only the URT and bursty traffic models for the performance evaluation of the packet switch architecture described in this thesis. In these traffic models, the distribution of the traffic across the outputs is uniform.

3.1.1 Uniform Random Traffic

The main advantages of using uniform random traffic (URT) are that less overhead is involved in the traffic generation compared to others, analytical modeling of URT is feasible for most of the switching architectures, and URT provides more realistic loads than permutation traffic. Here, overhead means hardware complexity of generating the traffic for test purposes. Generating uniform random traffic is very simple and requires less hardware. For bursty traffic, the traffic generator state is stored in registers, which increases the size of the traffic generator and slows down the process. Due to these reasons, URT is the most common traffic model used for evaluating the performance of packet switches and has been used for performance study of almost all proposed switching architectures [18].

In URT, a cell arrives at an input port during a timeslot with probability p, and the probability of no arrival is 1- p. Cells arrive at each input port with same probability. Therefore p represents the traffic load at each input port. The incoming cell selects its output destination randomly and independently from all the other cells arriving at the different inputs. Any output port can be selected with a probability of 1/N, where N is the size of the switch.

3.1.2 Bursty Traffic

As we mentioned in Chapter 1, an integrated broadband communications network is required to support different existing and emerging high bit rate services, such as videoconferencing, video on demand, medical imaging etc. The traffic originating from

these sources is usually bursty. A bursty source generates traffic at a peak rate for a short time and remains almost inactive between these slots. The bursty traffic model is considered to be more realistic than uniform random and permutation traffic models.

Several traffic models have been proposed in the literature to model this burtiness in the traffic arrival. Out of these, the ON-OFF model is the most popular and widely used to generate bursty traffic [18]. We use the ON-OFF model to generate bursty traffic and observe the behavior of the packet switch architecture presented in this thesis. In the ON-OFF traffic model, during the ON period the source sends the cells to the same destinations and the source is idle during the OFF period. The durations of both these ON and OFF periods are independently determined from two geometric distributions. Uniform random traffic is a special case of bursty traffic model with ON period length being 1. The duration of ON and OFF periods are evaluated in terms of time slots from two independent geometric distributions as [25]

$$L = 1 + \left\lfloor \frac{\ln(1-R)}{\ln(1-p)} - 1 \right\rfloor, \tag{3.1}$$

where L is the length in terms of timeslots, R, $0 \leq R < 1$, is the random number generated and p, $0 < p < 1$, is the inverse of mean burst length. The above equation is used for generating burst length for both ON and OFF periods for the specified mean burst length. Users are required to specify two parameters: link load ρ and the average burst length for ON or active period. The mean OFF or idle period length can be obtained from the following:

$$L_{OFF} = L_{ON} \times \frac{1-\rho}{\rho}. \tag{3.2}$$

Cells are continuously generated during the ON period and are destined to the same output port which is chosen uniformly among all the output ports and independently from other bursts. Each output port can be chosen with a probability of 1/N.

3.2 Multiple Input Queueing

To overcome the throughput limitation in input queued switches with a single FIFO queue at the input ports, many alternatives have been devised. Out of these the multiple input queueing (MIQ) technique is drawing much attention recently due to the high throughput and high-speed operation capability [26]. The main idea of multiple input queueing is to allow the cells behind the blocked head of line to be switched to a free output port. A MIQ switch deploys m, $1 < m \leq N$, queues at each input port. Each of these m queues stores packets destined for a particular group of the output ports. These groups do not overlap with each other. Each input port can transfer up to m cells to output ports during a time slot, however, each output port can receive only one cell during the same time. This is referred to as the free rule in MIQ switches. Its alternative, the restricted rule, restricts the number of cells switched from each input port during a time slot to one [27]. However, the total number of cells switched from all the inputs cannot exceed N for both free and restricted rules.

An 8 x 8 MIQ switch is illustrated in Figure 3.1. This switch has four FIFO queues at each input port and employs the free rule. The first queue stores the packet destined to output ports 1 and 2, the second queue stores the packets destined for output ports 3 and 4, and so on. If m is equal to one, there is a single queue for all the output

ports, and the switch is a conventional input queued switch. On the other hand, if the restricted rule is employed and m is equal to the size of the switch (m = N), each queue is dedicated to one output port, and the arrangement is referred to as virtual output queued (VOQ) switch.

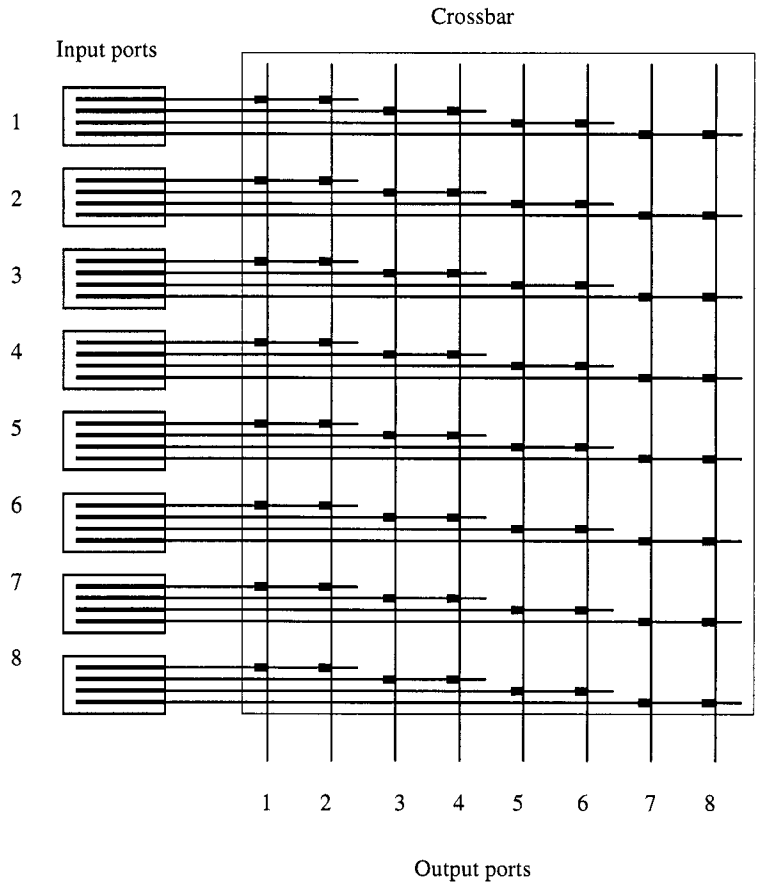


Figure 3.1 A 8 x 8 MIQ switch with m = 4

In [28], Christos Kolias et al. presented the throughput analysis and simulation results of a MIQ switch with crossbar switch fabric and showed that maximum throughput of a MIQ switch can be computed as

$$\gamma_{max} = m + 1 - \sqrt{1+m^2} \quad , \tag{3.3}$$

where γ_{max} is the maximum throughput of the switch. In this analysis, uniform random traffic model was used for modeling traffic arrival. Table 3.1 shows the throughput of MIQ switches for different numbers of queues at each input port for both restricted and free rule presented in [26].

One argument against the free rule is that it requires expansion in the switch fabric and thus increases the cost. However, from the table it is obvious that to achieve similar performance as for the free rule, the restricted rule application requires a large number of queues at the input ports. For example, to obtain a throughput in excess of 99%, 64 queues are required at each input port for the free rule but for the same performance 256 queues are required for the restricted rule.

The mean cell delay in the pure output queued (OQ) switches is shown to be [13]

$$D_{mean} = \frac{p}{2(1-p)} \quad , \tag{3.4}$$

where p, $0 \leq p < 1$, is the traffic arrival rate. A switch that has no interconnect contention and whose output ports are capable of accepting all the cells destined to them during a switching cycle, are known as an ideal output queued switches. The ideal output queued switches are known for their high throughput and are suitable for providing QoS guarantees.

Table 3.1 Maximum throughputs of MIQ switches for different number of queues at each input port

Number of Queues at each Input Port m	Maximum Throughput (free rule) γ_{max}	Maximum Throughput (restricted rule) γ_{max}
1	0.586	0.586
2	0.746	0.705
4	0.877	0.802
8	0.938	0.873
16	0.969	0.921
32	0.984	0.953
64	0.992	0.972
128	0.996	0.984
256	0.998	0.991

Figure 3.2 illustrates the delay performance a ideal output queued switch and MIQ switches with different numbers of queues at each input port. These results are obtained from simulation of a 128 x 128 MIQ switch with restricted rule and 2, 8, and 16 FIFO queues at each input port. The delay performance of the OQ switch is obtained from Equation 3.4.

From these results, we observe that the MIQ switch with 16 queues at each input port provides delay performance close to that of an ideal OQ switch. However, to approximate the performance of an ideal OQ switch, more than 16 queues are required at each input. This large number of queues at each input port increases the complexity of the input port controllers. Another problem with MIQ switches is that the provision of QoS guarantees in MIQ switches is too complex. Scheduling the switch fabric for packet

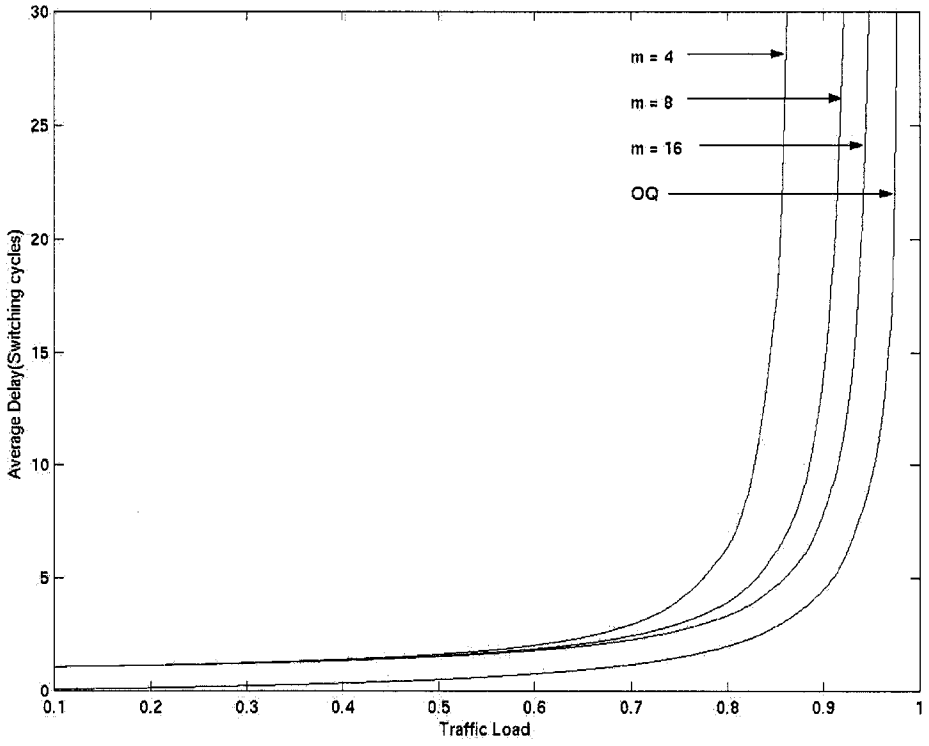


Figure 3.2 Delay performance of an ideal OQ and 128 x 128 MIQ switches

transfer to the desired output ports and scheduling the packets to provide QoS guarantees, simultaneously, is a two dimensional process.

Employing multiplane switch fabric or speeding up the switch fabric can also enhance throughput of the IQ switches. The problem associated with the speed up is that it makes the switch unsuitable for high speed switching. Thus, it is one of the important issues which is required to be resolved for high-speed switching. At the same time, speeding up the switch fabric or using multiplane switch fabric requires queues at the output ports. Therefore, the switch becomes a combined input-output queued switch.

3.3 Combined Input-Output Queuing

Combined input-output queuing (CIOQ) overcomes the problems posed by input queueing and output queueing by allowing the use of slow speed memories and mitigating the effect of HOL blocking. When more than one cell contend for the same output, some are sent and others are temporarily queued at the input before being transferred to the outputs. In CIOQ, the high throughput can be achieved by slight speedup or providing multiple paths from inputs to outputs or parallelism. The parallelism factor is the number of planes in a multiplane switch architecture. The speedup is obtained by operating the switch fabric S times faster than the line rate and parallelism is achieved by using multiple parallel switching planes as a switch fabric to improve the throughput.

The speedup or parallelism factor can be anywhere between 1 and N, however, usually it is a small constant. A switch with speedup or parallelism factor of 1, stores all the incoming packets at the input port and no queueing is required at the output port as only one cell can be transferred to each output during one switching cycle. Therefore it is a pure input queued switch. Similarly, a switch with the speedup or parallelism factor of N doesn't require any input buffer, as all the cells coming at the input ports can be instantaneously switched to the desired output ports, and therefore it is a pure output queued switch.

Employing speedup or a multiplane architecture improves the throughput. Thus the delay performance close to that of an ideal OQ switch can be achieved. In the following sections, we attempt to determine the speedup or parallelism required to

emulate the OQ switch. We analyze the performance of speeded up and multiplane switch architectures and compare this with the ideal OQ switch.

3.3.1 Multiplane Switches

Multiplane switches are constructed from multiple identical, space division, switches [29]. These multiple switching planes improve the throughput of the switches by reducing the effect of head of line blocking that limits the throughput of input queued switches. However, this improved performance comes at the cost of more hardware which increases the cost, and thus, use of multiplane switches is a tradeoff between performance and cost of the switch architecture.

In multiplane switches, the speed of switch fabric is the same as that of the line rate. Therefore, multiplane switches are an attractive alternative to speeded up switches [30]. The multiplanes allow multiple cells to be switched to the output ports simultaneously and independently. In multiplane switches cells arriving at the input can pick a switch plane randomly or switch planes can be loaded sequentially. In the multiplane switch fabric, m cells can be transferred to the same output, where m is the number of the switch planes. However, each input can send only one cell during each switching cycle.

The maximum throughput of a multiplane switch with crossbar as the switching plane and singe FIFO queue at each input is shown to be [29]

$$\gamma_{max} = m + 1 - \sqrt{1+m^2} \ . \qquad (3.5)$$

Table 3.2 Maximum throughput of multiplane switches

Number of Planes m	Maximum Throughput $\gamma_{\max}$
1	0.586
2	0.746
4	0.877
8	0.938
16	0.969
32	0.984
64	0.992

Table 3.2 provides the maximum attainable throughput for multiplane switches for different numbers of planes. We observe that to obtain a throughput in excess 99%, at least 64 parallel planes are needed, which requires substantial hardware.

The mean delay of the multiplane switch is the sum of the mean delay through input ports and mean delay through output ports

$$D_{mean} = D_{in} + D_{out}. \tag{3.6}$$

The mean delay through input ports for a multiplane switch with single FIFO queue at each input and crossbars as the switching planes is given by [29]

$$D_{in} = \frac{11\lambda^4 - (32m+6)\lambda^3 + (48m+30)m\lambda^2 - (24m+48)m^2\lambda + 24m^3}{6(m-\lambda)(2m-\lambda)(\lambda^2 - 2(1+m)\lambda + 2m)}, \tag{3.7}$$

where λ is the traffic arrival rate. Similarly, the mean delay through output ports of the multiplane switch is given by

$$D_{out} = \frac{\gamma(1-\frac{1}{m})}{2(1-\gamma)}. \tag{3.8}$$

Equation 3.6 provides the mean delay for multiplane switches when the arrival rate is less than the maximum throughput. As the throughput reaches the maximum attainable throughput, the average delay approaches infinity.

In Figure 3.3, we show the delay performances of multiplane switches with crossbars as switching planes and an ideal OQ switch, obtained from Equations 3.6 and 3.4, respectively. We observe that the delay performance of switches with 4 and 8 parallel planes is not comparable to that of the OQ switch at high traffic loads. However,

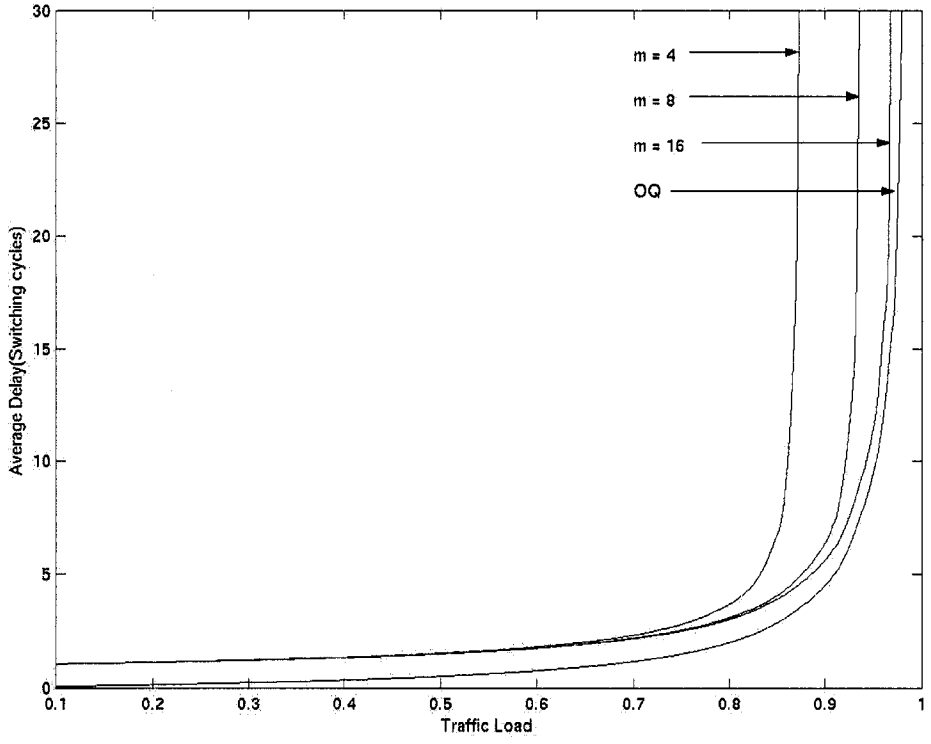


Figure 3.3 Delay performance of an ideal OQ and multiplane switches

as the number of switching planes is increased to 16 the performance of the switch is much closer to the OQ switch. Both, throughput and delay performance of the multiplane switches show that a large number of switching planes are required in multiplane switch for emulating the performance of the ideal output queued switch.

3.3.2 Speedup

The switch fabric can be speeded up to overcome the head of line blocking problem. A switch with speedup S transfers up to S packets from each input port to output ports. Similarly, each output port is capable of storing S packets. In speeded up switch fabric, the number of cells that can be sent from each input port and the number of cells that can be stored in each output port are the same. Therefore, for a single FIFO queue at each input and a speedup of 4 or 5, the impact of head of line blocking is minimal. However, another problem arises; that is, the switch fabric speedup can limit the speed of the switch and thus its performance. We observed from the simulation of a 128 x 128 switch with single FIFO queue at each input port for uniform random traffic model, a speedup of 2 or more provides throughput in excess of 99%.

Figure 3.4 shows the delay performance of the speeded up switches compared to the ideal OQ switch. The delay performance of speeded up switches is obtained from the simulation of a 128 x 128 switch for uniform random traffic and delay performance of the ideal OQ switch is obtained from Equation 3.4. From these results, we observe that for a speedup of 2 and 3, a delay performance close to that of the ideal OQ switch is achieved.

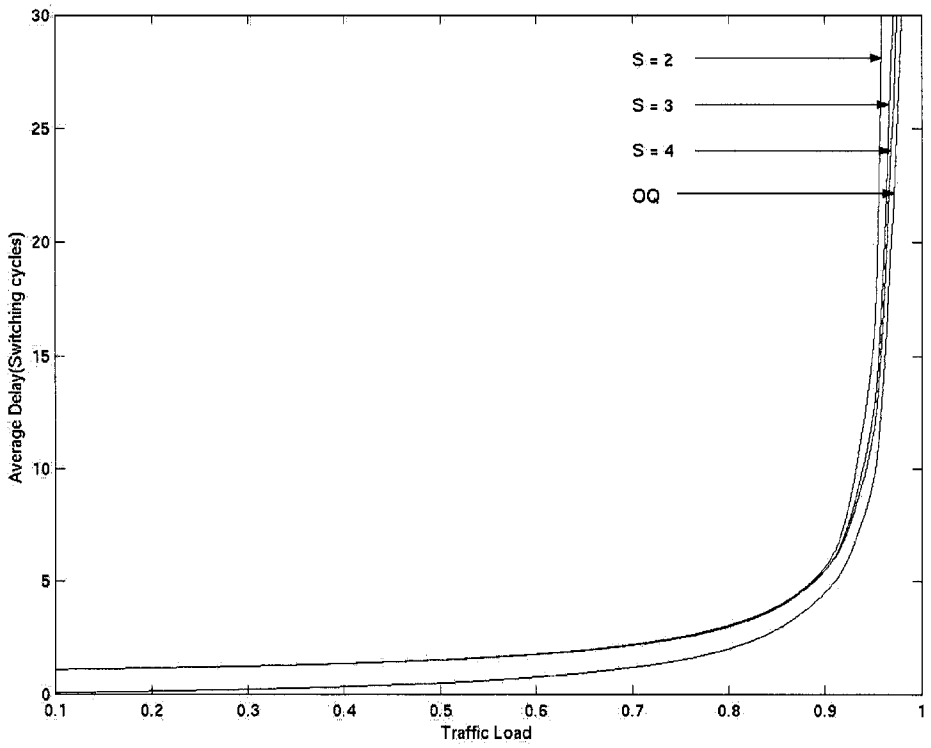


Figure 3.4 Delay performance of an ideal OQ and 128 x 128 speeded up switches

However, the throughput of the speeded up switch with S = 2 and S = 3 for bursty traffic with mean burst length of 15, is significantly degraded. On the other hand, the throughput of the speeded up switch with S = 4 even for bursty traffic remains almost the same as for uniform random traffic. Table 3.3 provides the throughput for speeded up switches for URT and bursty traffic. For bursty traffic the mean burst length equal to15 was used.

Figure 3.5 shows the average delay through input ports of speeded up switches with S = 2, S= 3 and S= 4. From the delay performance, we observe that the average delay through input ports of a speeded up switch with S = 2 and S = 3 increases with the

Table 3.3 Throughput of speeded up switches

Speedup S	Throughput for URT γ	Throughput for bursty traffic γ
2	0.999	0.982
3	0.999	0.988
4	0.999	0.996

increase in the traffic load, whereas, the average delay through input ports of a speeded up switch with S = 4, remains approximately the same for different traffic loads. Therefore packets suffer most of the delay at the output ports as the average delay through input ports is less than 2 switching cycles and remains so even for higher traffic loads. This small delay through input buffers, allows the provision of Quality of Service (QoS) guarantees in a similar fashion as in OQ switches by employing the well-known scheduling algorithms that exist for OQ switches, at the output ports of speeded up CIOQ switches.

Though speedup improves the performance of the switch significantly, speeding up the switch fabric can limit the size of the switch. At the same time, we observed that if we employ a multiplane switch fabric, a large number of planes is required to obtain a performance comparable to a speedup switch with S = 4. In the next section, we show how speedup can be achieved using a multiplane switch fabric.

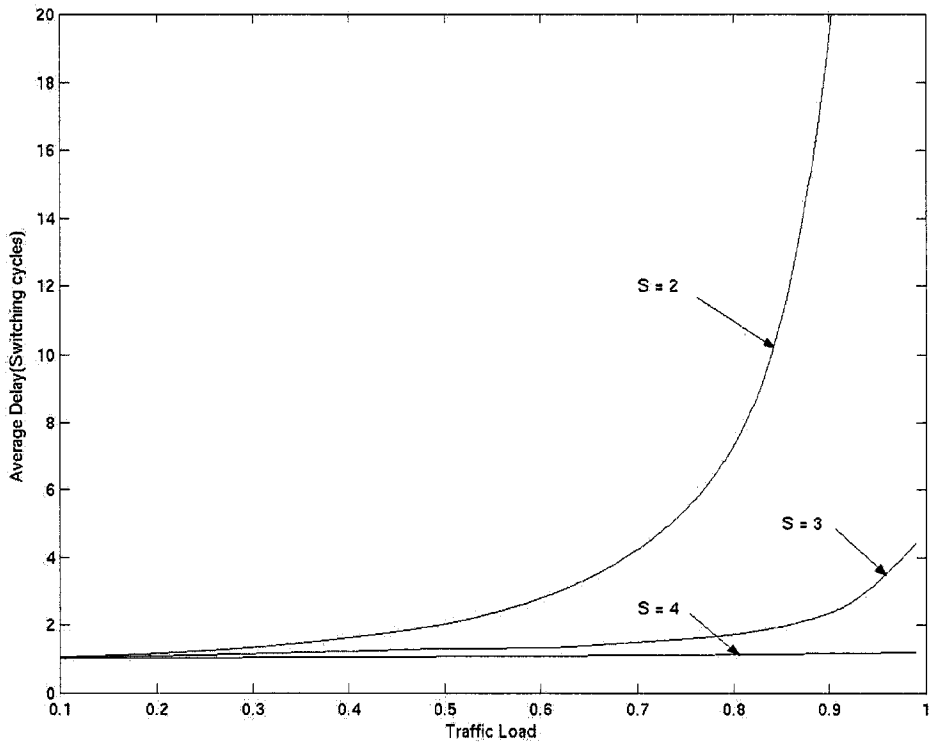


Figure 3.5 Average delay through input ports of 128 x 128 speeded up switches for bursty traffic

3.4 Speeding up using Multiplane Switch Fabric

As we described in the above section, speeding up the switch fabric provides the desired throughput and delay performance to emulate the throughput and delay performance of the OQ switch. However, speeding up the switch fabric is unsuitable for high-speed switches.

In this section, we describe how we obtain the same characteristics as for speeded up switch using multiple switching planes and eliminate the need for operating the switch

fabric faster than the line rate. We observed in the previous sections of this chapter that the multiplane switches using crossbars as the switching planes can provide the desired high throughput, however, it comes at the cost of extra hardware. To emulate the ideal output queued switches using a multiplane switch requires more than 64 switching planes. Similarly, we observed that a speeded up CIOQ can emulate an ideal output queued switch with a speedup of 4. However, this speedup may limit the speed of the switch.

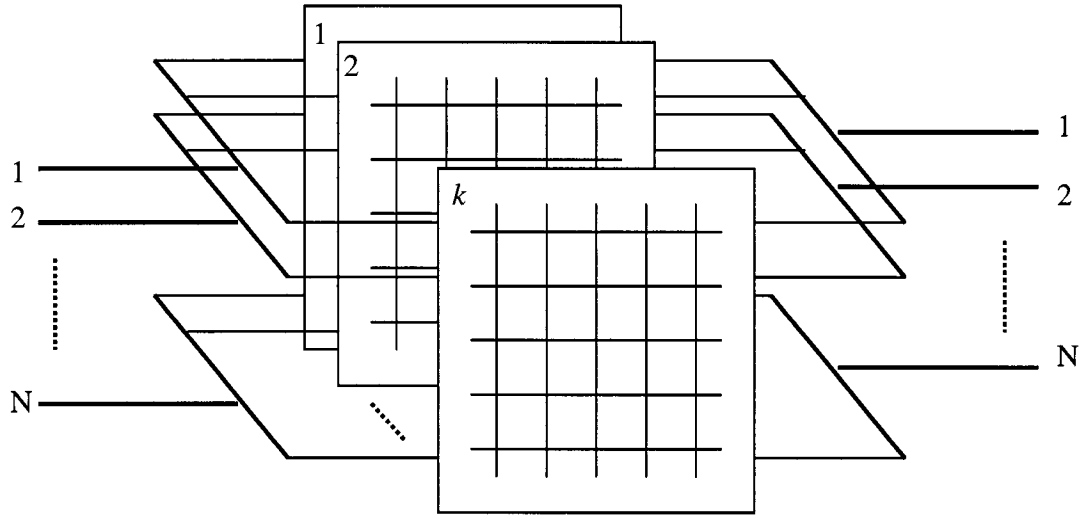


Figure 3.6 Switch fabric with k crossbars in parallel

If we use *k* crossbars as shown in Figure 3.6 and instead of switching more than one cell through separate planes to the same output port, a cell is transferred in parallel through all the *k* planes simultaneously, a complete cell transfer from input port to the

desired output port will take $1/k$ of the switching cycle. This parallel transfer provides the speedup equivalent to the number of planes without operating the switch fabric faster than the line rate. That is, for k parallel switching planes a speedup k can be obtained. A similar technique has been used in [31]; however, it was not employed for improving the throughput.

This design has two benefits. The first is that the limitation of only one packet from each input is eliminated. Therefore more than one cell can be switched from each input port and the second is that it provides the same delay performance as a speeded up switch while operating the switch fabric at the line rate.

In the distribution analysis of destinations performed by Li Cheng [32], he showed that under unicast uniform random traffic at full load, in any given switching cycle, the probability that there are more than four cells arriving in any given cycle at the input ports destined to a particular output port is approximately 1% and that this is almost irrespective of the size of the switch. Furthermore, we conducted simulations with different numbers of planes and found that the performance improvement became decreasingly significant as the number exceeded 4.

3.5 Scheduling

CIOQ switches require lower speed memories and switch fabrics, however, they require a traffic scheduler to configure the switch fabric before the transfer of packets from the input ports to the output ports. The switch fabric is required to be configured during each time slot to determine which cells are to be transferred to the output ports. In

crossbar based CIOQ switches we need a centralized or decentralized scheduler to provide the input-output pairs. These schedulers control the state of the respective crosspoints.

This scheduling process is an overhead and increases with the increase in the size of the switch and may limit the size of the switch. In Figure 3.7, *A* shows the conventional way of scheduling the switch fabric before transferring the packets from input ports to the output ports. As the switch size increase; the size of the scheduling window increases, which reduces the packet transfer window size. To compensate the decrease in packet transfer window, we need to speedup the packet transfer from input ports to the output ports or we can employ simple scheduling algorithms for decreasing the scheduling time window. These simple scheduling algorithms are fast but no fairness guarantees are provided. A scheduling algorithm is said to be fair, if it provides the same share of available resource, to all the input ports in a short time interval.

If the scheduling and packet transfer from input ports to the output ports are performed in parallel, the scheduling time problem can be limited. However, this single stage pipelined scheduling process will introduce a delay equivalent to the packet transfer time from input port to output port. In a speeded up switch architecture this delay would be 1/S times a switching cycle, which is not that significant. In Figure 3.7, *B* shows this single stage pipelining for S = 4. This provides a time window equivalent to 1/4 of a switching cycle, called subcycle, for scheduling the switch fabric. For a link rate of 1Gbps and packet size of 53-byte, a time window of 424ns is available for configuring the switch fabric.

With the above-mentioned parallelism, the same size time window is available for configuring the switch fabric as for the packet transfer. The fair scheduling algorithms presented in the literature for scheduling the switch fabric are slow and very complex [21][22], therefore this parallelism allows us to use those complex but fair algorithms for configuring the switch fabric.

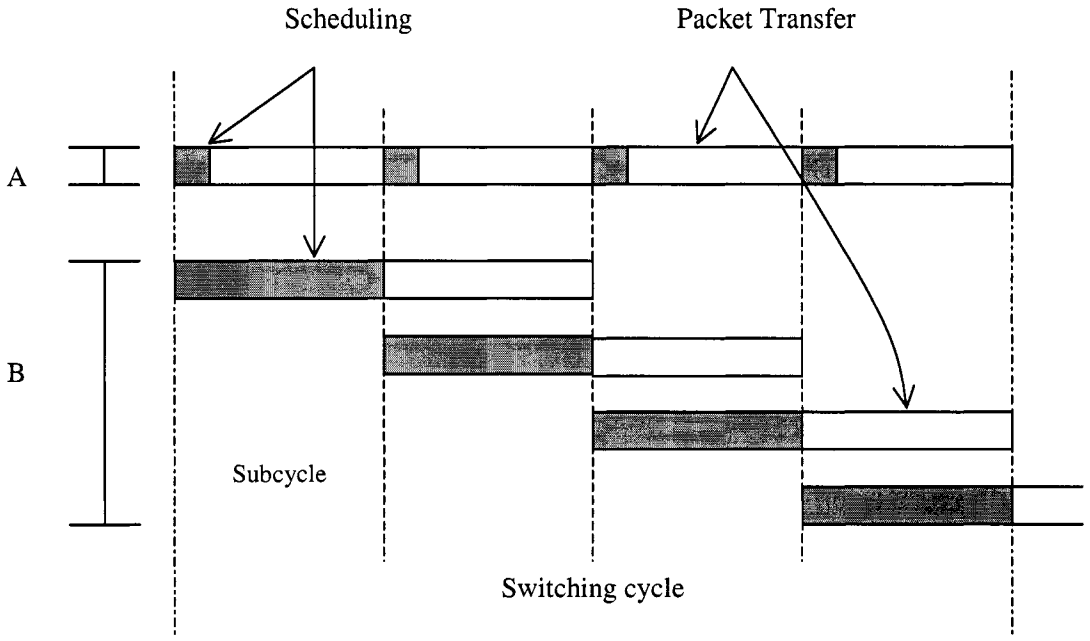


Figure 3.7 Single stage pipelining of the traffic scheduling

3.6 Summary

In this chapter, we described the techniques used for enhancing the throughput of the input queued switches such as multiple input queueing, multiplane switch fabric or speeding up the switch fabric, in detail. We started this chapter with the description of

traffic models used for simulations. We observed that speedup improves the performance of the switch significantly, however, speeding up the switch fabric can limit the size of the switch. At the same time, we observed that if we employ multiplane switch fabric, a large number of planes are required to obtain a performance comparable to a speeded up switch with S = 4. We showed how we can achieve the performance of a speedup switch using multiplane switch fabric. At the end of this chapter, we explained the scheduling problem in switches with buffers at the input ports and suggested the use of parallelism to eliminate the scheduling overhead. In the next chapter, we will describe the architecture of a packet switch which employs the combined input-output queueing, enhanced crossbar, and pipelined scheduling described in this chapter.

Chapter 4

Switch Architecture and Performance

In this chapter, we describe the architecture of switch that employs combined input-output queueing, enhanced crossbar, and pipelined scheduling that was described in the previous chapter. This switch architecture is suitable for use in routers as a high throughput back-plane or as an ATM switch. The switch architecture uses a multiplane switch fabric for improving the throughput. We begin this chapter with an overview of the design and move on to discuss the design of each individual block. We explain the scheduler design and describe the pipelining of the scheduling process in some detail.

Later in this chapter, the performance results of this switch architecture are presented. The performance of a switch is usually evaluated for the following three parameters: throughput, delay that cells suffer while traversing the switch, and cell loss rate or buffer occupancy. We obtained these results from simulations for both uniform random and bursty traffic arrival.

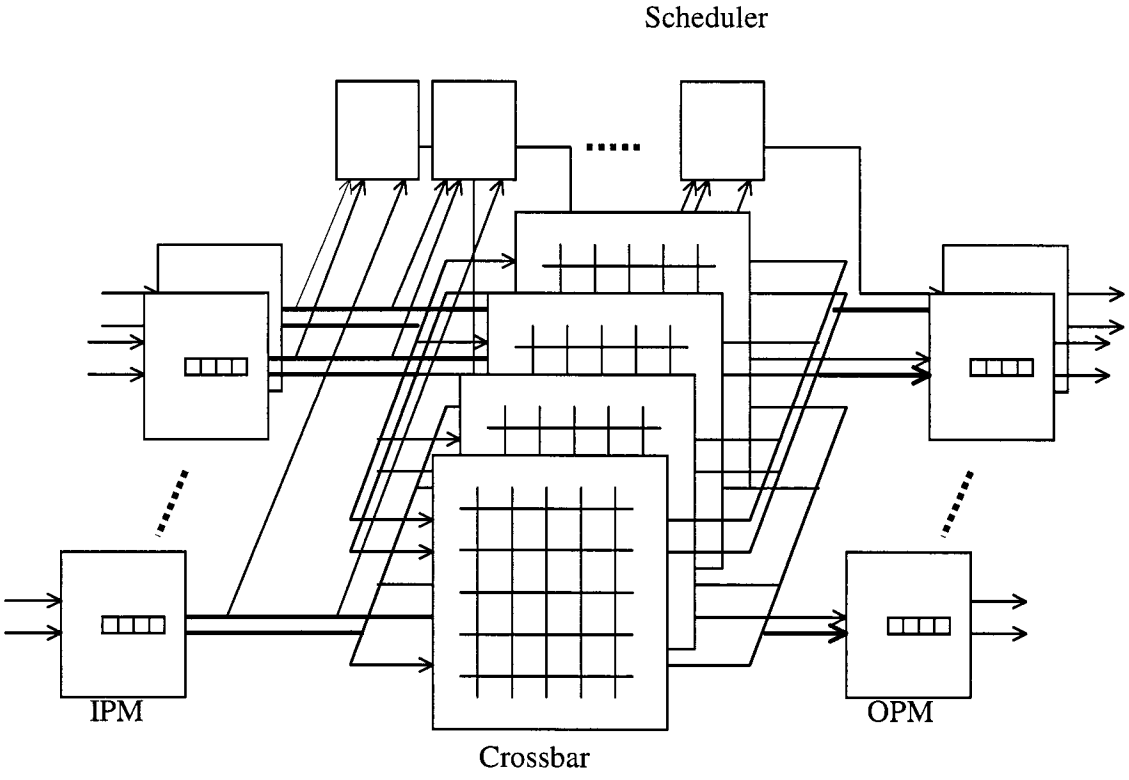


Figure 4.1 Enhanced crossbar CIOQ switch architecture

4.1 Switch Architecture

As illustrated in Figure 4.1, the main components of this architecture are the input port module (IPM), output port module (OPM), switch fabric and scheduler. The input and output port modules each have a single FIFO queue for storing the incoming packets. The switch fabric is the interconnection between input ports and output ports. The scheduler configures the switch fabric for transferring the packets from input ports to the desired output ports.

When a packet arrives at the input port, it is stored in a single FIFO queue available at the input port. The packet waits for a decision from the scheduler, which schedules the packet for transfer to the desired output port. This switch architecture has decentralized scheduling and is capable of handling fixed size packets only. Each OPM has its own scheduler to select one among the (1 to N) requesting input ports. All of these N schedulers work in parallel to select up to N input ports. At the start of each switching cycle, each input port requests the desired output port, each scheduler selects one among the requesting input ports. This selection result is passed back to the input ports and the switch fabric. Packets are then sent to the configured switch fabric and routed to the desired output ports. On arrival at the output port, packets are stored in the FIFO queues available at the output ports where they wait for transmission on the output link. When the output link is available, the head of line packet among the waiting cells is transmitted on the output link.

Two buffer control schemes are used to control buffers in combined input-output queued switches: the queue loss scheme and the backpressure scheme [33]. In the queue loss scheme a cell can be lost both at input and output buffers if the desired buffers are full. On the other hand, in the backpressure scheme cells are lost in input buffers only. If output buffers are full, the output port controller informs the input port and cells are buffered in input buffers instead of being lost at output buffers.

The backpressure scheme requires less buffer space compared to the queue loss scheme for providing the same performance. However, if the backpressure scheme is used, cell loss occurs only at the input ports and is independent of the session the cell

belongs to, which is unfair to a cell belonging to a well behaved session. On the other hand, in the queue loss scheme, a dropping scheme can be employed at output ports, which will drop cells belonging to the session whose session queues are full. Cells can still be lost at input ports if the input buffer size is too small. This architecture employs the queue loss scheme for dropping cells if desired queues are full. In the following we describe the major components of this switch architecture in some detail.

4.1.1 Input Port Module

The input port module is responsible for storing the incoming packet and requesting for the desired output ports from scheduler, and if granted, transmitting the cells to the desired output ports. The IPM is designed to be scalable in terms of packet size, however packets must be of the same size. We considered a packet size of 53 bytes throughout our work. The IPM in this architecture does not have any header processing capability for two reasons. First is that the header processing problem is well studied in the literature and a number of fast solutions exist [34][35]. Secondly, the delay associated with header processing is usually fixed, and therefore it does not affect the performance of the switch architecture.

In the speeded up switch architectures, packets are completely stored before sending them to the output ports. This architecture has only one packet queue at each input port; therefore, the input port controller can store the packet before determining the output port address to which the packet is destined. This allows the input port controller to use a large time window, equivalent to the time taken in completely storing a packet in

the queue, for determining the output port to which the cell is destined. Regardless of this, our main focus was to determine the performance of this architecture for different traffic arrival patterns; therefore, we excluded the header processing capability from the IPM.

On the arrival of the new packet, the input port controller stores the packet in the queue and requests for the packet transfer to the respective scheduler of the desired output port. If a grant is received from the scheduler, the input port controller dequeues the head of line cell and sends it to the switch fabric. Due to the high throughput provided by the enhanced crossbar switch fabric, most of the cells are switched to the output ports and very few are stored at the input ports. Therefore, the cell storing capacity required at the IPM is significantly less than that at the OPM.

4.1.2 Output Port Module

The OPM is responsible for storing the packet incoming from the IPM through the switch fabric and sending them on the output link on a first-come, first-served basis. The output port module has almost the same functionality as the IPM and is scalable in terms of packet size like the IPM. The main difference is that the OPM does not have to request for the transmission of the head of line cell waiting in a queue. Instead, if the output link is available, the head of line cell is transmitted in every switching cycle. As we shall see, the virtual speedup provided by the multiplane switch fabric allows the transfer of up to four cells to each output port during each switching cycle, therefore an output port is capable of storing up to four packets during the same period. The high

throughput provided by multiplane switch fabric significantly increases the requirement of buffer space at the output port.

The scheduler informs the OPM if a cell is scheduled for it. The output port controller stores the arriving cell in the only available queue, if there is space available. Otherwise the OPM discards the incoming cell. When the output link is free, the output port controller transmits the head of line packet in the queue on the link. Output ports do not have any QoS provision capability since we were mainly focused on the throughput and delay performance achievable with virtual speedup provided by the enhanced crossbar switch architecture.

4.1.3 Switch Fabric

In the last chapter, we showed how we can achieve the virtual speedup using multiple switching planes. We use the same switching technique in this architecture for providing the desired high throughput. The crossbar is the simplest and most widely used architecture for packet switches [36]. An N x N crossbar connects N input ports with N output ports. Any of the N input ports can be connected to any of the output ports through the crosspoint switches. A crossbar is interconnect contention free since every input-output pair has its own dedicated data path.

In this architecture, the switch fabric consists of four parallel crossbar planes. The cell switching is done in these four switching domains. The four domains transfer the cell to the output ports simultaneously and each operates at the line rate. Each IPM sends the data to the switch fabric through a 4 bit wide data bus and it is switched through each

plane bit by bit. As there are four parallel planes, 4 bits would be transmitted per clock cycle. This allows the transfer of up to four packets to each output port during each time slot. Similarly, up to four cells can be switched from each input port during the same period. The virtual speedup of four obtained through this four-crossbar arrangement provides a high throughput and limits the delay through input buffers. The crossbar is popular for the packet switches with a moderate number of ports. However, crossbars without any buffering capability at the crosspoints require schedulers for configuring them for packet transfer from input ports to the output ports.

4.1.4 Scheduler

Switches with input buffers require schedulers for configuring the switch fabric before transferring the packets from input ports to the output ports. The performance of schedulers is determined by their throughput and delay properties. However, their implementation complexity is the decisive factor in the selection of the scheduling algorithms for high-speed switches. The iSLIP scheduling algorithm is known for its high throughput, fairness and ease of implementation. Therefore we used an iSLIP scheduler in this switch architecture.

In [24], Mckeown showed that the iSLIP scheduling algorithm can achieve a high throughput for multiple iterations. However this scheduling algorithm is designed for switches with virtual output queueing at input ports where only one cell can be switched from each input port to each output port in every switching cycle. For this purpose, to decide fairly among the received grants, an accept arbiter at each input port is required to

select among the output ports which issued a grant. Due to this accept phase, multiple iterations are required to get the best match as only one grant can be accepted during a switching cycle. However, in our switch architecture every input port has only one FIFO queue therefore at the most only one grant can be received which will be certainly accepted. We called it SLIP because it is not iterative like the original iSLIP algorithm. SLIP does not need any accept arbiters. Therefore step 3 of iSLIP is eliminated in SLIP and grant arbiters update takes place in step 2 of the SLIP algorithm. This eliminates the need for these accept arbiters and multiple iterations. Therefore we use the SLIP algorithm with the first two steps, request and grant phase, for configuring the crossbars before packet transfer. Every IPM has separate request and grant lines to each scheduler. The following two steps perform the scheduling:

1. All the input ports which have cells, request for the output ports for which their head of line cells are destined.
2. The grant arbiters select one among the received requests in a round robin fashion. The grant signals are sent to the input ports and the pointers are incremented to one location beyond the selected input ports.

Figure 4.2 illustrates this scheduling process for a switch of size 4 and with a single FIFO queue at each input port. Each input port makes a request to the output port for which it has a packet stored. The output port selects the next requesting input port at or after the pointer in the round-robin schedule. Arbiters are shown here for output ports in Figure 4.2. Input ports 1, 2, 3, 4 request output ports 2, 3, 1, 3, respectively. Grant

arbiter 1 and 2 receive one request each and therefore they select the requesting input ports. However, output port 3 is requested by two input ports. Grant arbiter 3 selects one input port among the requesting input ports in a round-robin fashion as shown in Figure 4.2. Pointers are incremented to the locations beyond the selected input ports. The pointer at the output port 4 remains unchanged as none of the input ports requested the output port 4. At the completion of arbitration, three input-output pairs have been matched, which is the maximum size matching.

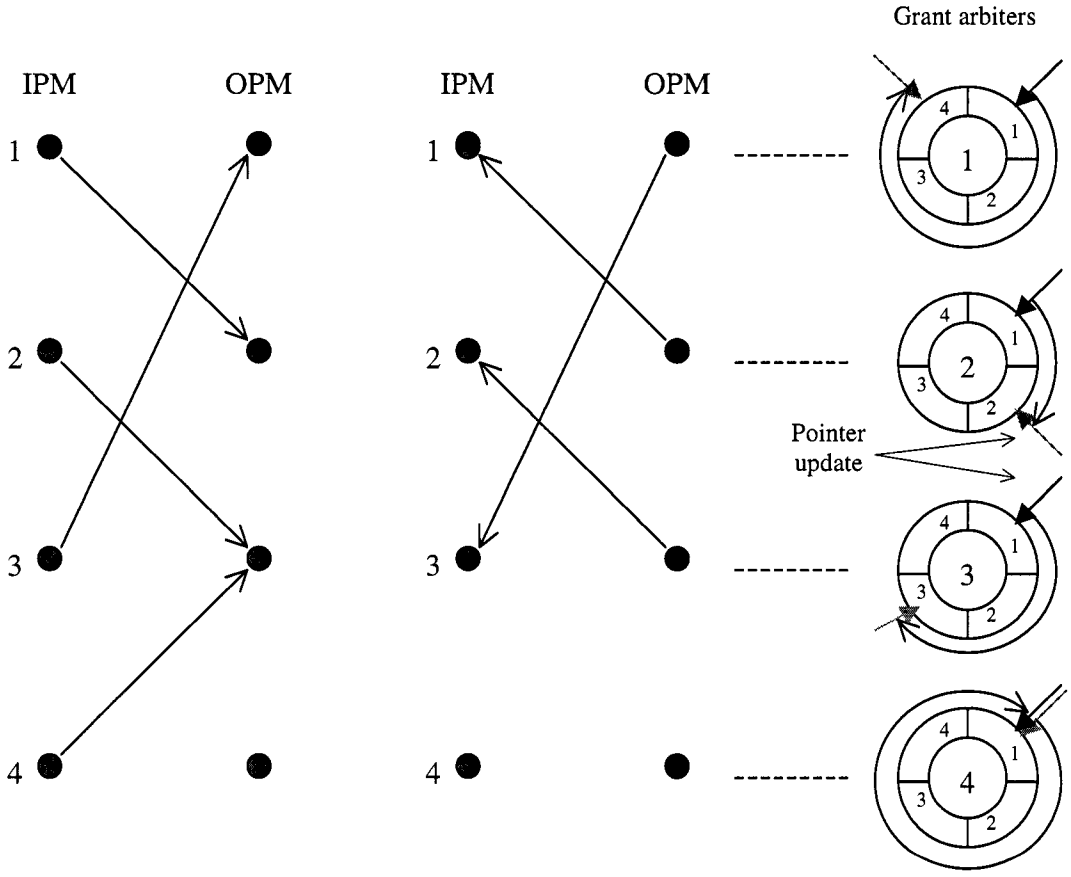


Figure 4.2 SLIP scheduling process for enhanced crossbar CIOQ switch architecture

4.2 Pipelined Scheduling

In the last chapter, we showed that by parallelizing the packet transfer from input ports to the output ports and scheduling process, the scheduling overhead can be eliminated and this single stage pipelining provides a large time window for slow and complex but fair scheduling algorithms. Similar approaches have been reported in [37][38] for CIOQ switches with multi-stage interconnection networks. This switch architecture employs this single stage pipelining for providing the large time window for schedulers for configuring the switch fabric for packet transfer from input ports to the output ports.

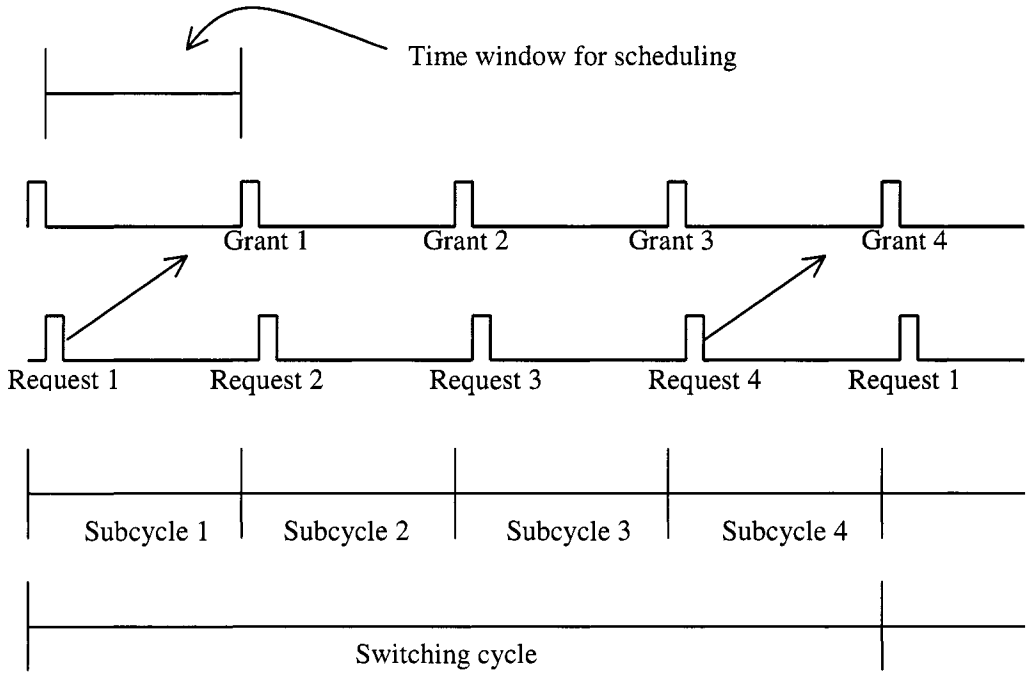


Figure 4.3 Pipelining of the scheduling process

The process is explained in the following. Each switching cycle is divided into four equal subcycles. Each input port requests for its head of line cell at the start of subcycle 1, and if selected, a grant is issued at the start of the next subcycle 2. The same process is repeated four times during each switching cycle as shown in Figure 4.3. The request 4 is granted in the first subcycle of the next switching cycle. This pipelined scheduling adds one subcycle delay for every cell traversing input buffers. For a line rate of 1Gb/s a time window of 106ns is available for the schedulers for configuring the switch fabric for cell transfer, which will decrease to 53 ns for a line rate of 2Gbps. This large time window allows the design and implementation of complex but fair scheduling algorithms. With the increase in the line rate, this time window will decrease, since an increase in line rate means a shorter switching cycle, which means a small subcycle.

The SLIP schedulers used in this architecture take less than 5ns for configuring a 16x16 switch. However with the increase in the switch size this delay would increase. For a line rate of 1Gbps, a scheduler 20 times more complex than SLIP can be implemented due to the large time window provided by pipelined scheduling.

4.3 Performance Evaluation

In this section, we present the software simulation results of the architecture described in the previous sections. The primary goal of simulation is to study the behavior of the presented switch architecture for both uniform random and bursty traffic. Performance of a switch is mainly determined by the overall throughput and the delay packets suffer while traversing the switch and the buffer requirement for achieving an

acceptably low cell loss rate. Delay performance is determined separately for input and output buffers.

We developed a C++ simulator on a UNIX platform for evaluating the performance of the presented packet switch architecture. Some of the modules were used from the existing simulator developed by Mehrotra [39]. The simulator is simple and modular and can be used for input queued, output queued and CIOQ switches. Throughout the simulations we used a 53 byte cell. However, the cell size can be changed but should be fixed for an entire simulation. We used the traffic models described in Chapter 3, for generating the traffic. All the simulations were run for 1000000 switching cycles unless specified otherwise. Large buffer size, were was used to ensure that no cell was lost during a given simulation. As the number of cells used in every simulation exceeded 10 million, this corresponds to a cell loss of less than one in ten million or better. All the queue sizes are given in terms of 53-byte cells e.g a queue size of 50 means that queue can hold 50 53-byte long packets. Throughout the simulations for bursty traffic load, a mean burst length of 15 was used.

Figure 4.4 shows the delay performance of the switch architecture for uniform random traffic and bursty traffic for different traffic loads. These results were obtained from the simulation of a 128 x 128 enhanced crossbar CIOQ switch architecture. From these results it is apparent that the delay performance of this architecture significantly degrades for bursty traffic.

For bursty traffic and an arrival rate greater than 0.8 the delay through the switch architecture is significantly high, therefore for determining the buffer space requirement,

we use a traffic load of 0.8. If we use higher traffic loads, to avoid cell loss we need a very large buffer space. At the same time for higher traffic loads (>0.8) delay through the switch architecture will increase significantly. Therefore we use a reasonable (0.8) traffic load such that delay through the architecture will be acceptable. We then determine the buffer space requirement for this load such that there is no cell loss. Throughput of this switch architecture is in excess of 99% and remains almost the same for both uniform random and bursty traffic. In the following we present the delay performance and buffer requirements for this architecture for different switch sizes.

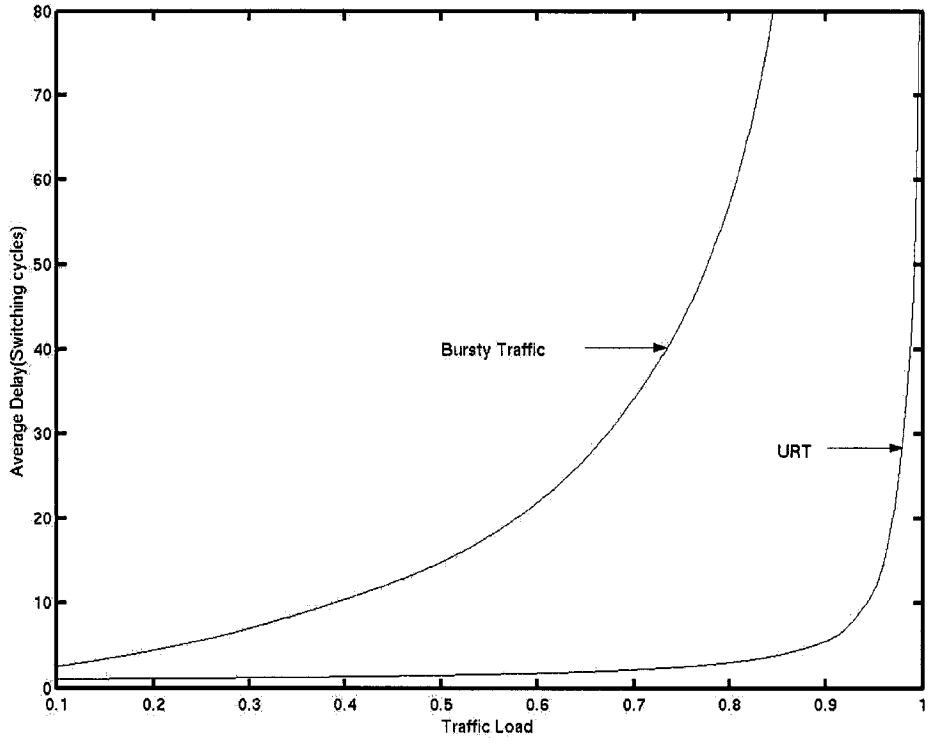


Figure 4.4 Delay performance of enhanced crossbar CIOQ switch architecture for different traffic loads

4.3.1 Delay Performance

We studied the simulation results for different switch sizes of the average delay of the speeded-up packet switches for both uniform random and bursty traffic arrival. We plotted the total average delay through the switch suffered by packet traversing the switch obtained from simulations. We have plotted the input and output queue delays separately. The delays shown in Figure 4.5 are in the number of switching cycles used to traverse through input ports, output ports or through the switch. It can be seen that the total delay is dominated

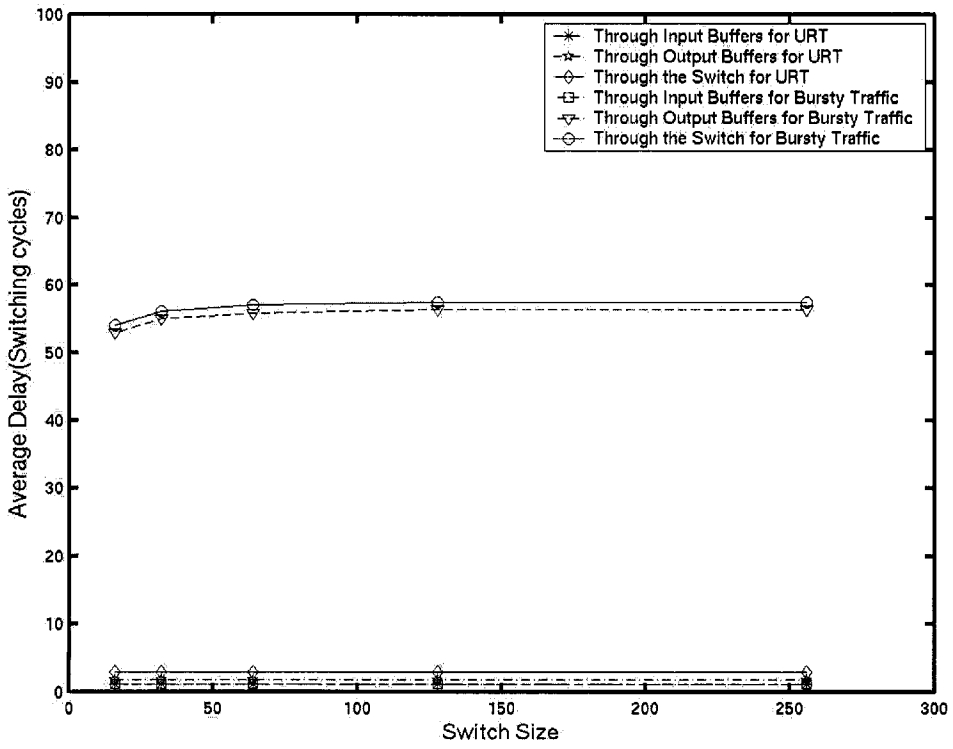


Figure 4.5 Average delay performance of enhanced crossbar CIOQ switch Architecture (traffic load = 0.8)

by the output queueing delay. It can be seen from the plot that the average delay that cells suffer while traversing the input port is significantly less than the average delay through output ports. The interesting result is that, even for bursty arrivals the average delay through the input ports is less than 2 switching cycles. This implies that the cells traversing the switch suffer most of the delay at the outputs. In other words, this is a predominantly output queued switch architecture. This is an important observation considering the fact that well-known scheduling algorithms exist for providing QoS guarantees in ideal output queued switches. These scheduling algorithms can be employed in this architecture in a similar fashion as in an ideal output queued switch for

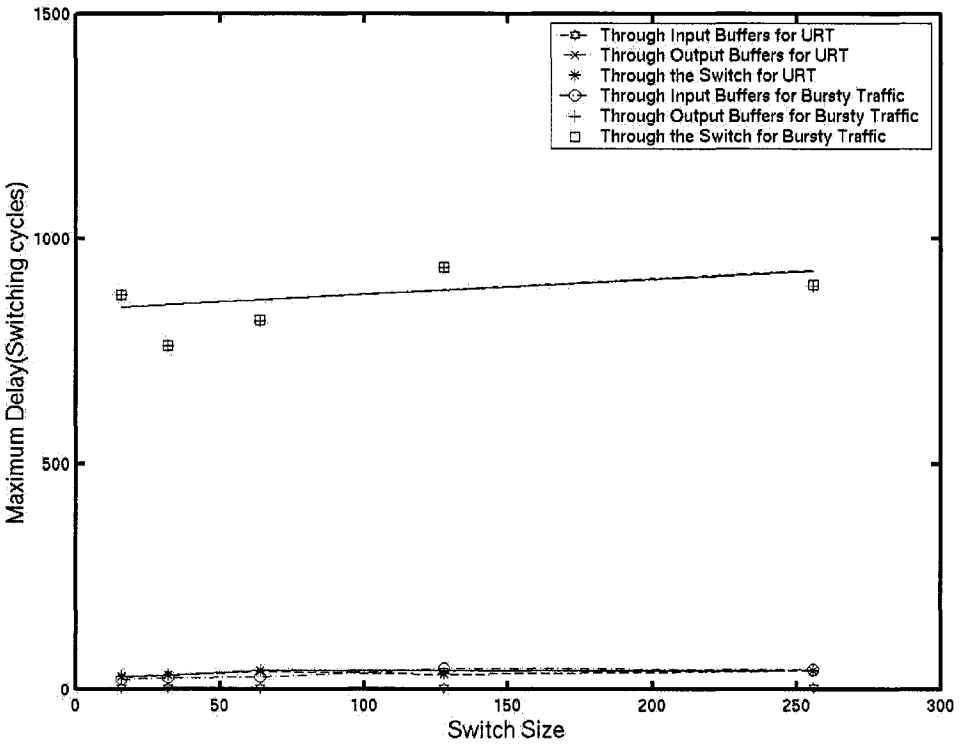


Figure 4.6 Maximum delay performance of enhanced crossbar CIOQ switch Architecture (traffic load = 0.8)

providing the QoS guarantees. The switch architecture overcomes the HOL blocking problem associated with pure IQ switches and the high switch speed requirement for the ideal OQ switches and closely approximates the best performance obtainable. From Figure 4.5 it is also obvious that the delay performance of these switch architectures remains the same for different switch sizes. This implies that this architecture scales well.

Most prior work compares the performance of the switches by measuring the average delay that packets suffer while traversing the switch for various traffic loads. Average delay serves as a good metric for studying the relative performance of two or more switch architectures; however maximum delay performance is a better parameter for determining the performance of any switch architecture. In Figure 4.6 we show the maximum delay performance of the switch architecture. Once again it can be seen that the total delay is dominated by the delay though output ports. Similarly, the maximum delay remains almost the same for different switch sizes. As expected the maximum delay for bursty traffic is significantly high.

4.3.2 Buffer Space Requirement

The buffer space requirement is one of the important factors in determining the performance of the switch architecture. Figure 4.7 presents the maximum buffer occupancy for both uniform random and bursty traffic arrivals. The buffer space requirements at the input port and output port are shown separately. It can be seen from Figure 4.7 that for bursty traffic the buffer requirement at the output port increases significantly. As we observed for delay performance the buffer space requirement is also

very high at the output port compared to at the input ports. This is because of the high throughput due to virtual speedup provided by the multiple switch fabric. It can be also seen that the buffer requirement at input port and output port for different size of switches remains the same. The increase in the buffer space requirement at the IPM is much less than the increase in the buffer space required at the OPM when traffic arrival changes from uniform random to bursty. Thus, this is a predominantly output queued switch.

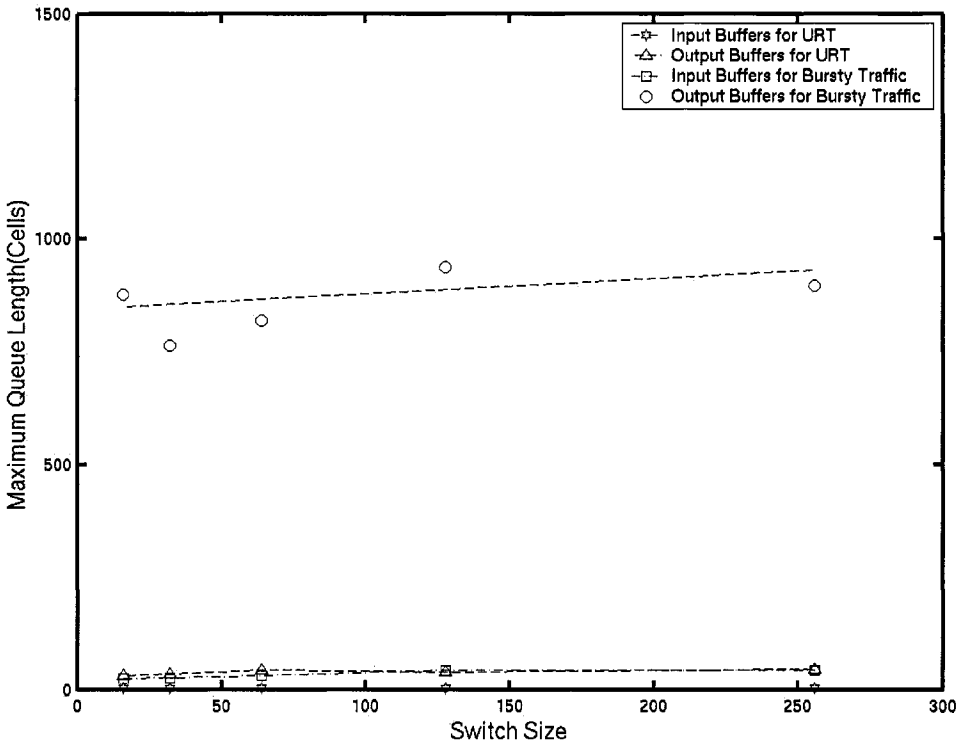


Figure 4.7 Buffer space requirement of enhanced crossbar CIOQ switch architecture

4.4 Summary

In this chapter, we described the design of enhanced crossbar combined input-output queued switch architecture which is capable of approximating the ideal OQ switch. We described each component of this architecture and discussed its design We also described the parallelism that is used to provide a large time window for implementing slow and complex but fair scheduling algorithms.

At the end of this chapter, we presented the performance of this architecture under uniform random and bursty traffic. The performance of the architecture is evaluated for the following parameters: the delay that packets suffer while traversing the switch and the buffer space requirement both at input and output ports for an acceptable performance. We observed that the average delay through input buffers is very small for both uniform random and bursty traffic and delay through output ports dominates the overall delay. The buffer requirement at the input ports is much smaller compared to that at the output ports. These results show that the performance of this switch architecture is close to that of an ideal output queued switch. In the next chapter we will VLSI implementation of this switch architecture.

Chapter 5

Design and VLSI Implementation

In the previous chapter, we presented the performance of the enhanced crossbar CIOQ switch architecture. In this chapter, we present the design and VLSI implementation of the same architecture to show that our work is easily implementable using readily available CMOS technology. The design is targeting 0.18-micron CMOS standard-cell ASIC technology. We describe the design methodology and implementation of input port module, output port module, switch fabric and scheduler in detail with high-level schematic diagrams. We report on the design complexity and discuss implementation results. This implementation is capable of handling a line rate of 622 Mbps. We begin with the input port module and describe all the major sub-blocks, and similarly, describe the output port module and its major sub-blocks. We then describe the implementation of the switch fabric and scheduler. At the end of the chapter, we present the total area and gate count for the design and each individual block.

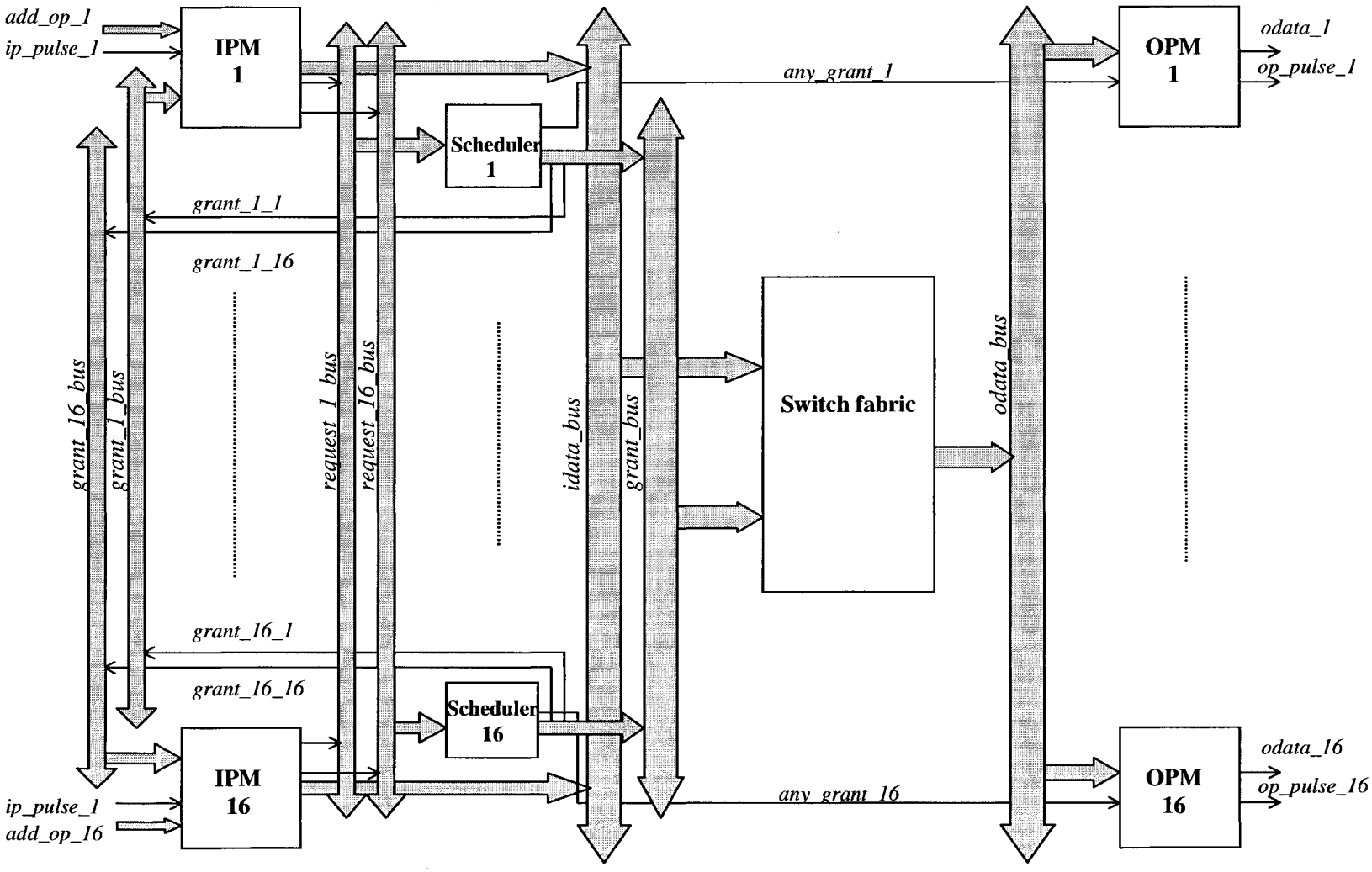


Figure 5.1 High-level schematic for a 16 x 16 enhanced crossbar CIOQ switch architecture

Figure 5.1 illustrates a high-level schematic of the enhanced crossbar CIOQ switch architecture described in the previous chapters. The *grant_bus* is a 16-byte wide bus, which carries the grant signal from all the 16 schedulers to the switch fabric. *idata_bus* is a 16-bit wide data bus that carries the data from IPMs to the switch fabric. *Odata-bus* is also a 16-bit wide data bus that carries the data from the switch fabric to the OPMs. Every scheduler has a 16-bit wide request and grant bus associated with it. The request bus carries the requests from all the IPMs to that scheduler and the grant bus carries the grants issued by all the schedulers back to the IPMs.

We determined from the software simulation of the architecture that for a bursty traffic load of 0.8 with mean burst length of 15, the input port should be capable of storing 50 cells and output port module should be capable of storing 900 cells to avoid cell loss both at input ports and output ports. Therefore we used this storage capability at each IPM and OPM, respectively.

The interface to the scheduler involves two lines for each input. One is dedicated to the request and the second is for the grant signal. Routing complexity increases with the switch size. Therefore this architecture with the current routing configuration may not be feasible for large sized switches, where by large size switches, we mean switches with more than 1000 ports. However with current VLSI technology, medium (50-500 ports) sized switches based on this architecture can be implemented.

The area of the design and blocks that is shown in this chapter is given in μm^2 and the gate count is given with reference to the area of a two-input NAND gate. Results presented later, do not include the area of memories, since for synthesis of memories,

usually a memory compiler is used, as it is the most area efficient approach [40]. This memory compiler can be called from within the synthesis tool or a component that has already been compiled can be instantiated. However, if a design compiler is used for synthesis of memories, it maps a bit cell to a flip-flop that results in very high gate counts. Due to unavailability of the memory compiler, we were unable to obtain the estimate of the gate count for all the buffers.

Table 5.1 Input port module implementation details

Block	Area	Gate count
Address memory pointer	2272	184
Data memory pointer	5338	432
6-bit counter	2273	184
Write controller	4793	388
Read controller	8724	706
6-bit up down counter	3263	264

5.1 Input Port Module

A high-level schematic of the Input port Module (IPM) is shown in Figure 5.2. To achieve a high-speed implementation, the control of the input port module is distributed between two controllers, the write and read controllers. As illustrated in the figure, each IPM has two dual-port Random Access Memories (RAMs) called the data buffer and address buffer, two 6-bit counters, four memory pointers, a 6-bit up down counter, a serial to parallel converter, and a parallel to serial converter. Table 5.1 summarizes the

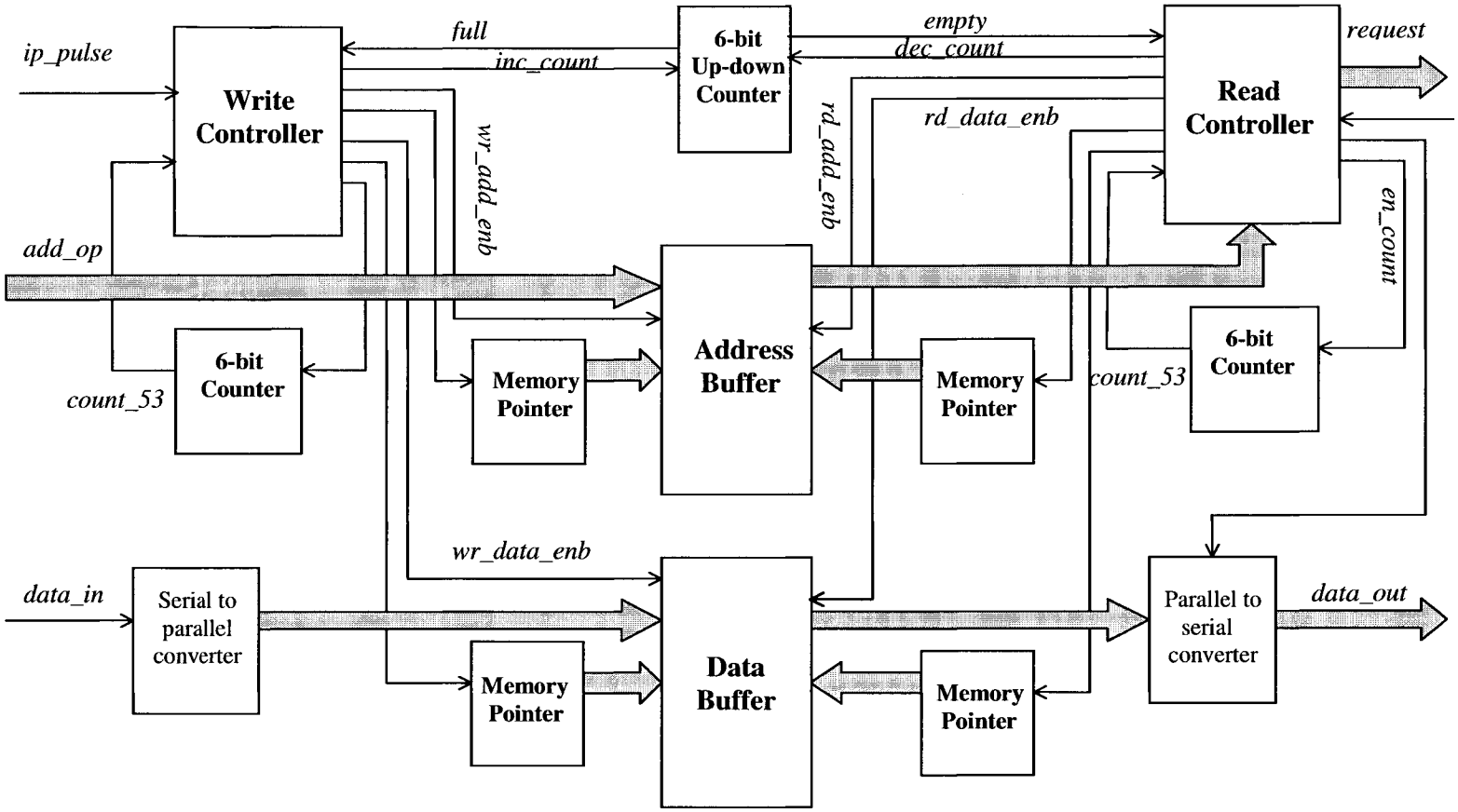


Figure 5.2 High-level schematic of input port module

area and gate count for each sub-block in the IPM for 0.18-micron CMOS technology. In the following we explain the functionalities of some of these blocks in some detail.

5.1.1 Write Controller

The write controller stores the incoming cells in the data buffer and stores the corresponding address of the output port to which the cell is destined in the address buffer. A serial to parallel converter converts the incoming data into a byte wide word and latches the word for the write controller, which stores the word in the memory.

The memory pointer for the address buffer is a 6-bit rollover counter, which provides the 6-bit address required to write to an empty location in the address buffer. Similarly, the memory pointer to the data buffer is a 12-bit rollover counter. The write controller increments these memory pointers after the respective write operations.

The write controller uses a 6-bit counter to count the number of bytes written to the data buffer. When 53-bytes have been written to the memory a *wr_count_53* signal is generated by the counter to inform the write controller about the completion of packet being written. A 6-bit up-down counter is used by both write and read controllers to keep track of packets stored in the data buffer. If the packet count is 50, a *full* signal is asserted to inform the write controller. The write controller samples the *full* signal before storing the cell to check if there is space in the data buffer for storing another cell. If the buffer is full the write controller drops the arriving cells. After storing the packet in the data buffer the write controller increments the counter.

5.1.2 Read Controller

The read controller samples the *empty* signal from the 6-bit up-down counter to check the status of the data buffer. If the *empty* signal is level high than there is no cell stored in the data buffer, thus the read controller waits for the arrival of a new cell. If the empty signal is level low, the read controller reads the corresponding address of the head of the line packet from the address buffer and requests the desired output after decoding the address. If a grant is received from the scheduler, it transmits the packet to the output through the switch fabric and decrements the up-down counter.

The read controller also uses a 6-bit counter, like the write controller, to count the number of bytes the read controller reads from the data buffer. When a complete packet has been read the counter generates a *rd_count_53* signal to inform the read controller about packet completion. A parallel to serial converter converts the read byte wide data into a 4-bit wide word and sends it to output port module through the switch fabric.

Like the write controller, the read controller also maintains two similar memory pointers to address and data buffers. These memory pointers provide the address required to read from the specific locations in the memory.

5.1.3 Buffers

The data buffer holds up to 2650 one-byte words, equivalent to 50 53-bytes packets. The address buffer holds up to 50 four-bit wide words, the output port addresses of up to 50 packets stored in the data buffer. The buffers are maintained as first-in-first-

out queues, that is the cell and address are stored at the end of the queue and read from the head of the queue.

Table 5.2 Output port module implementation details

Block	Area	Gate count
Data memory pointer	6826	552
6-Bit counter	2273	184
Write controller	2678	217
Read controller	3785	306
10-bit up down counter	4919	398

5.2 Output Port Module

The output port module is responsible for storing and transmitting the cell on the link. A high-level schematic of the output port module is shown in Figure 5.3. Like the input port module it has two controllers for storing and retrieving the cell from the data buffer. As illustrated in the figure, each IPM has a dual-port RAM called data buffer, two 6-bit counters, four memory pointers, a 10-bit up-down counter, a serial-to-parallel converter, and a parallel-in serial-out shift register. The data buffer holds up to 47700 one-byte words that is equivalent to 900 53-byte packets.

In the following we explain functionalities of the write and read controllers in some detail. Table 5.2 summarizes the area and gate count for each sub-block in the OPM for 0.18-micron CMOS technology.

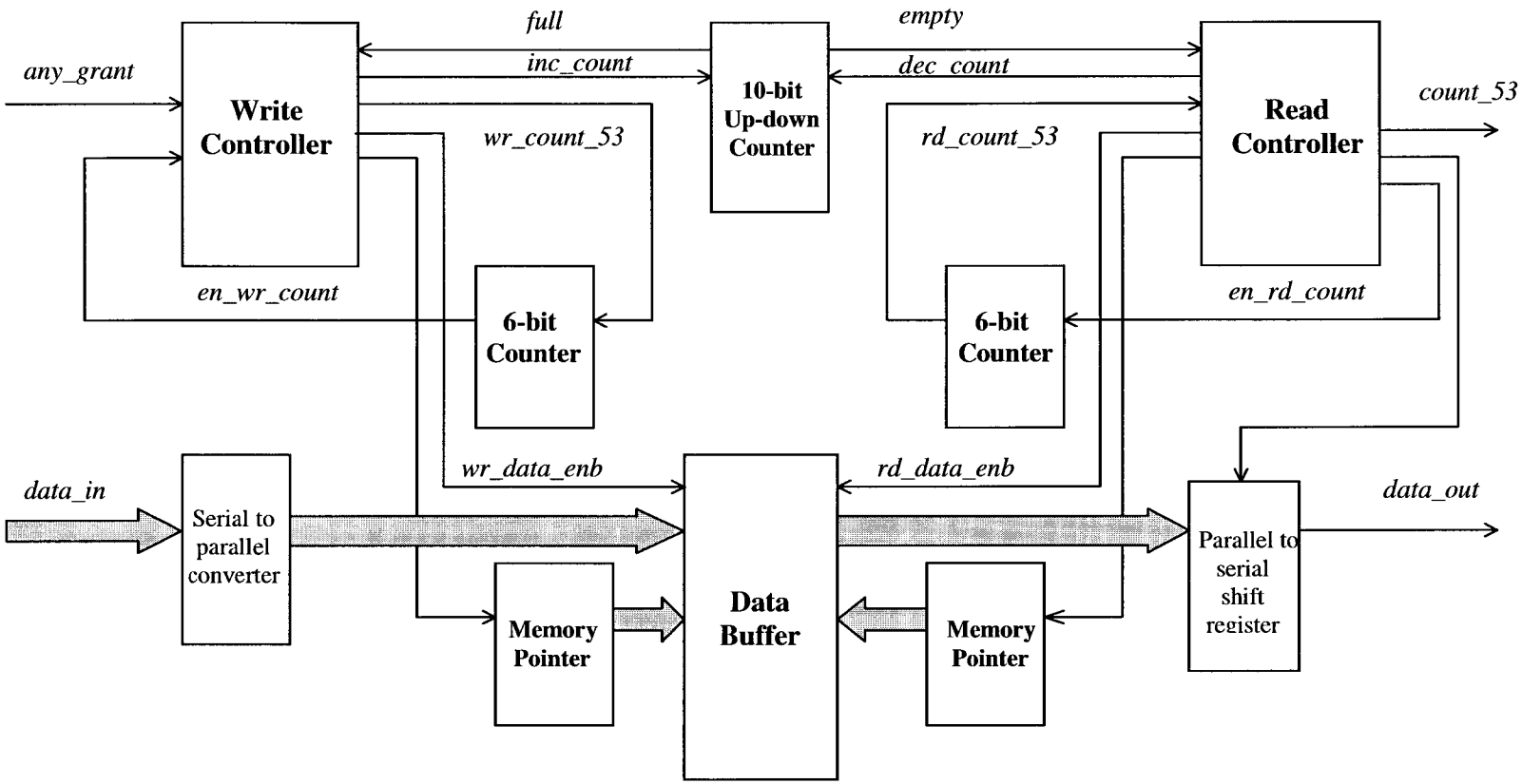


Figure 5.3 High-level schematic of output port module

5.2.1 Write Controller

The write controller stores the cells coming from the IPM in the data buffer. A serial to parallel converter converts the incoming 4-bit word into a byte wide word and the write controller stores the word in the data buffer. The memory pointer provides the address of the location where the packet is to be stored. This memory pointer is a 16-bit rollover counter, which is incremented by the write controller after every write operation.

The write controller uses one of the two 6-bit counters, to count the number of bytes written to the data buffer. When 53-bytes have been written to the memory a *wr_count_53* signal is generated by the counter to inform the write controller about the completion of packet being written.

A 6-bit up down counter is used to keep track of packets stored in the data buffer. If the packet count is 900, a *full* signal is asserted to inform the write controller. The switching cycle is divided into four equal subcycles and at the start of each subcycle the write controller samples the *full* signal and the *any_grant* from the scheduler associated with the OPM. If a grant has been issued and the *full* signal is not level-high then it stores the packet, else if the buffer is full the write controller discards the arriving cells as we employ a queue loss scheme in this architecture. On the other hand, if the data buffer is not full, the write controller stores the incoming packet from the IPM at the end of the buffer and increments the up-down counter.

5.2.2 Read Controller

Like the read controller at the IPM, the read controller at the OPM also samples the *empty* signal from the 10-bit up down counter to check the status of the data buffer. If the *empty* signal is level high then there is no cell stored in the data buffer, thus the read controller waits for the arrival of a new cell. If the empty signal is level low, the read controller generates a frame pulse and starts reading the packet byte by byte and stores a byte in an 8-bit parallel to serial shift register. This shift registers transmits the packet bit by bit on the link. The read controller also uses a 6-bit counter, like the write controller, to count the number of bytes, the read controller reads from the data buffer. When a complete packet has been read the counter generates a *rd_count_53* signal to inform the read controller about packet completion and the read controller decrements the up down counter. Like the write controller, the read controller also maintains two similar memory pointers to address and data buffers. These memory pointers provide the address required to read from the specific locations.

5.3 Switch Fabric

The switch fabric is the interconnection between inputs and outputs. Figure 5.4 illustrates a 4 x 4 crossbar that is a simple and area efficient implementation of crossbars of smaller sizes; however, large crossbars cannot be implemented using this design due to the fanout constraint. We used the same design for implementing 16 x 16 crossbars for the design we presented in this document.

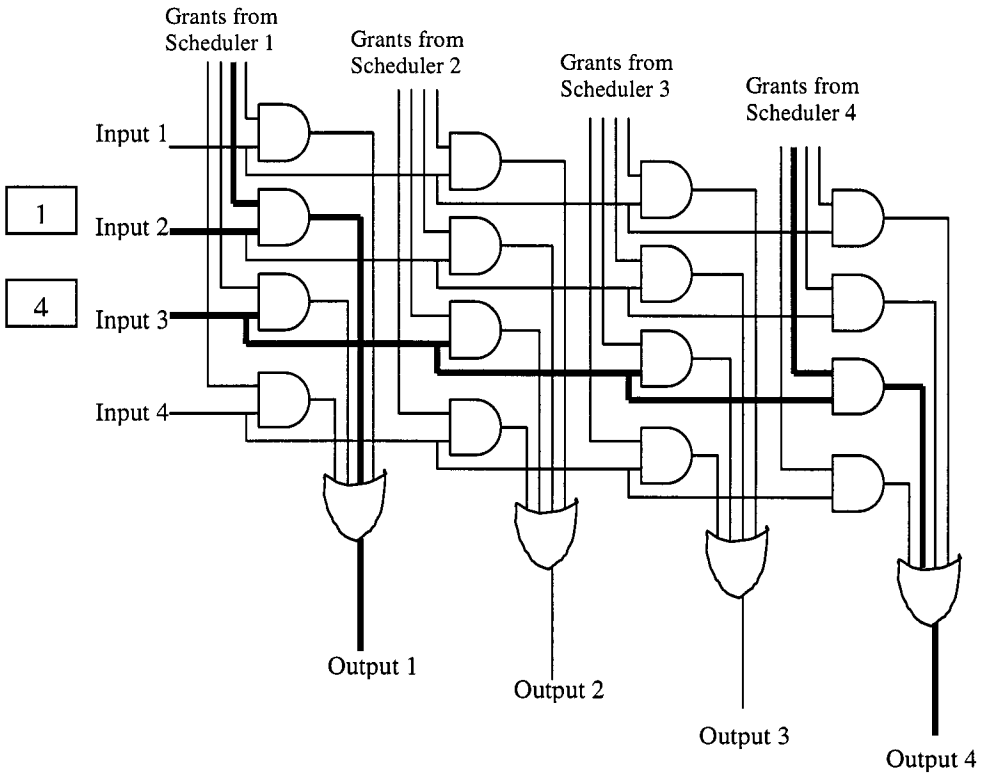


Figure 5.4 4 x 4 crossbar implementation

This architecture is fully controlled by schedulers. Schedulers are also responsible for informing the output port modules (OPMs) if cells have been scheduled for them. To understand the functionality properly we need to consider each column of AND gates in the crossbar shown in the Figure 5.4. Each scheduler controls a column of AND gates and at the most issues one grant signal during a subcyle. Therefore in each column only one input is connected to the output. As shown in Figure 5.4, if inputs 2 and 3 have cells for outputs 1 and 4 respectively, scheduler 2 will select input 2 and scheduler 4 will select

input 3. The grant signals are latched so that these input – output pairs are available for the entire subcycle. Cells are then routed to their respective outputs.

As we described in the previous chapter, four of these crossbars in parallel serve as the switch fabric for the switch architecture presented in this document. All four crossbars are configured with the same grant signal that is the overlapping crosspoints use the same latched grant signal to determine their states.

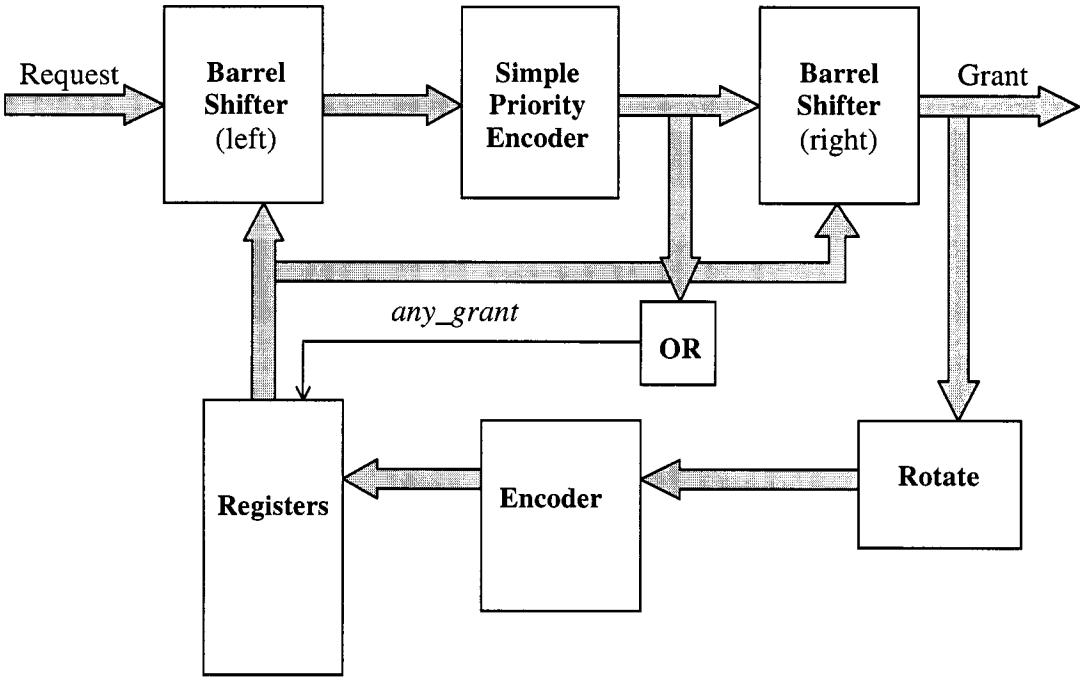


Figure 5.5 High-level schematic of scheduler

5.4 Scheduler

We used the iSLIP scheduling algorithm [24], and implemented it using barrel shifters and an encoder, as this is the most common design used for implementing round-robin arbiters [41]. One of the alternatives to the barrel shifter implementation is N simple priority encoders with an N:1 (N+1)-bit multiplexer implementation. This implementation is faster than the barrel shifter but results in large area compared to a barrel shifter implementation. In the barrel shifter implementation, there are two barrel shifters in the critical path. Therefore it is slower than the N simple priority encoder implementation. As well, in barrel shifter implementation only one simple priority encoder is required. A high-level schematic of the scheduler is shown in Figure 5.5. Two-barrel shifters in the critical path make this implementation of the scheduler slower than some other implementations presented in literature. However, in the presented switch architecture, for a line rate of 622 Mbps, a time window of 170 ns is available for the scheduling process, in the switch architecture presented in this document. This large time window allows us to use slow schedulers and focus on decreasing the area. Thus, we use a barrel shifter design, as it requires considerably less area than the other implementations.

In this implementation, a barrel shifter first rotates the incoming requests by P, where P is a 4-bit pointer used to remember the state of the scheduler and points to the input port, which had the highest priority at the end of previous scheduling cycle. The simple priority encoder selects one among the requesting input ports. The second barrel shifter than rotates the output of the simple priority encoder in the other direction by the

same P to obtain the final grant signals. If a grant is issued, P is incremented to the next location beyond the selected input port. This process of incrementing the pointer involves rotating the grant signal by one, then encoding that rotated signal and registering these encoded $\log_2 N$ bits. The state of the scheduler is changed only when there is a grant issued during the scheduling process.

5.5 Implementation Results

The architecture is designed in a top down fashion. However, the implementation was carried out in bottom up fashion. Each block was functionally verified and observed to be working as intended. However, we were unable to functionally verify the complete design because of the limitation of the simulation tools. Each block as well as the complete design is synthesized using Synopsys Design Compiler. The design met all the timing requirements. Packet transfer from the IPM and OPM and the serial to parallel and parallel to serial conversion of data are performed at the line rate. The corresponding blocks met the timing requirement that is 622 MHz. The rest of the design operates at half the line rate and successfully synthesized to meet the timings.

The maximum throughput of this 16x16 switch is close to 10Gbps with memories operating at 311MHz. Table 5.3 provides the area and its corresponding gate count for each individual block and total area of the 16x16 switch architecture. The area and gate count shown in the table exclude the area for the memories.

Table 5.3 Area and gate count for the design

Block	Area	Gate count
Input port Module	26755	2166
Output port module	29670	2402
Switch fabric	16652	1348
Scheduler	18343	1485
Total	1212940	98213

5.6 Summary

In this chapter we described the VLSI implementation of the 16 x 16 switch fabric presented in this document, using 0.18-micron CMOS standard cell technology. We described each block of the design with a high-level schematic and presented synthesis results. The control is distributed to achieve high speed. Similarly, instead of a centralized scheduler, each output port modules has a corresponding scheduler to schedule the traffic destined for it. Slow memory speed requirement for this architecture alleviates the size limitation problem, usually associated with speeded up switches.

Due to its simplicity and small gate count, the crossbar implementation presented in this chapter is suitable for small sized switches. The large time window available for scheduling due to pipelining of the scheduling process enabled us to use a slow but area efficient, barrel shifter and priority encoder, design for the scheduler. This implementation is capable of handling a line rate of 622 Mbps and the maximum

throughput of this 16x16 switch architecture is close to 10Gbps with memories operating at 311MHz. The throughput can be further improved with the use of 0.13 or 0.09-micron CMOS standard cell technology.

Chapter 6

Conclusions and Future Work

Speed and scalability are important measures in determining the overall performance of a high bandwidth packet switch. In this thesis, we addressed both these parameters and attempted to provide a solution, using combined input-output queueing and multiple crossbars. The resulting architecture is scalable and provides the throughput close to that of an ideal output queued switch. The switch architecture presented in this thesis is a combined input-output queued switch with four crossbar planes used for providing the virtual speedup of four. It employs pipelined scheduling for eliminating the main bottleneck of scheduler speed.

In Chapter 5, we have described the VLSI implementation of this architecture to show that this architecture is easily implementable in available VLSI technology. A 16x16 switch is implemented in 0.18-micron CMOS standard cell technology. This architecture is capable of achieving an overall throughput close to 10Gbps. Each port can

handle a line rate of 622Mbps which can be further improved by using 0.13 or 0.09-micron standard cell technology.

6.1 Contributions

The main contributions of this work are the throughput enhancement of the switch fabric such that it can emulate an ideal switch and elimination of the overhead associated with configuring the switch fabrics involving input buffers. Furthermore, this architecture employs distributed scheduling which makes the implementation of the schedulers simple. In the following we describe major contributions in some detail.

6.1.1 Output Queueing Emulation

Pure output queued switches which have no interconnect blocking and whose output ports are capable of accepting all the cells destined to them, are known for their high throughput and are suitable for providing QoS guarantees. However, due to switch fabric and memory speed limitations, the ideal output queued switches are no longer feasible for high-speed switching. Various studies have shown that an output queueing emulation approach is able to address the QoS issue. The idea is that if the switch fabric of an input-queued switch is moderately (by 4 or 5 times) speeded up, most of the incoming cells can be instantly sent to the output port. This requires queues at the output. This arrangement of speeded up switch fabric and combined input-output queueing makes the switch behave as an output queued switch. Each output port can provide QoS by using the well known scheduling algorithms which exist for this purpose.

In the packet switch architecture described in this thesis, we obtained the speedup using multiple switch planes and showed that this architecture achieves almost the same delay performance as an ideal output queued switch. We observed that the delay suffered by packets traversing the switch was dominated by the delay through the output buffers and the delay through the input buffers was very small compared to that through output buffers for both uniform random and bursty traffic. These results showed that this architecture can approximately emulate the ideal output queued switch. Hence the well-studied scheduling algorithms for providing QoS can be employed for this architecture at the output ports and similar QoS performance can be achieved. This architecture is suitable for switches which are required to support different services as it has the same throughput close to an ideal output buffered switch and similar QoS performance can be achieved.

6.1.2 Scheduling Overhead Elimination

Switches involving buffers at the input ports need schedulers for configuring the switch fabric before transferring the packet from input ports to the output ports. This scheduling is usually an overhead which increases with the increase in the size of the switch and may become a limiting factor in terms of size of the switch. In this architecture the scheduling and packet transfer are performed in parallel which eliminates this scheduling overhead. The single stage pipelining provides a larger time window for scheduling equivalent to that of the time required for a packet transfer from input port to

output port. The cost of this parallelism is that every cell traversing the input ports suffer a small delay equivalent to ¼ of a switching cycle.

With this pipelined scheduling, increasing switch size does not present a major problem in terms of the scheduling time because the scheduler has a large time window for configuring the switch fabric. The fair scheduling algorithms presented in the literature for scheduling the switch fabric are slow and very complex, therefore this parallelism allows us to use those complex but fair algorithms for configuring the switch fabric before transferring the packets from the input ports to the output ports.

6.1.3 Distributed Scheduling

Centralized schedulers face the scaling problem. Using distributed schedulers can mitigate this problem and each individual scheduler is much simpler. Distributed algorithms scale linearly with the switch size. One of the problems in designing a distributed algorithm is the exchange of information among all schedulers that may be at different input ports and output ports. In some scheduling algorithms multi-bit information, such as queue lengths, or waiting times, is required which increases the number of interconnections significantly. As well, the interconnection delay can increase the scheduling time significantly, especially when iterative scheduling algorithms are employed.

In this architecture, we employed schedulers at each output port, responsible for scheduling the traffic destined to that output port only. The scheduling algorithm employed does not require the queue length or waiting time information, therefore only

select and grant interconnections are required from each input port to each scheduler. The interconnection delay is not a problem in this architecture since a large time window is available for configuring the switch fabric due to pipelining of the scheduling process, described in the previous section.

6.2 Future Work

High-speed switches are required for emerging high bit rate applications. In addition to receiving high throughput, these applications often have other requirements such as a delay guarantee and a low packet loss rate. Currently, researchers are working in the direction of integrating QoS and switch matrix schedulers. In the following, we briefly describe some potential areas for modifications, which can be explored to improve the performance of this architecture.

6.2.1 Performance Comparison

An important task for the future is the comparison of this switch architecture with other existing or proposed similar switch architectures both in terms of performance and complexity. We compared the performance of this architecture to a hypothetical output queued switch which is referred to as the ideal output queued switch throughout this thesis. However this ideal output queued switch is impractical therefore for comparing the implementation complexity and related cost, a practical switch, whose performance is close to that of an ideal output queued switch is required.

6.2.2 Provision and Integration of QoS

Many emerging high bit rate application not only require high throughput but also need some delay and delay jitter guarantees. To support these emerging applications the packet switches should be capable of providing the QoS guarantees. For this architecture QoS scheduling algorithms can be implemented at the output port.

Integrating the functionality of the QoS scheduler into the switch matrix scheduler still remains a difficult task. Configuring the switch fabric for increasing the throughput and reducing the delay simultaneously becomes a two dimensional problem, since the delay and throughput in many cases are conflicting specifications. Some work is being carried out in this direction, but high throughput and fair scheduling algorithms are not yet available.

6.2.3 Dynamic Buffering

For a small sized switch, instead of maintaining one memory per port, one large multi port RAM can be used for several ports. The advantage of maintaining queues of several ports in the same RAM is that it may be dynamically partitioned, resulting in more efficient usage. This is particularly important, as the average memory utilization of this switch architecture is very low. Thus dynamic buffering can significantly reduce the memory requirement of this architecture.

6.2.4 Multicasting

The crossbar performs multicast by simultaneously delivering packets to multiple destinations. Two schemes can be adapted for this purpose: in the first, all the copies of packets are sent simultaneously. If the multicast packet does not get grants from all the desired outputs, it waits for the next cycle and tries again. In the second case, the copies of the packet can be sent to the outputs which have issued a grant. The second scheme is much simple and results in high performance.

References

[1] F. A. Tobagi, "Fast packet switch architectures for broadband integrated services digital network," *Proc. IEEE*, vol. 78, no. 1, pp. 133-167, January 1990

[2] Cisco Systems; "Fast switched backplane for a gigabit switched router," www.cnaf.infn.it/~ferrari/tfngn/doc/fasts_wp.pdf

[3] S. Chuang, A. Goel, N. Mckeown, and B. Prabhakar, "Matching Output Queueing with a Combined Input Output Queued Switch," *Infocom '99*, New York, USA, 1999

[4] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input vs. output queueing on a space-division packet switch," *IEEE Transaction on Communications*, Vol. 35, No. 12, pp. 1347-1356, 1987

[5] Y. Oie, M. Murata, K. Kubota, and H. Miyahara, "Effect of speedup in nonblocking packet switch," in *Proc. ICC '89*, Boston, MA, p. 410-14, June 1989

[6] M. A. Marsan, A. Bianco, E. Filippi, P. Giaccone, E. Leonardi, F. Neri, "On the behavior of input queuing switch architectures", *European Transactions on Telecommunications*, n.2, Feb./Mar.1999

[7] A. Mekkittikul & N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches," *INFOCOM '98*, 792-799, 1998

[8] B. Prabhakar and N. McKeown, "On the Speedup Required for combined Input and Output Queued Switching," *Automatica*, Vol. 35, no. 12, December 1999

[9] Y. S. Yeh, M. Hiuchyj, and A. Acampora, "The Knockout Switch: a Simple Modular Architecture for High Performance Packet Switching," *IEEE J. Selected Areas in Communications*, Vol. SAC-5, No. 8, pp. 1274-1283, October 1987

[10] J. S. C. Chen and T. E. Stern, "Throughput analysis, optimal buffer allocation, and traffic imbalance study of a generic nonblocking packet switch," *IEEE Journal of Selected Areas in Communications*, 9(3):439–449, April 1991

[11] K. Kar, T. V. Lakshman, D. Stiliadis, L. Tassiulas, "Reduced complexity input buffered switches," *Proceedings of Hot Interconnects VIII*, Palo Alto, August 2000

[12] A. H. Ahmadi and W. E. Denzel, "A Survey of modern high-performance switching techniques," *IEEE Journal on Selected Areas in Communications* , vol. 7, no. 7, pp. 1091-1103, Sep. 1989

[13] R.Y. Awdeh, and H.T. Mouftah, "Survey of ATM switch architectures," *Computer Networks & ISDN Systems*, vol. 27, pp. 1567-1613, 1995

[14] H. Zuzuki, H. Nagano, T. Zuzuki, T. Takeuchi and S. Iwasaki, "Output-buffer switch architecture for asynchronous transfer mode," *Int. journal of digital and analog cabled systems*, vol.2, 269-276, 1989

[15] C. Kolias and L. Klienrock, "Throughput Analysis of Multiple Input-Queueing in ATM switches," *Broadband Communications*, L. Mason and A. Casaca, pp 382-393, Chapman & Hall, London U.K.1996

[16] S. Nojima et al., "Integrated services packet network using bus matrix switch," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1284--1292, Oct. 1987

[17] F. A. Tobagi and T. C. Kwok, "The Tandem Banyan Switching Fabric: A Simple High- Performance Fast Packet Switch," *Proc. IEEE, Infocom' 91*, 1991

[18] Y. El-Sayed, "Performance analysis, design and reliability of the balanced Gamma network," Ph. D. Thesis, Memorial University of Newfoundland, 1999

[19] J. Giacopelli, M. Littlewood, and W. D. Sincoskie, "Sunshine: A broadband packet switch architecture," *In Proc. ISS'90*, Stockholm, May 1990

[20] N. McKeown, V Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *in Proceedings of INFOCOM'96*, pp. 296—302, March 1996

[21] D. S. Lee, "Generalized longest queue first: An adaptive scheduling discipline for ATM networks," *in Proc. IEEE INFOCOM'97*, vol. 3, pp. 1096-1104, 1997

[22] A. Mekkittikul and N. McKeown, "A starvation-free algorithm for achieving 100% throughput in an input-queued switch," *Proc. ICCCN'96*, Washington D.C., pp. 226-231, October 1996

[23] T. Anderson, S. Owicki, J. Saxie, and C. Thacker, "High speed switch scheduling for local area networks," *ACM Trans. Comput. Syst.*, vol. 11,no. 4, pp. 319-352, November. 1993

[24] N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188--201, April 1999

[25] C. Li, H. Heys, R. Venkatesan, "Traffic Generation for Broadband Switch Simulation," *NECEC2002*, St. John's, Canada, Nov. 2002

[26] H. Kim and K. Kim, "Performance analysis of the multiple input queued packet switch with the restricted rule," *IEEE/ACM Transactions on Networking*, vol.11, no.3, June 2003

[27] H. Kim, C. Oh, Y. Lee, and K. Kim, "Throughput analysis of the bifurcated input-queued ATM switch," *IEICE Trans., Commun.*, Vol. E82-B, pp. 768-772, May 1999

[28] C. Kolias and L. Klienrock, "Throughput Analysis of Multiple Input-Queueing in ATM switches," *Broadband Communications*, L. Mason and A. Casaca, pp 382-393, Chapman & Hall, London U.K.1996

[29] C. Kolias and L. Kleinrock, "Performance Analysis of Multiplane, Nonblocking ATM Switches," *Globecom'98*, pp. 356-362,Sydney, Australia, November 1998

[30] Y. Oie, T. Suda, M. Murata, and H. Miyahara, "Survey of Switching Techniques in High-Speed Networks and Their Performance," *International Journal of Satellite Communications*, 9, 1991

[31] M. Keyvani, "VHDL implementation of a high-speed symmetric crossbar switch," M.S. Thesis, Simon Fraser University, 1998

[32] L. Cheng, "Design, Modeling and Analysis of the Balanced Gamma Multicast Switch for Broadband Communications," Ph. D. Thesis, Memorial University of Newfoundland, 2004

[33] A. Pattavina and G. Bruzzi, "Analysis of Input and Output Queueing for Nonblocking ATM Switches," IEEE/*ACM Transactions on Networking*, Vol. 1, No. 3, June 1993

[34] A. Moestedt and P. Sjodin, "IP address lookup in hardware for high-speed routing," *In Proc. Hot Interconnects VI*, Stanford Univ., 1998

[35] D. Wu, "Modeling, Analysis and design of the Input Controller for ATM Switches," *M. Eng. Thesis*, Memorial University of Newfoundland, 2001

[36] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, and N. Chrysos, "Variable Packet Size Buffered Crossbar (CICQ) Switches," *Proc. of the Int. Conference on Communications*, Paris, France, 20-24 2004

[37] P. C. Wong and M. S. Yeung, "Design and Analysis of a Novel Fast Packet Switch – Pipeline Banyan," *IEEE/ACM Trans. on Networking*, vol. 3, no. 1, pp. 63-69, February. 1995

[38] R. Venkatesan and H. T. Mouftah, "Balanced gamma network – a new candidate for broadband packet switch architectures," *in Proceedings of INFOCOM, IEEE*, Florence, Italy, pp. 2482-2488, May 1992

[39] P. Mehrotra, "A Framework for Studying Work-Conserving and Non Work-Conserving Scheduling Disciplines," *Post doc. report*, Memorial University of Newfoundland, June 2000

[40] M. J. S. Smith, "Application-Specific Integrated Circuits," *Addison-Wesley*, 1997

[41] Gupta and N. Mckeown, "Designing and Implementing a Fast Crossbar Scheduler," *IEEE Micro*, pp. 20-28, 1999