PERFORMANCE ANALYSIS AND VLSI DESIGN OF A HIGH-SPEED RECONFIGURABLE SHARED QUEUE

LING WU

Your file Votre référence
ISBN: 978-0-494-33455-3
Our file Notre référence
ISBN: 978-0-494-33455-3

Performance Analysis and VLSI Design of a High-Speed Reconfigurable Shared Queue

by

© LING WU, B.ENG.

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

August 2007

St. John's Newfoundland

Abstract

Modern switches and routers require a large amount of storage space to buffer packets. This becomes more significant as the link speed increases and switch size grows. While DRAM is a good choice to provide capacity, the access time becomes a problem for high-speed applications. In this case, SRAM has to be used to match the link speed. However, SRAM is more costly and the density is low. The SRAM/DRAM hybrid architecture provides a good solution to meet both capacity and speed requirements.

To minimize packet loss and provide better quality of service (QoS), each switch port is normally equipped with a large amount of buffering resources, which is usually based on the worst case scenario. However, under normal load conditions, the buffer utilization is very low. Therefore, we propose a reconfigurable buffer sharing scheme in which a buffer controller can dynamically adjust the buffer size allocated for each port according to the parameters derived from the traffic pattern and buffer saturation status. The target is to improve the buffer utilization without posing much constraints on the buffer speed.

In our research, we study how buffer sharing architectures improve the switch performance, based on the results from both numerical analysis and simulations. The performance results obtained from both uniform and nonuniform traffics demonstrate that the proposed reconfigurable shared buffer can provide better queuing performance with a much smaller shared buffer. We further conduct research into the VLSI design of the proposed reconfigurable shared queue architecture using hardware description language VHDL and using 0.18um CMOS technology. The design result

indicates that the buffer sharing and control logic can be integrated into port controllers with a increasing of about $20,000$ gate-count for each 4-*port* group, while the memory size can be reduced into half of the dedicated buffer scheme.

Acknowledgements

First of all, the most sincere thanks is given to my supervisor Dr. Cheng Li, not only for his academic guidance, but also for his concern and help toward my family. With numerous meetings and patient instructions, he led me into this deep scientific area and directed me to the end of this work.

I would especially like to thank to Dr. Ramachandran Venkatesan, Dr. Howard Heys and the Faculty of Engineering and Applied Science for giving me this opportunity to pursue my graduate studies here; I would also like to give my appreciations to the School of Graduate Studies, and Memorial University of Newfoundland for providing such a fine research environment, facilities and technical support services.

I would like to thank Reza Shahidi and Mr. Nolan White; they were very helpful with setting up my computer systems and providing advice when ever I needed. Also thanks to Jonathan Anderson who newly became our computer administrator in the CERL lab. I would also like to express my appreciations to Colin Hodder from the writing center.

I would like to express my appreciations to all my colleagues: Zhiwei An, Weihua Gao, Tianqi Wang, Shi Chen, Liang Zhang, Pu Wang, Chuck Rumbolt, Dianyong Zhang, and Shenqiu Zhang. With all of you, the study and research work in this lab has been more than an enjoyable experience to me.

Most importantly, I would like to express my gratitude to my parents for their unconditional love and support; special thanks to my mother for her constant encouragement. I would like to dedicate this work to my mother, Sha Lu, and my father, Zhian Wu.

Another important person I am grateful to, is my sweetest little girl: Xinyu Huang, thank you for always staying at my side and willingly helping me at every moment.

Finally, I would like to thank another person. However, I could not pick up a single word to express my gratitude. For the past seventeen years, no matter it is sunny or rainy, stormy or windy, you are alway at my side. Love, encourage, support, they are not loudly spoken out, but they are all in your heart. Sincerely, I want to say thank you, my dearest husband: Yanping Huang.

Contents

List of Figures

List of Tables

List of Abbreviations

ATM	Asynchronous Transfer Mode
AQ	Active Queue
CI	Confidence Interval
CLR	Cell Loss Rate
CFDS	Conflict Free DRAM System
DOQ	Dedicated Output Queue
DRAM	Dynamic Random Access Memory
ECQF	Earliest Critical Queue First
FDM	Frequency Division Multiplex
FCFS	First-Come-First-Serve
FIFO	First-In-First-Out
FSM	Finite State Machine
HOL	Head Of the Line

IC	Integrated Circuits
ISDN	Integrated Service Digital Network
LSI	Large-scale integrated circuits
MDQF	Most Deficit Queue First
MMA	Memory Management Algorithm
OC	Optical Carrier
pdf	probability density function
PMU	Pointer Management Unit
QCU	Queue Control Unit
QDRSRAM	Quad Data Rate SRAM.
QoS	Quality of Service
RAM	Random Access Memory
RAU	Requests Arbitration Unit
RRR	Read Request Register
RTT	Round Trip Time
SE	Switch Element
SFU	Special Function Unit

SRAM	Static Random Access Memory
SONET	Synchronous Optical Network
SOQ	Shared Output Queue
TDM	Time Division Multiplex
VHDL	Very High Speed IC Hardware Description Language
VoIP	Voice over IP Network
VOQ	Virtual Output Queue
WRR	Write Requests Register

Chapter 1

Introduction

1.1 Brief History of Data Networks

Since the first data transmission network appeared in the late 1960s, data networks became a popular communication tool. Because of its great potential, in the past four decades, data communication networks have attracted extensive research. By pioneer researchers' tremendous efforts, the Internet development became one of the most successful and exciting phenomena in the history of technology. The merge of computer technologies and communication networks is so successful that there has a such valuable resource, the Internet, been available to everyone in the world. It changes our life styles dramatically, and also leads to numerous innovations of new technologies. As the data transmission speed keeps increasing, even the traditional telephone services are now considering to transmit telephone calls over the Internet, such as Voice over IP (VoIP).

The telephone network was established in the late nineteenth century. In the

beginning, the transmitted telephone calls were analog signals over wire lines [1]. At that time, the switching infrastructures were all electronic circuits to provide point-to-point connections with fixed bandwidth for each call.

Because of increasing demand, the switching networks were necessary to develop more efficient interconnection operations. Since 1960's, digital devices have been introduced to telephone network which provide higher service quality and speed [2]. In order to share expensive computer resources and exchange information, the data network was developed at around the 1960's [3]. The first nationwide data network ARPANET was established in 1969 by the pioneer research team at the Advanced Research Projects Agency, USA. These early data networks typically deployed modems to convert digital information into analog signals and transmitted over leased telephone lines [1]. The transmission bandwidth was typically limited by the voiceband channels, where the data rate was about $50Kbps$, with packet delay at the order of 50-100 ms [1].

Digital transmission has many advantages, such as, easier to regenerate, lower noise interference, and easier to multiplex, etc. As the digital technology progressed, Time Division Multiplex (TDM) brought the transmission bandwidth into a new level – T1-carrier, 24 voice channels were multiplexed to achieve 1.544 Mbps [1]. Later, the Frequency Devision Multiplex (FDM) was introduced to telephony switching systems. All these technologies brought to the Integrated Service Digital Network (ISDN) technology, which maximized the usage of digital infrastructures. In the mid-1980s, the bandwidth of data transmission had achieved $150Mbps$ and higher, and so-called Broadband ISDN (B-ISDN) appeared with more flexibilities and service categories [1].

The Synchronous Optical Network (SONET) standard appeared in the late 1980s and led to the world wide data transmission explosion [1]. It played an important role for the B-ISDN: the transmission bandwidth continuously increases and the transmission reliability and quality improves incredibly. In the SONET, Optical Carrier (OC) level describes the capacity range of an optic fiber to carry digital signals. As a road map, from the beginning of OC-1 ($1 \times 51.8\ Mbps$) to current OC-192 ($192 \times 51.8 Mbps$) and OC-768 ($768 \times 51.8\ Mbps$), a single optical channel can support 192 and 768 optical carriers and provide 10 *Gbps* and 40 *Gbps* of bandwidth. In the future, the OC-1536 and OC-3072 will provide 80 *Gbps* and 160 *Gbps* for faster data transmission [4].

1.2 Packet Switching

Since the network evolution through ISDN and SONET technologies, now the communication infrastructures can support a broad category of services, such as voice service, data service, image and video services, etc. From the network users' point of view, all computers send and receive messages through modems to their service provider's infrastructures. However, at each service network, hundreds of thousands of users could access the Internet at the same time. In addition, these accesses have various destinations, which could be local area applications or world-wide websites. All these applications require the network infrastructures to handle a large amount of messages efficiently, which puts great pressure on the network switching nodes.

The message switch is a core device which is responsible for transmitting data messages according to their required destinations [5]. It is distinguished from tele-

phone switch in numerous ways, such as, message transmissions are connectionless operations; the time delay can be tolerant compared with voice calls; and during the network congestion, it would hold data message temporarily for later transmission, and so on.

Packet switching has the same concept as the message switching except each message is limited by a packet length [5]. For a long message, it can be divided into a number of packets and sequentially transmitted over the networks. At the receiver side, these packets are reassembled together by a predefined order to restore the original message before passing onto the end user.

The packet switching, which is described in this chapter, usually refers to the packet distribution technology rather than a specific network switch node.

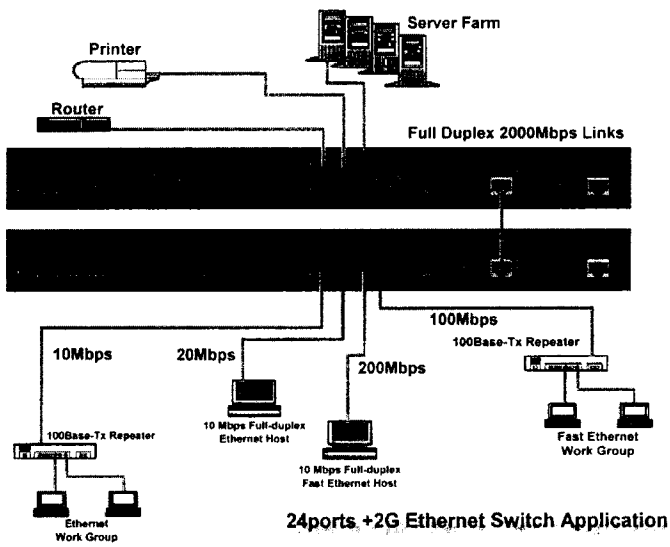

Figure 1.1: A 24 Ports Ethernet Switch (taken from www.netsys.com.tw)

As an example of the packet switch, Figure 1.1 shows a typical modern Ethernet

switch model, which has 24 ports with up to $2Gbps$ links.

The two basic functions of packet switching are store and forwarding (sometimes refer to as buffering and routing) of data packets [5]. Abstractly, packet switching can be described as a box with N inputs and N outputs. According to a packet destination, internal connection will be provided to the packet during its transmission. For the synchronized packet switching, during each time slot, the internal connections could be different according to the demands of incoming packets. Besides the two basic functionalities [5], some packet switches have priority service classes, which provide services with different resource allocation schemes according to the packet priority classes. Another functionality is broadcast and multicast, which are also commonly implemented in packet switches to distribute a data packet to multiple destinations. These special functionalities are important features to provide Quality of Service (QoS) to end users and improve infrastructure utilizations.

1.2.1 Packet Switching Architectures

Based on different criteria, packet switching can be classified into different categories. For example, based on how internal conflicts are handled, packet switches can be divided into internal non-blocking and blocking switches. Based on the method of switching operation, they can be classified into three types, shared memory type, shared medium type and space division type. An abbreviative description of these three types is provided here based on [3], [5], and [6].

1.2.1.1 Shared Memory Switch

Shared memory switch, sometimes referred to as the first generation packet switching, usually consists of a central computer with a dual-port memory shared by all switch inlets and outlets. Internally, the memory is divided into separate logic queues corresponding to each transmission link. At each time slot, all arriving packets from input links are multiplexed into a single stream and written to the shared memory; then, all packets at the head of each logic queue are retrieved and fed to their output links for departure. All receiving and departing operations are controlled by the central computer.

The shared memory switch appeared at the early stage of packet switching, while the data networks were still in small scale with less users and applications. The bandwidth of shared memory should be at least $2N$ times faster than link rate, and the central computer should run even faster to accomplish all operations in one time slot. These critical conditions limited the data transmission rate and their applications. As the network technologies progressed, the shared memory switch is no longer suitable for high speed data transmission applications.

However, the shared memory switch still has some remarkable advantages: Firstly, it is an internal non-blocking switch, because every incoming packet will be stored in memory unless memory is full; secondly, the memory size is the most optimized because the sharing mechanism eliminates memory redundancy; thirdly, it is easy to implement using single Large-Scale Integrated (LSI) circuit. These advantages enable the shared memory switches still exist in some small scaled data communication networks.

1.2.1.2 Shared Medium Switch

The shared medium switch is also known as the second generation packet switch. The shared medium switch usually deploys a shared high-speed medium, such as a parallel bus, that is capable to carry all arrived packets. At the output side, each link has an interface controller, which tests the identification on the head of each packet. If it has a match, the packet will be accepted and stored in its First-In-First-Out (FIFO) queue waiting for its turn to depart. This architecture, compared with shared memory switch, distributes the central computation functions to the interface controllers, which improves the operation speed. However, the speed of the shared medium and each buffer memory still should operate at N times faster than the link rate, and the same as the interface controllers, which is not feasible with the rapid development of computer communication networks.

1.2.1.3 Space-Division Packet Switch

In space-division packet switches, multiple concurrent paths can be established during each operation slot to transmit packets from their inlets to their destination outlets. This is also generally referred to as the third generation packet switch. In each time slot, the internal connections could adjust according to packet demands. Also, each Switch Element (SE) implements routing functionalities with distributed routing algorithms, so the centralized controller is not necessary anymore. The commonly used space division packet switch architecture is the multistage interconnection network. The crossbar switch belongs to this category.

- Crossbar Switch: For an $N \times N$ switch, the N inputs and N outputs can

be configured as a two-dimensional array with a switch at each cross-point on the grid. During each time slot, if the destinations of the N arrived packets are different from each other, N distinct paths will be setup by turning on N switches at the corresponding grid of each input-output pair, then all N packets will be transmitted in one time slot. However, when multiple packets are destined to the same output links, internal conflicts are not avoidable, and some queuing schemes must be deployed to improve the overall throughput of the switch. The major drawback of crossbar switch is its high complexity, which is in the order of $O(N^2)$ and is measured in terms of the number of cross-points. Therefore, the crossbar switch is not suitable for large switch fabric.

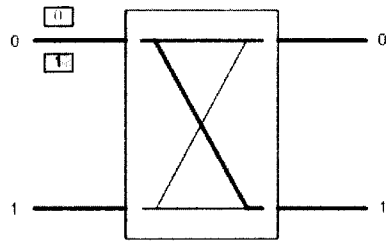

Figure 1.2: A 2×2 Switch Element

- Multistage Interconnection Network Based Switch: One typical multistage interconnection network is the Banyan-based network [5]. The Banyan network is constructed by interconnecting several switch elements as its basic building blocks. A 2×2 switch element is shown in Figure 1.2, which contains two inputs and two outputs. It has four internal paths for all possible combinations of a packet with different destinations. The routing bit for each stage at the

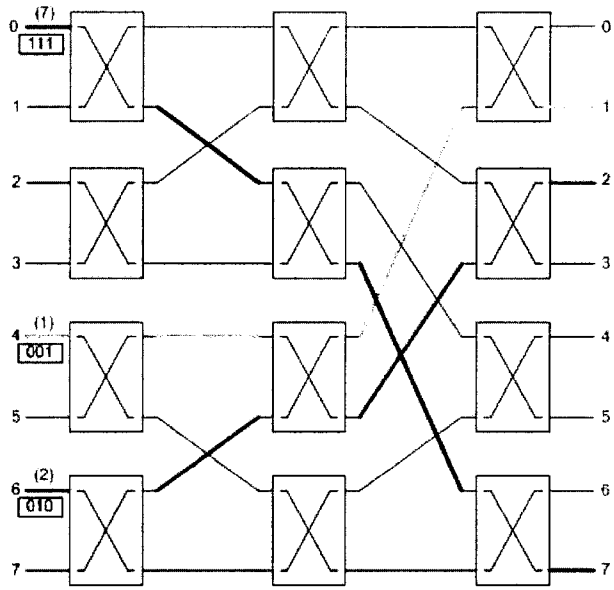

Figure 1.3: Multistage Switch Fabric: Banyan Network

packet header, either a 0 or 1, will determine whether the packet will be sent to upper or lower link. An 8×8 Banyan network switch is shown in Figure 1.3, which is comprised of three stages and with 4 switch elements in each stage. For example, as shown in the figure, input 0, 4 and 6 have packets destined to output 7, 1 and 2, respectively. The destinations are represented in their binary format as shown in the figure. The first bit (most significant bit) determines the routing at the first stage; the second bit is used in stage two; and the last bit is used in the last stage for routing decision. In this way, all three packets are routed to their destinations.

Other multistage interconnection network, such as, the *OMEGA* network and the *delta* network, etc., can be referred to [5] for more details.

- Multi-Path Switch: A Balanced Gamma Switch [7], [8] is used to illustrate the multi-path switch as shown in Figure 1.4. In the Balanced Gamma network, packets can be routed through even in the presence of failures of some of the SEs in the network [8]. Because of its multi-path property and efficient routing algorithm, Balanced Gamma network can provide outstanding performance, single fault-tolerant and robust when multiple faults exist, along with highly reliable and scalable properties.

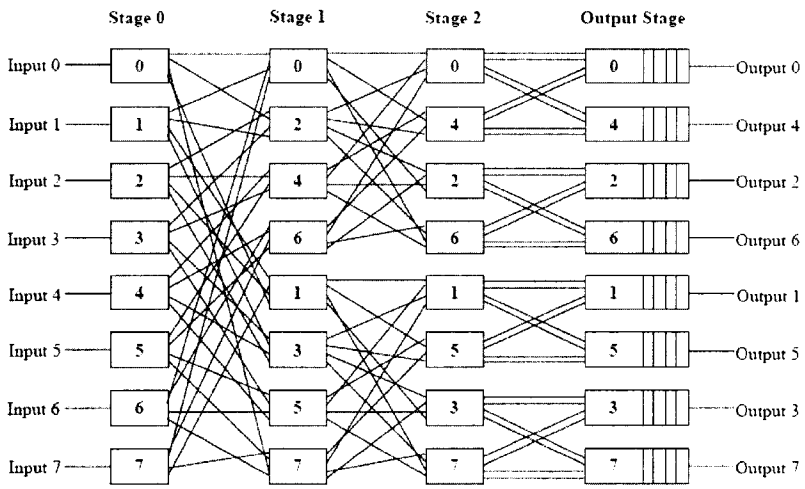

Figure 1.4: Multiple-Path Switch Fabric: Balanced Gamma Switch

1.2.2 Queuing Strategy

In order to store packets in switch fabric, packet buffering, also known as queuing, is usually implemented inside the network switches. Queuing systems can be classified

into input queue, output queue and shared queue depending on the different buffer locations.

1.2.2.1 Input Queuing

For input queuing, packet buffers are implemented at the input side of a switch fabric [9]. When packets arrive at inlets with more than one packet destined to a same outlet, they will compete for transmission at the switch fabric. Usually, only one packet will be switched, others will be blocked and have to be buffered in their corresponding buffers for temporary storage and will be tried again.

The most commonly known service discipline in a queuing system is First-Come-First-Served (FCFS), that is, if the queue buffer is not empty, the newly arrived packets have to stay in the line to wait for their turns. Here, a problem arises that if a packet at the head of line experiences a longer waiting time, then, all other packets behind it have to wait, even though they are destined to idle output links. This is usually known as the Head Of the Line (HOL) blocking. Because of that, the maximum throughput of a purely input buffered switch is about 58% [3] under uniform random traffic. The performance becomes even worse when the applied traffic is getting burstier.

To improve the throughput, some techniques have been developed to solve the HOL problem, such as, the Virtual Output Queue (VOQ) scheme. The VOQ divides the input buffer into a number of virtual output queues, and all arriving packets will be enqueued to their corresponding virtual output queues. In this way, the switch can maximize the throughput of the switch.

1.2.2.2 Output Queuing

The output queued switch buffers packets at output side after they are switched through the switch fabric [9]. In the case of multiple packets destined to the same outlet at the same time, each output queue usually accepts up to k packets simultaneously, while $1 \leq k \leq N$. Generally, k is called the knockout factor and its value can be engineered according to different switch sizes and traffic conditions.

As is known, if the k is equal to link number N, then the switch are totally non-blocking, which is an ideal but impractical for large switches. However, if k is properly chosen, the switch can still achieve close to 100% throughput [9]. Moreover, with an output queue, packets are only waiting in the buffer for unavoidable congestions because of multiple arrivals or previous arrivals, so the best delay performance can be achieved compared with other queuing approaches.

The drawback of this scheme is the potential speed up of k [3]. Another technique to reduce memory speedup limitation is to use concentrator which increases the hardware complexity, and increases cell loss in the concentrator [6] and [9].

1.2.2.3 Shared Queuing

The shared queuing has similar architecture as mentioned in shared memory switch, that is, all packets are stored in a shared memory [9]. At each time slot, all arrivals are multiplexed into a single stream and written to the memory; all outgoing packets are retrieved from the memory and demultiplexed to feed into corresponding output links. The shared queue achieves 100% throughput and optimized delay performance. However, the cost is unaffordable - $2N$ times speed up for both the shared memory

and the central computer.

From the previous discussion, it is clear that each queuing strategy has its advantages and drawbacks. In practice, schemes which combine different strategies may be deployed in switch fabric to achieve high performance. For example, some network switches use parallel switch planes to transmit packets, whereas some others implement recycle paths for those unsuccessful packets resulting from the port competitions.

References [1], [6], and [9] provide more detailed information for different switch models and queuing architectures.

1.3 Motivation

For many packet switches, output queuing has been one of the most popular buffering strategy and it has been widely used, such as in the knockout switch [10], Pinium switch [11], etc. It has been recognized that the output-queued switch possesses the best throughput and delay performance [12]. Therefore, we decide to conduct our research based on an output queued switch. However, if one output queue is dedicated to each port, the buffering space for each queue usually should be allocated based on the worst case traffic requirements. While, under normal traffic conditions, the applied load is far less from its peak load. Moreover, for a switch with a large number of ports, it is quite possible that the momentarily unbalanced traffic experienced in the switch would cause some output queues suffer from overly intensive load where packets discarding is unavoidable, while other queues may remain in the normal condition or even in the idle state.

This thesis will show that how the shared queue will provide better performance on the time delay and cell loss rate. Cell discarding will only occur when the entire shared buffer space is full. In this way, it can smooth out the unbalanced traffic load. The delay performance is optimized because a cell is only delayed by unavoidable waiting time in the FIFO. In addition, since the shared queue can accommodate more cells, the buffer utilization will be improved.

1.4 Thesis Organization

New generation routers and packet switches usually have a large number of ports with very high link rates. With the rapid increases of network applications, data traffics on the Internet are difficult to measure and cope with. As a result, a large queuing buffer is usually equipped in order to accommodate the worst-case scenario. However, under normal traffic conditions, buffer utilization is rather low. Improving the buffer efficiency and minimize possible packet loss has become a major concern for broadband switch design and implementation.

In our research, we proposed a reconfigurable buffer sharing scheme, which follows the advanced hybrid SRAM/DRAM approach to enhance the queuing performance for high speed switches and routers. It is known that, this architecture is the fastest buffering scheme with the worst case bandwidth guarantees [13]. The proposed scheme targets at improving buffer utilization, while not posing much constraints on the buffer speed by dynamically adjusting buffer spaces for individual ports according to traffic patterns and buffer saturation status.

The organization of the thesis is as follows: Chapter 2 provides a technical review

of various queuing systems and involved memory technologies. A brief introduction of our proposed shared queue architecture is presented too. Chapter 3 provides a numerical model to analyze the SOQ performance under uniform traffic. In Chapter 4, we present the performance comparison for different shared queue schemes under non-uniform traffic. Chapter 5 describes the design and implementation of the re-configurable SOQ system using hardware description language VHDL and 0.18 *um* CMOS technology. Chapter 6 gives the thesis conclusions and future work.

Chapter 2

Memory Technology and Network Queuing Systems

Modern memories can be simply classified into *volatile* and *nonvolatile* memory which depends on whether a constant power is supplied for data sustainment [14]. Currently, two widely available memory, *Static Random AccessMemory* (*SRAM*) and *Dynamic Random Access Memory* (*DRAM*) [15] are volatile memory, which can perform both reading and writing operations. This chapter firstly provides a brief introduction of SRAM and DRAM technologies, then proceeds with their deployments in network switches. The proposed reconfigurable shared output queue architecture is presented in the last section.

2.1 Random Access Memory Technologies

Random Access Memory (RAM) is initially used to hold program codes and data for computers. Random access means that any locations in the memory can be written

to or read from at any time, regardless of the last accessed location [16]. RAM is classified as a volatile memory, that is, data stored in its memory cell will be maintained upon a persistent power supply.

2.1.1 Static Random Access Memory (SRAM)

An SRAM chip contains an array of memory cells with support circuitry such as address decoders. The memory array is organized in rows and columns which called word lines and bit lines respectively. During a write or read operation, the address decoder provides the row address and column address, then the corresponding SRAM cell is selected, and the desired data value is written into or read from this location. Figure 2.1 shows a SRAM cell, which is implemented by using 6 transistors.

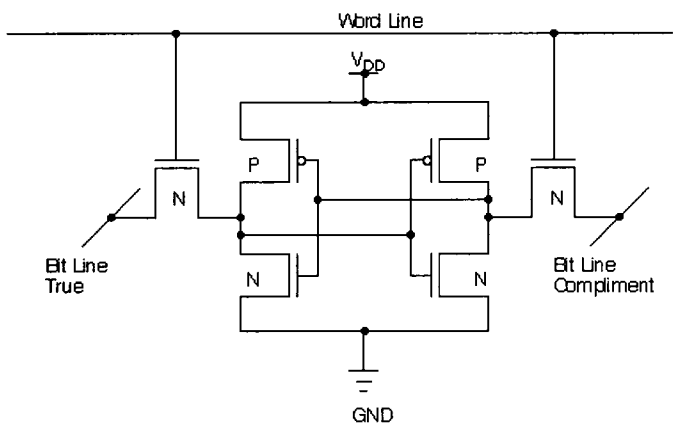

Figure 2.1: An SRAM Cell (taken from http://parts.jpl.nasa.gov/asic)

The performance of SRAM is usually measured in access time and cycle time [17]. The access time specifies a minimum amount of time required to read a bit from the memory, from the initiation of the read operation to when the bit appears on the

data bus. The cycle time indicates the time interval from the end of the previous operation to when the next operation is ready. SRAMs usually responds quickly to the required operations because their access time and cycle time are generally less than or equal to one clock cycle - considering a SRAM runs at $250MHz$, the duration is 4 ns. However, because the leakage and standby currents exist for all transistors, the SRAMs usually consume a large silicon area and power supply.

2.1.2 Dynamic Random Access Memory (DRAM)

Dynamic Random Access memory is probably the most widely used semiconductor memory because of its higher storage density and advantage on cost per bit. A DRAM cell is usually implemented using a storage capacitor and a single transistor as a switch [18]. The charge presented on the capacitor indicates a logical value 1, and its absence is 0. However, DRAMs have more complex operation modes and require additional circuitry to sustain the data value. During a reading process, the data bus which is connected to the corresponding data cell should be pre-charged. Then, the stored data on its capacitor leads to a very small voltage change, which causes a response on its sense amplifier. In this way, the data will appear on the data bus. A writing operation involves charging or discharging of the capacitor to the desired voltage level. Because of the leakage current, the value stored in the capacitor degrades gradually even with a constant power applied to it. Therefore, constant refresh operation has to be performed at a periodic interval to preserve its value.

The Random Access Cycle Time (T_{RAC}) is defined as a minimum time interval

for two consecutive accesses to a DRAM device [17]. This time interval includes the time to complete the first operation, the pre-charge or recovery time, and the address select re-active time. The T_{RAC} is an important DRAM measurement parameter because it determines the memory operation speed. Compared with the computer clock speed, the DRAM access time is usually very slow. To improve the DRAM performance, instead of faster retrieving of data, the trend of the DRAM technology is to provide a large volume of data during each access, which means that a wider data bus and memory bank interleaving have to be used to increase the data transmission parallelism.

Table 2.1 provides a simple comparison about current SRAM and DRAM parameters including access time, density, and power consumption per 100 *Gbit* of storage. The information for SRAM is taken from [19], and that for DRAM is from [20]).

Table 2.1: Typical SRAM/DRAM Parameters

	Access Cycle Time (Approximately)	Max. Density/Chip	Power Consumption/100Gbit (Approximately)
SRAM	4 ns	18 Mbit	10 kW
DRAM	50 ns	1 Gbit	10 W

2.2 RAM in High-Speed Network Applications

Switches and routers are the central components in communication networks whose basic function is to forward data packets. Buffering is a fundamental requirement

to temporarily provide data storage during network congestions. As a widely used rule-of-thumb [21], for TCP flow to work well, packet switch has to buffer all packets passing through in Round Trip Time (RTT), which implies that the required buffer space is usually huge.

The queuing system for high-speed switches/routers is based on memory technologies. Usually, SRAMs are employed to provide faster operation speed. However, its density is low and it is more costly. Therefore, as the memory capacity requirement becomes significant, DRAMs are getting attractive. Although DRAMs can provide massive and dense storage, the memory access time is usually slow. Much research has been conducted in past decade on the above mentioned issues for memory technologies.

It is obvious that modern memory technologies have become more and more sophisticated. New techniques, such as cache and memory bank interleaving, have greatly improved the performance of modern computer systems. These techniques had been introduced to packet switches too.

2.2.1 Pipelined Memory Shared Buffer - 1995

The basic concept of memory bank interleaving can be found as early as in 1988 for the Prelude switch [22] and in 1993 for Turner's multicast switch [23]. In 1995, Manolis Katevenis *et al.* proposed the concept of the Pipelined Memory Technique [24], which employed interleaved memory banks as shown in Figure 2.2.

In this figure, the interleaved memory banks take advantage of the fact that the packet access to the memory bank is in a sequential manner. This is because the

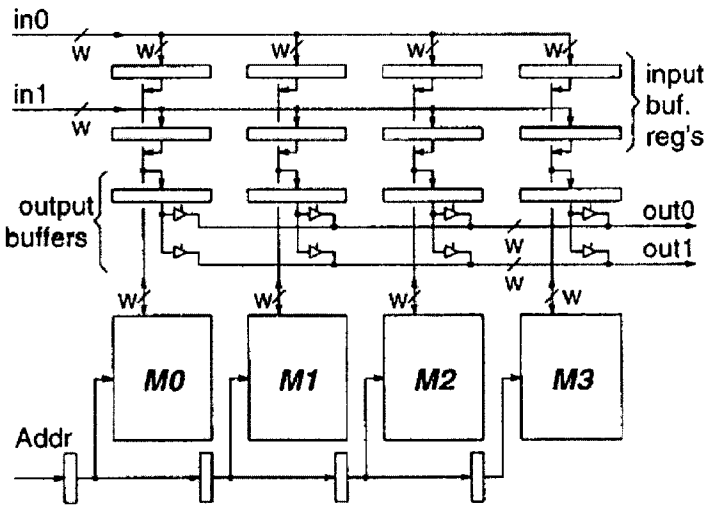

Figure 2.2: The 2 × 2 Shared Buffer using Pipelined Memory (taken from [24])

incoming and outgoing link width is equal to w and packet width is equal to a multiple of w. So, when the first word $w0$ is received by the registers on the left most column, they are ready to be transferred to the memory bank $M0$, while subsequent *words* are in the progress of being registered by $w1, w2$, and $w3$. Because this is a 2-*input* and 2-*output* system, the memory operations have been divided into 4 pipeline stages. As long as the packets are wider than 4 *words*, the memory operations and packet receiving/transmitting can work perfectly without any stall.

2.2.2 DRAM-based Shared Memory - 1997

Later, Chiueh and Varadarajan proposed the architecture of the *Beluga* shared memory ATM switch in 1997, which employed centralized and heavily interleaved DRAMs for a shared queue, together with input and output buffers at the receiving and transmitting ports [25].

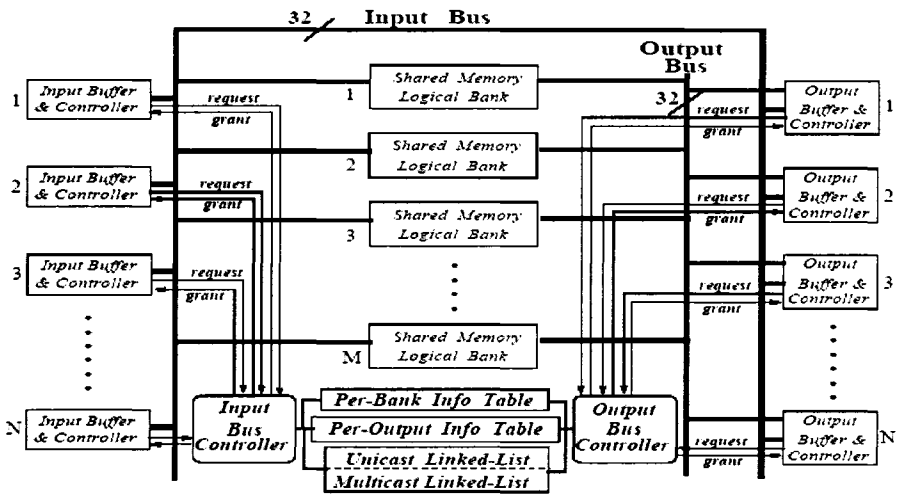

Figure 2.3: The Beluga System Architecture (taken from [25])

The overall *Beluga* system architecture is shown in Figure 2.3, with N connections and M shared memory logical banks. To implement the shared memory with DRAMs, a small amount of SRAMs are used as writing and reading buffers to match with the data bus speed. Inside the *Beluga*, if an arrival packet needs to be stored in the memory, a memory bank will be selected to be activated and receive the entire packet. This clearly indicates that, as the link number N increases, the required memory bank M has to be increased to ensure no bank conflict if multiple packets arrive simultaneously. Because the shared memory banks are basically single port memory devices, the reading and writing contention still exists when both try to access the same memory bank. Under this situation, the writing process will have a higher priority to ensure no packet loss occurs. The reading process has to be delayed. As a result, the corresponding transmission port will be stalled for one round robin

interval.

2.2.3 Hybrid SRAM/DRAM Architecture - 2001

In 2001, S. Iyer and N. McKeown proposed a new hybrid SRAM/DRAM architecture for fast packet switches, which has been recognized as the fastest packet buffer with the worst-case bandwidth guarantee [13]. In the paper, the authors proposed a hierarchical memory architecture, which is shown in the Figure 2.4, and demonstrated the SRAM size calculations and Memory Management Algorithm (MMA).

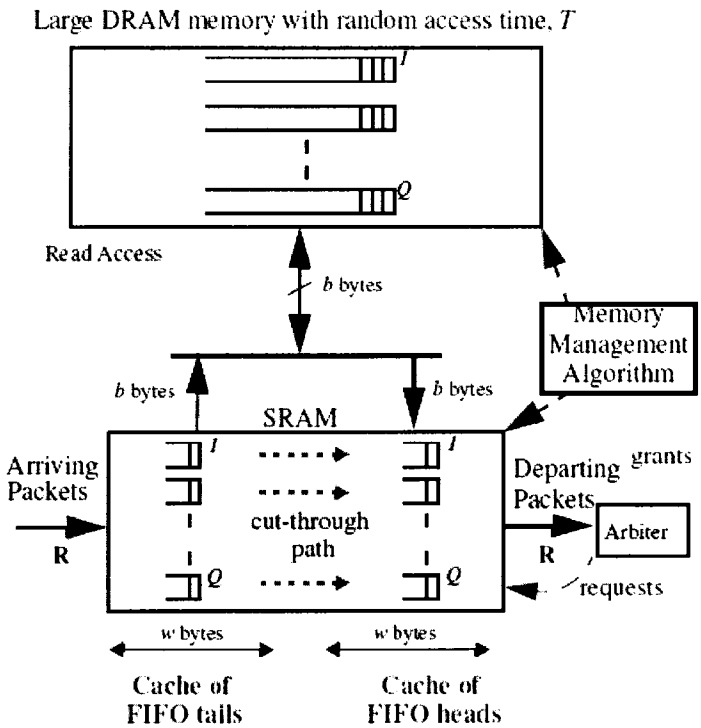

Figure 2.4: Hybrid SRAM/DRAM Packet Buffer (taken from [13])

In their design, an input queue was implemented at each line interface. To improve the performance, the input queue is organized in Virtual Output Queue fashion, where the input queue is virtually divided into N output buffers corresponding to all output links. For each interconnection, two small SRAM buffers are implemented at arriving and departing interfaces to store the head and tail cells and all the rest cells are stored in the central DRAM memory. As the figure shows, the access to DRAM for writing and reading is organized in a block of b cells. The value of b is decided by several factors: the DRAM access time, link speed, memory hierarchy and MMA. After b is derived, the tail and head buffer sizes can be determined to achieve the most optimized SRAM size in the system. The memory management algorithm presented in the paper is called the Earliest Critical Queue First MMA (ECQF-MMA). During every time slot, the central switch scheduler issues a request to each input queue for transmission, and all requests are stored in a *Lookahead Buffer*. Then, all transmissions are executed in the pipeline fashion. The ECQF-MMA has an advantage that the SRAM size is minimized, which is on the cost of an additive pipeline delay to the packets.

Additional efforts have been put forward to improve the performance and control algorithms based on this innovative system. [26] introduced new memory management algorithms, for example, the Most Deficit Queue First MMA (MDQF-MMA) is proposed to guarantee zero pipeline delay for larger SRAM size, and the Most Deficit Queue First is proposed as a compromised solution between ECQF-MMA and MDQF-MMA.

In [27], J. Garcia, M. Valero, and et.al., implemented an input queuing system using the same SRAM/DRAM architecture. The system could support up to $OC-3072$ ($160Gbps$) link rate with about one hundred ports and several service

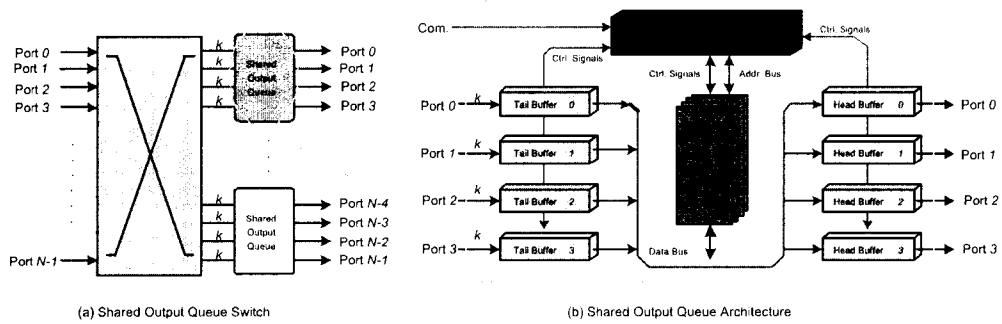

Figure 2.5: Proposed Shared Output Queue Architecture

classes. In their system design, heavy DRAM bank interleaving was introduced, which is called Conflict Free DRAM System (CFDS), with reduced block size of cells (smaller data granularity). To ensure the conflict free access to DRAM banks, out-of-order execution and bank-renaming were deployed. However, the CFDS increases the system complexity and introduces DRAM fragmentations during the system operation.

2.3 Proposed Shared Output Queue (SOQ)

2.3.1 Proposed SOQ Architecture

Based on the previous discussion, we propose a reconfigurable buffer sharing scheme for an $N \times N$ output-queued switch, which is shown in Figure 2.5(a) and 2.5(b). To make it more general, we denote k in the figure as the knockout factor, where $1 \leq k \leq N$, and we use fixed length data cells as our traffic load. Although, data packet could be variable length, we assume that they have been segmented in advance.

In Figure 2.5(b), we show a 4-port shared buffer architecture: each port has

dedicated tail and head buffers which use fast SRAM memory to match with the high link speed. A block of cells buffered in a tail buffer will be transfered to the shared central memory which will be implemented in DRAMs. To improve DRAM efficiency, multiple interleaved memory banks are deployed. Similarly, the reading process will transfer a block of waiting cells to their corresponding head buffers for transmission to downstream nodes. The block transfer of cells between the main memory and those tail/head buffers must obey two rules:

- The tail buffers should not experience overflow before the main memory is full;
- The head buffers are never empty if there are waiting cells in the main memory and tail buffers.

Also, the transmission of the block of data should match the speed and bandwidth requirements between SRAMs and DRAMs. Furthermore, in case that a tail buffer has less than a block of cells but the head buffer is empty, a cut-through transmission will be performed through data bus.

2.3.2 The SOQ High Level Control Algorithm

The top level control algorithm is implemented inside the SOQ, and its pseudo code is shown below. Basically, in a while loop, the algorithm checks the read/write requests from the head buffers and tail buffers, then the corresponding operations are performed. The details of these operations are discussed in *Chapter* 5 when functional units design and implementation are presented.

```
while SystemOperating do
  if WriterRequest then
    if !QueueFull then
      write-transmission
    else
      set-full-flag
    end if
  else if ReadRequest then
    if !QueueEmpty then
      read-transmission
    else if !T-BufferEmpty then
      cutthrough-transmission
    else
      null
    end if
  else
    null
  end if
end while
```

The proposed shared output queue scheme deploys the hybrid SRAM/DRAM architecture, which follows a similar memory technique mentioned in [27] and [13]. In these previous works, the input queue, more precisely, the virtual output queue, is implemented for the switch fabric. The major concern of the research is to replenish the head buffers according to the requests from the switch scheduler. The results

indicate that various memory management algorithms provide different performance on time delays and SRAM size requirements. In the proposed SOQ scheme, we are more focused on the shared memory management to improve the queuing performance on cell loss rate and delay time, which will be discussed in the next two chapters. In *Chapter* 5, the detailed architecture design and implementation will be studied.

2.4 Summary

In this chapter, we have introduced some basic semiconductor memories and their characteristics. In particular, how these characteristics affect their applications in high-speed networks. From the literature review, we find that the hybrid SRAM/DRAM scheme exhibits excellent performance, not only for computer system designs, but also for high speed network applications. It has been recognized as the fastest buffering strategy with guaranteed worst-case bandwidth. Therefore, we propose the shared output queue scheme based on this advanced architecture. In fact, this architecture can be used not only for the design of the input and output queues in broadband switches, but also for the shared memory queue with multiple ports inside the switch fabric.

Chapter 3

Performance Analysis of the Shared Output Queue under Uniform Random Traffic

The numerical methods and analytical methods are theoretical foundations for the performance analysis of queuing systems. In our study, an analytical model is developed under uniform random traffic which provides numerical results of performance of our proposed SOQ scheme.

3.1 Analytical Modeling

3.1.1 Review

The analytical methods provide theoretical foundations for the modeling of network queues. Many studies have been conducted on the subject of various shared queuing

schemes. The computational complexity is considered to be an important factor for the feasibility of a model in real applications.

In [28], Turner proposed an analytical model, referred to as the *scalar model*, in which the state variables ($s_{i,t} \in \Theta$) are used to keep track of the overall buffer occupancy and determine the status of each virtual queue. However, the calculated results of this model are overly optimized when compared to those from simulations. Then, the *vector model* [29] was proposed, which provided better accuracy than the *scalar model*. Its state, $s = (s_1, s_2, ..., s_b)$, is a vector with b as the number of logic queues, and s_i gives the number of packets destined to a specific output port i. This model considered the correlations among packets inside the shared queue, however, the computational intensity increases severely as the buffer size and switch size increase.

Turner's *bidimensional model* [30] keeps track of both the number of packets in the shared queue and the number of active outputs. On the other hand, Giannatti and Pattavina's *bidimensional model* [31] provides relatively better accuracy and tractability. In this model, the shared queue is studied via a tagged queue (representing a logic queue) and all other logic queues are considered together as an untagged queue. The two variables of the switch state keep track of the content of the tagged queue and the cumulative content of the untagged queue. In this way, the state space is kept reasonable low, even for a large switch size and large number of logic queues. Based on this model, in [32], Abonamah and Dang developed their bidimensional scheme to analyze the behavior of channel grouping with shared queues for ATM switches.

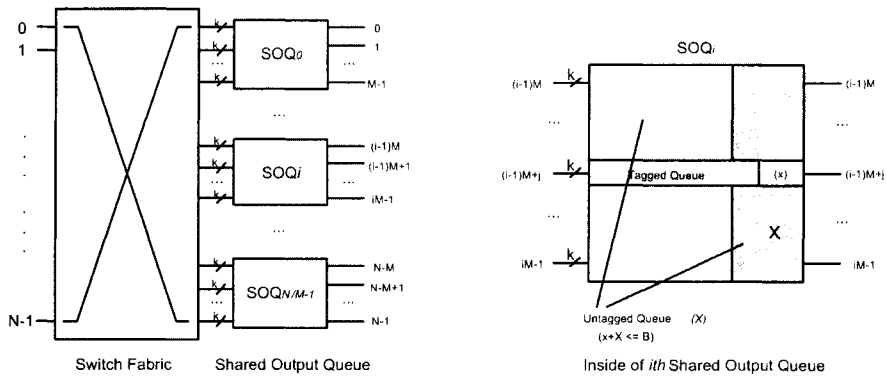

Figure 3.1: Shared Output Queue Model

3.1.2 Performance Analytical Model

In this subsection, we present our performance model of the shared output queue under uniform random traffic, which is modified from Pattavina's *bidimensional model*. The illustrative architecture of the shared output queue (SOQ) is shown in the Figure 3.1. Because of the homogeneity among all SOQ modules, we only focus on one of the SOQ modules, the SOQ_i, where $0 \leq i \leq N/M - 1$, and M is the number of logic queues sharing a common buffer with a total of B cell spaces. Because cell destinations are uniformly distributed, any queue in SOQ_i can be chosen as the tagged queue, which is denoted by ϕ, the other $M - 1$ logic queues are grouped together as the untagged queue which is denoted by Φ.

For a stable queue, its behavior can be described by the state probabilities and state transitions. It is clear that two operations are involved, that is, writing cells into the buffer during an arrival process and reading cells from the buffer during a departure process. Therefore, we define the queue state (x, X) as the state of the system with x cells in the ϕ and X cells in the Φ. Because the cell number x and

X could be arbitrary numbers within the boundary B, the state at a given time t is $(x, X) \in \Theta$, where Θ is the collection of all valid states.

The following notations are defined for later use:

- $\Psi_t(x, X)$: the state probability of SOQ_i at time slot t with x cells in the ϕ and X cells in the Φ.
- $\Psi_{t'}(v, V)$: the intermediate state probability after the arrival process, with v cells in the ϕ and V cells in the Φ.

By introducing the intermediate state, we can decompose the operation in one time slot into two parts: first, the arrival process and followed by the departure process. Given the initial SOQ_i state probability $\Psi_t(x, X)$, after the arrival process, the intermediate state probability is

$$\Psi_{t'}(v, V) = \sum_{\forall (x,X) \in \Theta} \Psi_t(x, X)\gamma_A(a, A), \tag{3.1}$$

where $\gamma_A(a, A)$ is arrival probability that a cells arrives to ϕ and A cells arrives to Φ, and state $(v,\ V)$ is equal to $(x + a,\ X + A)$.

Similarly, after the departure process, the SOQ state probability is given by

$$\Psi_{t+1}(y, Y) = \sum_{\forall (v,V) \in \Theta} \Psi_{t'}(v, V)\gamma_D(d, D), \tag{3.2}$$

where $\gamma_D(d, D)$ is the departure probability that d cells depart from ϕ and D cells depart from Φ, and the next state $(y,\ Y)$ is equal to $(v - d,\ V - D)$.

Next, we examine and analyze the two processes in detail.

3.1.3 Arrival Process

We assume that cell arrival is an independent and identically distributed (i.i.d.) random process with the probability of p across all switch inlets. Cell destinations are uniformly distributed to all outputs. We further assume that the switch fabric is ideal and nonblocking, so that cell loss is only caused by buffer overflow. Under such circumstances, a simple Queue Loss (QL) mechanism will be deployed so that all exceeded cells will be dropped. If we define Θ as the set of all valid states, $\forall(x, X) \in \Theta$, then,

$$x \leq B,\ X \leq B,\ x + X \leq B, \tag{3.3}$$

where B is the total buffer size for the share queue.

Clearly, under the i.i.d. random traffic, cell arrival to ϕ and cell arrival to Φ are independent random process, so the joint probability of $\gamma_A(a, A)$ can be written as

$$\gamma_A(a, A) = \gamma_a(a)\gamma_A(A), \tag{3.4}$$

where $\gamma_a(a)$ is the probability of having a cells to tagged queue ϕ and $\gamma_A(A)$ is the probability of having A cell to untagged queue Φ.

Let us define $\beta(N, k, p)$ as the probability that there are k cells destined to an output queue in a time slot, given that p is arrival rate at each switch inlet and N is the switch size, and we have

$$\beta(N, k, p) = Pr\left[K = k\right] = \binom{N}{k} (p)^k (1 - p)^{N-k}. \tag{3.5}$$

Therefore, under random traffic condition, arrival process for output queues can be modeled by the binomial distribution. Let us denote S_t the available cell space inside

the SOQ_i at time t, that is,

$$S_t = B - (x + X),$$

the probabilities $\gamma_a(a)$ and $\gamma_A(A)$ can be written as

$$\gamma_a(a) = \beta(N, a, p/N), \text{ while } (a + A) \leq S_t \tag{3.6}$$

$$\gamma_A(A) = \beta(N, A, p(M-1)/N), \text{ while } (a + A) \leq S_t. \tag{3.7}$$

During a time slot, the number of cells arrives at SOQ_i must not be larger than N (100% traffic load has N cells arrival), then all pairs of (a, A) satisfy following constraints

$$\begin{cases} a + A \in [0, N] \\ a \in [0, N] \\ A \in [0, N]. \end{cases} \tag{3.8}$$

When $(a + A) > S_t$, the exceeded cells will be discarded, which means that the state transition from (x, X) to (v, V), where $v + V = B$, will occur for all ($a + A = S_t$+1, S_t+2, ..., N). In this case, if we define Δ as a set including (S_t+1, S_t+2, ..., N), then cell arrival probabilities to the tagged and untagged queue when $(a + A) \in \Delta$, γ_a and γ_A respectively, can be calculated by

$$\begin{cases} \gamma_a(a_s) = \sum\limits_{\forall(a)\in\Delta} \beta(N, a, p/N) \\ \gamma_A(A_s) = \sum\limits_{\forall(A)\in\Delta} \beta(N, A, p(M-1)/N). \end{cases} \tag{3.9}$$

where $a_s + A_s = S_t$.

3.1.4 Departure Process

Next, we examine the departure process. In our switch model, one cell departs from each logic queue unless its buffer is empty. Because the departure processes for all

logic queues are independent, the probability of having d and D cells departing from the tagged queue and the untagged queue, respectively, can be write as

$$\gamma_D(d, D) = \gamma_d(d)\gamma_D(D). \tag{3.10}$$

The probability of d cells departing from the tagged queue ϕ is

$$\gamma_d(d) = \begin{cases} \gamma_d(d=1) = 1, & \text{while } v > 0 \\ \gamma_d(d=0) = 1, & \text{while } v = 0, \end{cases} \tag{3.11}$$

which indicates a deterministic cell departure of one will occur when the tagged queue ϕ is not empty. Then, the next state of tagged queue is $y = v - 1$.

For the untagged queue Φ, at time $t^{'}$, V cells are stored in the $M-1$ logic queues. It is possible that all V cells belong to one logic queue, or they could be distributed to the $M-1$ logic queues. It is necessary to find out how many cells will depart in the given time slot. We define a queue as Active Queue (AQ) which contains at least one cell in its logic buffer. In this way, if we know the number of AQ within $M-1$ logic queues, we know the number of departing cells in the current time slot. The approximation of the AQ value is based on [30], and the essential assumption is that all combinations of the V cells existing in all $M-1$ logic queues are equiprobably distributed. Then, the problem is to find out the probability of AQs in the untagged queue Φ with V cells in it.

We define the function

$$\sigma(m, n) = \binom{m+n-1}{n-1}, \tag{3.12}$$

which gives all possible combinations that the sum of n nonnegative integers is equal to m [30], [32]. For example in the case of $m = 3$ and $n = 2$,

$$\sigma(3, 2) = \binom{3+2-1}{2-1} = \frac{4!}{(4-1)!1!} = 4.$$

That is to say, there is a total of 4 ways for 2 nonnegative integers to sum up to 3. Based on this, we calculate the probability of D active queues, where $1 \leq D \leq M-1$, and V cells in all untagged queue,

$$Pr(D) = \frac{\binom{M-1}{D}\binom{V-1}{D-1}}{\sigma(V, M-1)}. \tag{3.13}$$

In this equation, the denominator gives all possible ways that V cells are distributed among the $M-1$ logic queues, and the numerator gives the total number of ways that V cells are distributed to D active queues times the binomial coefficient D chosen from $M-1$. With the number of AQs derived, the probability of having D cells departing from the untagged queue is given by

$$\gamma_D(D) = \begin{cases} \gamma_D(D=0) = 1, & \text{while } V = 0 \\ Pr(D), & \text{while } V > 0, \end{cases} \tag{3.14}$$

and the next state of Φ is $Y = V - D$.

Therefore, the joint probability of having d and D cells depart from ϕ and Φ, $\gamma_D(d, D)$,respectively, can be written as

$$\begin{cases} \gamma_d(d=0)\gamma_D(D=0) = 1, & \text{while } v = 0 \text{ and } V = 0 \\ \gamma_d(d=1)\gamma_D(D=0) = 1, & \text{while } v > 0 \text{ and } V = 0 \\ Pr(D), & \text{otherwise.} \end{cases} \tag{3.15}$$

After all the transition probabilities have been derived, we are ready to compute state probabilities and performance parameters.

3.1.5 Steady-State Analysis and Performance Parameters

In the performance evaluation, a stable queue and a steady-state computation are necessary, which is assumed for our analytical model. In our analysis, the probabilities

of arrival γ_A and departure γ_D are pre-calculated and stored in two tables. After the initial state is chosen, we compute and record state probabilities slot by slot. In this way, at the end of the desired number of iterations, the corresponding performance parameters can be obtained. For reasonable B and M values, the required space to store the state probabilities is usually reasonably small for our analytical model. In most cases, convergence can be achieved with around 50 iterations, which indicates that this iterative analytical approach is efficient. The following parameters, which include cell loss rate and average queue occupancy, are studied and compared.

3.1.5.1 Average Queue Occupancy

After the SOQ_i steady-state probabilities have been worked out, the average number of cells in queue, $\overline{N}$, can be obtained as

$$\overline{N} = \frac{1}{M} \sum_{j=0}^{B} \sum_{J=0}^{B-j} (j + J)\Psi(j, J). \tag{3.16}$$

The average delay time can be computed by applying the well-known Little's formula, which stated that the average number of customers in a stable system, Nu, is equal to their average arrival rate, λ, multiplied by their average time in the system, Tw, that is

$$Nu = \lambda \times Tw.$$

It should be noticed that in our calculation, the service time, which is constant, is not included.

3.1.5.2 Cell Loss Rate (CLR)

At any given time slot t, if the remaining buffering space, $S_t = B - (x + X)$, is not enough to accommodate all incoming cells, up to S_t cells will be accepted, and the excessive ones will be discarded. For a given state (x, X), the corresponding Cell Loss Rate $\pi_{(x,X)}$ can be calculated as

$$\pi_{(x,X)} = \Psi(x, X) \sum_{a+A=S_t+1}^{N} \eta\gamma_A(a, A), \tag{3.17}$$

where the factor

$$\eta = \begin{cases} 0 & \text{while } (a + A) \leq S_t \\ \frac{a+A-S_t}{a+A} & \text{while } (a + A) > S_t \ . \end{cases} \tag{3.18}$$

Therefore, the total cell loss rate π can be derived as

$$\pi = \sum_{(x,X)\in\Theta} \pi_{(x,X)}. \tag{3.19}$$

3.1.6 Performance Comparison

Here, we compare the performance results from the analytical model with those from simulations. An ideal 128×128 non-blocking switch is used for the study. The sharing factor M is set to 4 and the shared buffer space B is 32 cells. The cell arrival rate p at switch inlet ranges from 0.6 to 0.9 and the departure rate is 1. For the analytical model, we chose the initial state is $(0, 0)$. Figure 3.2 and 3.3 compare the cell loss rate and average queue occupancy performance for the SOQ-4 scheme. The results from different iterations, $40, 50$ and 60 rounds as in the figure, are almost same, which indicate the queue has been in the steady state. Therefore, we can conclude that

the analytical model can converge in about 50 round iterations, and it is efficient to compute the performance results.

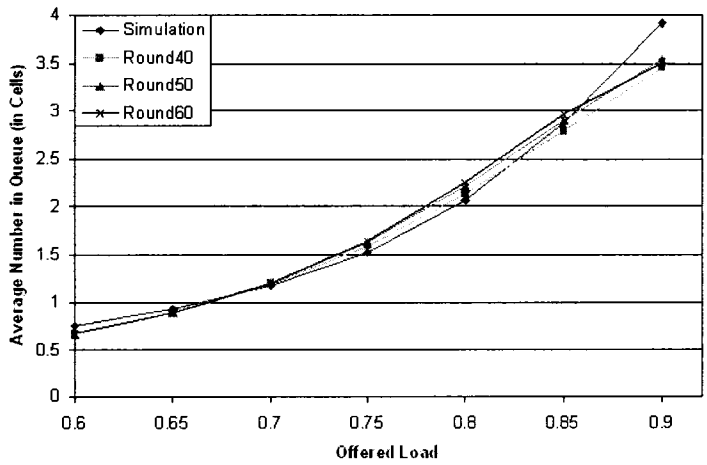

Figure 3.2: Average Number In Queue vs. Offered Load for SOQ-4

In both figures, the results from the analytical model match the simulation results very well under various load conditions. With an efficient buffer sharing scheme, fewer buffer would be required, the state calculation from our model can be be done efficiently.

To demonstrate the advantage of using shared queue scheme, we compared the analytical results of various SOQ schemes with that from the Dedicated Output Queue (DOQ) scheme. Figure 3.4 compares the cell loss rate performance of different schemes. It is clear that all SOQ schemes have better cell loss performance over the DOQ scheme.

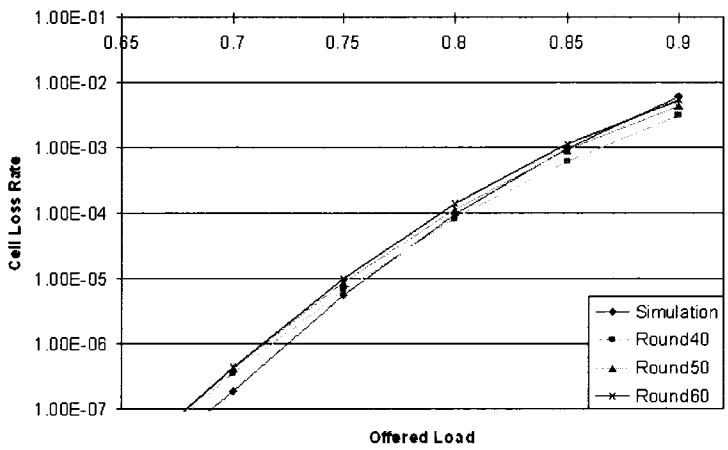

Figure 3.3: Cell Loss Rate vs. Offered Load for SOQ-4

3.2 Confidence Interval Analysis

The Confidence Interval (CI) is an efficient method for statistical analysis of simulation results, which estimates the result reliability and approximates its distribution range. Confidence interval is defined as an estimated range of values which are likely to include an unknown population parameter, the estimated range being calculated from a given set of sample data. Reference [33] provides the basic concepts of CI and its applications, which is briefly introduced here.

3.2.1 Confidence Interval Basics

From the basic statistics theory, for a set of samples $\{Z_1, Z_2, ..., Z_n\}$, the mean value of this set is defined as

$$\bar{Z}(n) = \frac{\sum_{i=1}^{n} Z_i}{n}, \tag{3.20}$$

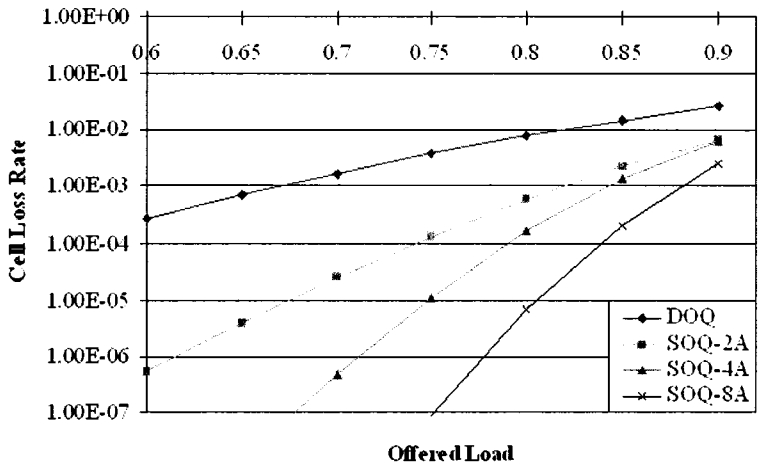

Figure 3.4: Cell Loss Rate Performance for Analytical Models of DOQ and Various SOQ Schemes

and the sample variance is

$$S^2(n) = \frac{\sum_{i=1}^{n}[Z_i - \bar{Z}(n)]^2}{n-1}. \tag{3.21}$$

If the simulation result is $z \subset \{Z_1, Z_2, ..., Z_n\}$, and the set $\{Z\}$ contains independent and identically distributed variables (i.i.d.) with $n \to \infty$, then the sample mean $\bar{z}$ has a normal distribution with Probability Density Function (pdf)

$$P(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-(z-\mu)^2/2\sigma^2}, \tag{3.22}$$

where its mean value is μ and its variance is σ^2, which is shown in Figure 3.5.

We can see, if a is properly chosen, then the covered area could be 90, 95 or 98 percent, as defined as the desired confidence interval $1 - \alpha$. If the whole area under

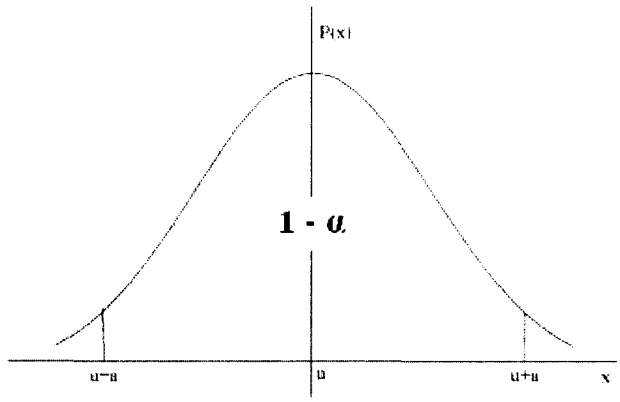

Figure 3.5: Normal Distribution

the curve represents as 1, then the area between $\{-a, a\}$ is:

$$P(u-a<z<u+a)=\frac{1}{\sqrt{2\pi\sigma^2}}\int_{u-a}^{u+a}\exp^{-(z-\mu)^2/2\sigma^2}dz=1-\alpha. \tag{3.23}$$

By replacing $w=(z-\mu)/(\sigma/\sqrt{n})$, so

$$P(-w_{\alpha/2}<\frac{z-u}{\sigma/\sqrt{n}}<w_{\alpha/2})=1-\alpha. \tag{3.24}$$

Here the desired value of this integral, can be calculated be choosing the point $a=w_{\alpha/2}$. So the value of z is,

$$z=\mu\pm w_{\alpha/2}\frac{\sigma}{\sqrt{n}} \tag{3.25}$$

However in practice, it is difficult to use this formula since the Variance σ^2 is usually unknown in advance. In order to estimate confidence interval, the sample's standard deviation S is used under the condition that the set contains large sample

elements. So the formula is modified to

$$z = \mu \pm t_{\alpha/2,n-1}\sqrt{\frac{S^2(n)}{n}}, \tag{3.26}$$

where $t_{\alpha/2,n-1}$ is a value corresponding to the desired α and sample space n. So Equation 3.26 gives the $100(1-\alpha)$ percent confidence interval of z with $n-1$ degrees of freedom. Generally, some typical values of t have been calculated and stored in a table for readers to retrieve, which is also provided in the previously mentioned reference [33].

Now, we conduct the confidence interval analysis of our simulation results.

3.2.2 Confidence Interval Analysis of SOQ Performance

In this subsection, we study the SOQ performance by analysis its confidence interval for the simulation results.

In the simulation, with uniform random traffic, the offered load ranges from 60 to 90 percent is applied. For each round of every experiment trial, the switch runs for 5 million cycles. A total of 50 rounds is repeated to collect the output data. We provide the confidence interval analysis of the average delay and cell loss rate.

3.2.2.1 Delay Performance

Table 3.1 provides the 95% confidence intervals for average delay performance for various offered load conditions. The upper and lower bounds of the interval are shown in the table for various queue sharing schemes, which indicate that 95 percent of simulation results are located within the range. This also means that, in any

number of simulation rounds, only 5 percent of the results will be distributed outside of the range in the table.

Table 3.1: 95% Confidence Interval of Delay Performance

Offered Load	DOQ		SOQ-2		SOQ-4		SOQ-8	
	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower
0.6	0.7362813	0.7363421	0.7440419	0.7441183	0.7441234	0.7441885	0.744099	0.7441632
0.65	0.8993704	0.8994493	0.9207768	0.9208733	0.9213091	0.9214073	0.921233	0.921326
0.7	1.1000013	1.1000906	1.1544122	1.154515	1.1574216	1.1575402	1.1574761	1.1576038
0.75	1.3442313	1.3443317	1.471593	1.471749	1.4879049	1.4880666	1.4881909	1.488359
0.8	1.6352177	1.6353167	1.9048013	1.9049961	1.9787996	1.9790815	1.9842544	1.9845221
0.85	1.9710783	1.9711622	2.4743716	2.4745836	2.733775	2.7341673	2.8055281	2.8059845
0.9	2.3424462	2.3425582	3.1542365	3.1544308	3.7751787	3.7755865	4.1920849	4.1926083

3.2.2.2 Cell Loss Rate performance

The 95% Confidence Interval of Cell Loss Rate is listed in the Table 3.2. It is noticed that zero cell loss is encountered under low traffic load conditions and when a larger queue sharing scheme is used.

Table 3.2: 95% Confidence Interval of Cell Loss Rate

Offered Load	DOQ		SOQ-2		SOQ-4		SOQ-8	
	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower
0.6	0.0004615	0.0004626	0.0000028	0.0000028	-	-	-	-
0.65	0.0011242	0.0011259	0.0000178	0.000018	-	-	-	-
0.7	0.0025299	0.0025319	0.0000973	0.0000978	0.0000002	0.0000003	-	-
0.75	0.0052763	0.0052796	0.0004501	0.000451	0.0000057	0.0000058	-	-
0.8	0.0102256	0.0102299	0.0017481	0.0017501	0.0000934	0.0000939	0.0000004	0.0000004
0.85	0.0184646	0.018469	0.0056218	0.0056252	0.0009864	0.0009883	0.0000533	0.0000537
0.9	0.0311044	0.0311098	0.0147878	0.0147934	0.0060762	0.0060809	0.0019289	0.0019314

3.3 Summary

In this chapter, we have studied some mathematical methods for the performance analysis of queue sharing scheme. Firstly, we have introduced our shared output queue analytical model based on discrete-time Markov chain analysis. Secondly, we briefly reviewed the confidence interval method for verifying simulation results.

In our analytical model, a discrete-time Markov chain model with two dimensional state variables is used to evaluate the Shared Output Queue buffer behavior. The state probabilities and state transition probabilities are derived. According to the traffic arrival rate, the arrival probabilities toward both the tagged queue and the untagged queue are developed. The analysis of departure process has been conducted. With all the aforementioned analysis, the state probabilities and state transition probabilities of the bidimensional Markov chain can be determined.

The advantage of this bidimensional Markov chain model is that, the probability computation converges quickly (within about 50 iterations and reasonable state space), which shows the efficiency of this model. Also, the model provides accurate performance evaluations for average waiting time for cells in the queue, and the estimation of cell loss rate under heavy traffic.

The confidence interval method is a powerful tool for conducting simulation analysis. The confidence interval indicates that as simulation runs for finite times, the results eventually distribute within a specific range. In our research, we calculated the 95% CI range, which can be easily extended to other ranges. Due to the requirement of huge number of simulation runs, the confidence interval analysis is only carried out for the purpose of simulation model construction and initial simulation

results verification analysis. Once verified, we will just use the model to generate the performance results based on the rule of thumb: for example, to claim a better than 10^{-9} cell loss ratio, a minimum of ten times the required packets will ge gererated, that is, at least ten billion cells will be generated and studied.

Chapter 4

Performance Analysis under Non-Uniform Traffic

An analytical model under bursty traffic is usually complex due to the correlation between arrivals, hence, computer simulations are commonly used to evaluate system performance. This chapter studies the performance of our proposed Shared Output Queue under various non-uniform traffic through a serial of simulations.

4.1 simulation Environment

In our research, we develop a simulation platform using the Object Oriented Programming Language *Java*. The simulation platform includes Traffic Generation class, Switch Fabric class and Output class with queuing buffers. For the shared queue, a Controller class is placed in front of the shared buffer. Figure 4.1 and 4.2 shown the relational diagrams for various classes used in the Dedicated Output Queue scheme and the Shared Output Queue scheme, respectively.

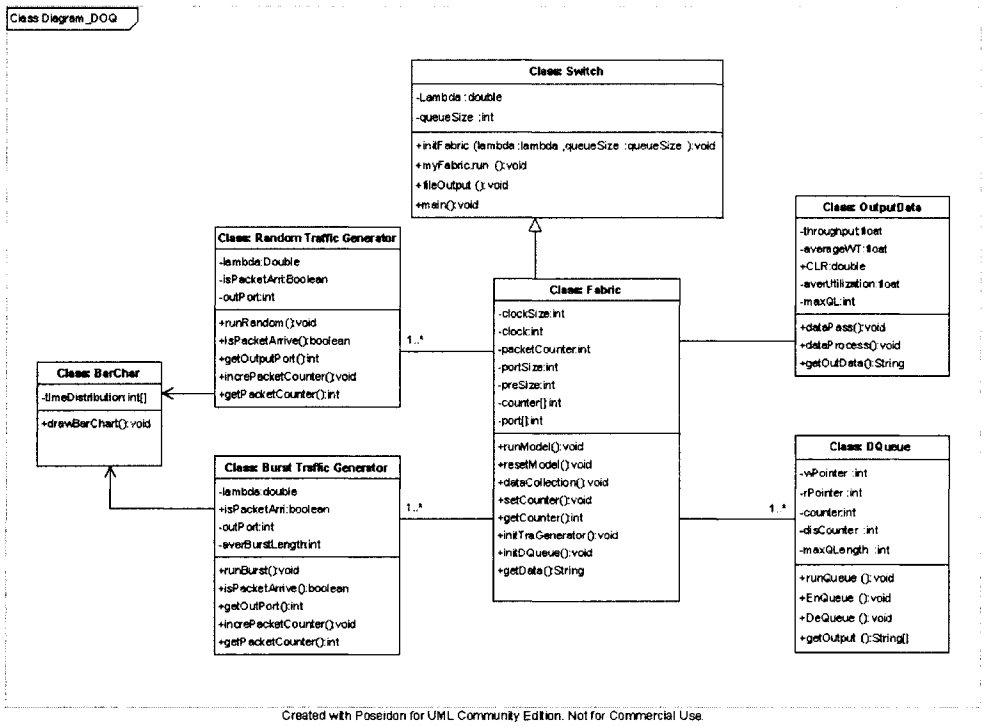

Figure 4.1: Class Diagram for DOQ Scheme

An ideal, non-blocking, 128×128 switch fabric is used for our study. The output buffers are assumed to be ideal, that is, all incoming cells would be enqueued if there are cell spaces available. The traffic source generates bursty traffic with arriving rate λ. A departure rate of one cell per time slot is assumed, i.e., $\mu = 1$, and the offered load in switch fabric is $\rho = \lambda/\mu$. The duration for each simulation lasts for a period of 5 million slots, which yields about 640λ million cells switched through the switch fabric.

It is well-known that the initial conditions affect the precision of the performance data collected in simulations. In our study, a warm-up technique has been used

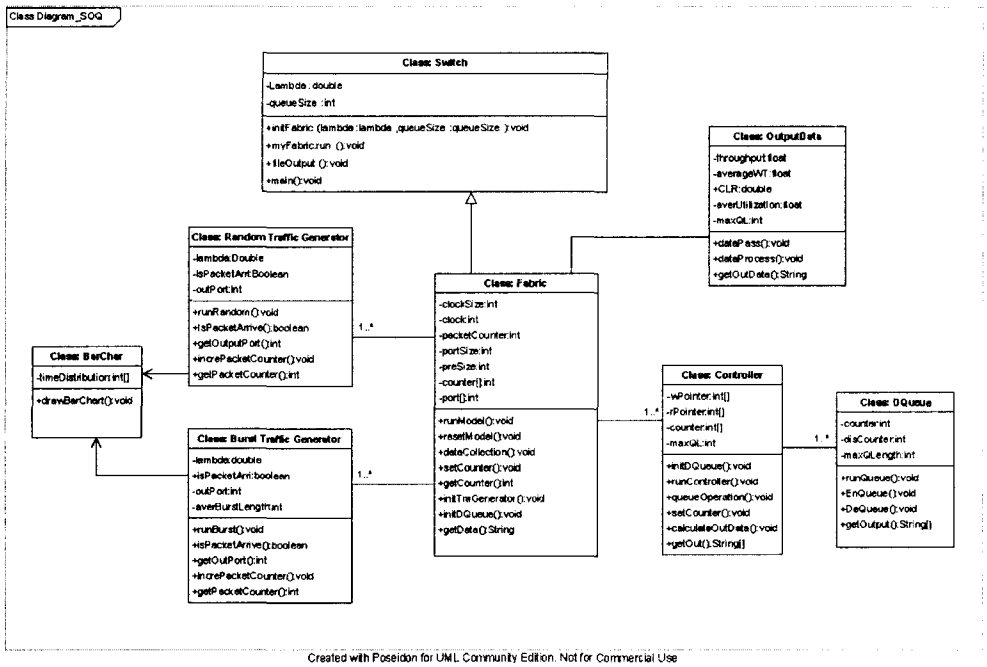

Figure 4.2: Class Diagram for SOQ Scheme

to provide an initial condition to the system. Figure 4.3 shows the queue length parameter for different warm-up periods, and it clearly indicates that the queue length becomes stable if the warm-up period is large enough. According to this figure, we chose 50 thousand as our warm-up period to make sure that all queues have been pre-loaded.

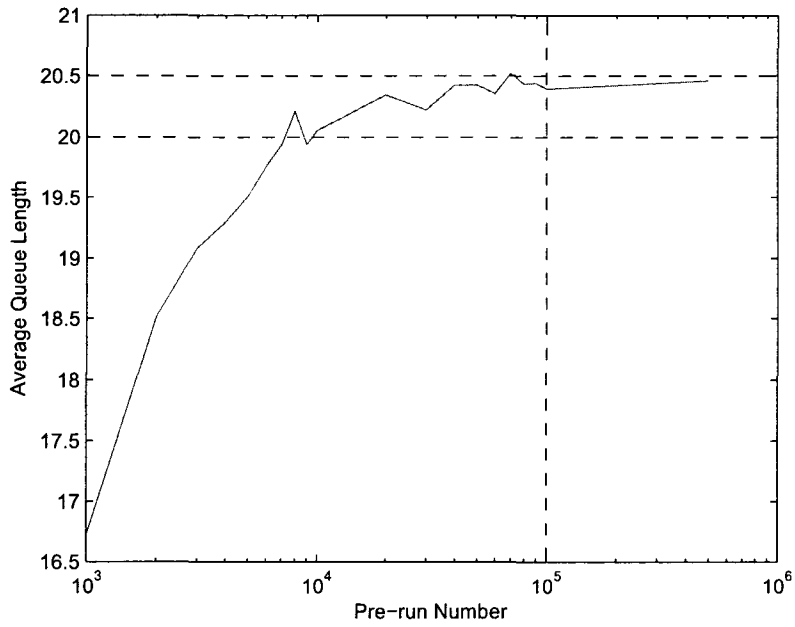

Figure 4.3: Warm-up Period

4.2 Performance under Bursty Traffic

4.2.1 Bursty Traffic Generation

Bursty traffic provides a better resemblance of traffic in real networks. In general, more buffer space would be required under bursty traffic. Therefore we assume 80-cell space is equipped to each port, hence, a total of 10240 cell space is used for the whole switch.

Bursty traffic can be modeled using the ON/OFF Markov-Modulated model [34], which is shown in Figure 4.4 for the state transition diagram. During the busy period (ON state), cell arrives in each time slot with the same destination. Then followed by the idle period (OFF state), during which no cell arrival occurs. For each input

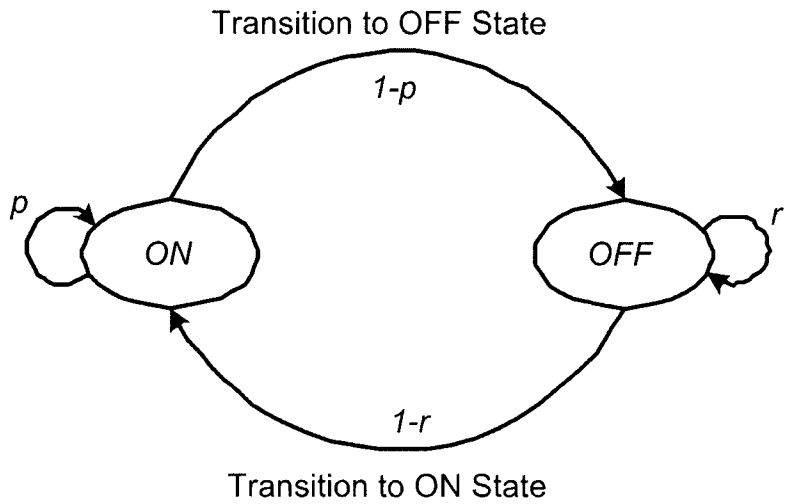

Figure 4.4: ON-OFF Markov Modulated State Machine

link, a new cell will arrive with a probability of p, and the probability of having no cell coming is $1 - p$. Similarly, for idle state, after a slot with no cell arrival, the probability of no cell arrival for the next slot is r, and the probability of having a cell coming is $1 - r$. The duration of busy and idle periods can be modeled as two geometric distributions. The probability of having a burst of s slots is given by

$$Pr_B(s) = p(1-p)^{s-1}, \tag{4.1}$$

where s is burst length, which could be $1, 2, ...$ While the probability of having t consecutive idle slots is

$$Pr_I(t) = r(1-r)^t, \tag{4.2}$$

where t is idle length, which could be $0, 1, 2, ...$ [35]. The slight difference of these two equations indicates that burst length s should not be zero, which also means no two consecutive idle periods existing in this model.

The offered load to the link, ρ, is given by

$$\rho = \frac{\overline{s}}{\overline{s} + \overline{t}}, \tag{4.3}$$

where $\overline{s}$ and $\overline{t}$ are the average length of ON and OFF periods. For the geometric distribution, the mean burst length, $\overline{s}$, is $\frac{1}{p}$, and the mean idle length, $\overline{t}$, is $\frac{1-r}{r}$. Based on the offered load ρ and the chosen average burst length $\overline{s}$, we can calculate the value of p and r. Then the runtime burst length and idle length can be calculated in the simulation using the Inverse Transform Technique [36].

4.2.2 Performance Evaluation

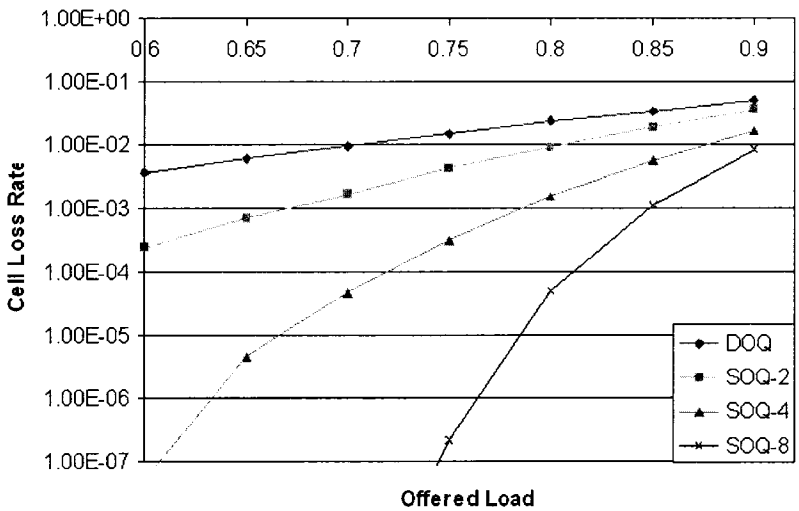

Figure 4.5: Cell Loss Rate vs. Offered Load

In the following discussion, we assume the average burst length $s = 10$ time slots

unless otherwise indicated. An 128×128 switch is assumed, with traffic load λ ranges from 0.6 to 0.9. The buffer size is set to 80 cell space per port.

Figure 4.5 shows the Cell Loss Rate for different traffic load. It is clear that the performance of a switch under bursty traffic has degraded significantly when compared to those from the random traffic, although the buffer size per port is already much larger. Nevertheless, the performance enhancement via buffer sharing is obvious, for example, under 80% offered load, the cell loss rate for the DOQ scheme is about 2.7×10^{-2}. However, the number is reduced to 1×10^{-2}, 3.1×10^{-3}, 4.2×10^{-4} when 2-port, 4-port and 8-port sharing schemes are used, respectively. Table 4.1 shows the 95% confidence interval for the cell loss rate performance under bursty traffic.

Table 4.1: 95% Confidence Interval of Cell Loss Rate

Offered Load	DOQ	SOQ-2	SOQ-4	SOQ-8
0.6	0.0049992 - 0.0050079	0.0004246 - 0.0004284	0.0000052 - 0.0000057	-
0.65	0.0079848 - 0.0079980	0.0010609 - 0.0010664	0.0000327 - 0.0000337	-
0.7	0.0123672 - 0.0123833	0.0024672 - 0.0024767	0.0001785 - 0.0001812	0.0000012 - 0.0000015
0.75	0.0186487 - 0.0186665	0.0053597 - 0.0053725	0.0008144 - 0.0008207	0.0000294 - 0.0000309
0.8	0.0273647 - 0.0273837	0.0107947 - 0.0108099	0.0030776 - 0.0030886	0.0004225 - 0.0004274
0.85	0.0390632 - 0.0390922	0.0200768 - 0.0201043	0.0092631 - 0.0092831	0.0033558 - 0.0033671
0.9	0.0542670 - 0.0542921	0.0344335 - 0.0344631	0.0220594 - 0.0220843	0.0139168 - 0.0139428

Because larger buffer size is used for bursty traffic, the average number of cells in the queue increases for all schemes, as shown in Figure 4.6. Under light traffic condition, the results for all schemes are close. When the offered load increases, that performance using the SOQ schemes increases faster than that of the DOQ scheme. For example, at 90% offered load, the average number of cells using the SOQ-8 scheme is over 60 cells, whereas it is around 30 cells for the DOQ scheme.

Reasons that account for such observation are: the performance of average number of cells in the queue is measured based on the cells that are successfully accepted by the output queue; During the *ON* period under the bursty traffic, it is very easy for the buffers dedicated to certain output ports (in the case of the DOQ scheme) to experience overflow which results in significant amount of cell loss, while other ports have relatively low utilization. Therefore, average number of cells in the queue is low. However, with shared buffer schemes, significant enhancement in buffer utilization can be achieved by distributing the excessive cells for one port to other ports, which yields a large number of cells in the queue. Table 4.2 shows the 95% confidence interval for this performance measurement.

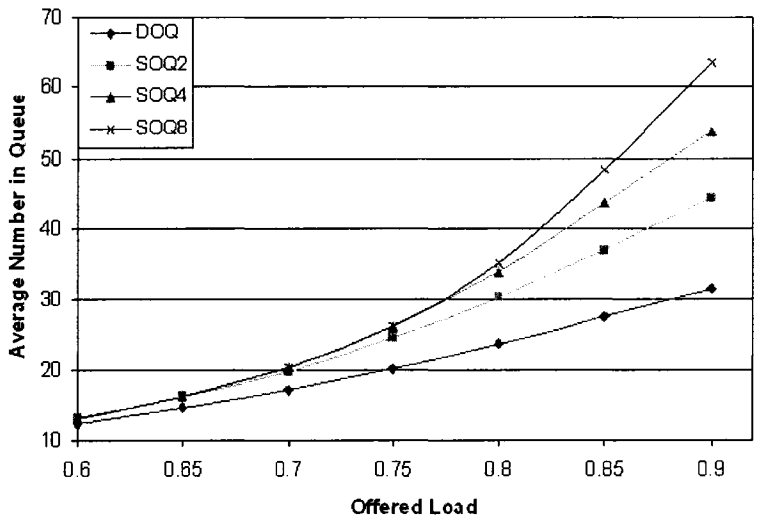

Figure 4.6: Average Number in Queue vs. Offered Load

In Figure 4.7, we show the cell loss rate for different buffer sizes under 85% of-

Table 4.2: 95% Confidence Interval of Average Number in Queue Performance

Offered Load	DOQ	SOQ-2	SOQ-4	SOQ-8
0.6	12.8593 - 12.8625	13.9600 - 13.9648	14.1031 - 14.1091	14.1018 - 14.1077
0.65	15.2036 - 15.2075	17.0615 - 17.0674	17.4504 - 17.4576	17.4618 - 17.4698
0.7	17.8673 - 17.8700	20.8745 - 20.8814	21.8492 - 21.8597	21.9534 - 21.9641
0.75	20.8647 - 20.8690	25.4819 - 25.4893	27.7072 - 27.7218	28.1997 - 28.2153
0.8	24.1853 - 24.1893	30.8644 - 30.8690	35.2943 - 35.3096	37.2567 - 37.2748
0.85	27.7997 - 27.8033	36.8118 - 36.8216	44.1814 - 44.1970	49.3780 - 49.3994
0.9	31.6443 - 31.6480	42.9790 - 42.9863	53.0251 - 53.0371	61.3065 - 61.3272

fered load. This figure can help us to determine the necessary buffer size to achieve desireable cell loss rate. For example, in order to achieve a cell loss of 10^{-6}, the switch using the SOQ-8 scheme will require 192 cell buffers for each port, whereas the number reaches 320, 400 and 600 cell spaces in the case of SOQ-4, SOQ-2 and DOQ schemes, respectively. It is clear that a switch using the SOQ schemes requires much smaller buffering resources than the DOQ scheme.

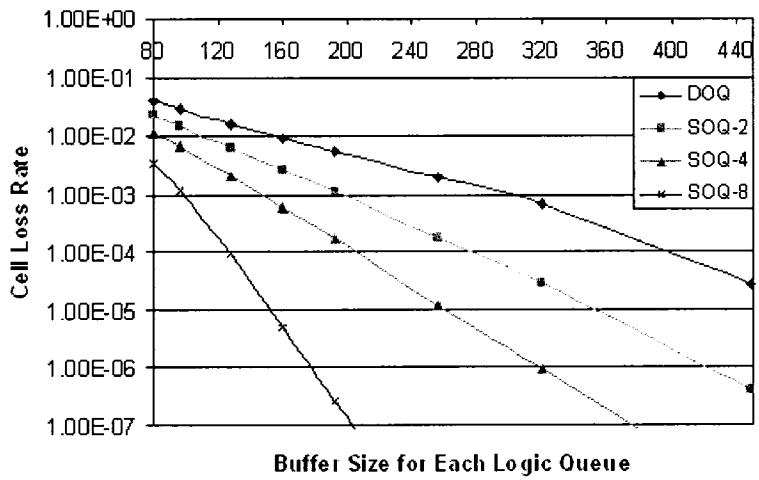

Figure 4.7: Cell Loss Rate vs. Buffer Size ($\lambda = 0.85$)

From the above analysis, it is clear that when a more realistic traffic model, for example, bursty traffic, is used, all SOQ schemes demonstrate better performance than the DOQ scheme. Using the same assumptions, for the 128×128 switch to achieve a better than 10^{-6} cell loss rate, the total required memory for the SOQ-8 scheme is approximately $128 \times 192 \approx 25K$ cells, while the number for the DOQ scheme is around $77K$ cells, which is three times larger.

4.3 Performance under Hot-Spot Traffic

Hot-Spot traffic generally exists in data networks which represents highly demanded applications and services. As more traffic is destined to these hot-spot nodes, more pressure is induced to switching nodes along the path.

4.3.1 Hot-Spot Traffic Model

Assume a single hot-spot exists among all output ports, which will be referred to as *hot port*[37]. Other output ports are considered to be *cold ports*. The *hotport* has a higher chance to be requested by incoming cells. The probability of an incoming cell requesting the *hotport* is

$$p_h = \rho(f_h + \frac{1 - f_h}{N}), \tag{4.4}$$

and the probability of a cell requesting one of the cold outputs is

$$p_c = \rho(\frac{1 - f_h}{N}), \tag{4.5}$$

where f_h is the fraction of the hot traffic. Obviously, the total input traffic load $p_h + (N-1)p_c = \rho$.

4.3.2 Performance Evaluation

Assume the fraction of hot traffic $f_h = 1\%$, our simulation results show some interesting phenomenon. For the DOQ scheme, the traffic filled up the buffer of hot port quickly. As expected, high cell loss rate is experienced. Because the buffer is always full, the average number of cells in the hot port is equal to its buffer size. On the other hand, the cold ports operate in normal conditions, since the hot traffic does not influence them. Hence, the cell loss rate and average number of cells in the queue almost remain the same as what have been obtained in the previous section.

For the SOQ schemes, any shared queue which does not include the hot port has similar performance as in the previous section. So, we will focus on the hot port and those cold ports within the same shared group.

Figure 4.8 shows cell loss rate for the hot port and the cold ports within the shared group (with suffix G), which are compared with the non-grouped cold ports (with suffix c) for the DOQ and SOQ. Interestingly, hot port cell loss rate is very high for all schemes (DOQ and SOQs). This is because, for the 128×128 switch, with 1% hot traffic, the hot port traffic load becomes (76.8% to 115.2%). Such overflow makes the queue become unstable. Under this situation, no matter how big the buffer is, the hot traffic always saturate the buffer space which leads to extremely high cell loss. Consequently, those cold ports within the same shared buffer space experience higher cell loss as well.

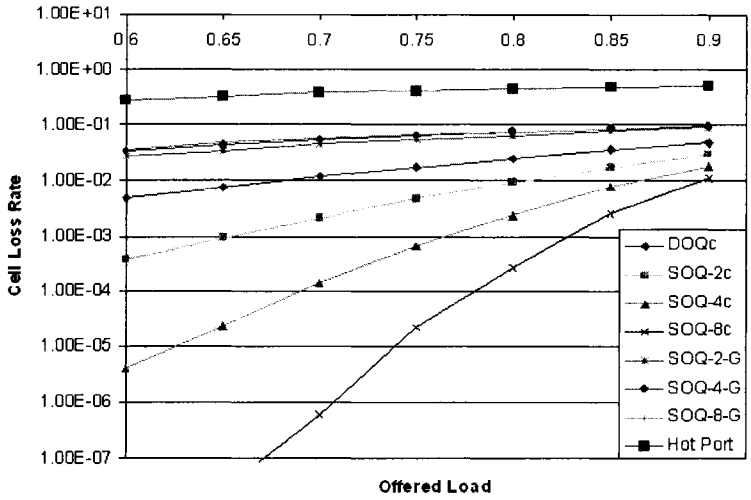

Figure 4.8: Hot Port and Its Shared Group

Figure 4.9 shows average number of cells performance. The similar observation obtained: the hot traffic cells always saturate the available buffer space, so the average number of cells for hot port is equal to the total buffer size. For those cold ports sharing the same queuing buffers, they only take a small portion of buffer space, which show much smaller average number performance compare with those cold ports.

With our proposed reconfigurable SOQ scheme, the buffer sharing parameters, such as sharing level thresholds, can be adjusted in runtime. In this section we show the performance of the fully shared queue scheme. To isolate the hot-spot traffic from those cold traffic, user can configure the shared queue into the dedicated output queue, which will reduce the hot-spot traffic influence to those *cold ports* within the same shared group.

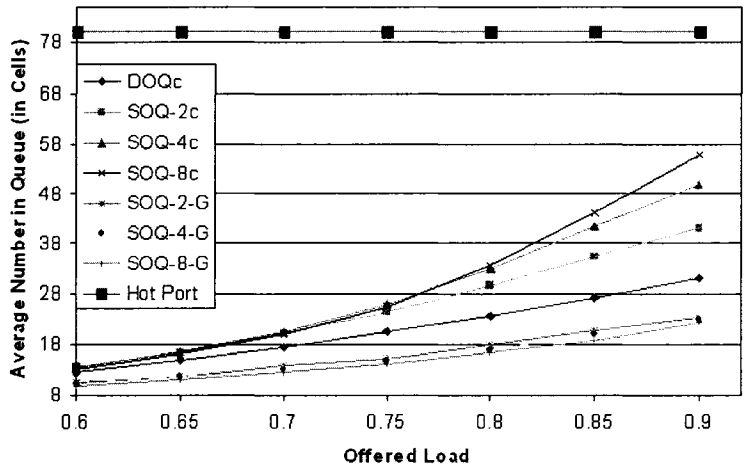

Figure 4.9: Average Number of Cells in Queue under 1% Hot Spot Traffic

4.4 Performance under Prioritized Traffic

Network traffic has various characteristics, some may require better transmission quality, such as network control messages and real-time applications. So the prioritized traffic is likely to be considered in real data networks. This section studies the performance of the proposed SOQ scheme under prioritized traffic.

We assume that incoming traffic has three priority classes [3]. *Class* 0 is the highest priority which may contain the network control information, and *Class* 2 is the lowest, which represent the best effort data traffic. In our simulation, cell priority is considered only during traffic congestion, that is, when cell drop needs to be considered. In that case, cell with the lowest priority will be dropped first, and then move to cell with higher priorities if the available room is still not sufficient to accommodate all incoming cells.

In the simulation, the generated traffic is a mixture of 10% *Class* 0 traffic,

20% *Class* 1 traffic, and 70% *Class* 2 traffic. The average number of cells in queue is very little affected by priority classes, because newly arrived cells are always enqueued at the tail of buffers which do not influenced by their priorty classes. So we focus on the cell loss rate performance for different buffer sizes under 85% traffic load.

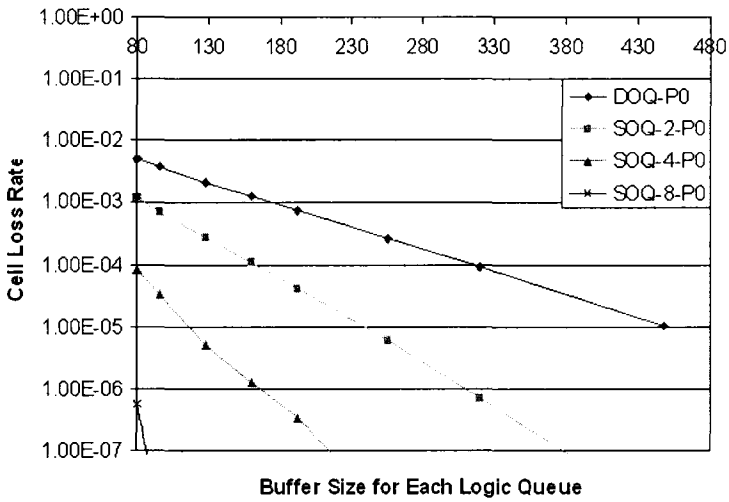

Figure 4.10: Cell Loss Rate for Priority Class 0 ($\lambda = 0.85$)

Figure 4.10 to Figure 4.12 compare the cell loss rate for traffic belong to the three priority classes using various sharing schemes. As expected, better performance can be achieved for traffic associated with higher priority, for example, with 80 cell space allocated in SOQ-4 scheme, cell loss performance of 8.3×10^{-5}, 1.4×10^{-3},1.6×10^{-2} can be achieved for *Class* 0, *Class* 1 and *Class* 2, respectively. Furthermore, better performance can be achieved for schemes with more ports sharing.

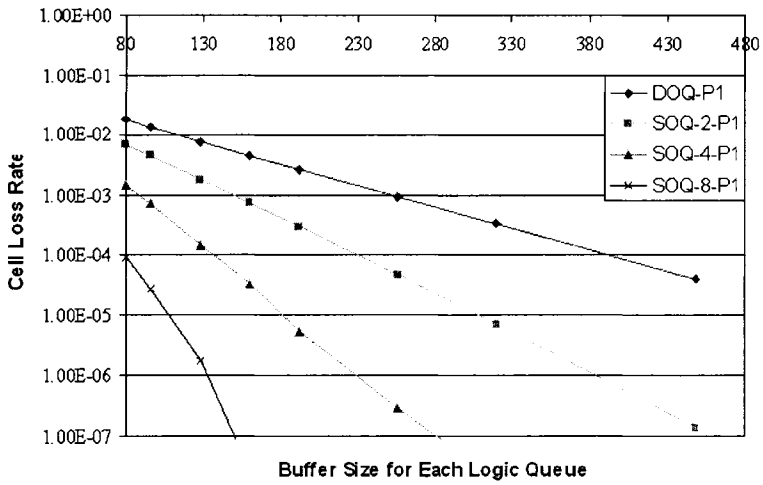

Figure 4.11: Cell Loss Rate for Priority Class 1 ($\lambda = 0.85$)

4.5 Performance under Set Assignment

In our switch model, the queuing buffers are shared by multiple output ports. During the arrival process, a memory address should be provided immediately upon each arrival. However, for a high speed network switch with multiple simultaneous arrivals at each logic queue, such speed requirement introduces much pressure to the implementation of address pointers.

Set assignment means that a bunch of contiguous buffer space will be assigned to a logic port when required, where *Set* is used to denote the bunch of buffering space. The details of the set assignment scheme and its control logic are introduced in the next chapter. Because the set assignment scheme only influences the performance for the shared queuing schemes, the delay and cell loss performance are studied only for the SOQ schemes with different set sizes.

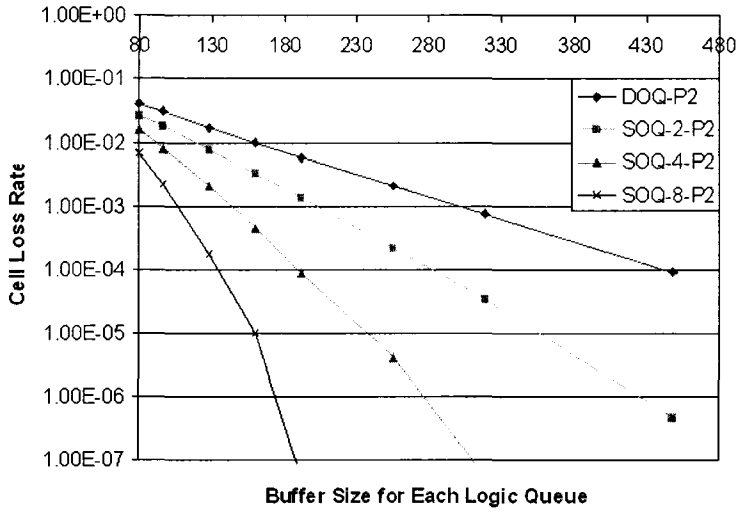

Figure 4.12: Cell Loss Rate for Priority Class 2 ($\lambda = 0.85$)

To distinguish the set sizes with different sharing schemes, we use $2Share$ to indicate SOQ-2 scheme and use the suffix to represent the set size (4, 8, 16). Therefore, the $2Share - 1$ scheme in the figure indicates two ports sharing with the set size of one, which is the same results of SOQ-2. Similar notations are used for SOQ-4 and SOQ-8 schemes.

4.5.1 Average Delay

Figure 4.13 to 4.15 show that for different set sizes, the numbers of cells in queue are almost the same, especially under small set sizes such as 4 and 8, under various traffic loads. However, when the set size is 16 and the traffic load is heavy, (90%), the delay performances are reduced by about 5.3%, 6% and 6.5% for SOQ-2, SOQ-4 and SOQ-8 schemes, respectively. The main reason of this deduction is because of

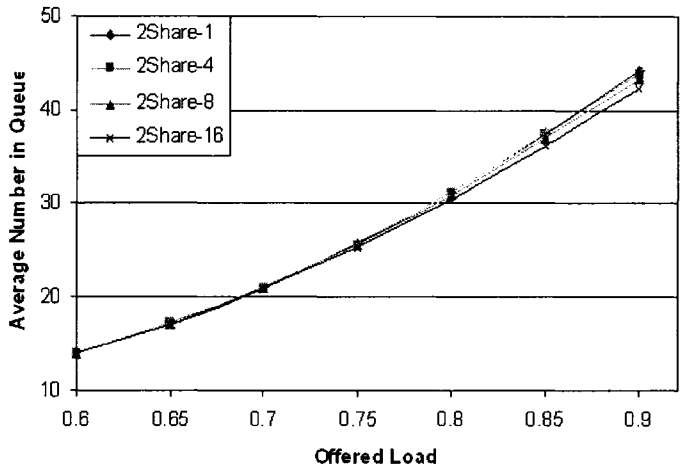

Figure 4.13: Average Number of Cells in Queue for Different Set Size - SOQ-2

its increased cell loss rate, where more cells are discarded because buffers are locked with each set, they could not be shared any more.

4.5.2 Cell Loss Rate

Figure 4.16 to 4.18 show that cell loss performance under set assignment degrades as the set size increases. This is because, for the set assignment, when a bunch of buffers is assigned to a specific output port, these buffers are no longer being shared by other ports. Even if only a small portion of the set is occupied, the set is still dedicated to its assigned port. At this moment, if other ports require more buffering space, it is very likely that there is no available set for assigning.

Although the increased buffer granularity in set assignment degrades the shared queue performance, the trade off between hardware complexity and performance has to be considered during the SOQ system implementation. From the previous discus-

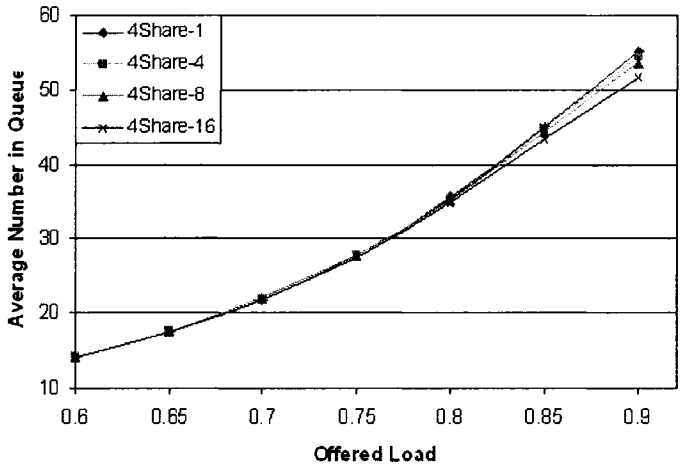

Figure 4.14: Average Number of Cells in Queue for Different Set Size - SOQ-4

sion, it is noticeable that, if the set size is not very large, the performance remains in an acceptable range. Therefore, for our implementation which will be discussed in the next chapter, a set size of 8 is chosen for the system.

4.6 Performance Scalability

In this section, we study how the switch performance scales with switch size and the burstiness of network traffic.

4.6.1 Different Switch Sizes

In this subsection, we study how the switch performance scales with switch sizes. Modern high-speed switches support hundreds of incoming and outgoing ports, the performance scalability is an important aspect for any proposed scheme.

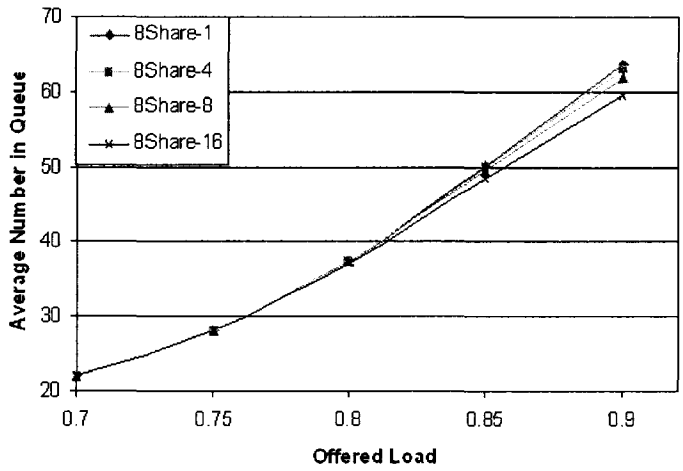

Figure 4.15: Average Number of Cells in Queue for Different Set Size - SOQ-8

Figure 4.19 compares the cell loss rate for the DOQ scheme and the SOQ-8 scheme for various switch sizes under bursty traffic with an average burst length of 10. The impact of switch size on the loss performance is minor, particularly under heavy load conditions. This implies that the loss performance scales well with switch sizes. Similar observation is obtained for average number of cells in queue performance as shown in Figure 4.20.

4.6.2 The Impact of Traffic Burstiness

In this subsection, we examine how traffic burstiness affects the performance of the proposed schemes. A 128 × 128 switch fabric with 80 cell buffers per port is used for the experiments.

Table 4.3 to 4.5 list the cell loss rate for average burst length of 5, 10 and 15, respectively. It is clear that, the cell loss rate performance for the *DOQ* scheme degraded

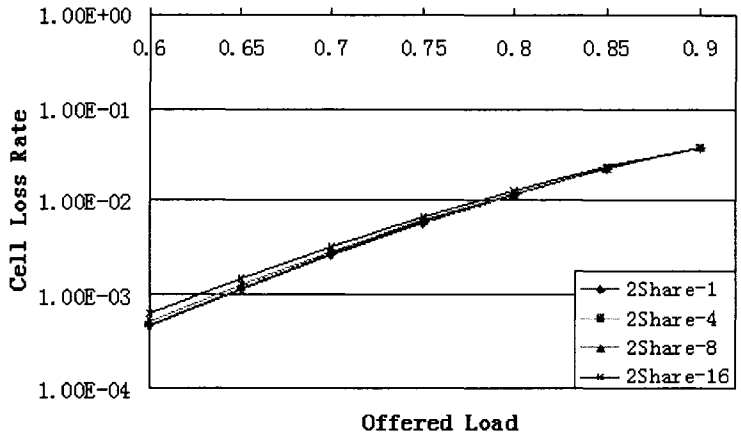

Figure 4.16: Cell Loss Rate for Different Set Size - SOQ-2

dramatically as traffic burstiness increases, whereas the SOQ-8 scheme manages to maintain its cell loss rate performance reasonably well. For example, with an average burst length of 10 and 15, the SOQ-8 scheme can still achieve 10^{-6} cell loss rate with 75% and 65% offered load, respectively. However, for the DOQ scheme, at 60% offered load it has 1.07×10^{-4} cell loss rate, even with only a burst length of 5. When the burst length increases to 10 and 15, the cell loss rate droped quickly to 3.79×10^{-3} and 1.63×10^{-2}, respectively. In general, traffic burstiness has significant impact on the switch performance.

4.7 Summary

In this chapter, the performance of the proposed SOQ scheme has been studied under bursty traffic. Various non-uniform bursty traffic patterns, such as, hot-spot traffic, prioritized traffic, and a shared memory management algorithm - set assignment, are

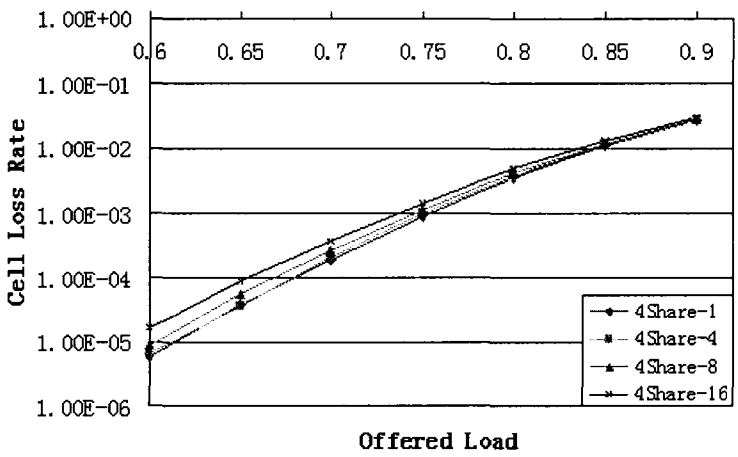

Figure 4.17: Cell Loss Rate for Different Set Size - SOQ-4

applied to the switch model to obtain their performance. Issues related to traffic generation, such as, bursty traffic modeling, hot-spot traffic generation, and prioritized traffic generation, are discussed. Furthermore, the scalability issues of our proposed shared queue are studied with the growing switch sizes and bursty lengths.

We analyzed the requirement of buffer size to achieve the desired cell loss performance. As expected, the SOQ-8 only requires one third of the buffer space of the DOQ scheme, which implies the total memory size has been reduced dramatically. In addition, the longer average delay performance, especially under heavy traffic, which, as we mentioned earlier, is unavoidable queuing delay because more cells are accommodated in the shared buffer. The hot-spot traffic causes higher cell loss rate for those shared ports because hot traffic saturates all cell buffers. The main reason of the saturation, as we analyzed, is the total combined traffic load in the hot port has exceeded 100%. In general, the performance obtained in this chapter clearly shows

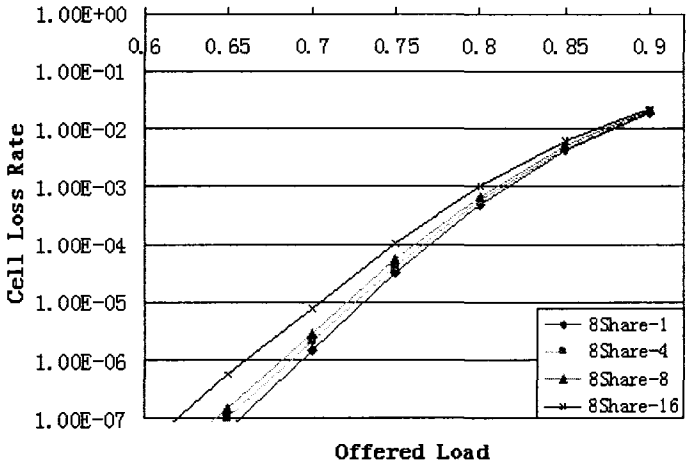

Figure 4.18: Cell Loss Rate for Different Set Size - SOQ-8

that, the shared output queue provides better performance on cell loss rate under bursty traffic and non-uniform traffic conditions.

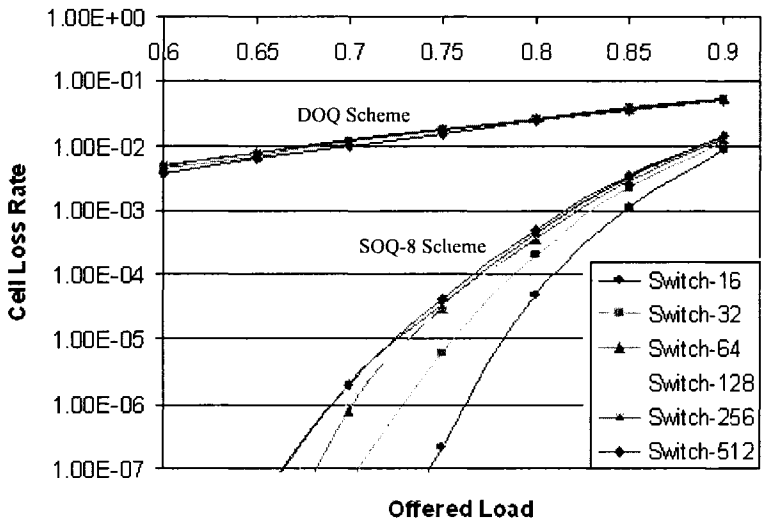

Figure 4.19: CLR for DOQ and SOQ-8 vs. Offered Load

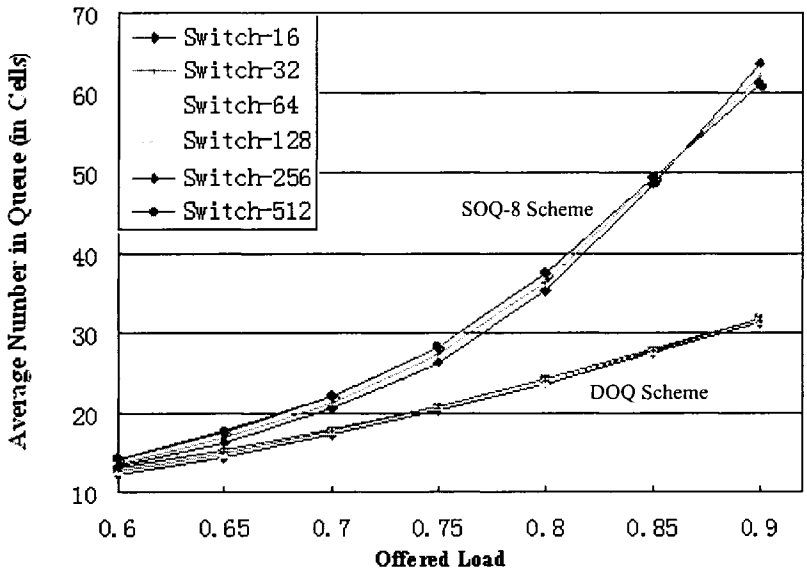

Figure 4.20: Average Number of Cells in Queue vs. Offered Load

Table 4.3: Cell Loss Rate for Burst Length of 5

Offered Load	Cell Loss Rate(B5)			
	DOQ	SOQ2	SOQ4	SOQ8
0.6	1.08E-4	1.54E-7	-	-
0.65	2.90E-4	2.77E-6	-	-
0.7	7.08E-4	1.67E-5	-	-
0.75	1.67E-3	9.09-5	8.71E-7	-
0.8	3.72E-3	4.75E-4	2.60E-5	-
0.85	6.26E-3	1.34E-4	1.85E-4	4.24E-7
0.9	6.25E-3	1.33E-4	1.96E-4	4.71E-7

Table 4.4: Cell Loss Rate for Burst Length of 10

Offered Load	Cell Loss Rate(B10)			
	DOQ	SOQ2	SOQ4	SOQ8
0.6	3.79E-3	2.45E-4	-	-
0.65	6.25E-3	7.10E-4	4.69E-6	-
0.7	9.95E-3	1.66E-3	4.69E-5	-
0.75	1.54E-2	4.28E-3	3.26E-4	2.17E-7
0.8	2.35E-2	9.33E-3	1.58E-3	4.94E-5
0.85	3.44E-2	1.89E-2	5.89E-3	1.13E-3
0.9	4.92E-2	3.50E-2	1.73E-2	8.61E-3

Table 4.5: Cell Loss Rate for Burst Length of 15

Offered Load	Cell Loss Rate(B15)			
	DOQ	SOQ2	SOQ4	SOQ8
0.6	1.63E-2	3.83E-3	2.88E-4	2.30E-6
0.65	2.27E-2	7.15E-3	9.01E-4	3.14E-5
0.7	3.09E-2	1.25E-2	2.61E-3	2.13E-4
0.75	4.14E-2	2.08E-2	6.63E-3	1.28E-3
0.8	5.41E-2	3.31E-2	1.41E-2	5.36E-3
0.85	6.95E-2	5.00E-2	2.65E-2	1.56E-2
0.9	8.78E-2	7.20E-2	4.45E-2	3.34E-2

Chapter 5

Shared Output Queue System Design and Implementation

Chapter 2 provides a brief introduction to the proposed Shared Output Queue architecture, and this chapter presents the detailed system designs and implementations in various hierarchical levels.

5.1 System Level Design

The proposed Shared Output Queue system mainly contains four subsystems: tail buffers, head buffers, main controller, and memory banks. The tail buffers are responsible to accept arrived cells and store them in their buffers temporarily. The head buffers store the outgoing cells for transmitting. The main controller controls all these processes, and provides commands for main memory writing, reading or cut-through transmissions according to current conditions.

Figure 5.1 shows a 4-port sharing system architecture and internal data trans-

mitting processes on the data bus. This is the basic SOQ system which will be implemented.

During every main memory writing and reading, a block of b (b is block size) cells is transferred on the data bus. Considering the value of b could be quite large, a wider internal data bus is necessary, and it has to be shared by all subsystems as the figure shows. As a result, a tri-state bus driver is implemented for each tail and head buffer. However, the main memory has its own chip-select (cs) signal, so it is not necessary to implement tri-state driver for main memory.

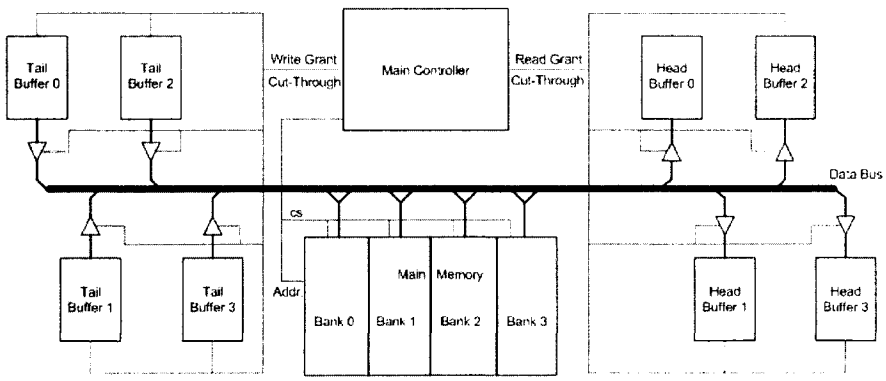

Figure 5.1: Data Bus Architecture

In the system implementation, data cell arrives sequentially with one bit at a time from the switch fabric. Also, the transmitting process at the head buffer is a sequential process with one bit departing at every clock cycle.

Next, we examine each subsystem in more detail about its functional descriptions, its architecture design and functional simulation results. In real implementation, the main memory bank is massive and is usually handled differently, either by using stan-

dard commercial parts, or synthesized by a different tool, e.g., memory compiler. So the implementation of the main memory bank will not be included in our discussion. Instead, we use data files to emulate memory banks during our simulation.

5.2 Tail and Head Buffer

As described in chapter 2, DRAMs provide better storage capacity because a DRAM cell is usually implemented using a single transistor with a capacitor. However, the access time to DRAM is relatively long. On the other hand, the SRAMs are faster, but they consume larger area and power. Therefore, only a small amount of SRAMs will be used in the proposed shared queue architecture.

5.2.1 Memory Size

In our design, a small amount of SRAMs are used for Tail Buffer and Head Buffer implementations, whereas massive DRAMs are used for main memory bank. In order to match the speed between faster links and slower main memory bank, wider data bus and memory bank interleaving have been used to achieve better data transmission parallelism.

To determine the sizes of Tail and Head buffers, some important factors should be considered. Because the DRAM access time is the most critical factor, we define a time interval T for the DRAMs to perform one memory access, either for a writing or a reading operation. Given the link rate R, the maximum number of bits coming in from a link is $2RT$. Considering that up to k cells can be accepted by each port, and the number of shared ports is M, the data block between tail/head buffer and

main memory is $2kMRT$ bits, which is referred to as b cells.

Because the successive transmissions from the same tail/head buffer will only be performed in M intervals in the worst-case scenario, a minimum of $2b - 1$ cells for the tail buffer should be sufficient. Similarly, for the head buffer, in the worst-case scenario, it should have a size of $(b - 1) + 2M$ cells. This kind of arrangement will ensure that the tail/head buffer satisfies the block transfer rules which defined in *Chapter* 2, and meanwhile, the buffer sizes are minimum.

For example, if we chosen a $50ns$ access time DRAMs for the main memory, then $T = 50ns$. If the link rate $R = 10Gbps$, $k = 4$, and $M = 4$, then the block size is 16000 bits. If a data cell is equal to 64 bytes, then the block size is equal to 32 cells. For the link rate of $5Gbps$, the block size is 16 cells. In our implementation, the block size will be chosen as 16 cells, however, the tail buffer and head buffer sizes will be calculated using 32 cells for possible future expansion.

Because the successive transmissions from the same tail/head buffer will only be performed in M time intervals in the worst-case scenario, a minimum of $2 \times b - 1$ bits for the tail buffer size should be sufficient, where in our example it is 63 cells. Similarly for the head buffer, in the worst case scenario, it should have a size of $(b - 1) + 2M$ cells, where it equals to 39 cell spaces in our example. In addition, the cut-through transmissions between tail buffers and head buffers can be any number of cells, but the maximum value is equal to b cells to avoid head buffer overflow.

The size of the main memory is engineered by many factors, such as the Quality of Service (QoS) requirements. With different data traffic requirements, the main memory should be expandable. In our implementation, a buffer of 256 cells are equipped for each port as in the DOQ scheme, which yields a total of 1024 cells for

the main memory bank in the case of the SOQ-4 scheme.

Inside each tail and head buffer, cells are arranged in the First-Come-First-Served (FCFS) fashion and a simple control algorithm will be implemented. An internal counter is used to track the number of cells in the buffer, and when the number reaches the block size, a write transmission request is sent to the Main Controller. Similarly, when a head buffer has space for a block of data cells, a read transmission request is sent out.

5.2.2 Tail Buffer Implementation

The tail buffer is responsible to receive arrival cells from the switch fabric, keep track of the number of cells in its buffer, send transfer requests to the main controller, and manage the writings and readings from its buffer memory. So, the tail buffer can be constructed by these functional blocks which are organized into two groups: the Datapath, which includes most combinational circuits; and all control functionalities inside the tail buffer, which is usually implemented using Finite State Machine.

5.2.2.1 Tail Buffer Datapath

The tail buffer datapath includes Receive Registers, a Packet Counter, and Address Pointers to provide addresses during the tail buffer write and read operations.

1. Receive Register

Although in previous simulation study, the switch is assumed to be an ideal non-blocking, it is not practical for implementation. A knockout factor of 4 is commonly used, such as in [38] and [39]. This is because under uniform random traffic, the arrival probability in an output queue is modeled by binomial distribution. The

probability of having more than 4 cells destined to a specific queue is very low, for example, under 70% random traffic, this probability is about 0.00074. Under such circumstances, only the exceeded cell would be dropped. Therefore, the cell loss rate caused by knockout factor would be considerably small. For this reason, in our design, we choose the knockout factor $k = 4$ for implementation.

Two methods can be used to fulfill such requirement. One is to use faster memory to construct the receiving buffer, e.g., the buffer could operate k times faster than the regular link rate. However, this is not practical for modern switches which are already running at very high speed. The other method is to implement k receiving buffers in parallel with each one operates at the same speed as a regular link. In our implementation, the second approach is used, which is shown in Figure 5.2. Arriving data cells will be immediately written to tail buffer for storage.

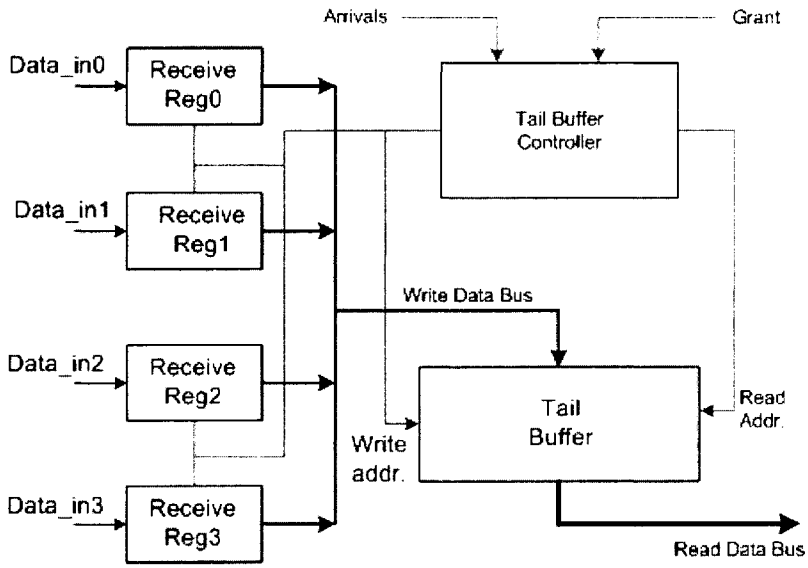

Figure 5.2: Tail Buffer Architecture

The data cells arrive in each register sequentially, but writing into the tail buffer is performed on a wider internal data bus. In the worst case scenario, k cells are written in round robin fashion. The register is 8-bit wide and the data bus is 4-bit wide, so 2 bus transmissions are needed to finish one cell writing. During the busy period, a new receiving process is overlapped with the previous transmission process to achieve higher system efficiency.

2. Cell Counter

An internal cell counter keeps tracking of the number of cells in the tail buffer is employed. When the number exceeds the block size, a write request will be issued to the main controller. During data transferring from the tail buffer, either the operation is main memory write or cut-through operation, the cell counter decreases by one upon each cell reading. When the cell counter reaches the block size or equals to zero, then the *data valid* signal will be de-asserted, which indicates the end of current data transfer operation. Two flag signals are also provided by the counter: tail buffer empty or full, and write transmission request.

3. Address Pointers

Because of the parallel operations of writings and readings, two pointers are implemented in the tail buffer which provide the write address and the read address. In the write address unit, double buffering has been used because parallel and overlapped operations for receiving and writing processes, which is shown in the tail buffer FSM design below.

5.2.2.2 Finite State Machine(FSM)

A reading operation from the tail buffer is granted by the main controller, while writing to the tail buffer should never be blocked for data receiving process. This means that the tail buffer should be capable of writing and reading at the same time. Currently, QDR SRAM [19] could meet these requirements because it has two separate data buses and two address buffers for writing and reading operations. An arbiter is required to ensure no address conflict occurring. Because reading and writing operations can be conducted simultaneously, two independent Finite State Machines (FSMs) are designed to control the tail buffer for reading and writing operations.

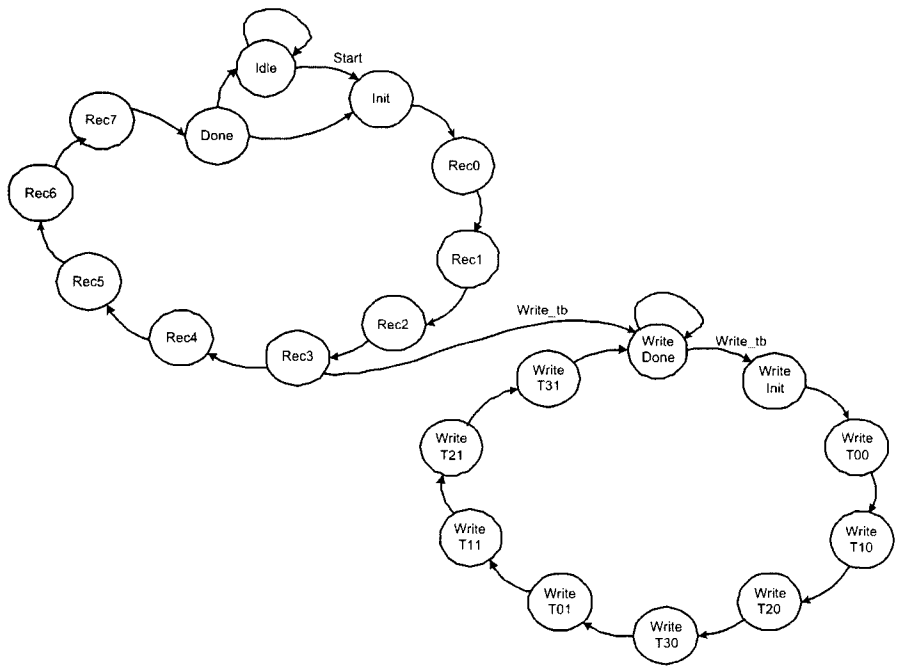

Figure 5.3: Tail Buffer Finite State Machine

The main state diagram of the tail buffer is shown in Figure 5.3, and it is clearly

indicated that the receiving process is overlapped with the tail buffer writing process. The receiving process of one packet is assumed in eight steps, once half packet is buffered in the receiving register, write operation will be started which will write the half packet into tail buffer. The four parallel receiving registers will be read out in round robin fashion as indicated as state $Write\ Tij$, while i is the index of receiving registers, and j indicates the first half or second half of a packet. During the writing of first half of packets from all registers, the rest half of packets continuously arrive at registers. The writing of the second half of packets will probably overlapped with the receiving process of next arrival cycle.

The reading process, which is another independent process, has a separate finite state machine, and it performs read operations under the *grant* or $cut - through$ signal from the main controller.

5.2.2.3 Synthesized Tail Buffer

The designed tail buffer has been synthesized using *Synopsys* (Design Compiler), and the resulting circuit at block level is shown in Figure 5.4, which is comprised of a datapath and two Finite State Machines.

5.2.3 Head Buffer Implementation

The design of the head buffers is similar to the tail buffers, except that they include only one transmitting register. Cells coming from the tail buffer or main memory will be temporarily stored in the head buffer, which will be continuously read out and transmitted to the output link. The Cell Counter keeps track of the number of cells in the buffer. If the available space is larger than the block size, a read request will be

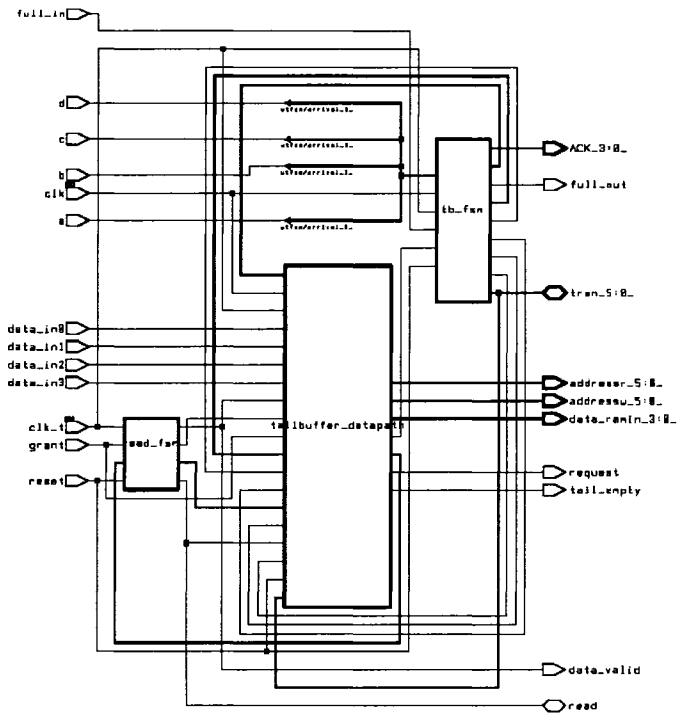

Figure 5.4: Synthesized Tail Buffer Circuit

issued to the main controller. Similar to the tail buffer, the head buffer can perform writing and reading operations simultaneously.

The synthesized head buffer circuit is shown in Figure 5.5 which includes four main blocks: Cell Counter, Address Pointer, Transmitting Register and Head Buffer FSM.

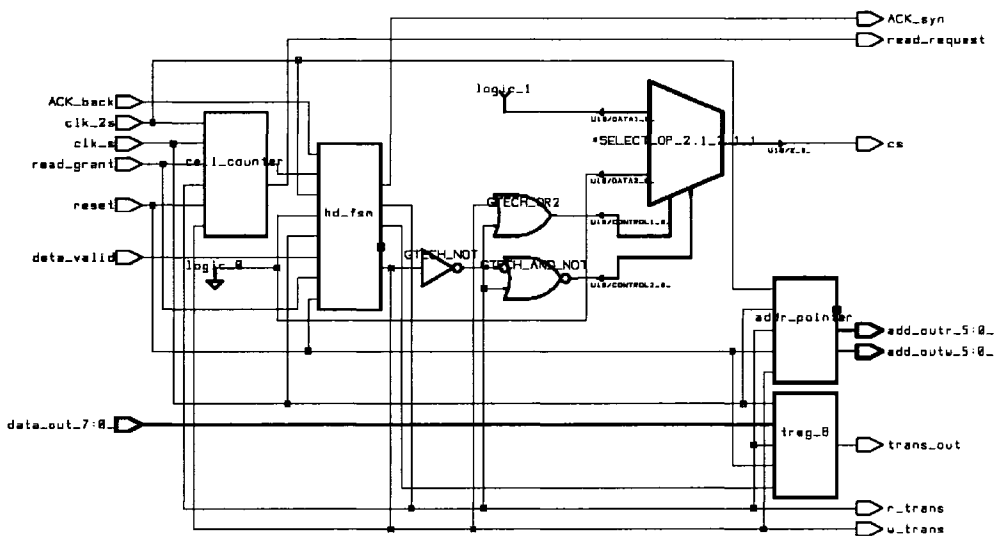

Figure 5.5: Synthesized Head Buffer Architecture

5.3 Reconfigurable Main Controller

Being the most critical unit in the system, the main controller performs the following tasks: i) processing requests from tail and head buffers, ii) managing the main memory operations, iii) manipulating shared queuing operations, and iv) performing reconfigurations functionalities. In order to accomplish these tasks, the controller is partitioned into several functional units, and these units work together under the control of a Main Finite State Machine.

Four major functional units are designed, which cooperate to perform the reconfigurable buffer sharing functionality. They are:

- Request Arbitration Unit(RAU);
- Pointer Management Unit(PMU);

- Queue Control Unit(QCU);
- Special Function Unit(SFU).

As an example, Figure 5.6 shows a block write operation. In the figure, the first write request in the RAU is *Port* 0 and the corresponding address in the PMU is 9. Hence a block of cells are transfered from the tail-buffer of *Port* 0 to the central memory and stored at the location of 9. Then the request register will be updated and the next write request from *Port* 3 will be processed. The block read operation is executed in a similar manner.

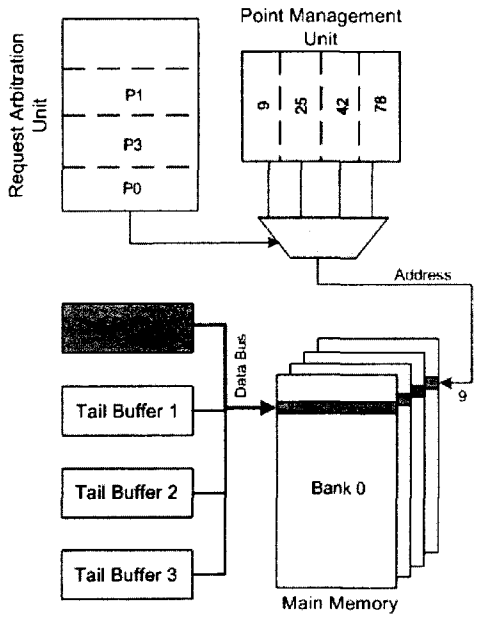

Figure 5.6: Write Operation

5.3.1 Main Controller Datapath Design

This section introduces the detailed functionalities, architecture designs and implementations of each functional unit inside the SOQ main controller.

5.3.1.1 Request Arbitration Unit(RAU)

Inside the RAU, two sets of shift registers are used to track writing and reading requests from each tail/head buffer. They are named as Write Request Register (WRR) and Read Request Register (RRR). The main control process checks the write requests and read requests alternatively, and a *grant* signal is issued upon the positive response from the request registers. The grant signal includes *write grant* corresponding to WRR, *read grant* and *cut throughgrant* corresponding to RRR. The pseudo code of the main control algorithm has been given in *Chapter* 2.

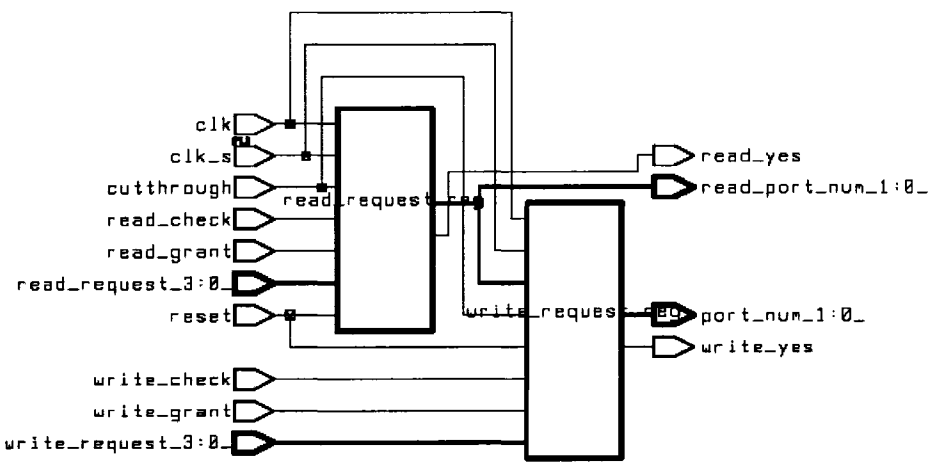

Figure 5.7: Synthesized Request Arbitration Unit

The functional simulation will be presented later in the entity testing subsection, and the synthesized result of the request Arbitration Unit is shown in Figure 5.7.

5.3.1.2 Pointer Management Unit (PMU)

The PMU is responsible to provide the address for main memory write and read. The addressing scheme is decided by the main memory organization. In our design, four memory banks are used to construct the main memory and each memory bank is comprised of several memory blocks. To differentiate from cell blocks, we use the term *set* to describe a block of addresses to accommodate blocks of cells. As a result, there are three addresses for the memory access: *Set Address*, *Pointer address*, and *Burst Address* as shown in Figure 5.8. Because the granularity of each memory operation will be in a block of cells, both *set address* and *pointer address* are remained the same during each operation.

It is very common that burst read or burst write will be required for DRAM access. A burst length of $2, 4$ or 8 can be used in a burst operation. In our implementation, the burst length of 4 is chosen which means a 2-bit burst address is required. We choose the block size $b = 16$ cells, which means the main memory of 1024 cells is organized into 64 blocks with each set containing 8 blocks of cells, and a total of 8 sets are available. Therefore, we need 3 bits for set address, 3 bits for pointer address and 2 bits for burst address.

The implementation of the PMU contains three components: address pointer, set list, and set pool. One address pointer and one set list are dedicated to each link, so, in total 4 address pointers and 4 set lists are implemented. There is only one set pool which stores all available sets. Two operations are involved between the set pool

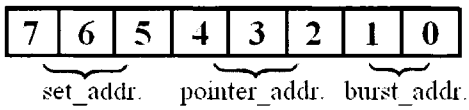

Figure 5.8: DRAM Addressing Scheme

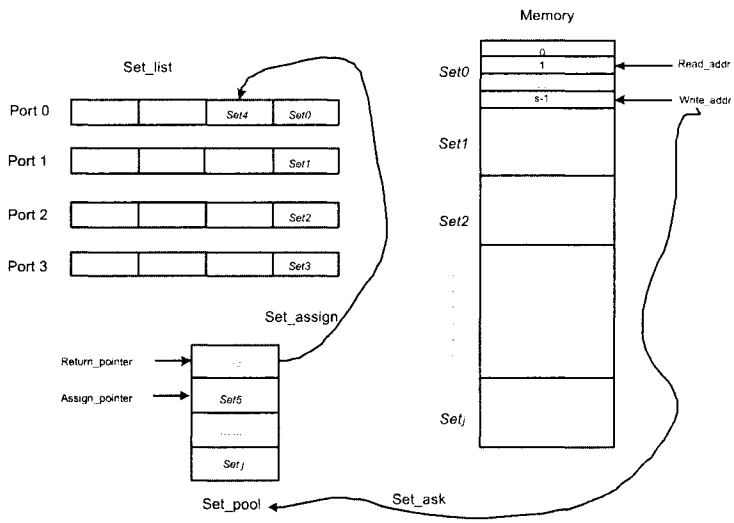

Figure 5.9: Memory Management - Set Assignment

and each logic link. Firstly, when a write address pointer indicates that current set is full, it requires for assigning a new set as shown in Figure 5.9. If there is an available set in the set pool, then one set is assigned to the required link. Secondly, when a read address pointer reaches the set boundary which indicates the set is empty, a *returnset* signal is issued to both its set list and the set pool. Upon a set return operation, the corresponding set list drops the set, and the set pool adds it into the pool of available sets.

Figure 5.10 shows the synthesized result of the Pointer Management Unit. But the burst address component is not included because it is allocated more closely to

the main memory banks as a separate unit.

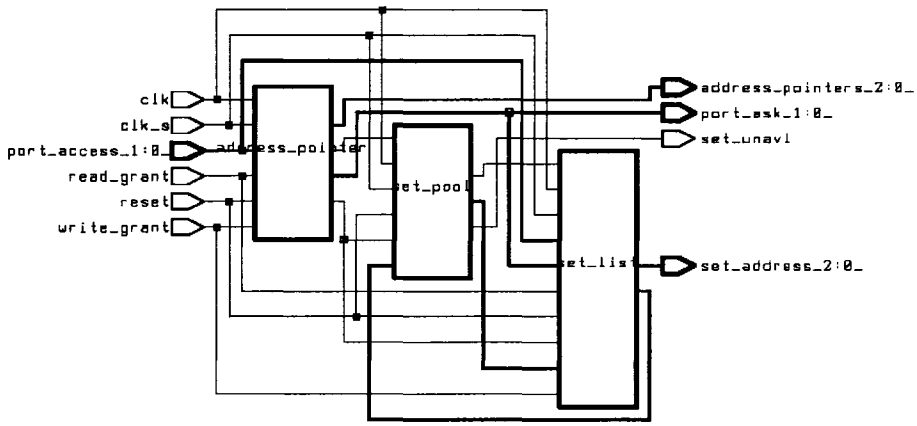

Figure 5.10: Synthesized Address Pointer Management Unit

5.3.1.3 Queue Control Unit(QCU) and Special Function Unit(SFU)

The functional components inside the Queue Control Unit include counter based queue length registers for each port, a total queue length register, and a queue full/empty flag register. The queue length register is used to report the dynamic queue status, with which the main memory controller can make corresponding operation decisions.

The Special Function Unit is a unique design in the proposed SOQ scheme. It stores the configuration parameters and various parameters for different traffic patterns, such as the thresholds of queue length for each port. The value for the parameters can be reconfigured during the runtime based on the information and commands given by the network processor at the port controller. In this project, we only focus

on the basic infrastructure for runtime reconfiguration. The SFU unit is implemented in the datapath together with the QCU.

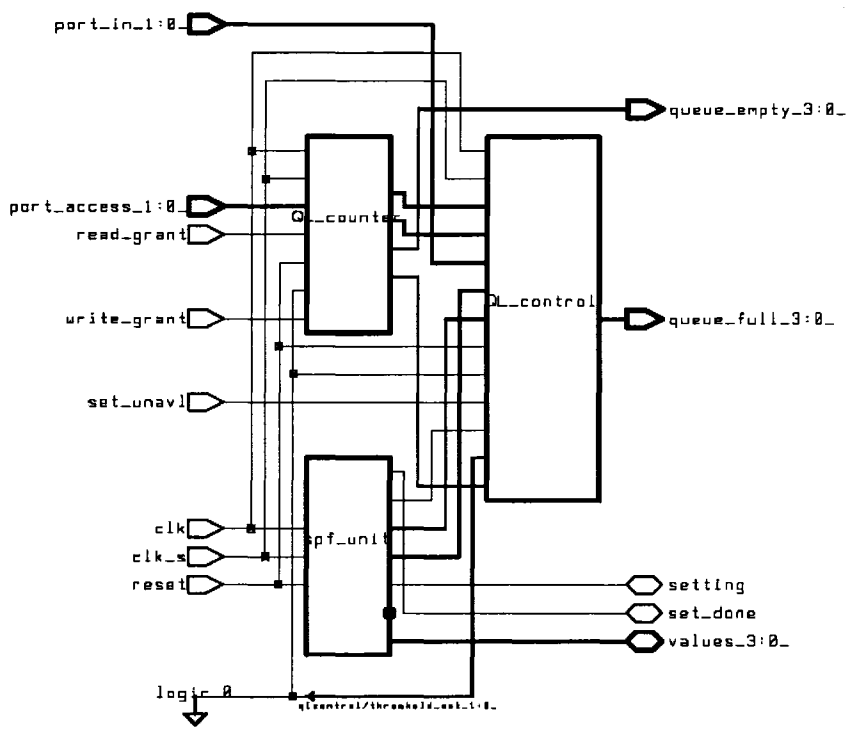

Figure 5.11: Synthesized Queue Length Controll Datapath

The synthesized QCU and SFU datapath is shown in Figure 5.11 and its functional simulation is shown in the main write and read operation figures below.

After all these functional units perform their designed functionalities correctly, they are integrated together as one subsystem which is called the main datapath, and will be used for higher level synthesis.

5.3.2 Main Finite State Machine

The state transition diagram of the main controller is shown in Figure 5.12. When the system is powered on, the DRAMs will be first initialized. After it is done, the system enters the initial state which provides ready signals to the switch fabric. In its normal operation mode, the controller alternatively checks the write request and read request, and conduct corresponding processing according to different system states, traffic conditions, and the configurable parameter settings. Cut-through operation, in which the user data do not need to go through the main memory, is considered and handled using a separate state. When the corresponding operation has been issued, the system will enter the *wait state* in which the *data_valid* signal is checked to terminate current transmissions, and make the controller move on to the next checking state.

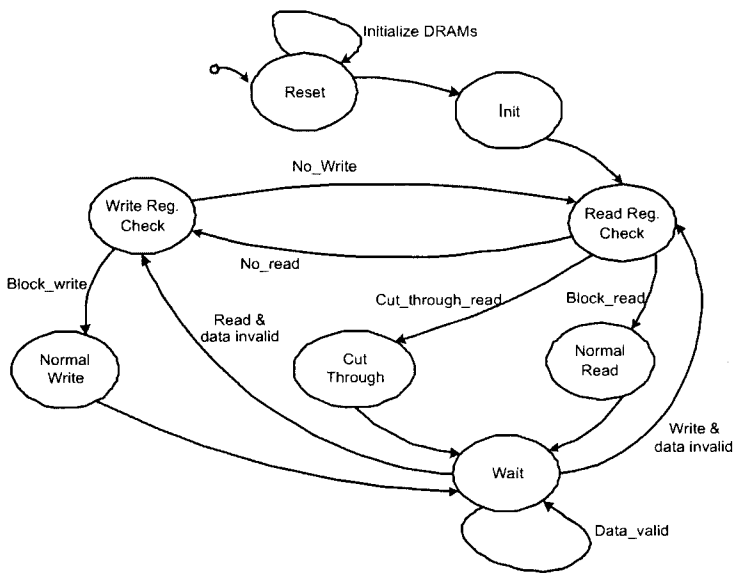

Figure 5.12: State Diagram of Main-FSM

5.3.3 Synthesized Main Controller

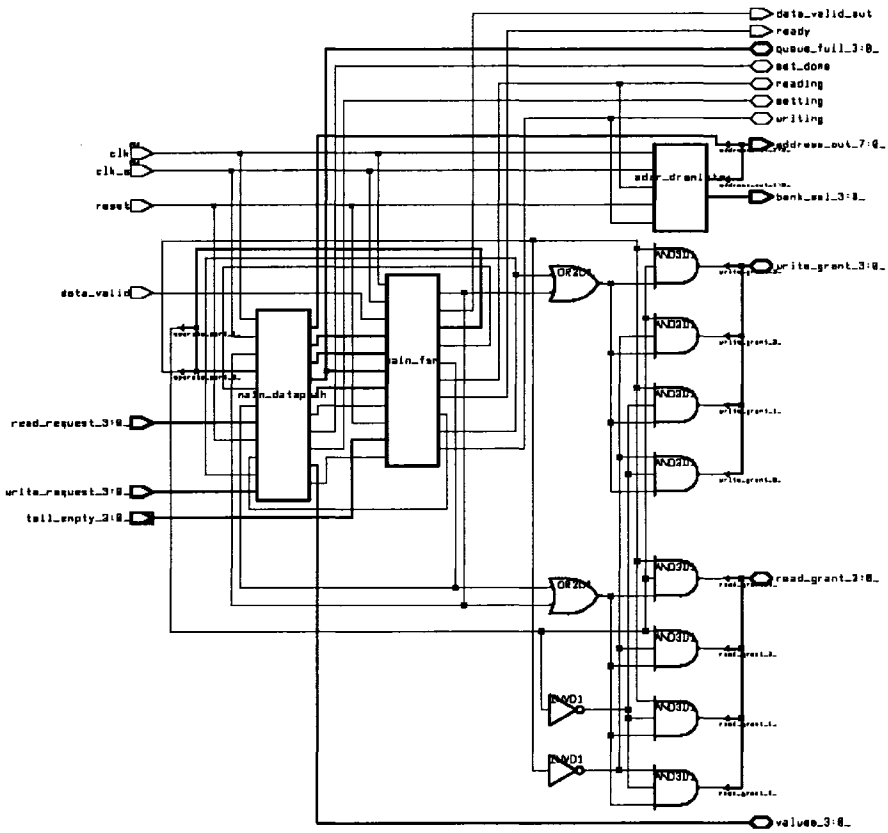

Figure 5.13: Synthesized Main Controller

Figure 5.13 shows the synthesized top level main controller circuit, which includes the main datapath, the main FSM and the DRAM address unit. The main datapath contains four functional units as we mentioned in the previous section: RAU, AMU, QCU and SFU.

5.4 System Implementation

A top-down design and bottom-up implementation approach is followed in our research. A divide-and-conquer strategy is used in our design until all the leaf components become manageable. As shown in Figure 5.14, after three levels of partitioning, the implementation hierarchy of the SOQ system is obtained.

In our design and verification, all components are properly designed and tested before the 4-port shared output queue is constructed. DRAMs and SRAMs are considered for system design and functional simulations, however, they are not included in the system final synthesis because different CAD tools are usually used for memory design, which is beyond the scope of our research.

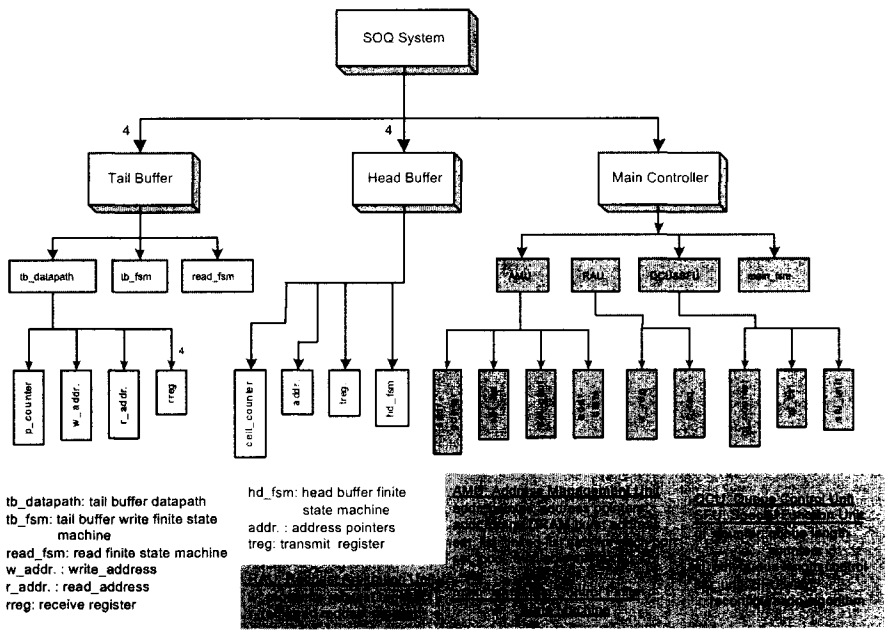

Figure 5.14: Hierarchical SOQ System Architecture

The functional simulations are conducted on the *ModelSim XE III* simulation platform. The test vectors are generated using software programs and recorded in four data files which will feed to four input ports. Bursty traffic is considered in the hardware simulation. When testing, for each iteration the *ModelSim* simulator reads in a test vector to the receiving register according to the generated cell arrival status information. Similarly, on the output side, all transmitted cells are recorded into a result data file. When the simulation is completed, all the data files are sent to another software program for final verification. In our experiment, after about 8000 simulation rounds, around 5800 cells per link are transmitted over the output link. As expected, no cell loss has occurred as long as they got queued in the buffer.

The whole 4-port sharing system is synthesized using Synopsys Synthesizer DC Shell, with the $0.18um$ CMOS technology. The total area for the whole control logic is $24.79um^2$, which is approximately $20,000$ gates in terms of 2-input NAND gates. The total dynamic power consumption is merely $24.12mW$. The whole design can comfortably operate using the $10ns$ system clock.

5.5 Entity Functional Testing

5.5.1 Tail and Head Buffer Functional Simulation

The functional simulation results of the tail buffer are shown in Figure5.16. Three state transition processes: main receive process (state), write to tail buffer (tstate), and read from tail buffer (rstate) are shown in the waveform. The cell counter counts the number of cells in the tail buffer. When a grant signal *read* is given, which is

shown in the figure for a cut-through operation, six cells have been read from the tail buffer to the data bus, and the cell counter is updated to zero afterward.

Figure 5.17 shows the functional simulation results of the head buffer. On the waveform, the departing cell appears as the *trans_out* signal. Two state transition processes are shown as transmission state(tsstate) and writing to head buffer state(wstate). In this figure, it is shown that a total of 12 cells has been written into the head buffer, and meanwhile, two of them have departed, and ten cells remain in the buffer.

5.5.2 Main Controller Functional Simulation

The functional simulation of the PMU is shown in Figure 5.18, which includes assigning a set to one set list and returning a set to the set pool.

When writing a block of cells to main memory, the controller issues a *write grant* to both the selected tail buffer and the address pointer management unit. Immediately following the grant signal, data streams are put on the data bus, and a write address is also available on the address bus. The memory write operation is interleaved to each individual memory banks with the *bank select* signal which is issued by DRAM address unit, which shows in Figure 5.13 as the separate unit (addr_draminte). It also provides 2-bit burst address to the address bus. The memory read operation is similar, except that the data streams are from main memory to a head buffer.

For a cut-through operation, a *cut through* signal is issued to the selected tail and head buffers. However, no signal is sent to the address pointer management unit to ensure no main memory involved in the process. Data stream is directly

transmitted from the tail buffer to the head buffer, and when the cell counter in the tail buffer reaches zero, the *data valid* signal becomes false and terminates the transmission process. In this way, it is more flexible to control the number of cells for each transmission, while the maximum value is equal to block size to avoid the head buffer overflow.

The functional simulation waveform in Figure 5.19 shows a main memory write operation, with a cut-through process; and Figure 5.20 shows a main memory read operation.

5.5.3 Internal Delay

Because the internal data bus is shared by all links, an additional waiting delay is introduced at the very beginning when the system comes out from the reset state, while the head buffers are empty and wait for cells to be write in through the shared data bus. Once the head buffers are filled up, they would send data cells out serially and require for cell replenishment before their buffers become empty. So the internal waiting delay is observed as pipeline delays which would happen when head buffers are empty.

According to Figure 5.21, the delay time between the completion of receiving at tail buffers to the starting of transmissions at head buffers varies for all links. For the first port ($Port0$), the delay time is about 7 clock cycles, however, for the last port ($Port3$), the delay time is about 28 clock cycles. Although, this delay time affects the overall SOQ performance, as we stated, it is introduced by the hardware implementation of the shared data bus which is considered as one of the hardware

implementation and performance trade-offs.

5.6 Summary

In this chapter, we have studied the Shared Output Queue system design and implementation issues. We have presented the detailed functionalities for each unit. According to its functional specifications, the architectural design has been provided. A top-down design and bottom-up implementation approach has been followed in this study, and divide-and-conquer strategy has been used for system implementation. The hierarchical system design was shown in Figure 5.14, and along with each basic component design, functional simulation and synthesized circuit diagram has been shown. Then at a higher level, a structural subsystem has been derived. Both the functional verification and synthesis results are produced for further integration. Finally, based on all subsystems, the top level SOQ system was constructed. In order to verify its functionalities, we developed a computer program to generate test vectors, which are applied to all system inlets. At the outlets, the transmitted data cells are verified with their input vectors using another program. As indicated in the previous section, the final results is promising: the system is working exactly as we expect.

The hardware implementation of proposed SOQ-4 scheme assumes that a symbolic data cell is 8-*bit* long and incoming and outgoing links are 1-*bit* wide. In this condition, totally 8 cycles will be required to receive or transmit one data cell. In addition, the purposes of the implementation in our research are:

- Verify the proposed shared buffer scheme has better performance;

- The shared memory control algorithms are feasible under the critical speed requirements for modern high-speed switches and routers.

These goals have been accomplished by testing the implemented SOQ-4 with optimized results.

Because the main memory controller has been implemented, SOQ system can provide dynamic adjustments and reconfigurations to the shared memory according to traffic characteristics and commands from the central switch controller or port controller. Compare with traditional FIFO queue, the SOQ-4 has added about $20,000$ gates (in term of NAND gate), which would be $5,000$ gates or $6um^2$ for each port. As a result, the increased control circuit has significantly improved the buffer performance, which is shown in previous chapters. The employment of the advanced hybrid SRAM/DRAM architecture also provides the worst case bandwidth guarantee and optimized memory utilization.

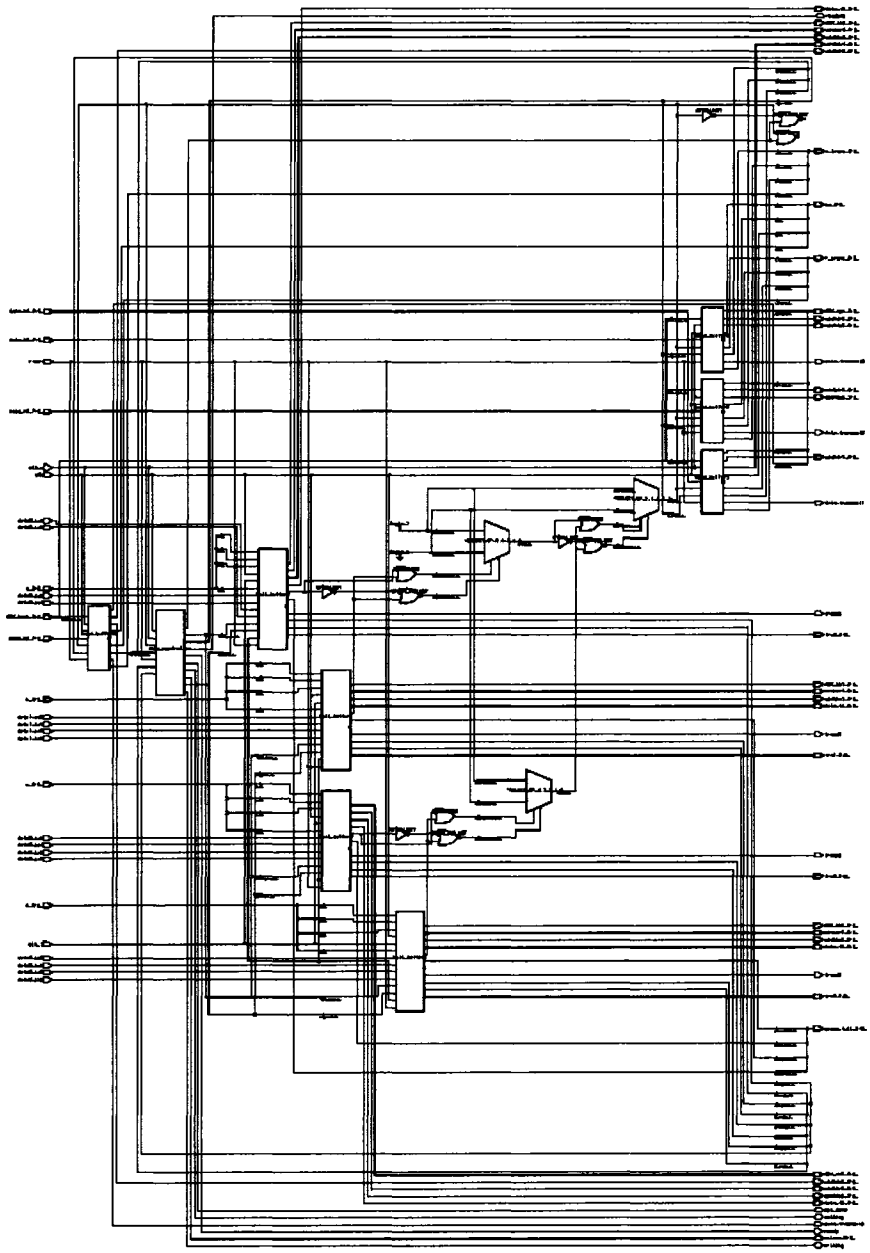

Figure 5.15: Synthesized SOQ System Architecture

Figure 5.16: Tail Buffer Function Simulation

Figure 5.17: Head Buffer Function Simulation

Figure 5.18: Waveform of Set Assign and Return

Figure 5.19: Main Controller Function Simulation - Writing

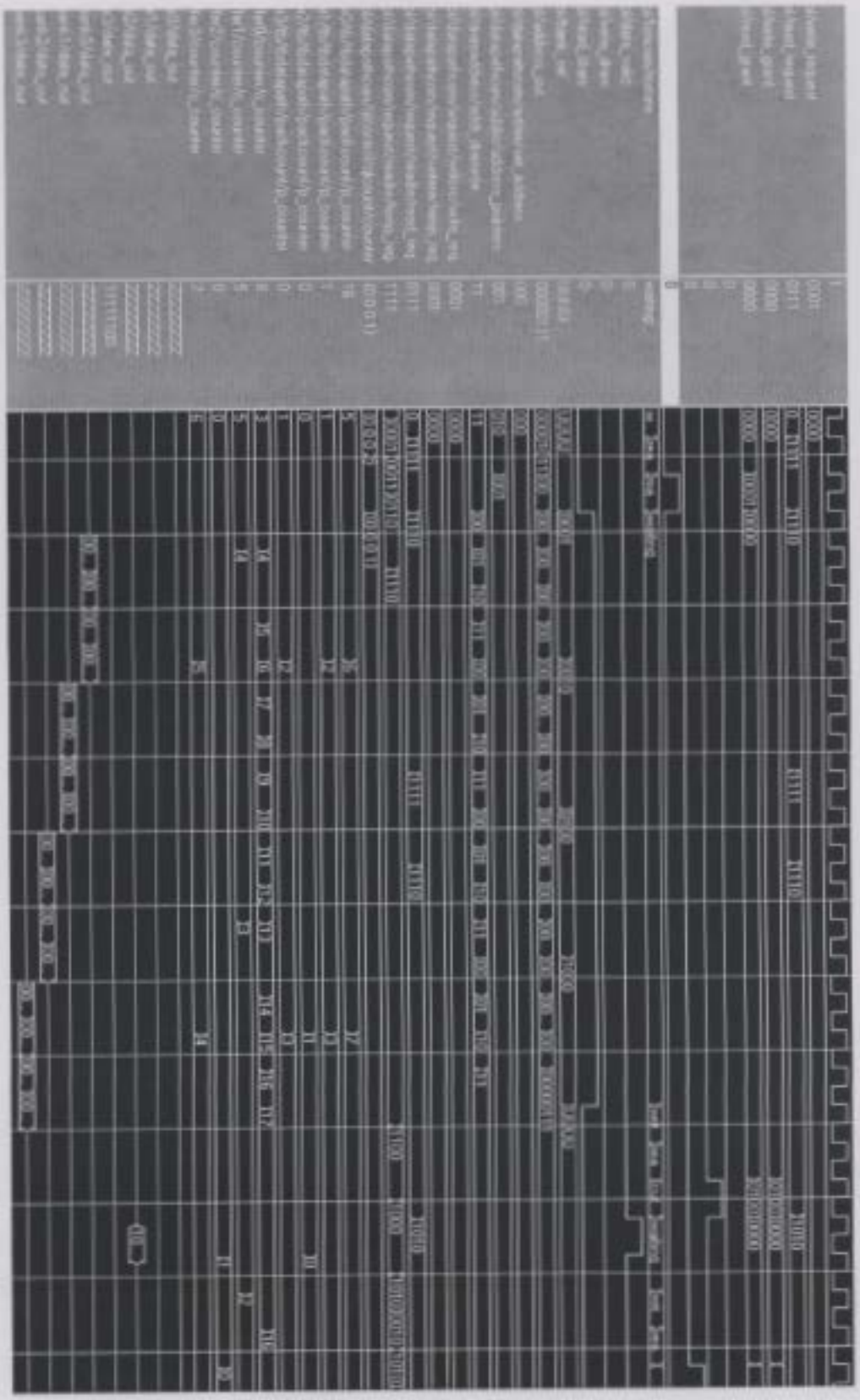

Figure 5.20: Main Controller Function Simulation - Reading

Figure 5.21: Internal Delay

Chapter 6

Conclusion and Open Issues

6.1 Summary of Thesis

In this thesis, we reviewed the fundamental issues related to high-speed switches and routers, which include the history of data networks, network switch architectures and their working principles, and the classification of high-speed network switches which include the shared memory switch, the shared medium switch, and the space-division multi-stage interconnection network based switch. We further reviewed memory technologies and their applications in the high-speed data networks. From early stage queuing systems to the advanced hybrid SRAM/DRAM architecture, which has the worst case bandwidth guarantee and optimized memory size. Based on the review and analysis, we proposed the reconfigurable Shared Output Queue(SOQ) architecture.

In order to analyze the performance of the proposed SOQ, we proposed a numerical model under uniform, random traffic. Detailed analysis of this model is presented based on a two-dimensional Markov chain model. The computed results from the

model are compared with those from simulations, which proved that this numerical model is accurate and efficient.

Then we further examined the shared queue performance under non-uniform and bursty traffic through simulations. Various bursty traffic patterns were applied to the simulation platform to obtain a comprehensive set of performances. The results indicate clearly that the shared queue scheme provides better performance. Much lower cell loss rate with a small increased delay time, which is unavoidable because more cells are accommodated in the shared queue. Moreover, we showed that, to achieve the same level of cell loss rate, the shared queue requires much less memory space, for example, the 4-port sharing scheme can reduce the memory size by half to that of the dedicated queue. This number can be as low as one third in the case of 8-port sharing.

Finally, we conduct the Shared Output Queue architecture VLSI design and implementation with a symbolic cell of 8-bit wide. From functional specifications to top down design hierarchies, we recursively decomposed the system into subsystems until down to the component level. Then from the bottom level and up, we implement each component with its functional simulation and its synthesized circuit. By constructing the subsystems and the whole system in a structural way, we successfully complete the whole design and verification process. The final implemented SOQ system is working properly: with about 75% traffic load, no cell loss is experienced during the whole hardware simulation process.

6.2 Major Contribution of Thesis

Some major contributions of this thesis are as follow.

- A Reconfigurable Shared Output Queue Architecture for High-Speed Switched/Routers

We have studied the advantages of the hybrid SRAM/DRAM architecture for data network applications. This architecture can provide both higher operation speed and massive storage space, which are desirable characteristics for queuing systems of high speed network switches and routers. Based on this observation, we proposed the reconfigurable Shared Output Queue architecture for an output queued switch system. The applications of the proposed hybrid SRAM/DRAM architecture is not limited to the output queue system, but can be extended to all high-speed network storage applications.

- Analytical Model under Uniform Random Traffic

In this thesis, an analytical model is developed based on the two-dimentional Markov chain under uniform random traffic, which provides the theoretical foundations for analyzing the behaviors of the shared output queue. This model gives accurate results and it is very efficient for computation. The results obtained from both analytical model and simulation are very close, which clearly indicates that the proposed SOQ scheme provides better performance under uniform random traffic.

- Comprehensive Analysis under Non-Uniform and Bursty Traffic

Also the thesis provides a comprehensive analysis of the proposed shared queue system performance under non-uniform bursty traffic. We adopted the simulation method to determine the system performance, and with various bursty traffic patterns, the SOQ scheme displays the superior performance on cell loss rate which is

one of the most important measurement for network applications. In general, the Shared Output Queue scheme provides much better cell loss rate performance compared with those from the dedicated queue. It also minimizes the required memory size and improves the buffer utilization.

- VLSI Design and Implementation

Finally, we proceed to conduct the architectural implementation of our proposed reconfigurable Shared Output Queue. From the functional descriptions of the SOQ system, we decompose the whole system into hierarchical levels. At each level, all components are implemented along with their functional design, verifications and synthesis. In the end, we obtain the top level structural SOQ system which performs the designated functionalities as we expected. The system provides a basic reconfiguration infrastructure for higher level controller (switch central controller or port controller) to access for run time reconfiguration.

6.3 Open Issues for Future Work

Although, we have provided a comprehensive performance study on the proposed Shared Output Queue and architecture implementation of the basic SOQ system, there are still some aspects that can be further improved in the future, which include the development of the memory management algorithm and a reconfiguration scheme.

- Shared Memory Management Algorithm

Inside the shared memory system for network applications, memory management is to determine logic queue's status and allocate buffer space to each logic queue during system operation. An efficient memory management algorithm is difficult to

derive because too many parameters should be considered, such as, different QoS requirements, different priority classes, and different traffic patterns. Furthermore, the computational complexity also affects the time for making decisions, which is critical for high speed network switches. Further research on these parameters is required to derive the efficient shared memory management algorithm.

- Better Network Traffic Modeling

As network applications are continuously exploding, numerous new applications, such as network streaming media [35] and VoIP, are appearing on the Internet. These new traffic patterns have some unique characteristics, and will lead to new challenges to traditional network infrastructures. So, a thorough study on these traffic patterns are necessary for developing an efficient shared memory management algorithm.

- Runtime Reconfiguration

As new traffic patterns appear on the Internet, the QoS requirements vary from each other. How to efficiently adapt to these demands becomes a great challenge to the design of network switches and routers. A runtime reconfiguration scheme can provide more adaptability and flexibility to adjust the system with different traffic demands. However, as the network speed continuously increaseing, the reconfiguration time and the period between two reconfigurations are critical factors, and they should be carefully managed. Hence, the study of runtime reconfiguration scheme is required in future research.

References

[1] Toomas M. Chen, Stephen S. Liu, *ATM Switching Systems.* Norwood: Artech House, 1995.

[2] "Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/digital_telephony," 2007.

[3] Srinivsan Keshar, *An engineering approach to computer networking : ATM networks, the internet, and the telephone network.* Boston, MA: Addison-Wesley, 1997.

[4] "Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/optical_carrier," 2007.

[5] Fouad A. Tobagi, "Fast packet switch architectures for broadband integrated services digital networks," in *Proceedings of the IEEE*, vol. 78, pp. 133–178, Jan 1990.

[6] H. Jonathan Chao and Cheuk H. Lam and Eiji Oki, *Broadband Packet Switching Technologies.* New York: A John Wiley & Sons, 2001.

[7] Cheng Li, *Design, Modeling and Analysis of the Balanced Gamma Multicast Switch for Broadband Communications.* PhD thesis, Memorial University of Newfoundland, Canada, St. John's, CA, 2004.

[8] Ramachandran Venkatesan and Hussein T. Mouftah, "Balanced gamma network - a new candidate for broadband packet switching architectures," in *in Proceedings of the IEEE INFOCOM'92*, vol. 3, pp. 2482–2488, 1992.

[9] Achille Pattavina, *Switching Theory.* New York: John Wiley & Sons, 1998.

[10] Yu-Shuan Yeh, Michael G. Hluchyj, and Anthony S. Acampora, "The knockout switch: A simple, modular architecture for high-performance packet switching," *IEEE Journal on Selected Areas in Communications*, vol. 5, No. 8, pp. 1274–1283, Oct 1987.

[11] K. Eddie Law, and Alberto Leon-Garcia, "A large scalable atm multicast switch," *IEEE Journal on Selected Areas in Communications*, vol. 15, No. 5, pp. 844–854, Jun 1997.

[12] H. Jonathan Chao, Byeong-Seog Choe, Jin-Soo Park and Necdet Uzun, "Design and implementation of abacus switch: A scalable multicast atm switch," *IEEE Journal on Selected Areas in Communications*, vol. 15, No. 5, pp. 830–843, Jun 1997.

[13] Sundar Iyer, Ramana R. Kompella and Nick McKeown, "Analysis of a memory architecture for fast packet buffers," in *IEEE Proceedings and Workshop, High Performance Switching and Routing*, (Dallas, USA), pp. 368–373, May 2001.

[14] William Stallings, *Computer Organization and Architecture.* Upper Saddle River, NJ: Prentice Hall, fifth ed., 2000.

[15] Sundar Iyer and Nick McKeown, "Techniques for fast shared memory switches," *Technical Report - TR01-HPNG-081501, Stanford University*, Aug 2001.

[16] "Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/static_random_access_mem 2007.

[17] Ashok K. Sharma, *Advanced Semiconductor Memories: Architectures, Designs, and Applications.*

[18] "Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/dram," 2007.

[19] IDT Datasheet, *18Mb Pipelined QDRII SRAM Burst of 4.* IDT Inc., San Jose, USA, May, 2006.

[20] Macron Datasheet, *Double Data Rate (DDR) SDRAM - 1Gb.* Macron Technology Inc., Pasadena, USA, May, 2006.

[21] Sundar Iyer, Ramana R. Kompella and Nick McKeown, "Design packet buffers for router linecards," *Technical Report - TR02-HPNG-031001, Stanford University*, Mar 2002.

[22] Michel Devault, Jean-Yves Cochennec and Michel Servel, "The 'prelude' atd experiment: Assessments and future prospects," *IEEE Journal on Selected Areas in Communications*, vol. 6, No. 9, pp. 1528–1537, Dec 1988.

[23] Jonathan J. Turner, "An optimal nonblocking multi-cast virtual circuit switch," *Technical Report WUCS-93-30, Washington University*, vol. 1, pp. 298–305, Jun 1994.

[24] Manolis Katevenis, Panagiota Vatsolaki and Aristides Efthymiou, "Pipelined memory shared buffer for vlsi switches," in *Proceedings of the ACM SIGCOMM' 95 Conference*, (Cambrige, USA), Aug 1995.

[25] Tzi-Cker Chiueh and Srinidhi Varadarajan, "Design and evaluation of a dram-based shared memory atm switch," *ACM SIGMETRICS Performance Evaluation Review*, vol. 25, Issue 1, Jun 1997.

[26] Gireesh Shrimali and Nick McKeown, "Building packet buffers using interleaved memories," in *IEEE Proceedings and Workshop, High Performance Switching and Routing*, (Hong King, PRC), pp. 1–5, May 2005.

[27] Jorge Garcia, Jesus Corbal, Llorenc Cerda and Mateo Valero, "Design and implementation of high-performance memory systems for future packet buffers," in *Proceedings of the 36th IEEE/ACM International Symposium on Microarchitecture (MICRO-36'03)*, (San Diego, USA), pp. 372–384, Dec 2003.

[28] Jonathan S. Turner, "Queueing analysis of buffered switching networks," *IEEE Transactions on Communications*, vol. 41, No. 2, Feb 1993.

[29] Alberto Monterosso and Achille Pattavina , "Performance analysis of multistage interconnection networks with shared buffered switching elements for atm switching," in *Proceeding of IEEE INFOCOM'92*, (Florence, Italy), May 1992.

[30] Giuseppe Bianchi and Jonathan S. Turner, "Improved queueing analysis of shared buffer switching networks," *IEEE/ACM Transactions on Networking*, vol. 1, No. 4, Aug 1993.

[31] Stefano Gianatti and Achille Pattavina, "Performance analysis of atm banyan networks with shared queueing - part i: Random offered traffic," *IEEE/ACM Transactions on Networking*, vol. 2, No. 4, pp. 398–410, Aug 1994.

[32] Abdullah A. Abonamah and Xyan-H. Dang, "Queueing analysis of shared-buffer atm switches with grouped output channels," *International Journal on Communication Systems*, vol. 14, 2001.

[33] Averill M. Law and W. David Kelton, *Simulation Modeling and Analysis.* Columbus, OH: McGraw Hill, third ed., 2000.

[34] "Network processing forum, http://www.npforum.org/benchmarking," 2006.

[35] "Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/internet," 2006.

[36] Jerry banks and John S. Carson II and Barry L. Nelson and David M. Nicol, *Discrete-Event System Simulation.* Upper Saddle River, NJ: Prentice Hall, 2001.

[37] Mahmoud Saleh, Mohammed Atiquzzaman, "Buffer occupancy in atm switches with single hot spot," in *Electronics Letters, IEEE*, vol. 31, Issue 1, pp. 13–15, Jan 1995.

[38] Cheng Li, Ramachandran Venkatesan and Howard M. Heys, "Design of a scalable switch architecture for efficient high-speed data multicasting." *International Journal on Communication Systems*, Aug. 2006.

[39] Ramachandran Venkatesan, Yaser El-Sayed, Rajagopalan Thuppal and H. Sivakumar, "Performance analysis of pipelined multistage interconnection networks," *Informatica: An International Journal of Computing and Informatics*, vol. 23, No. 3, pp. 347–357, Sep. 1999.

Appendix

```
****************************************
Report : cell
Design : soq_nomem
Version: 2001.08-SP2
Date   : Wed Nov 15 17:03:49 2006
****************************************

Attributes:
    b - black box (unknown)
    c - control logic
    h - hierarchical
    n - noncombinational
    r - removable
    s - synthetic operator
    u - contains unmapped logic

Cell                    Reference       Library              Area  Attributes
------------------------------------------------------------------------------
U7                      GTECH_OR2       gtech                0.00  c, u
U8                      GTECH_NOT       gtech                0.00  c, u
U9                      GTECH_AND_NOT   gtech                0.00  c, u
U10                     *SELECT_OP_2.1_2.1_1                 0.00  s, u
U18                     GTECH_OR2       gtech                0.00  c, u
U19                     GTECH_NOT       gtech                0.00  c, u
U20                     GTECH_AND_NOT   gtech                0.00  c, u
U21                     *SELECT_OP_2.1_2.1_1                 0.00  s, u
U29                     GTECH_OR2       gtech                0.00  c, u
U30                     GTECH_NOT       gtech                0.00  c, u
U31                     GTECH_AND_NOT   gtech                0.00  c, u
U32                     *SELECT_OP_2.1_2.1_1                 0.00  s, u
U40                     GTECH_OR2       gtech                0.00  c, u
U41                     GTECH_NOT       gtech                0.00  c, u
U42                     GTECH_AND_NOT   gtech                0.00  c, u
control                 main_controller                  52145.25  h, n
hdbuffer0               head_buffer0                      8106.80  h, n
hdbuffer1               head_buffer1                      8106.80  h, n
hdbuffer2               head_buffer2                      8106.80  h, n
hdbuffer3               head_buffer3                      8106.80  h, n
tbuffer0                tail_buffer0                     40834.77  h, n
tbuffer1                tail_buffer1                     40834.77  h, n
tbuffer2                tail_buffer2                     40834.77  h, n
tbuffer3                tail_buffer3                     40834.77  h, n
------------------------------------------------------------------------------
Total 24 cells                                           247911.56
```

Figure 1: SOQ System Synthesis Report

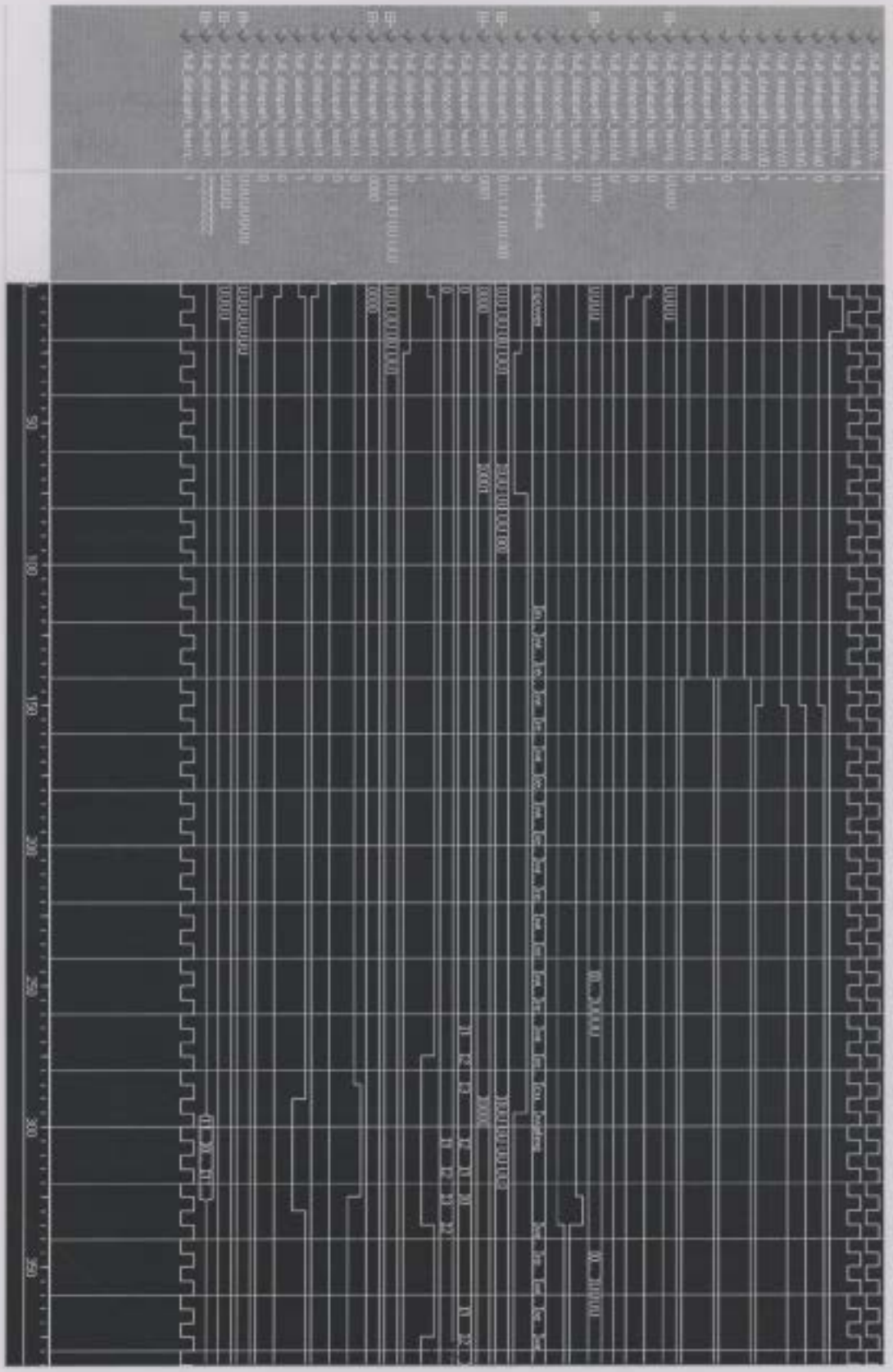

Figure 2: System Delay Time

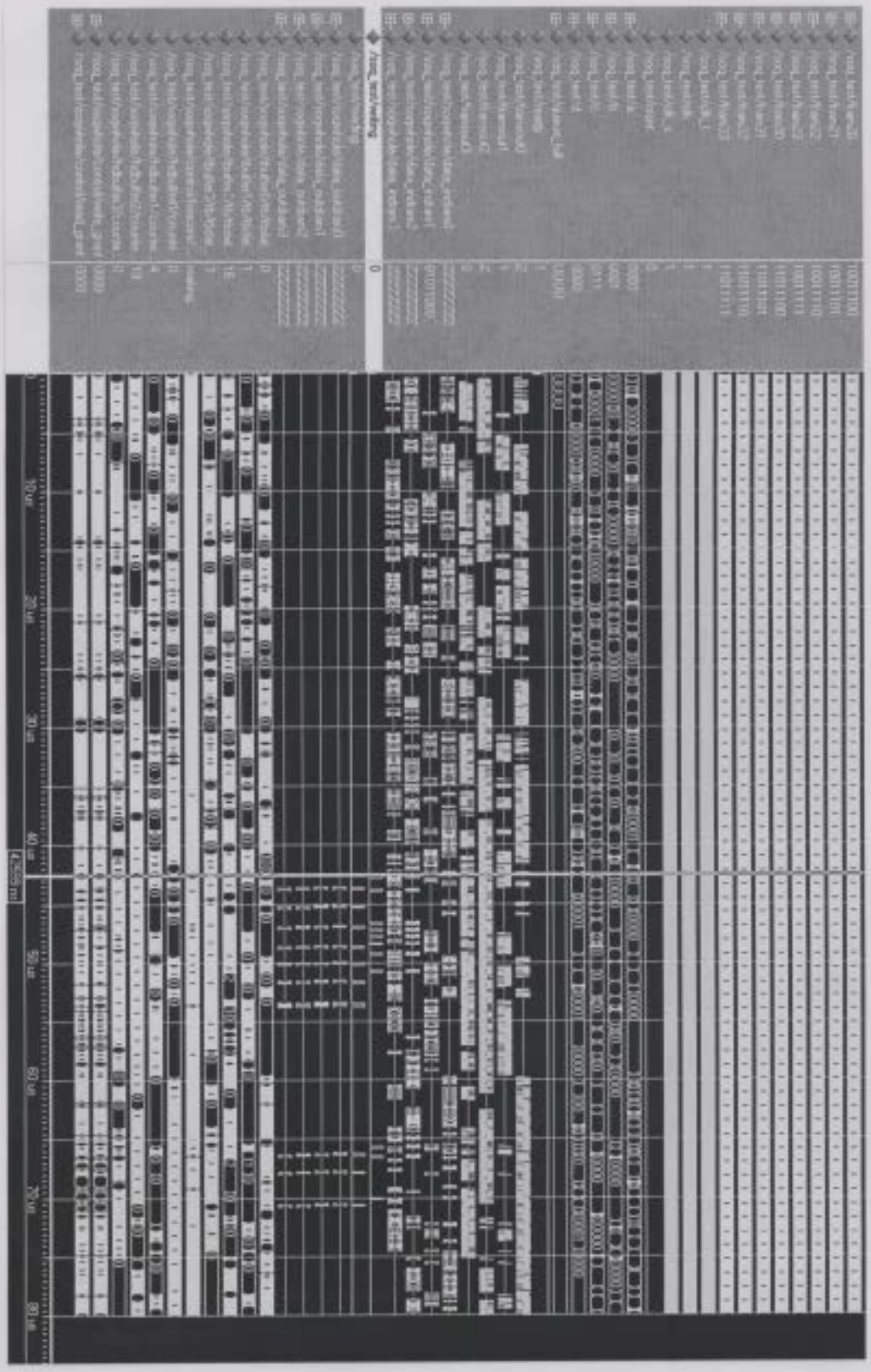

Figure 3: Waveform under 60% Traffic Load

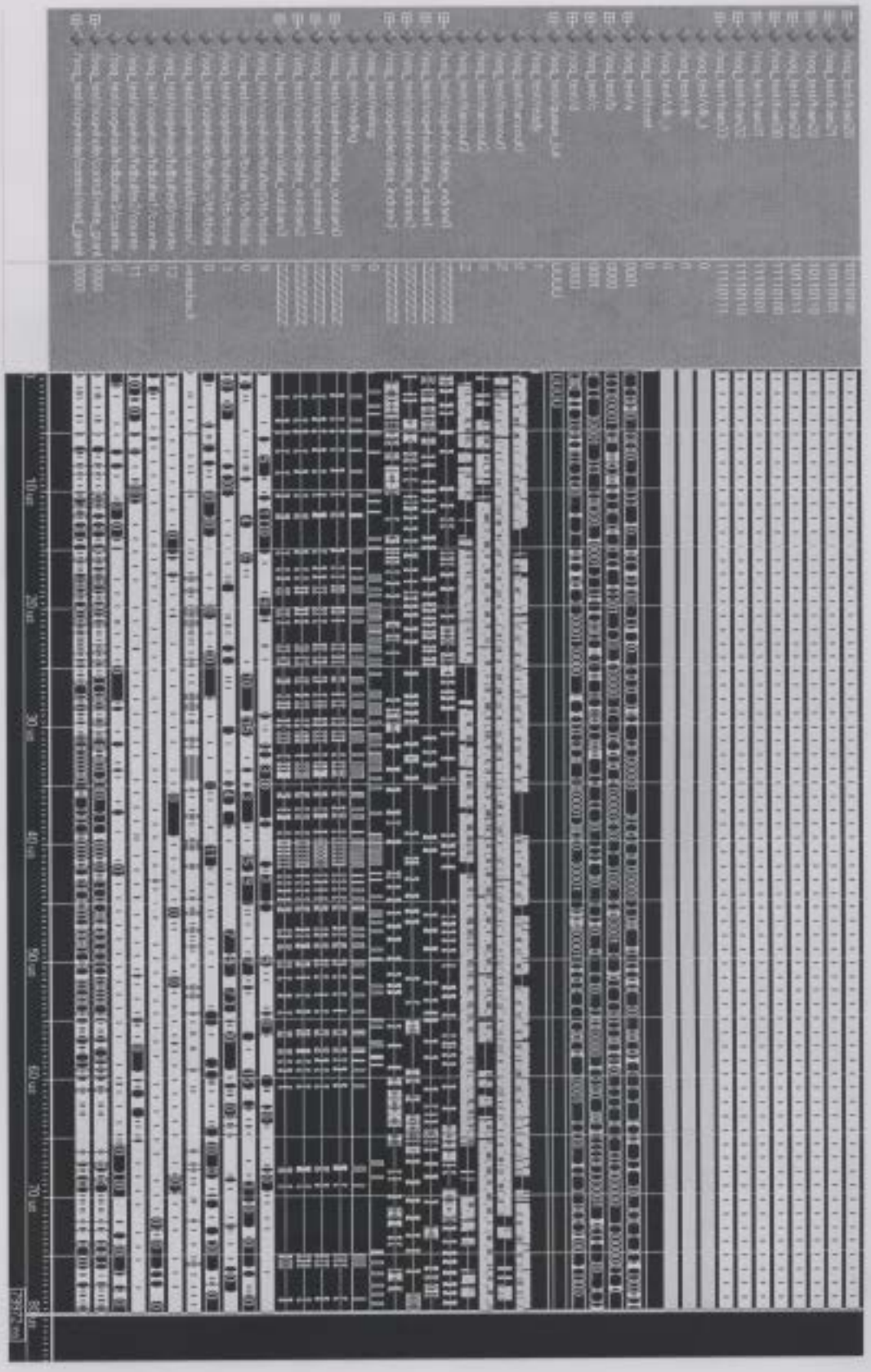

Figure 4: Waveform under 75% Traffic Load

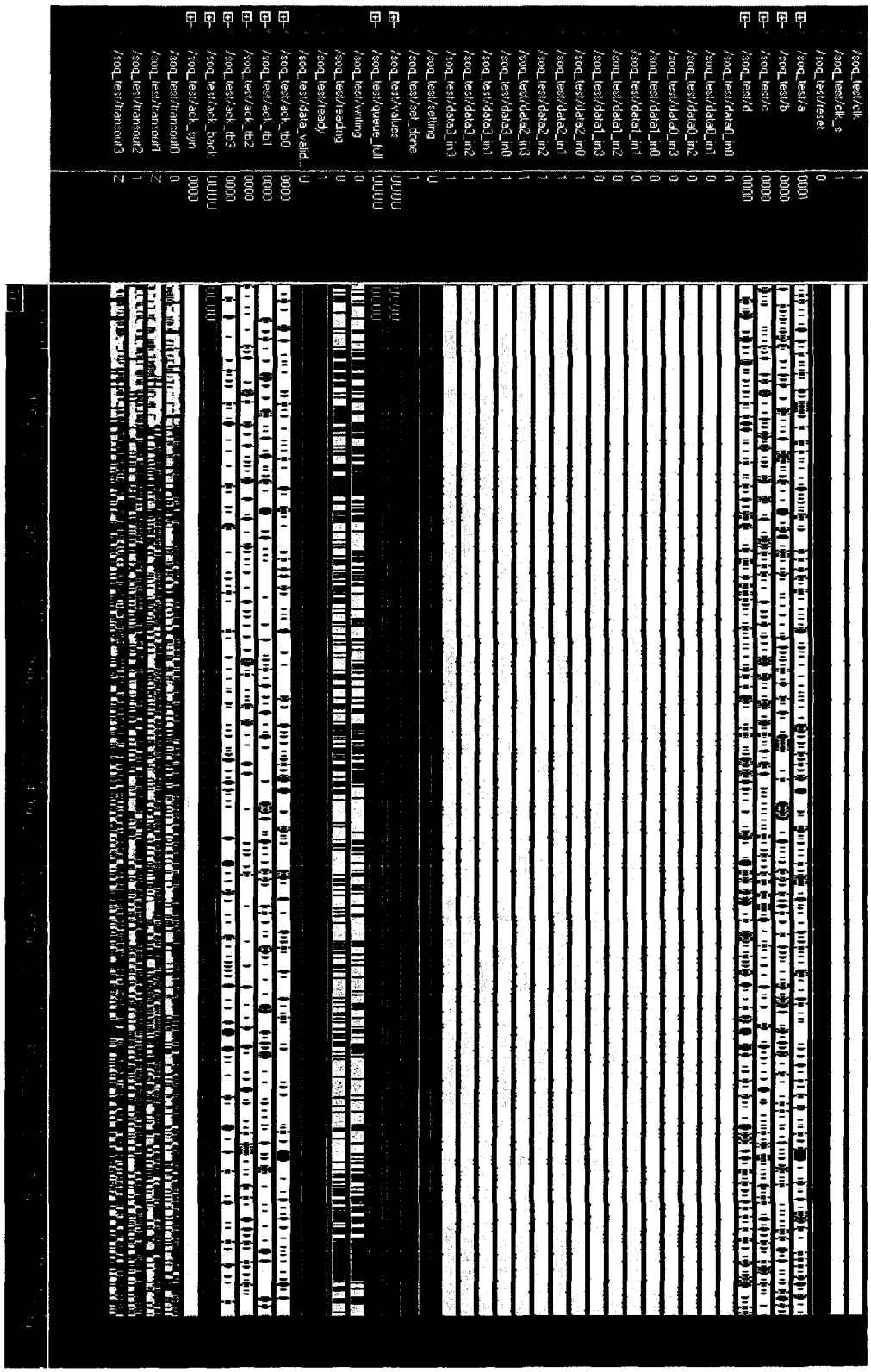

Figure 5: Whole Simulation Waveform