Design and evaluation of high-performance packet switching schemes

Taner Doganer
University of Texas-Pan American

DESIGN AND EVALUATION OF HIGH-PERFORMANCE PACKET SWITCHING SCHEMES

A Thesis

by

TANER DOGANER

Submitted to the Graduate School of the
University of Texas Pan-American
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

April 2004

Major Subject: Electrical Engineering

DESIGN AND EVALUATION OF HIGH-PERFORMANCE PACKET SWITCHING SCHEMES

A Thesis
by
TANER DOGANER

Approved as to style and content by:

[illegible]

Dr. Sanjeev Kumar, Ph.D
Chair of Committee

[illegible]

Dr. Jae Sok Son, Ph.D
Committee Member

[illegible]

Dr. Richard H. Fowler, Ph.D
Committee Member

April 2004

ABSTRACT

Doganer, Taner, Design and Evaluation of High-Performance Packet Switching Schemes. Master of Science (MS), April 2004, 108 pp., 46 figures, 26 tables, references, 39 titles.

The design of high-performance packet switches is essential to efficiently handle the exponential growth of data traffic in the next generation Internet. Shared-memory-based packet switches are known to provide the best possible delay-throughput performance and the lowest packet-loss rate compared with packet switches using other buffering strategies. However, scalability of shared-memory-based switching systems has been restricted by high memory bandwidth requirements, segregation of memory space and centralized control of switching functions that causes the switch performance to degrade as a shared-memory switch is grown in size. The new class of sliding-window based packet switches are known to overcome these problems associated with shared-memory switches. This thesis presents different schemes proposed earlier by Dr. Kumar for use in the sliding-window switch to allocate self-routing parameters. Comparative performance of these schemes have been evaluated in this thesis. The results show the scalability of the switch that can be achieved with different parameter assignment schemes. It is shown that not all assignment schemes have same performance. With appropriate assignment scheme, it is possible to achieve very high throughput-performance and switch size for sliding-window switches.

DEDICATION

This thesis is dedicated to my family, my mother Meryem, my father Yasar, and my sister Tulin, for their guidance, support, love and enthusiasm. Without these things this thesis could not have been possible. I dedicate this work to Dr. Ercan Nasif without who's help I would not be where I am.

ACKNOWLEDGEMENTS

I would like to thank all those who assisted me in writing this thesis. I would like to express sincere appreciation to Dr. Sanjeev Kumar for defining the research problem and for his guidance and assistance in completing this research work. It was a great opportunity to work on newly invented architecture. Thanks to Dr. Jae Sok Son and Dr. Richard H. Fowler for their support and their willingness to serve as the committee members. I am deeply thankful to fellow researcher Yavuz Tor for his assistance with computer programming. I am also very thankful for support from CITeC.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

1.1 Introduction

The replacement of copper with fiber and the advancements in digital communications and encoding are at the heart of several developments that will change the communication infrastructure. The world of broadband data communications is based on interconnecting small networks to larger ones. Whatever the protocols with which the interconnection is achieved - be it the traditional X.25, Frame Relay, or the modern ATM - at the lowest level we always find fast hardware-based switches, capable of delivering information from the various interconnected networks to their correct destinations [1].

The scalable and distributed nature of the Internet continuously contributes to a wild and rapid growth of its population, including the number of users, hosts, links, and emerging applications. Internet growth measurement by the number of computers attached to the Internet in each year from 1981 through 2003 has shown in Figure 1.1 [2]. The great success of the Internet thus leads to exponential increases in traffic volumes, stimulating an unprecedented demand for the capacity of the core network.

Network providers therefore face the need of providing a new network infrastructure that can support the growth of traffic in the core network. Advances in fiber throughput and optical transmission technologies have enabled operators to deploy capacity in a dramatic fashion. However, the advancement in packet switch/router technologies [3] is rather slow, so that it is still not able to keep pace with the increase in

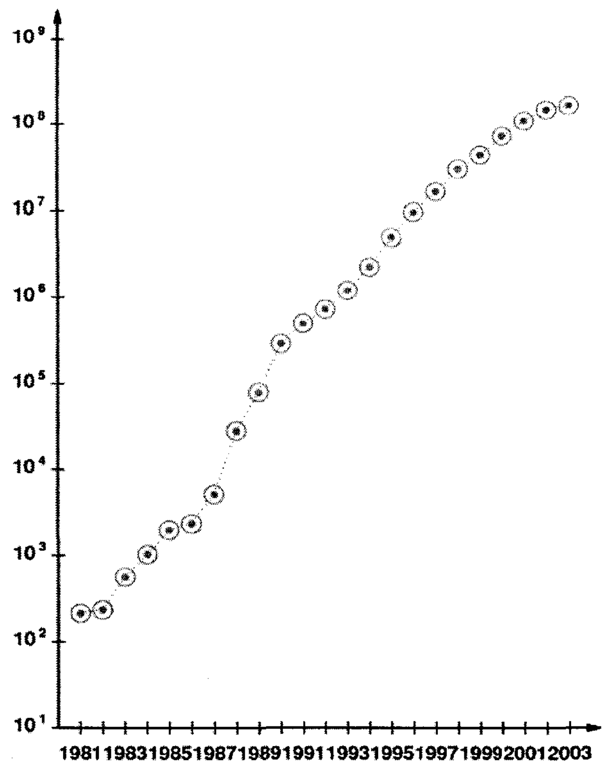


Fig. 1.1 Internet growth measured by the number of computers attached to the Internet

link transmission speed. The enhancements offered by a suitable switching mechanism should include: increased flexibility, high traffic capacity, enhanced bandwidth efficiency for ‘bursty’ traffic, inherent rate adaptation, and service independent support of multi-service traffic.

1.2 Fast Packet Switching

Packet switching allows statistical multiplexing and bandwidth on demand [4]. Conventional packet switching handles bursty traffic on packet-by- packet basis, and offers variable bandwidth channels [4].

An integrated switching mechanism nowadays is capable of the efficient support of multi-service traffic (voice, data, video, text, image, etc.). Such a switching mechanism must handle bursty traffic efficiently by offering connections with low packet loss ratio and high throughput capacity for a given resource deployed at the switch. A switch design capable of growth to a very high traffic capacity will also be required to support the expected growth in multi-service traffic. Fast packet switching has been proposed as an integrated switching mechanism to satisfy these requirements. It attempts to retain the flexibility of conventional packet switching while reducing the delay and increasing the maximum switch capacity for Internet traffic. Fast packet switches usually deal with high bandwidth and low bit error rate offered by optical fiber transmission.

The essential difference between fast packet switching and conventional packet switching is that every port of a fast packet switch handles several orders of magnitude more packets per second than that of a conventional packet switch. In a network with fast packet switching, the link-by-link protocol functions of error control are minimized or removed, and they are rather implemented on an end-to-end basis. Fast packet switching avoids complex routing algorithms that act on a per packet basis [4] and rather deploy switching of packets for faster packet transfer.

1.3 Fast Packet Switch Architecture

A number of approaches to the design of a fast packet switch have been implemented but some basic concepts are common to most designs. The general structure of a fast packet switch is given in Figure 1.2.

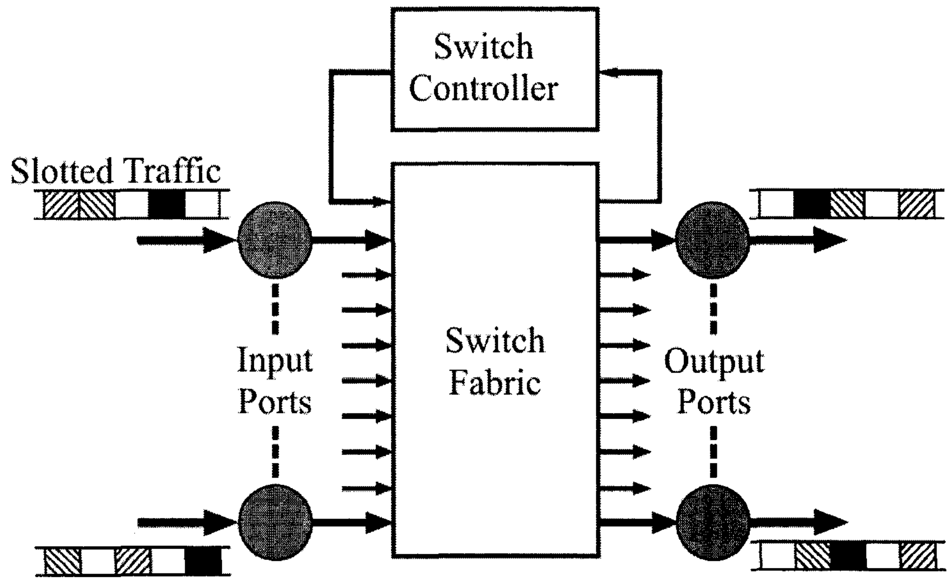


Fig. 1.2 General structure of a fast packet switch [4]

All of per packet processing functions are performed by the input and output ports, which work concurrently and possibly independent of one another. The switch controller is required only for higher-level functions such as connection establishment, bandwidth reservation, management, and congestion control. The switch controller may communicate with the input and output ports either directly or via packets across the switch fabric. External connections to the switch are generally required in the form of bi-directional links that are formed by grouping an input and an output port together.

Many fast packet switches are designed only to handle short fixed length packets. In such designs the bandwidth on both input and output lines is divided into time-slots, each of which may carry a single packet (cell) or may be empty (i.e. slotted traffic). All lines must be synchronized and this is accomplished either by means of a frame structure, as in Time Division Multiplexing (TDM), or by filling empty packets with a synchronization pattern.

Fast packet switches with a total traffic capacity of up to several Gbits/sec may be constructed using a single path switch fabric such as shared memory [5]. Beyond this capacity a multi-path design must be adopted where the traffic capacity is no longer limited by the bandwidth of a single path (or shared medium) switch fabric, but grows with the number of paths in the switch fabric. A multi-path switch design may be achieved by implementing the switch fabric with a self-routing multi-stage interconnection network of simple switching elements. Alternatively, a number of complete fast packet switches, based upon a single path switch fabric, may be interconnected to form a larger structure. Both approaches achieve a multi-path fast packet switch by interconnection of smaller switching elements.

A range of self-routing multi-stage interconnection networks is available offering a choice of complexity against throughput [6-9]. A self-routing switch fabric will require a tag to be prefixed to each packet, which specifies the required destination output port number. This may be implemented by table look-up in the input port. The self-routing property of the interconnection network ensures that each packet will be routed towards the correct output port by a simple decision made by each switching element in the path. This decision is based only upon the packet tag. It may be implemented without reference to the position of the switching element within the network and it does not depend on the address of the input port from which the packet originated.

1.4 A Simple Classification of Switch Designs

Two fundamental components are required to construct a fast packet switch: switching and buffering; and relative positioning of these components permits a simple classification of fast packet switch design. If the buffering remains external to the switch fabric the design is based upon a non-buffered switch fabric. Else, if the buffering is implemented within each of the switching elements forming the switch fabric a buffered switch fabric (or internally buffered) design results. Of the designs based upon a non-buffered switch fabric, if the buffering precedes the switch fabric, the switch is classified as input buffered. Else, if the buffering follows the switch fabric, the design is output buffered.

1.5 Buffering Strategies

In this section, fast packet switches are classified to their buffering schemes. Main types of buffering schemes are described and below, their advantages and disadvantages are discussed.

1.5.1. Internally Buffered Switches

Internally buffered switches are those that employ buffers within switching element. An example of this type is the buffer banyan switch [10]. The buffers are used to store internally blocked packets so that the packet loss ratio can be reduced. Scalability of the switch can be achieved by replicating the switching elements. However this type of switch suffers from low throughput and high transfer delay that is caused by the delay from multiple stages. To meet QoS requirements, some scheduling and buffer management schemes need to be installed at the internal switching elements, which will increase the implementation cost.

1.5.2 Input Buffered Switches

An input buffered switch buffers incoming packets at the input ports of the switch before transmitting them across the switch fabric. The input buffered switch suffers from the Head-of-Line (HOL) blocking problem and limits the throughput to 58.6 % [16] for uniform traffic. If a packet at the head of the input queue at any input port find that the output port it requires is busy then it must wait at the input port until the required output port becomes free. While it is waiting at the head of the queue it prevents other packets behind it in the queue from being serviced. In order to increase the switch's throughput, a technique called "windowing" can be employed, where multiple packets from each input buffer are examined and considered to transmission to the output port. But a most one packet will be chosen in each time slot. The number of examined packets determines the window size. It has been shown that by increasing the window size to two, the maximum throughput is increased to 70 % [18-20]. Increasing the window size does not improve the maximum throughput significantly, but increases the implementation complexity of input buffers and arbitration mechanism. This is because the input buffers cannot use simple FIFO memory any longer, and more packets need to be arbitrated in each time slot. Several techniques have been proposed to increase the throughput [11-20].

1.5.3 Output Buffered Switches

An output buffered switch allows all incoming packets to arrive at the output port. Because there is no HOL blocking, the switch can achieve 100 % throughput. However, since the output buffer needs to store N packets in each time slot, its memory speed will limit the switch size. A concentrator can be used to alleviate the memory speed limitation problem so as to have a larger switch size. The disadvantage of this remedy is the inevitable packet loss in the concentrator.

The Batcher-Banyan switch fabric may also be employed in an output buffered switch. An example of such a switch design is provided by the Starlite switch [21]. In this switch contention resolution is performed by arbitration within a trap network located between the Batcher and Banyan switch fabrics. All packets that loose arbitration are directed to a set of shared buffers on the output side of the switch fabric. These packets are recirculated back into the input side of the switch fabric at the beginning of the next time slot. In this design, the size of the switch fabric must be greater than the number of input and output ports in order to accommodate the recirculation buffers. Analysis has shown that the number of switch fabric ports devoted to the re-entry buffers must be two to three times the number of switch input/output ports in order to ensure an acceptably low probability of packet loss [22]. Thus the ideal performance of output buffered switch is gained at the expense of a much larger and more complex switch fabric.

The Knockout switch is another example of an output buffered switch but differs from Starlite in that buffers are dedicated to specific output ports are not shared [23]. All incident packets are broadcast concurrently to every output port. Each output port inspects the destination tag of every packet and selects those with a tag, which matches its own output port number. Up to L packets arriving at the same output port within the same time slot may be handled by each output port and stored in an output buffer. For random traffic, the probability of more than L packets arriving in the same time slot routed to the same output port is very low and excess packets are simply discarded. The most obvious difficulty with the Knockout Switch arises from the use of a fully interconnected broadcast switch fabric. This result in an increase in the hardware complexity and interconnections required of at least one order of magnitude when compared to input buffered switches of the same size. It does, however, offer the advantage of incremental growth allowing the switch to grow by one switch port at a time. The buffer requirement is approximately twice that of an input buffered switch for a packet loss probability on the region of 10^{-6} for traffic with a random destination distribution.

1.5.4 Input and Output Buffered Switches

Input and output buffered switches are intended to combine the advantages of input buffering and output buffering. In input buffering, the input buffer speed is comparable to the input line rate. In output buffering, there are up to L ($1 < L < N$) packets that each output port can accept at each time slot. If there are more than L packets destined for the same output port, excess packets are stored in the input buffers instead of discarding them as in the concentrator. To achieve a desired throughput, the speedup factor L can be engineered based on the input traffic distribution. Since the output buffer memory only needs to operate at L times the line rate, a large-scale switch can be achieved by using input and output buffering. However, this type of switch requires a complicated arbitration mechanism to determine which of L packets among the N HOL packets may go output port.

Another kind of speedup is run the switch at a higher rate than the input and output line rate. In other words, during each packet slot, there can be more than one packet transmitted from an input to an output.

1.5.5 Shared Buffered Switches

In shared buffer switches, all packets are stored in a common buffer shared by all inputs and outputs. One can expect that it will need less buffer to achieve a given packet loss ration, due to the statistical nature of packet arrivals. Output queuing can be maintained logically with link list, so that no packets will be blocked from reaching idle output ports, and still can achieve the optimal throughput-delay performance, as with dedicated output queuing.

Shared buffer switches, their advantages and disadvantages will be discussed in detail in Chapter 2.

CHAPTER 2

SHARED-MEMORY SWITCHES

2.1 Shared Memory Switches

In this section some of the shared-memory switches and a new class of packet shared-memory switch with plural memory modules and decentralized control will be investigated.

In a shared-memory switch, as shown in Figure 2.1, incoming packets are time-division multiplexed into a single data stream and sequentially written to shared-memory. The routing of packets is accomplished by extracting stored packet to form a single output data stream, which is in turn demultiplexed into several outgoing lines. The memory addresses for both writing incoming packets and reading out stored packets are provided by a control module according to routing information extracted from the packet headers.

Shared-memory switches that operate without blocking, all input and output ports have access to a shared-memory module are able to write up to N incoming packets and to read out N outgoing packets in a switching cycle, so that as in output-buffered switches, throughput is not reduced by output port contention, and an optimal throughput/delay performance is achieved. Furthermore, to provide a specified level of service (i.e., packet loss rate) a small number of buffers is required (in comparison with the amount buffering schemes) for all traffic patterns.

A shared-memory switch also has some additional features, e.g., its basic architecture can be easily modified to handle several service classes through priority control functions to meet different service requirements, multicasting and broadcasting can be also easily implemented in contrast to other types of architectures. Shared-memory type switches are, therefore, very attractive for the industry and majority of implemented prototypes use them as the building blocks for larger switching fabrics [25].

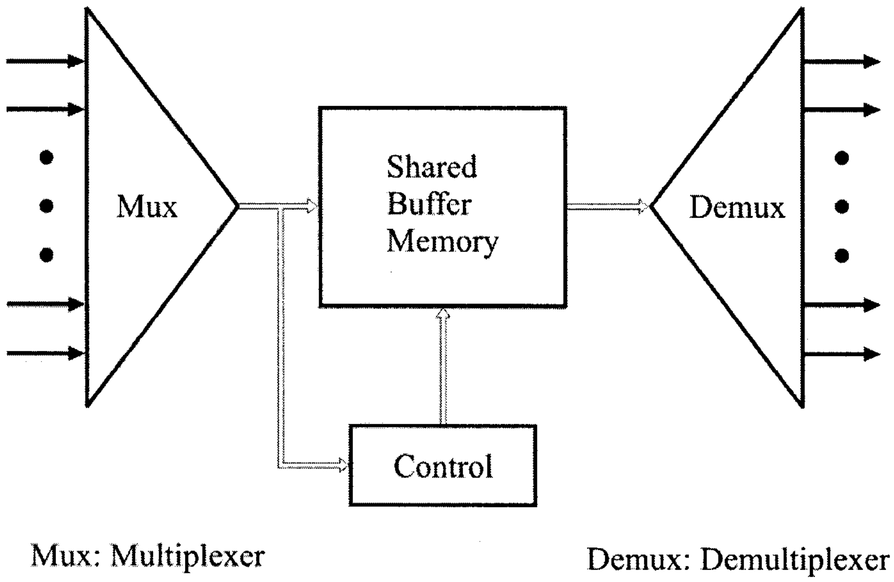


Fig. 2.1 Basic architecture of shared-memory switches.

The advantage of the shared-memory switch type is that it provides the best memory utilization, since all input/output ports share the same memory. The memory size should be adjusted to keep the packet loss rate below a chosen value. There are two different approaches in sharing memory among the ports: complete partitioning and full sharing. In complete partitioning, the entire memory divided into N equal parts, where N is the number of input/output ports, and each part is assigned to a particular output port. In full sharing, the entire memory is shared by all output ports without any reservation. Some mechanisms, such as putting an upper and a lower bound on the memory space, are needed to prevent monopolization of the memory by some output ports [38].

2.2 Shared-Memory Switching Architectures

In this section, some of different approaches to organize the shared-memory switching architectures will be discussed.

2.2.1 The Prelude Switch

The Prelude was the first shared-memory type switch architecture using the ATM-like concept, described in the literature [26]. Because it was the pioneering shared-memory ATM switch and since some features of this architecture are still used in various switches today. The Prelude switch is a modified conventional time-division switch. The switch operates synchronously with the time cycle equal to the duration of an octet. The switch can be divided into parts performing the following functions (Figure 2.2): serial/parallel conversion, clock adaptation and phase alignment, supermultiplexing, control, buffering, demultiplexing and parallel/serial conversion.

After a serial to parallel conversion of the incoming packets (to reduce the required physical memory speed), packets join the clock adaptation queues from where they will be extracted and aligned in such a manner that they present a shift of one octet, so the headers on different inlets arrive at consecutive time slots.

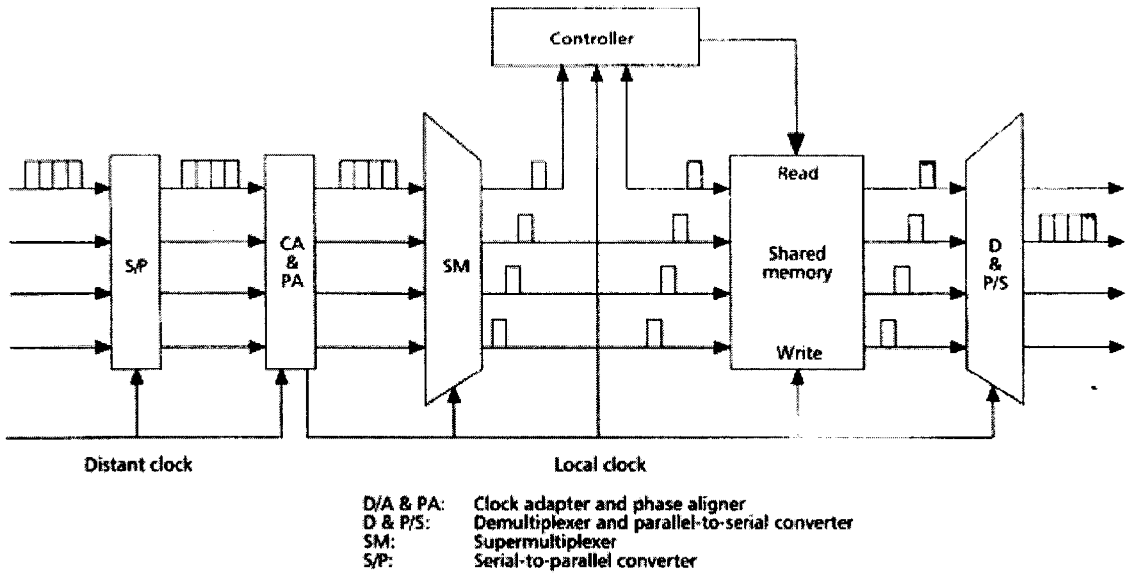

Controller
Read
Shared memory
Write
S/P
CA & PA
SM
D & P/S
Distant clock
Local clock
D/A & PA: Clock adapter and phase aligner
D & P/S: Demultiplexer and parallel-to-serial converter
SM: Supermultiplexer
S/P: Serial-to-parallel converter

Fig 2.2 The Prelude Switch [26]

The supermultiplexing function performed by a 16 x 16 space-division switch that sets its internal paths every time cycle according to a cyclic modulo N algorithm, that is, in state 1, input i is connected to output i, in the following state input i is connected to output (i + 1) mod N, and so on, where N is the switch size. As the result, all the headers of different input packets are multiplexed on the first output of the supermultiplexing stage, all the first user information octets on the second output, etc. After this so-called parallel-diagonal transformation, header octets are successfully processed and translated by a central controller to define the switch action. The packets are stored diagonally in the buffer memory, which is organized in N banks, each bank being dedicated to an octet of the packet, i.e., the header stored in an empty location of bank 1, and the remaining octets of same packet are stored in consecutive banks. The address register is increased by one for each successive octet. The address where the packet header is placed is stored by the controller in a dedicated queue for the destined output port.

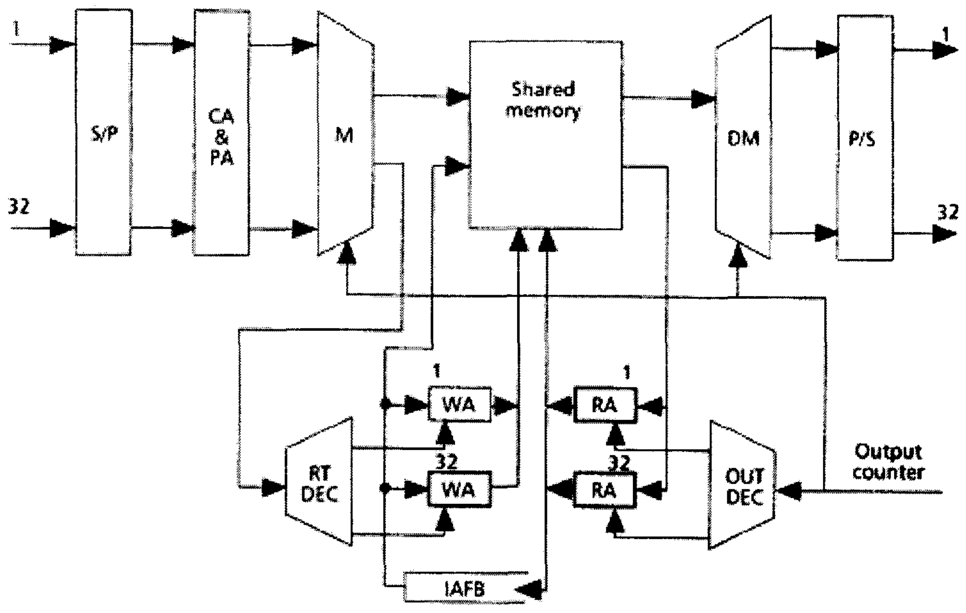
To read out the memory packets to the output links, the controller, at each time cycle, delivers address A in the first bank where a packet header of a packet to be extracted is stored, the reminder of the packet is easily retrieved from the other banks during the following time cycles incrementing A. Retrieved packets are fed to a rotative space-division switch which acts in a reversed order as in the supermultiplexing stage, reconstructing the packets. Finally, the packets are converted from parallel to serial form and are transmitted to the corresponding output lines.

The output control queues have fixed size, which means that the memory is not completely shared. However, since all input and output lines have access to the shared-memory, it would not be difficult to extend this design to achieve a higher degree of sharing.

2.2.2 Link-List-Based Shared-Memory Switch

Another group of shared memory switches employs a link list to logically organize the buffering system. In this switching system, the buffer memories for output queueing are completely shared by all the switch output ports and can be assigned to any output ports and can be assigned to any output port as required by traffic conditions.

The switch architecture is illustrated by Figure 2.3, and its basic operation is as follows. First packets are converted from serial to parallel format; the packet headers are analyzed and appropriately translated by header converters. Subsequently, packets from all input ports are multiplexed and written into the shared buffer memory. The buffer memory is logically organized as N (N=32) linked list, one for cache output line. A linked list is a set of chained memory locations that indicate the place where successive



DM:	Demultiplexer
HC:	Header converters
IAFB:	Idle address FIFO buffer
M:	Multiplexer
OUT DEC:	Output decoder
P/S:	Parallel-to-serial converters
RA:	Read address register
RT DEC:	Route decoder
S/P:	Serial-to-parallel converters
WA:	Write address register

Fig. 2.3 Link-list-based shared memory switch [27]

packets for a particular output are stored, maintaining, in that way, the packet sequence. A pair of address registers (one to write WA and one to read RA) control the proper operations on the shared memory. Write register i indicates the end of the chain and so designates an empty memory location where the last packet that arrives for output i can be stored. Read register i indicates the beginning of the list, and therefore, the location of the first packet that can be delivered to output port i. Both the packets and the linked list structures are stored in the same buffer memory.

For each input packet, the appropriate WA register (designated by the route decoder RT DEC) is accessed to get the memory location for that packet. Simultaneously, a new address of an empty buffer (IAFB), which keeps a pool of empty buffer locations, to update the content of the WA register. Analogously, at each switching cycle a packet from each linked list is identified through the content of RA registers, retrieved, demultiplexed, and transmitted. At the same time, the contents of RA registers rejoin the idle address buffer and are replaced by the next chain addresses obtained from the same addresses as the output packets, thus updating the pointers.

The basic design can be easily modified to accommodate different service classes organizing the packets addressed to a given output port into multiple linked lists, one for each service class and serving these lists according to a prioritized scheme. This requires the addition of a pair of WA and RA registers to the control block for each service class and each output port.

2.2.3 Hybrid Shared and Dedicated Output Buffer Switch

In this approach the control mechanism to route the packet in the shared memory to its output port uses N dedicated FIFO buffers, one for each output port [28]. Thus, instead of arranging packets, which have the same output port destination in a linked list by the address chain pointer, their addresses are stored into a FIFO buffer, which is dedicated to a specific output port.

The switch architecture is shown in Figure 2.4. Serially incoming packets are converted to bit parallel data to reduce the required internal speed of the switch, sequentially multiplexed in time and stored in a shared buffer memory. Their respective buffer addresses are recorded on separate FIFO buffers associated with the specific destinations of every packet. A particular FIFO buffer is selected by examination and

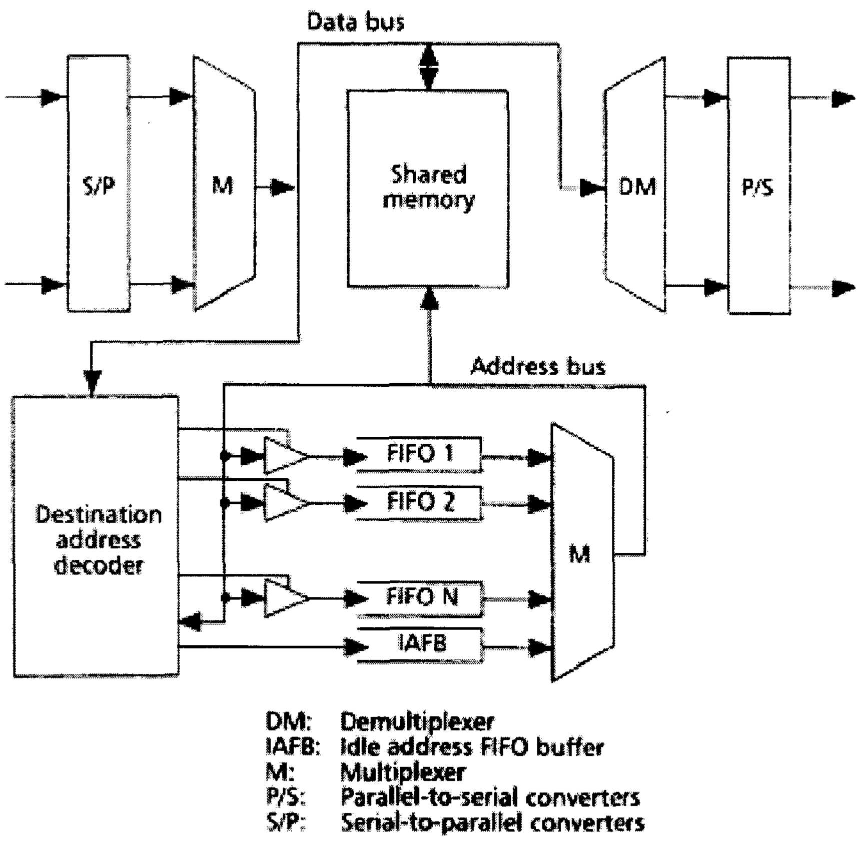

Data bus
S/P
M
Shared memory
DM
P/S
Address bus
Destination address decoder
FIFO 1
FIFO 2
FIFO N
IAFB
M
DM: Demultiplexer
IAFB: Idle address FIFO buffer
M: Multiplexer
P/S: Parallel-to-serial converters
S/P: Serial-to-parallel converters

Fig. 2.4 Hybrid Shared and Dedicated Output Buffer shared memory switch [28]

conversion of the routing header information of each packet. The write addresses to store the incoming packets into the memory pool are provided by an idle address FIFO buffer that holds all the empty buffer locations of the shared memory. The addresses in the FIFO buffers are used read the packets out from the shared buffer memory, demultiplex and reconvert them to serial format, and finally to send them to their destinations. The new free addresses are returned to the idle address FIFO queue.

The performance of the switch is the same as that of the shared switch using linked lists, but the necessary amount of buffering required for the FIFO queues.

Nevertheless, the implementation is simple and the multicasting and priority control easy to implement.

2.2.4 Space-Time-Space (STS) Shared-Memory Switch

Separate buffer memories are shared among all input and output ports via crosspoint space division switches. Figure 2.5 depicts an 8 x 8 example to show the basic configuration of the space-time-space (STS) shared-memory switch [30-31]. The multiplexing and demultiplexing stages in the traditional shared-memory switch are placed with crosspoint space switches; thus the resulting structure is referred to as STS-type. Since there is no time-division multiplexing, the required memory access speed may be drastically reduced.

The WRITE process is as follows. The destination of each incoming packet is inspected in a header detector, and is forwarded to the control block that controls the input-side space switch and thus the connection between the input ports and the buffer memories. As the number of shared-buffer memories (SBMs) is equal to or greater than the number of input ports, each incoming packets can surely be written into an SBM as long as no SBM is full. In order to realize the buffer sharing effectively, packets are written to the least occupied SBM first and the most occupied SBM last. When a packet is written into an SBM, its position (the SBM number and the address of the packet in the SBM) is queued in the address queue. The read address selector picks out the first address from each address queue and controls the output-side space switch to connect the picked packets (SBMs) with the corresponding output ports.

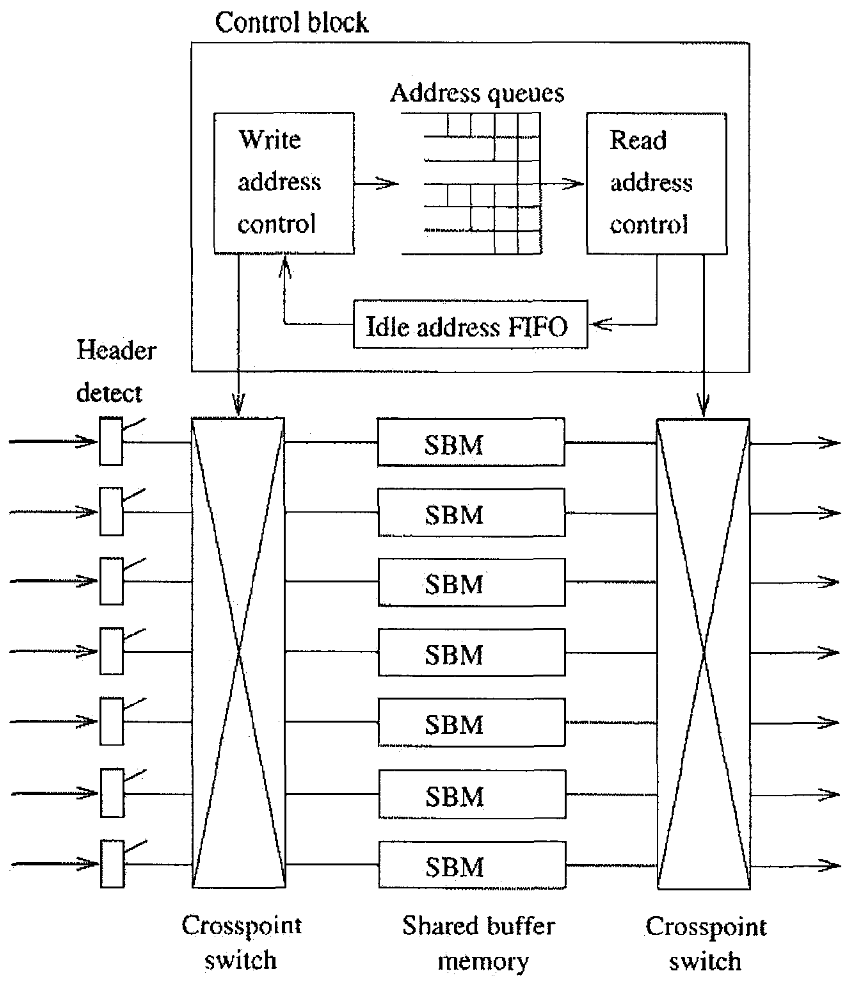


Fig. 2.5 Space-Time-Space (STS) shared memory switch [30-31]

It may occur that two or more packets picked for different output ports are from the same SBM. Thus, to increase the switch's throughput it requires some kind of internal speedup to allow more than one packet to be read out from an SBM in every packet slot. Two methods can be used to alleviate the effects produced by this type of contention: one is to increase the number of SBMs, the other is to accelerate the memory read out process. Both methods are evaluated in [30] and it is concluded that the speedup method gives better result. A speedup factor of 3 is chosen, therefore, an SBM has to be able to perform one write and three read operations in a packet time. Another disadvantage of this switch is the requirement for searching for the least occupied SBM (may need multiple searches in a time slot), which may cause a system bottleneck when the number of SBMs is large.

2.2.5 The Sliding-Window Shared-Memory Switch

Growth in size of the switch in [30-31] would make central controller and the switch more complex and therefore, limit the capacity of a STS-type switch. Contrary to the switch in [30-31], the Sliding-Window (SW) switch's memory modules are not controlled by any centralized memory controller [32]. In the SW switch [32], memory modules are independent and they use their local memory controllers to perform WRITE and READ operations for data packets based only on the information available locally. The SW switch provides a way to reduce the performance bottleneck created by centralized controllers in typical shared-memory switch architecture. The main objectives [32] of the SW switch architecture are to:

- facilitate global sharing of physically separate memory modules among all the input and output ports of the switch to reduce packet loss under bursty traffic conditions;
- alleviate the need for a centralized memory controller;
- partition the overall switching function into multiple independent stages;
- allow multiple independent stages to make switching decisions based only on the information available locally;
- operate multiple independent stages in a pipeline fashion in order to enhance packet switching speed;

- provide maximum output utilization even when backlog occurs due to burstiness; and
- provide various memory sharing schemes for finite memory space deployed in the switch.

2.2.5.1 Components of the sliding-window switch

Figure 2.6 shows the overall architecture of the SW switching system with decentralized pipelined control. The SW based switching system consists of the following independent stages, namely: (1) the self-routing parameter assignment circuit; (2) the input interconnection network; (3) shared parallel memory modules used for write and read of data-packets; and (4) the output interconnection network. Input lines carry the incoming data packets and the output lines carry the outgoing data packets after being

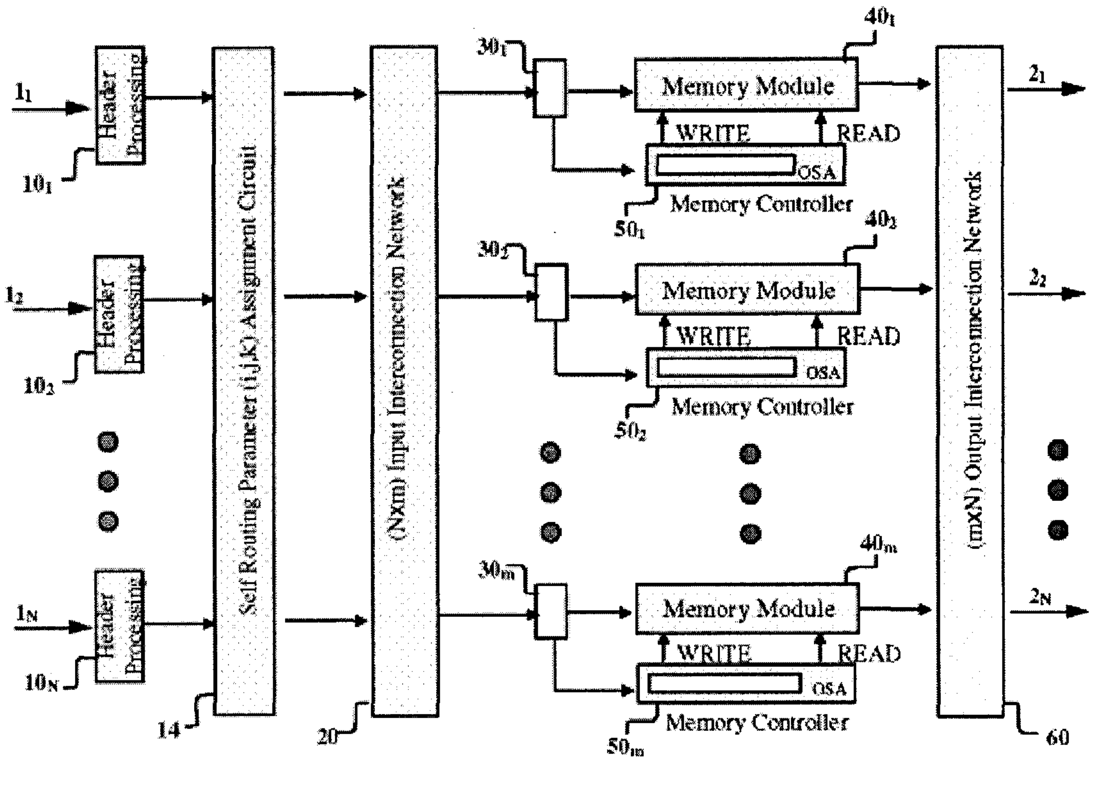

Header Processing
Self Routing Parameter (i,j,k) Assignment Circuit
(Nxm) Input Interconnection Network
Memory Module
WRITE
READ
OSA
Memory Controller
(mxN) Output Interconnection Network
14
20
60

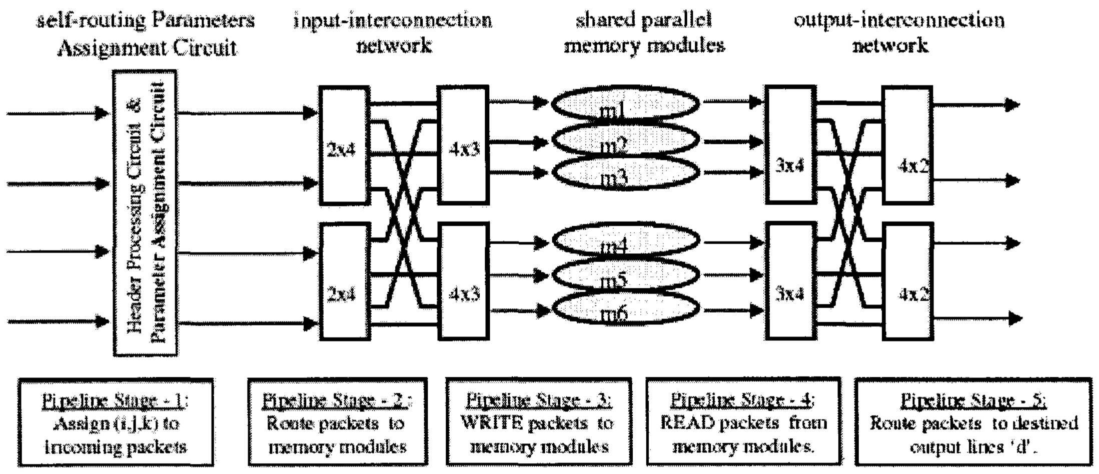
Fig. 2.6 Schematic diagram of the SW-based switch architecture [32]

switched to their output destination by the SW switching system of Figure 2.6.The incoming packets are processed by header processing circuit for extraction of the output-port destination address denoted by d, the destination address of incoming packets is forwarded to a self-routing parameter assignment circuit. The self-routing parameter assignment circuit uses output destination information d and a parameter assignment method to produce self-routing parameters (i, j, k) for incoming data packets. The self-routing parameters (i, j, k) are then attached as a self-routing tag to the incoming data packets. Thereafter, incoming packets use the attached self-routing tag (i, j, k) to propagate independently through various stages of the SW packet switching system of figure 2.6.

The input interconnection network uses the parameter i of the routing tag of an incoming packet to route the packet on a given input line to its i^{th} output line which in turn is connected to the respective i^{th} memory module. The memory controllers use the parameter j to write the received packet in the j^{th} memory location of the corresponding memory modules. Corresponding to each memory controller there is one output scan array (OSA) each with σ locations. The j^{th} location of the OSA holds the scan value of a received packet stored in the corresponding j^{th} location of its memory module. OSA of each memory controller is updated at the time of write and read of data packets to and from the respective locations in the memory modules.

During the packet WRITE cycle of an incoming packet to j^{th} memory location in a given memory module i, the associated scan-plane value (k) of the received packet is stored in the corresponding j^{th} location in the OSA of the corresponding memory controller. During the READ cycle of a packet from the j^{th} location of a memory module, the corresponding jth location in the OSA is set to zero to indicate empty memory location in the corresponding memory module. During the packet READ cycle, the data packets are output from parallel and independent memory modules and are finally routed to the respective output destinations by the output interconnection network.

The output interconnection network makes use of the output destination information d stored in a packet's header to route each packet to a final output destination. An example of a 4x4 SW switch with decentralized control is shown in figure 2.7 [32].



self-routing Parameters Assignment Circuit
input-interconnection network
shared parallel memory modules
output-interconnection network
Header Processing Circuit & Parameter Assignment Circuit
2x4
2x4
4x3
4x3
m1
m2
m3
m4
m5
m6
3x4
3x4
4x2
4x2
Pipeline Stage - 1: Assign (i,j,k) to incoming packets
Pipeline Stage - 2 : Route packets to memory modules
Pipeline Stage - 3: WRITE packets to memory modules
Pipeline Stage - 4: READ packets from memory modules.
Pipeline Stage - 5: Route packets to destined output lines 'd'.

Fig. 2.7 Example of a 4x4 SW switch architecture deploying 6 parallel memory modules [32]

The assignment of self-routing parameters (i, j, k) to the incoming packets is achieved by parameter assignment circuit (PAC). There are several possible ways of assignment of self-routing parameters for incoming packets. The speed of PAC will determine not only the size of PAC, but also the switch size. Therefore, the goal of this research is to enable faster assignments to the incoming packets, design PAC in order to maximize higher switch capacity by using different techniques solving the problems related the scalability.

CHAPTER 3

ARCHITECTURE OF PARAMETER ASSIGNMENT CIRCUIT

3.1 Self-Routing Parameters (i, j, k)

Self-routing parameters help data packets to self propagate through different stages of the switching system. The assignment of self-routing parameters (i, j, k) to the incoming packets is performed by the parameter assignment circuit. As explained in chapter 2, the self-routing parameter assignment circuit uses destination d of an incoming packet to produce a self-routing tag (i, j, k). In this chapter the speed of the Parameter Assignment Circuit (PAC) is evaluated for different assignment schemes. It is possible to design a Fast PAC with known hardware components or ASIC [35-36]. In this chapter we evaluated assignment schemes using general purpose processor e.g. Pentium 4 and also using known hardware components.

3.2 Determination of Self-Routing Parameters (i, j, k)

Determination of self-routing parameters (i, j, k) by assignment circuit for an incoming packet is shown by the flowchart in figure 3.1 [32]. The symbols used in figure 3.1 are described as follows:

- d is the packet's output-port destination which resides in the header portion of the incoming packet.

- j_d is the assigned output slot vector (OSV) in the global memory space for an incoming packet destined to output-port d.
- k_d is the assigned scan plane in the global memory space for an incoming packet

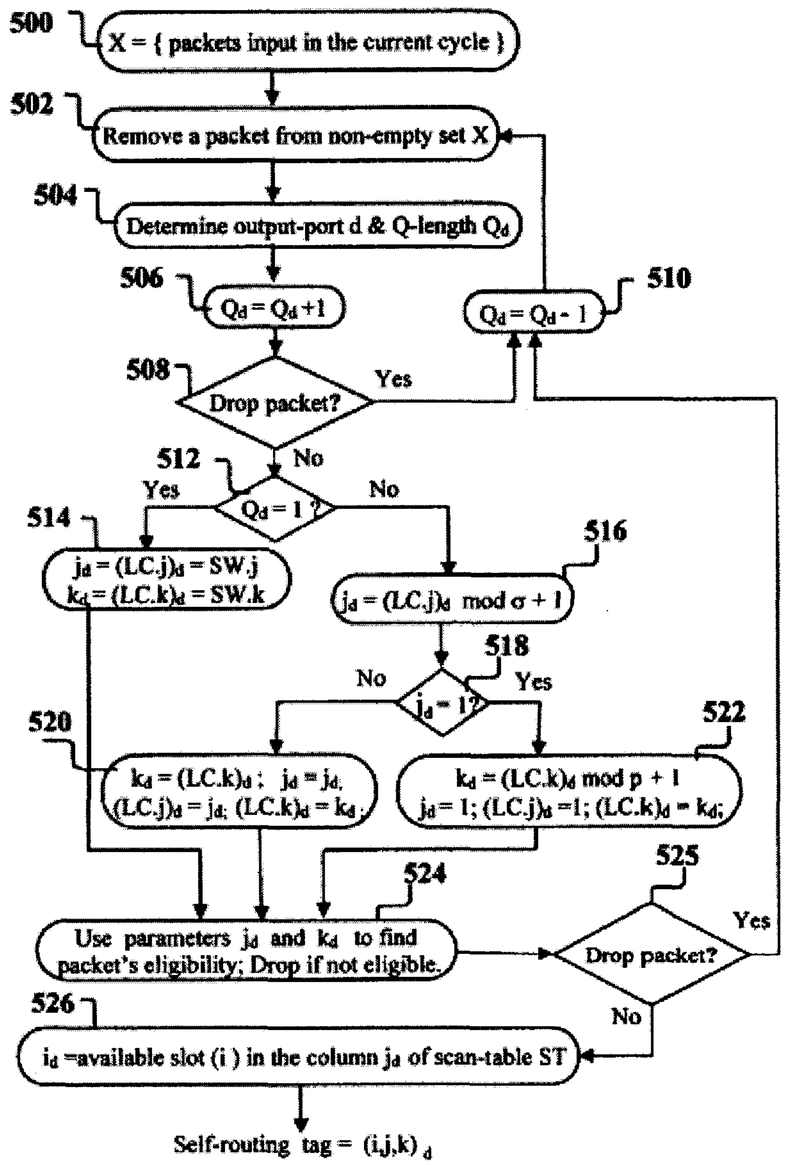

500
X = { packets input in the current cycle }
502
Remove a packet from non-empty set X
504
Determine output-port d & Q-length Qd
506
Qd = Qd + 1
510
Qd = Qd - 1
508
Drop packet?
Yes
No
512
Yes
Qd = 1 ?
No
514
jd = (LC.j)d = SW.j
kd = (LC.k)d = SW.k
516
jd = (LC.j)d mod σ + 1
518
No
jd = 1?
Yes
520
kd = (LC.k)d ; jd = jd,
(LC.j)d = jd; (LC.k)d = kd;
522
kd = (LC.k)d mod p + 1
jd = 1; (LC.j)d = 1; (LC.k)d = kd;
524
525
Use parameters jd and kd to find packet's eligibility; Drop if not eligible.
Drop packet?
Yes
No
526
id = available slot (i) in the column jd of scan-table ST
Self-routing tag = (i,j,k) d

Fig. 3.1 Assignment process for self-routing parameters (i , j, k) [32]

destined to output-port d.

- i_d is the assigned memory slot in the assigned OSV, j_d above. i_d designates one of the parallel memory modules.
- σ is the maximum number of OSVs present on the scan planes of the global memory space.
- p is the maximum number of scan planes in the global memory space.
- Q_d is the queue length inside the global memory space for the packets destined to output-port d.
- X is a set of data packets input during a given switch cycle, $0 \leq |X| \leq N$, where N is the number of input ports.
- $(LC.j)_d$ and $(LC.k)_d$ are j & k values for the last packet admitted in the global memory space for destination port d.

The assignment circuit and flow diagram in figure 3.1 use a set of counters and tables to facilitate determination of self-routing parameters. Figure 3.2 provides one of the several possible ways to implement self-routing parameter assignment circuit of the packet switching system. To enable faster assignments, parameter assignment circuit uses two separate processors. The first processor receives the destination address of the incoming packets from the header processing and uses the flowchart shown in figure 3.1 to assign j and k parameters. While processor 2 works to find ith parameter for a packet in step 526 of figure 3.1, the processor 1 works in parallel to determine j and k parameters for the next packet. In effect, processor 1 and processor 2 of figure 3.2 work in pipeline fashion to determine j, k parameters, and the corresponding ith parameter for incoming packet in a given cycle.

Once the packet qualifies to enter the assigned Output Slot Vector OSV j, then its j and k parameters are forwarded to processor 2 for determination of parameter i in the two-dimensional array called scan table (ST). The slots of the ST are designated by ST(i,j), wherein i denotes the memory module and j denotes an OSV. The row number i of an empty slot (i,j) in the pre-assigned column j of the ST is assigned as the parameter i of the self-routing tag. The parameter i can have a value from 1 to m, where m is the number of memory modules. The parameter j can have a value from 1 to σ, where σ is the number of packet locations in the memory modules. The value of ST(i,j) holds the

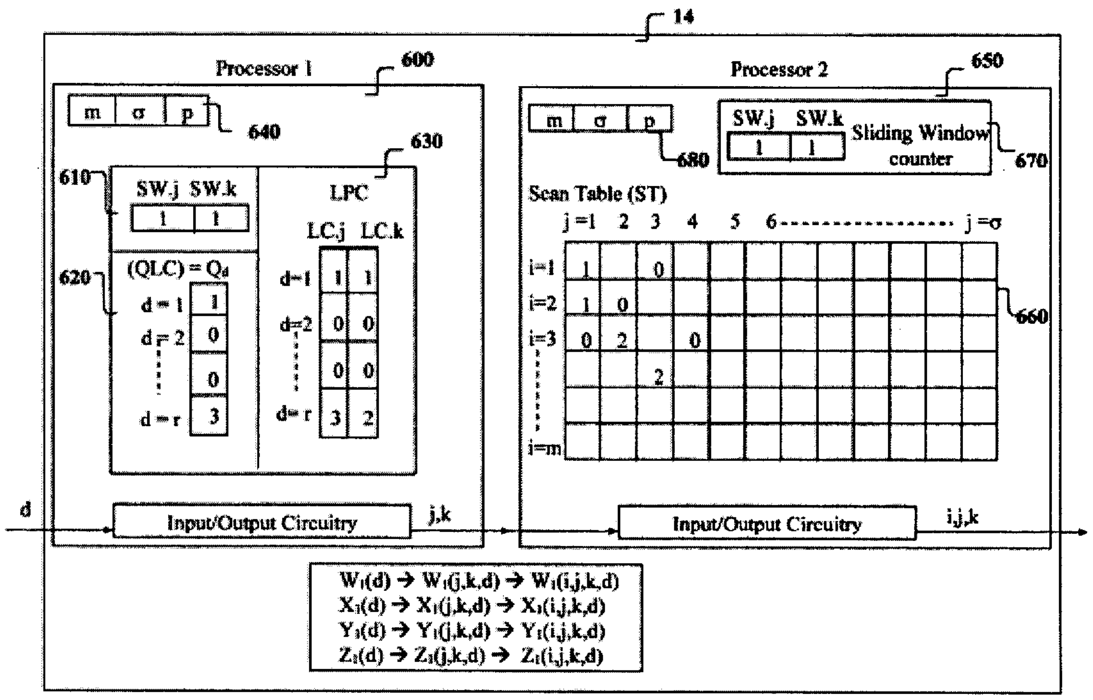


Fig 3.2 A self-routing assignment circuit [32]

scan variable k of the packet, ST(i,j) = k, with k > 0 indicates that the jth location of the ith memory module holds a valid packet whose scan-plane value is k. Whereas, ST(i,j) = 0 indicate that the jth location of the ith memory module in the global memory space is empty and does not hold a valid packet.

3.3 Parameter Assignment Circuit of SW Switch

Each packet assigned parameters for self-routing (i,j,k). According to SW algorithm [32], first j, k values are calculated and then (j, k) values are used to calculate i-value. It is possible to use known basic hardware components to design PAC that can easily handle 500 OC-3. It is possible to design various parallel assignment circuit hardware components such as parallel comparators, multiplexers, encoders / priority encoders to build fast special purpose hardware for PAC. Special purpose hardware with known basic components [35-36] or CMOS ASIC can speed-up the assignment process in PAC. However, due to lack of synthesis resources, in this thesis we investigate PAC performance with general purpose Pentium-4 processor [34], and provide performance estimate for PAC when it is implemented in hardware using known hardware component.

3.3.1 Assignment Process for Parameters j, k

In this section, each process for determination of j and k parameters is investigated to implement on an Intel Pentium 4 General-Purpose Processor. The reason for implementing on Pentium 4 processor is that instructions and latencies are publicly known [34]. Those latencies are used to determine the speed of parameter assignment circuit and the number of lines that can be supported.

Processor 1 receives the destination address of the incoming packet from header processing circuit and uses steps from 500 to 525 in figure 3.1 [32] to assign j and k parameters. In order to find processing-time that it takes, from step 500 to 525 to assign j and k parameters, the longest path (worst case) should be tracked. Longest path is going to be steps 504, 506, 508, 512, 516, 518, 522, 524, 525 in figure 3.1.

It is assumed that all the counters of processor 1 (SWC-Sliding-window counters, QLC-Queue Length Counter of each Queue, LPC-Last Packet Counter) shown in figure 3.2 are kept in the cache memory of Intel Pentium 4 processor. Step 500 shows the initial state, where X packets are input in a given cycle through the incoming ports. Step 502 shows removal of a packet for the purpose of determining its destination information as output port d in step 504. Following instructions will be performed in processor 1.

- The queue length counter (QLC) in figure 3.2 helps determine the queue length Q_d for a packet destined to output port d in step 504 of the flowchart in figure 3.1. Determination of the output port and getting Q_d-length process in step 504 takes 2.5 clock cycles. It is assumed that all the routing tables corresponding output port d and Q-lengths are kept in cache memory of the Pentium 4 processor (instruction: MOV Register <- [Memory]).

- Step 506 increments the value of Q_d to take into account the new arrival. Increment of the Q_d process in step 506 takes 1 clock cycle. Because Q_d-length has already been stored in one of the register of Pentium 4 processor in previous process. (instruction: INC Register,1)

- According to step 508, if ($Q_d > p.\sigma$), then the packet destined to port d is dropped and the Q_d value is decreased by one in step 510 and the assignment process loops back to step 502 to process another packet. Here, $p.\sigma$ is a predetermined upper limit imposed on the length of an output queue inside the global memory space. If the Q_d is bigger than Q_{max} ($p.\sigma$), packet should be dropped in step 508 figure 3.1. Therefore, comparison has to be done. Decision of dropping packet or not is basically compare operation which takes 0.5 clock cycle. (instruction: COMP Q_d, Q_{max})

- In step 512, the queue length of a given destination port is compared. If $Q_d = 1$, then it means it is the only packet for a given destination port "d" in the global memory space and it need not wait as there are no other packets waiting for that destination port. In such case, step 514 is followed according to which the OSV and the scan plane value of the current location of the SW counter 610 is assigned as j and k parameter for the incoming packets. However, step 514 will not take account to find longest path scenario because step 514 is not in the longest path. Therefore, comparison has to be done in step 514. Compare operation is basically takes 0.5 clock cycle (COMP Q_d, 1). Then Q_d value should be updated in the cache memory, which takes 2.5 clock cycles (instruction: MOV [Memory] <-Register).

- According to step 516, $j_d = (LC.j)_d \bmod \sigma + 1$ which means consecutive OSV, i.e., OSV following destination's last packet's OSV is assigned as the j parameter for the incoming packet. If $j_d = 1$ in step 518, then the assigned OSV is on a new scan plane and the scan plane value assigned to the incoming packet is increased by 1, in step 522 as $kd = (LC.k)_d \bmod p + 1$. On the contrary, if $j_d \neq 1$, then the assigned OSV is on the same scan plane as the last packet admitted for that destination's output queue and the same value of LC.k from the counter 630 in figure 3.2 is assigned as the k parameter for the incoming packet in step 520 of the flowchart of figure 3.1. Mod σ operations can be performed by a counter that counts up to σ, such as 8-bit counter. First 2 most significant bits can be assigned to k_d, least 6 significant bits can be assigned as j_d for each destination port. And this 8-bit counter assumed to be kept in the cache memory of Pentium processor for each destination port. Then that counter should be updated in the cache memory. Therefore, step 516 takes total of 6 clock cycles (instructions: MOV Register <- [Memory]; INC Register,1; MOV [Memory] <- Register).

- There is no need to perform step 522 in figure 3.1 because all j_d and k_d values are already assigned in previous process.
- In step 525, if r output ports have $Q_d < \sigma$, then only $(m - r)$ slots in OSV(j) can be occupied by the packets of other ports of which $\sigma \leq Q_d \leq p.\sigma$. A counter r is used to count the number of output ports with $Q_d < \sigma$. An array ES(j) is used to indicate number of empty slots for OSV(j) in ST(i,j), where $j = 1$ to σ. If a packet is qualified in step 525, then it is admitted to proceed the next step 526, else the packet is dropped in step 525. These processes take total of 8.5 clock cycles (instructions: COMP register, σ ; COMP register, Q_{max} ; MOV register <- [ES(j)] COMP register, r ; DEC register, 1 ; MOV [ES(j)] <- register ; DEC register, 1).

By now in the flowchart in figure 3.1, an incoming packet destined its d, has obtained two out of its three routing parameters, i.e., for OSV as j_d and the scan plane k_d. Once j and k values determined, then the processor 1 sends j and k parameters to processor 2 (shown figure 3.2) for determination of the parameter i.

In order to find processing-time it takes to assign j and k parameters for an incoming packet, all latencies should be added from step 500 to 525. Total of 21.5 clock cycles are needed to find j and k parameters. Instructions and latencies are summarized in table 3.1.

Table 3.1 Summary of instructions used in processor 1

Step No	Instructions	Latencies (clock cycles)
504	MOV	2.5
506	INC	1
508	COMP	0.5
512	COMP, MOV	3
516-522	MOV, INC, MOV	6
524-525	COMP, COMP, MOV, COMP, DEC, MOV, DEC	8.5
		Total = 21.5

Intel Pentium 4 General Purpose Processor is assumed to assign j, k parameters for processor 1. Processor's clock rate is taken 3.2 GHz so that one-clock cycle 0.3125 ns. Processor 1 requires maximum total of 21.5 clock cycles to assign j and k parameters. One packet (typical 64-byte, 64 x 8 = 512 bits) transmission takes 3.3 μs for each OC-3 (155 Mbps line-speed) interface. So, the number of lines (N) that can be supported can be calculated as following,

$$N \times (21.5 \times 0.3125\times10^{-9}) \leq 3.3\times10^{-6}, \qquad (1)$$

$N \leq 498$ (with Pentium-4)

N, the number of maximum input lines is approximately 498 solving formula 1. That means Intel Pentium 4 processor with 3.2 GHz clock rate can assign j and k parameter to total 498 OC-3 lines of incoming packets using flowchart in figure 3.1 in one input cycle.

3.3.2 Assignment Algorithms for Parameter i

The assignment of parameter i is done for the incoming packets based on known values of output slot vector j and scan-plane k that has been determined earlier using self-routing parameter assignment scheme [32]. Three different algorithms have been proposed for assignment of parameter i to an incoming packet.

3.3.2.1 Parameter-i Assignment Algorithm 1

Algorithm 1 is shown in figure 3.3. A packet is received from the processor 1 to be assigned i parameter. The near value of i is obtained by increasing the previously assigned i value to previous packet 1 in MOD fashion. If ST(i,j)=0 for near value i, then that means the memory location (i,j) is available and that location is assigned. The last i value, which is assigned for previous packet in the scan-table is increased by 1 in MOD m fashion. Then, the value of ST(i,j) is compared with zero for new value.

If the ST(i,j) is zero, means that the j^{th} location of the i^{th} memory module in the global memory space is empty and does not hold a valid packet. The next step will be assigning the new value of i along with j, and k parameters to build a self-routing parameters $(i, j, k)_d$ for the incoming packet.

If the ST(i,j) is not zero, means that the j^{th} location of the i^{th} memory module holds a valid packet whose scan-plane value is k. Then i value will be increased by 1 in

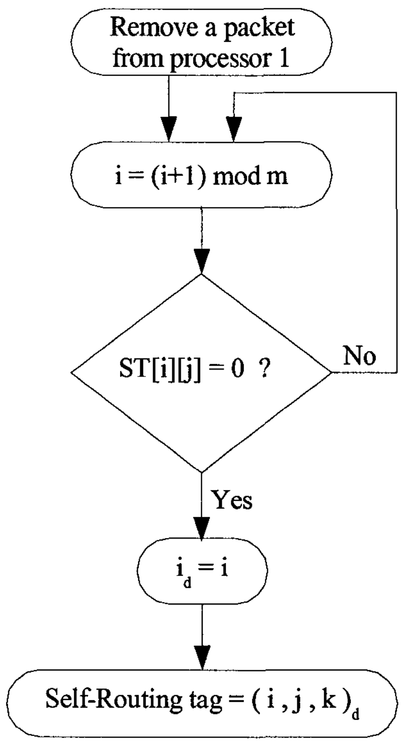

Remove a packet from processor 1
i = (i+1) mod m
ST[i][j] = 0 ?
No
Yes
i_d = i
Self-Routing tag = (i , j , k)_d

Fig. 3.3 Flowchart of algorithm 1

MOD m fashion and compare if the value of ST(i,j)=0. This process is repeated until the jth location of the ith memory module in the global memory space is empty.

The packets belonging to the same input cycle are assigned different values of i (i.e. the ith memory) module in an increasing order. As an example, if $i = 3$ has been assigned to the first incoming packet of the cycle then for the next incoming packet an attempt is made to assign $i > 3$ for that assigned OSV j, if none of the greater values of i are available for that OSV j then only the smaller values of i are chosen in that given OSV j in a MOD fashion.

It is assumed that all the counters of processor 2 (SWC-Sliding-window counters, ST-Scan Table) shown in figure 3.2 are kept in the cache memory of Intel Pentium 4 processor. Mod m operation (shown in figure 3.3) can be performed by a counter that counts up to m and kept one of the general-purpose register, such as 4-bit counter. Determination of ST(i,j) slot is either empty or not process can be found out with a comparison. If it is 0, then the current i assigned as the parameter i of the self-routing tag, If it is not 0, then the assignment process loops back to mod operation until empty slot found. The number of loops back can be maximum of the number of memory module. Because step 525 in figure 3.1 ensures there is available empty slot in current OSV.j.

Therefore, total number of clock cycles to assign i parameter for algorithm 1 is depended on the number of memory modules (m). Hence, algorithm 1 requires maximum total of (1.5 clock cycles) x (number of memory modules), which is 1.5 x m clock cycles

Table 3.2 Summary of instructions used in processor 2 for algorithm 1

Step	Instructions	Latencies (clock cycles)
i = (i +1) mod m	INC	1
ST [i][j] = 0 ?	COMP	0.5
		Total = 1.5

to assign i parameter for an incoming packet. (instructions: INC register, 1 ; COMP [ST(i,j)], 0). Summary of instructions and latencies for algorithm 1 are shown in table 3.2. One packet (64-byte) transmission takes 3.3 μs for each OC-3 (155 Mbps line-speed) interface. So, the number of lines (N) can be found following formula,

$N \times (1.5 \times 0.3125 \times 10^{-9} \times m) \leq 3.3 \times 10^{-6}$, (2)

m, number of deployed memory modules is given 2N,

$N \times (1.5 \times 0.3125 \times 10^{-9} \times 2N) \leq 3.3 \times 10^{-6}$, (3)

$N \leq 60$ (with Pentium-4)

N, the number of maximum input lines is approximately 60 solving formula 3. That means Intel Pentium 4 processor with 3.2 GHz clock rate can assign i parameter to total 60 lines of incoming packets using algorithm 1 (shown in figure 3.3) in one input cycle.

It is possible that some of the incoming packets in a given cycle may be assigned to same memory module using algorithm 1. The algorithm does not examine the pre-assigned i values for packets arriving in a given cycle. This situation can cause multiple packets to be stored in the same memory module and thus increasing the number of memory cycles for storage and hence the memory bandwidth requirement. In order to decrease the number of packets assigned in one memory module, called the memory bandwidth requirement, i parameter assignment algorithm 2 is developed and discussed next section.

3.3.2.2 Parameter-i Assignment Algorithm 2

Algorithm 2, shown in figure 3.4, uses more parameters and variables than algorithm 1. A new variable, FAS (first available slot) is used to keep available first memory module to store the packet. Once FAS is set to the ith memory module, FAS will not change during the parameter i search for an incoming packet. The reason for having FAS is that if all available memory slots on different memory modules are pre-assigned to previous packets then assignment has to be made to first available slot in a given OSV j without new search. In that case there will be additional memory cycle in a given cycle. Counter will assist to detect that kind of case. Counter will be increased by 1 when the current memory module is pre-assigned, and when the current memory slot holds valid packet. Each time counter will be compared to m if it has searched all the memory modules to find available memory slot in a given OSV j. If counter has reached the value of m then value of FAS will be assigned to parameter i.

PO[i] array holds the pre-occupied memory modules in a given cycle. If the value of PO[i] is 1, that means i^{th} memory module pre-assigned by previous packets in a given cycle. If the value of PO[i] is zero, that means i^{th} memory module is not assigned by previous packets in a given cycle.

Algorithm starts with receiving a packet from processor 1. FAS and counter variables is set to zero. The last i value, which is assigned for previous packet in the scan-table, increased by 1 in MOD m fashion. Then, the value of ST(i,j) is compared with zero. If the ST(i,j) is not zero, means that the j^{th} location of the i^{th} memory module holds a valid packet whose scan-plane value is k. Then counter will be increased by 1 and i value will be increased by 1 in MOD m fashion and compare the value of ST(i,j) until the j^{th} location of the i^{th} memory module in the global memory space is empty. That empty space will be the first available slot when the algorithm moves along. If the ST(i,j) is zero, means that the j^{th} location of the i^{th} memory module in the global memory space is

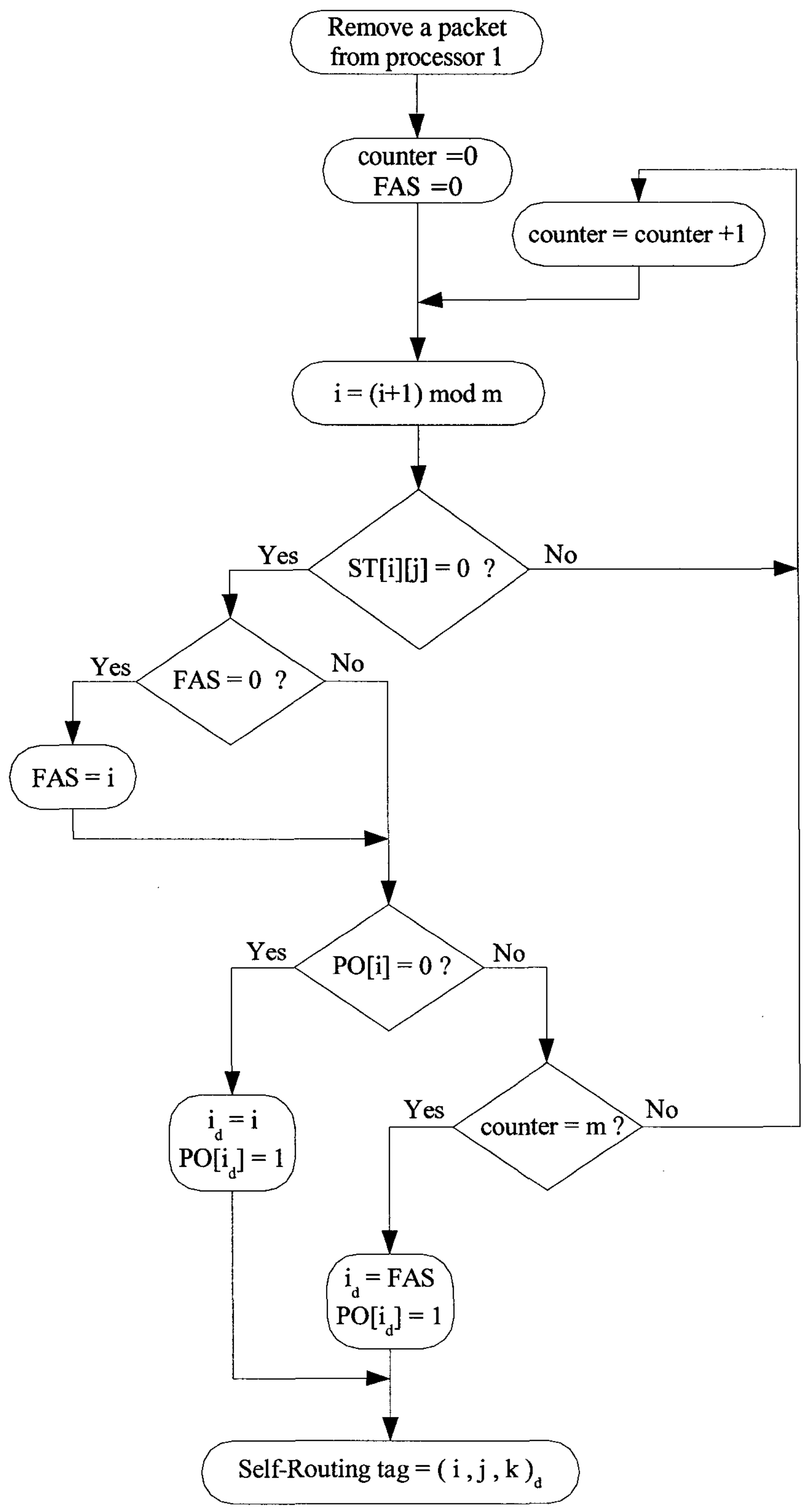


Fig. 3.4 Flowchart of algorithm 2

empty and does not hold a valid packet. Next step will be setting that i value to FAS. Once FAS is assigned to i value, it is not going to change during the algorithm. Then there will be comparison the PO[i] value with zero in order to find out current slot was pre-assigned by previous packets in a given cycle. If PO[i] value is zero then the j^{th} location of the i^{th} memory module in the global memory space is empty and does not hold a valid packet and does not pre-assigned. Next step will be assigning the i parameter along with j, and k parameters to self-routing tag $(i, j, k)_d$. If PO[i] value is 1 (that slot was pre-assigned) then the counter will be examined if all memory modules have been searched. If the counter value is m, then i parameter will be assigned to FAS (first available slot). If the counter value is less than m, search for i parameter will continue until empty space will be found.

It is assumed that PO[i] array in figure 3.4 is kept in the cache memory, FAS and Counter in figure 3.4 are kept in two different general-purpose register, and all other main counters of processor 2 (SWC-Sliding-window counters, ST-Scan Table) shown in figure 3.2 are kept in the cache memory of Intel Pentium 4 processor. Following processes (longest path) will be performed to find out service rate of Processor 2.

- Mod m operation (shown in figure 3.4) can be performed by a counter that counts up to m and kept one of the general-purpose register, such as 4-bit counter. Increment operation is basically takes 0.5 clock cycle. (instruction: INC Register,1).
- Determination of ST(i, j) slot is either empty or not process can be found out with a first move ST(i, j) value from the cache and then comparison operation have to be done in this step. Move operation is basically takes 2.5 clock cycle. Then compare

operation takes 0.5 clock cycles (instruction: MOV Register <- [Memory] ; COMP Register, 0).

- In order to find out whether the current memory slot was pre-occupied or not first move PO(i) value from the cache and then comparison operation has to be done in this step. Move operation is basically takes 2.5 clock cycle. Then compare operation takes 0.5 clock cycles (instruction: MOV Register <- [Memory] ; COMP Register, 0).
- If each slot either full or pre-occupied, then there must be a Counter that count up to the number of memory modules. Therefore, compare and increment operations have to be done in this step. Compare operation takes 0.5 clock cycle, Increment operation takes 0.5 clock cycle (instructions: COMP Register, m ; INC Register, 1).

The number of loops back can be the maximum of the number of memory module. Because step 525 ensures there is available empty slot in current OSV.j. Summary of instructions and latencies for algorithm 2 are shown in table 3.3.

Therefore, the number of clock cycles to assign i parameter for this algorithm is depended on the number of memory modules (m). Hence, this algorithm can take

Table 3.3 Summary of instructions used in processor 2 for algorithm 2

Step	Instructions	Latencies (clock cycles)
i = (i +1) mod m	INC	1
ST [i][j] = 0 ?	COMP	0.5
PO [I][j] = 0 ?	COMP	0.5
Counter = m ?	COMP	0.5
++ Counter	INC	1
		Total = 3.5

maximum total of (3.5 clock cycles) x (number of memory modules), which is 3.5 x m clock cycles. One packet (64-byte) transmission takes 3.3 μs for each OC-3 (155 Mbps line-speed) interface. So, the number of lines (N) can be found following formula,

$N \times (3.5 \times 0.3125 \times 10^{-9} \times m) \leq 3.3 \times 10^{-6}$, (4)

m, number of deployed memory modules is given 2N,

$N \times (3.5 \times 0.3125 \times 10^{-9} \times 2N) \leq 3.3 \times 10^{-6}$, (5)

$N \leq 39$ (with Pentium-4)

N, the number of maximum input lines is approximately 39 solving formula 5. That means Intel Pentium 4 processor with 3.2 GHz clock rate can assign i parameter to total 39 lines of incoming packets using algorithm 2 (shown in figure 3.4) in one input cycle.

It is observed that process time of i parameter assignment algorithm 1 and 2 are dependent on switch size which is limitation of SW switch architecture. Switch size cannot exceed 60x60 using parameter-i assignment algorithm 1, 39x39 using parameter-i assignment algorithm 2.

3.3.2.3 Parameter-i Assignment Algorithm 3

In order to improve processing-time for parameter-i assignment, we propose a new algorithm called Queue-based parameter-i assignment algorithm (algorithm 3). Design of Queue-based parameter-i assignment circuit is shown in figure 3.5. Queue-based algorithm does not perform any search to assign i parameter for incoming packet. All available i values for each OSV.j have queued. Therefore, the number of queues is equal to number of memory locations (σ). For instance, there are 16 queues if σ = 16; if the number of memory modules is 8 (m = 8), capacity of each queue is 8.

Queue-based assignment algorithm starts with initialization of queues. After initialization, the value stored in front of queue(j) will be assign as i parameter for each incoming packet. When the packet, stored in i^{th} memory module, sent out to output interconnection network, then i will be added at the end of the corresponding queue(j) meaning that (i, j) memory module slot is available for next incoming packets.

Advantage of this algorithm is that there is no search involved for parameter-i assignment for each incoming packet. Processes that require in order to get an i value from the queue and write-back the parameter i when the packet send out

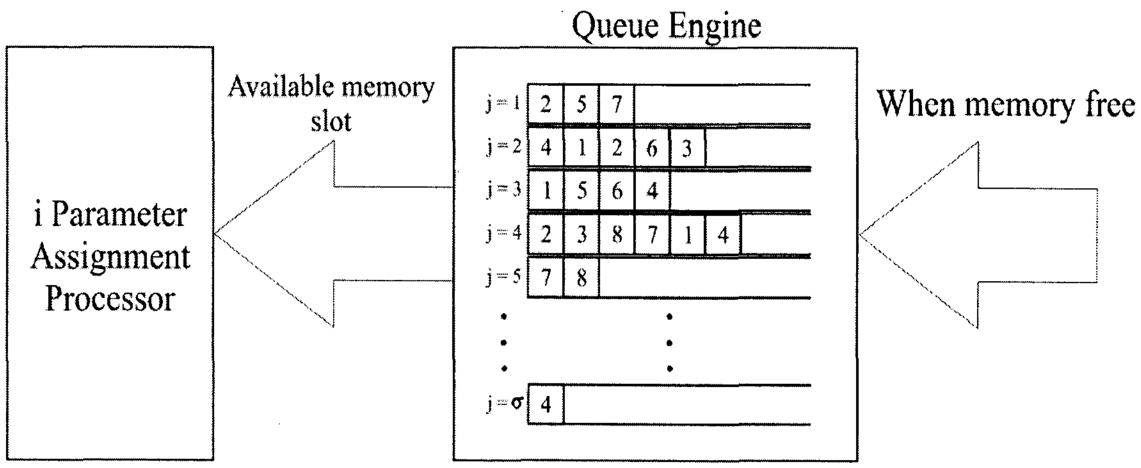


Fig. 3.5 Queue-based i parameter assignment circuit

can be implemented in Intel Pentium 4 General Purpose Processor. Summary of instructions and latencies for algorithm 3 are shown in table 3.4.

Table 3.4 Summary of instructions used in processor 2 for algorithm 3

Step	Instructions	Latencies (clock cycles)
Get an i value from given OSV.j Queue	IN	10.5
Send an i value to given OSV.j Queue	OUT	10.5
		Total = 21

Those processes take 21 clock cycles for each packet. Hence, processor 2 can handle 502 lines in one packet (64-byte) transmission time for OC-3 interface.

Processor 2 requires maximum total of 21 clock cycles to assign i parameter for an incoming packet. One packet (64-byte) transmission takes 3.3 μs for each OC-3 (155 Mbps line-speed) interface. So, the number of lines (N) can be found following formula,

$$N \times (21 \times 0.3125\text{x}10^{-9}) \leq 3.3\text{x}10^{-6}, \qquad (6)$$

$N \leq 502$ OC-3 lines (using Pentium-4)

N, the number of maximum OC-3 input-lines is approximately 502 solving formula 6. That means Intel Pentium 4 processor with 3.2 GHz clock rate can assign i parameter to total 502 OC-3 lines of incoming packets using algorithm 3 in one input cycle.

The only disadvantage of Queue-based assignment algorithm is the average memory bandwidth requirement is higher when it is compared to algorithm 1 and 2 at any switch size. This is because there are no control mechanism and i values for all packets in an input cycle are assigned in probabilistic fashion. This situation can be improved by increasing the memory-speed of parallel modules. This issue will be discussed in Chapter-4.

3.3.2.4 Special Purpose Hardware

It is also possible to use known basic hardware components [35], such as parallel comparators, multiplexers, encoders / priority encoders [36] to build fast special purpose hardware. One possible design way is shown in figure 3.6. The idea of this hardware is to

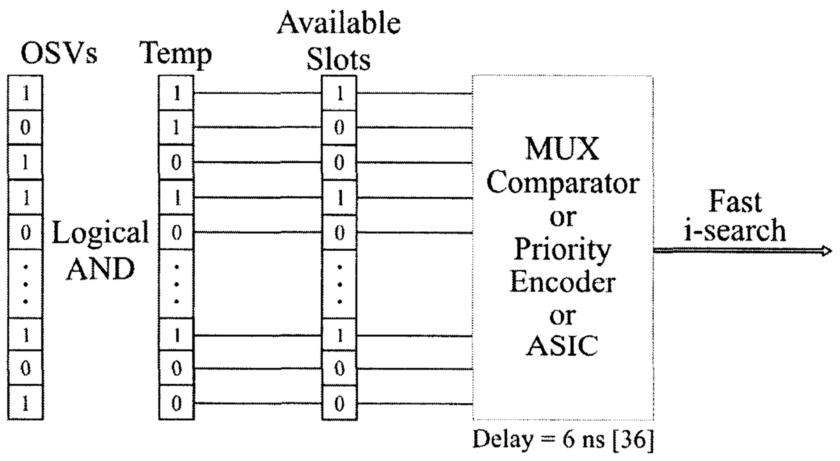


Fig. 3.6 Special purpose parameter-i assignment circuit

keep all OSV vectors in bit-wise which indicates that location is either available or occupied from the previous cycles. Multiplexer or encoder ASIC design will determine only the available slots in parallel fashion and assign those available slots for each incoming packet with estimated 6 ns delay. Therefore each i-value will be assigned for incoming packets in an input cycle in 6 ns. So, the number of lines (N) can be found following formula,

$N \times (6\times10^{-9}) \leq 3.3\times10^{-6}$, (7)

$N \leq 550$ OC-3 lines (using Special purpose hardware)

3.4 Summary of Self-Routing Parameter Assignment Circuit

In this chapter, several possible ways to implement self-routing parameter assignment circuit are described. Processes for determination of j and k parameters [1] are fixed and implemented in Processor 1 of PAC. It is observed that Processor 1 can assign j and k parameters to total 498 incoming packets in one input cycle. Performance of i parameter assignment depends on which algorithm is implemented. Using Pentium-4, 60 input lines can be deployed if algorithm 1 is implemented, 39 input lines can be deployed if algorithm 2 is implemented, and 502 input lines can be deployed if algorithm 3 is implemented. Using hardware-based design, up to 512 lines can be supported. Table 3.5 summaries switch sizes depend on 3 different algorithms. Processor 1 and processor 2 of figure 3.2 work in pipeline fashion to determine j, k parameters, and the corresponding i^{th} parameter for incoming packet in a given cycle. Therefore, switch size and PAC size

Table 3.5 Summary of switch size depends on 3 different algorithms

Parameter Assignment Circuit				Switch & PAC Size (OC-3 lines)
Processor 1 (Pentium-4)	Processor 2			
Pentium-4	Pentium-4		Special Purpose Hardware	
498 OC-3 lines	Algorithm 1	60 OC-3 lines	550 OC-3 lines	60x60
	Algorithm 2	39 OC-3 lines		39x39
	Algorithm 3	502 OC-3 lines		498x498

is determined by the slowest processor; processor 2 for algorithm 1, processor 2 for algorithm 2, and processor 1 for algorithm 3.

The maximum number of OC-3 lines that can be supported by a general purpose Pentium processor = 498 OC-3 lines.

Total capacity = 498 x OC-3 lines = 77 Gbps (using Pentium-4).

Total capacity = 512 x OC-3 lines = 80 Gbps (using specialized known hardware components).

CHAPTER 4

PERFORMANCE OF PARAMETER ASSIGNMENT SCHEMES

4.1. Self-Routing Parameter Assignment Circuit (PAC) of Sliding Window Switch

In this chapter, memory bandwidth of sliding-window packet switch and its performance under traffic with varying burstiness is investigated.

The self-routing parameter assignment circuit uses the sliding-window switching scheme [32] to compute the self-routing parameters (i,j,k) to be attached to the incoming packets. The parameter j in a packets self-routing tag designates the packet location in i^{th} memory module, and parameter k designates a packet's turn to go out of the memory. In this chapter, the self-routing parameter assignment schemes given in chapter 3 are used for determination of parameters i, j and k.

4.2. Parameters of Performance Evaluation

4.2.1. Memory Bandwidth of Sliding-Window Switch

The parameter i denotes the memory module where an incoming packet is stored. The parameter assignment circuit first determines the j and k parameters, and uses the j and k values to determine the value of ith parameter i.e. the memory module (i) where an incoming packet is stored. If packets, arriving in a given input cycle, are assigned to different memory modules then it is possible for all the incoming packets belonging to that cycle to be stored in parallel to different memory modules requiring just one memory cycles. However, due to different traffic patterns and distribution of output destinations among the incoming packets, it is possible that the some of the incoming packets in a given cycle may not be assigned to different memory modules. This will require the packets of an input cycle to be stored in memory modules in more than one memory cycle and hence increasing the memory bandwidth. The memory bandwidth for the sliding-window switch is defined as the number of memory WRITE cycles needed to store incoming packets in an input cycle. (The average memory bandwidth of a given sliding-window packet switch and its performance under traffic with varying burstiness is measured.)

4.2.2. Ratio of Multiple Packets Stored in 1 Memory Module

Number of memory-write cycles in an input cycle can be determined by the maximum number of packets is assigned to in one memory module. Ratio of multiple packets in 1 memory module is given by,

Ratio of multiple packets going to same memory module =

$$\frac{\text{Number of instances of multiple packets going to same memory module}}{\text{Total number of packets input}} \tag{1}$$

If ratio is 0, that means there is no memory conflict that all incoming packets are assigned to different parallel memory modules. If ratio is greater than 0, there are some memory conflicts that some of the incoming packets are assigned to same memory module. Ratio is always less than 1 because it is not possible that all incoming packets are assigned to same memory module.

4.2.3. Bursty Traffic Model

To study performance of memory-bandwidth for the switching system, a bursty traffic is generated using a two state ON-OFF model i.e. by alternating, a geometrically distributed period during which no arrivals occur (idle period), by a geometrically distributed period during which arrivals occur (active period) in a Bernoulli fashion and vice versa figure 4.1.

If p and r characterize the duration of the active and idle period respectively, then the probability that the active period lasts for i time slots is given by,

$P(i) = p(1-p)^{i-1}$ for $i \geq 1$ (2)

and the corresponding average burst length is given by,

$E_B[i] = 1 / p$ (3)

Similarly, the probability that the idle period lasts for j time slots is given by,

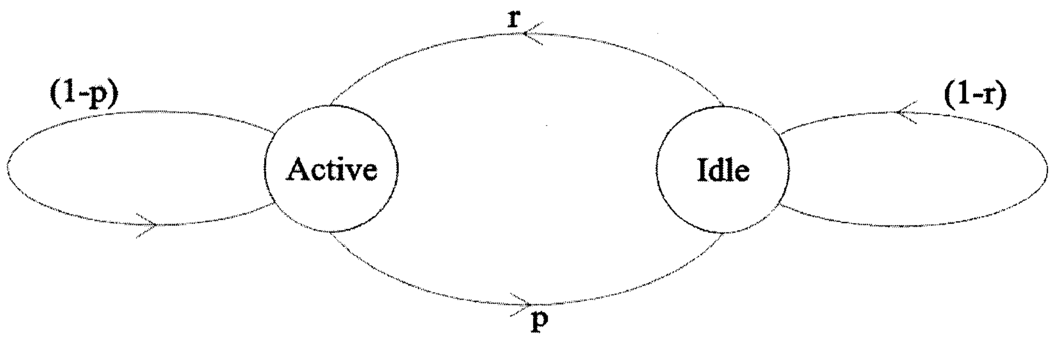


Fig. 4.1 A two-state ON-OFF model

$R(j) = r(1-r)^j$ for $j \geq 0$ (4)

and corresponding mean idle period is given by,

$E_I [j] = (1 - r) / r$ (5)

Hence, for a given p and r, the offered load L is given by,

$L = E_B [i] / (E_B [i] + E_I [j]) = r / (r + p - r.p)$ (6)

4.2.4. Simulation Setup

The measures of interest considered in the simulation studies are the offered load for a bursty traffic of a given average burst length (ABL), and memory bandwidth of the memory modules required to store incoming packets for switching purposes. The simulation experiments started out with empty memory modules and the incoming bursts of packet were uniformly distributed to all the outputs. The switch size considered for the sliding-window switch for this evaluation was 4x4, and then switch size increased by 2 times, i.e. 8x8, in order to have more result. Depending on the offered load, first a maximum of 4×10^6 packets were generated for evaluation of the memory-bandwidth of the switch. Then number of generated packets was increased depending on the different switch sizes. Three different types of bursty traffic were generated. First bursty traffic had an average burst length (ABL) of 8 packets, second bursty traffic had an ABL = 16 packets, and the third bursty traffic had an average burst length (ABL) of 32 packets. The number of scan-length $\sigma = 16$, the number of scan-planes $p = 2$. Different numbers of memory modules (m) deployed for the experiments, first m = 6 modules ($m = 2N - 2$), second m = 8 modules ($m = 2N$), and the rest depends on the switch size. In this simulation experiments, the self-routing parameter assignment scheme [32] is used to first determine j and k values for the incoming packets and then the j and k values are used to determine the i th parameter using two different search algorithm.

4.3. The Effect of Different Burst Lengths on Switch Performance

The plot shown in figure 4.2 is the simulated Average Memory-Bandwidth Required versus Offered Load using algorithm 1. First bursty traffic had an average burst length (ABL) of 8 packets (shown blue solid line), second bursty traffic had an ABL = 16 packets (shown green dash line), and the third bursty traffic had an average burst length (ABL) of 32 packets (shown stared red line). A maximum of $4\text{x}10^6$ packets were generated for this experiment of the memory bandwidth of the switch. The switch size is 4x4. The number of memory modules is 6, The number of scan-length $\sigma = 16$, the number of scan-planes $p = 2$.

In the simulation, it is seen that multiple packets stored in one memory module, which requires faster memory-write compared to line speed. It is possible that 3 packets (worst case) can be stored in one memory module shown in table 4.1. That means memory-write speed has to be 3 times faster than the line speed so that 3 packets can be written in one write cycle. The effect of different burstiness (8, 16, 32 packets) on average memory bandwidth required versus offered load is depicted when 8 memory modules deployed to the sliding window switch in figure 4.3. It is also possible that 3 packets (worst case) can be stored in one memory module shown in table 4.2. That means memory-write speed has to be 3 times faster than the line speed so that 3 packets can be written in one write cycle. It is observed that average memory bandwidth requirements in two figures (figure 4.3 and 4.4) are almost same up to 85 % load. When the load is higher

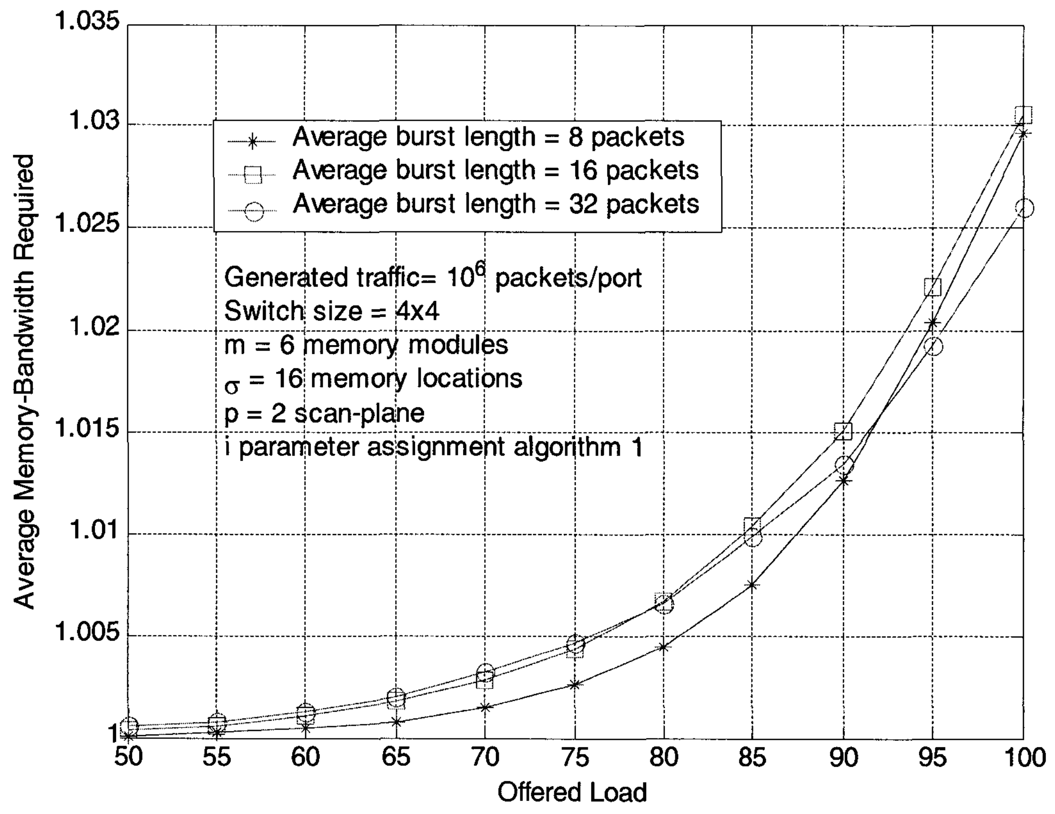


Fig. 4.2 Average Memory-Bandwidth Required vs. Offered Load at 4x4 switch with 6 memory modules using i parameter assignment algorithm 1

Table 4.1 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.2

Average Burst Length	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
8 packets	7.8×10^{-3}	0	0	0
16 packets	7.7×10^{-3}	5×10^{-7}	0	0
32 packets	6.7×10^{-3}	2.5×10^{-7}	0	0

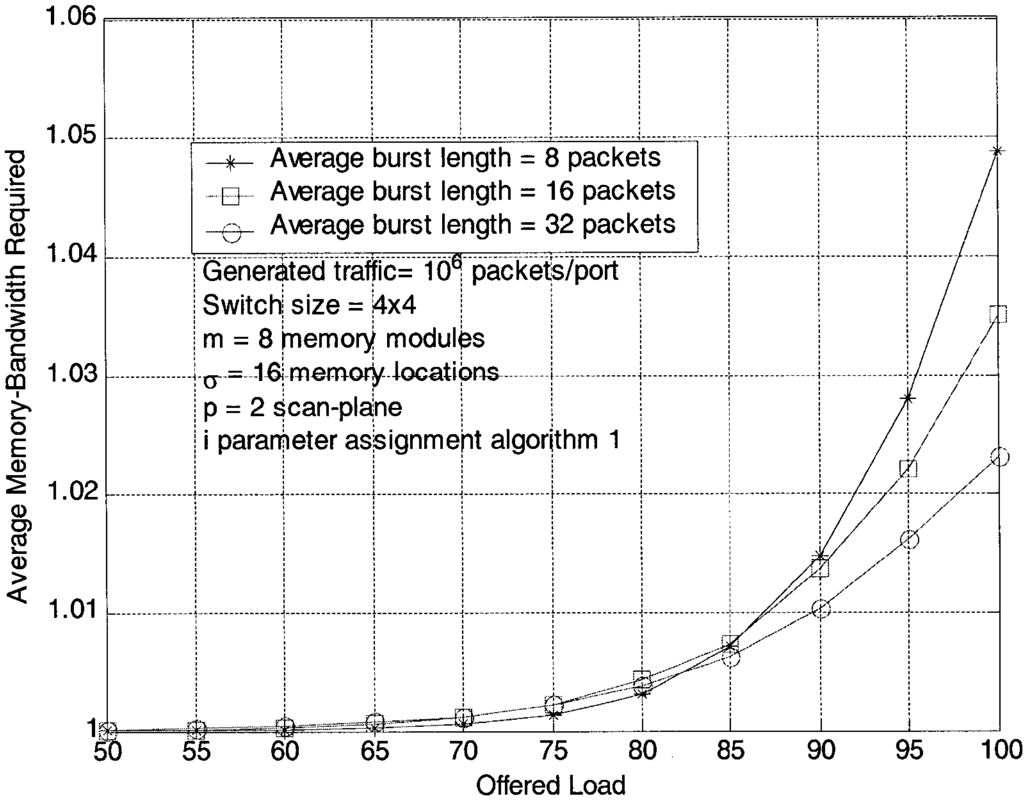


Fig. 4.3 Average Memory-Bandwidth Required vs. Offered Load at 4x4 switch with 8 memory modules using i parameter assignment algorithm 1

Table 4.2 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.3

Average Burst Length	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
8 packets	1.2×10^{-2}	1×10^{-6}	0	0
16 packets	8.5×10^{-3}	2.5×10^{-7}	0	0
32 packets	6×10^{-3}	0	0	0

than 85 %, the average memory bandwidth differentiated in figure 4.3. This can be explained by the way packets are stored in parallel memory modules of the switch.

Traffic with higher burstiness brings longer burst of packets all destined to same output port. The self-routing parameter assignment scheme [32] assigns successive OSVs for successive packets of an output queue. Furthermore, the assignment for parameter i for packets of an input cycle is such that they are allocated to different memory modules. This causes the packets in longer bursts to be stored diagonally on a two-dimensional scan-plane in the memory space of switching system. This is true with the packets belonging the smaller bursts; however diagonal storage footprint will be much smaller than the traffic with higher burst length. The diagonal storage of packets can be done more readily in parallel, requiring only one memory cycle. Hence, the traffic with higher burstiness produces larger diagonal storage-footprints that mostly require only one memory cycle for packets arriving in one cycle, and hence requiring smaller average memory bandwidth.

Traffic with smaller burstiness has more occurrences of patterns that do not have diagonal storage-footprints. This is because the smaller bursts of packets brought in successive cycles are more likely to belong to different output-ports and many of which will be required to be stored in the current OSV or near current OSV [37]. This will cause assignment of multiple packets arriving in the same cycle to acquire slots in the same memory module and thus increasing the average memory-bandwidth for traffic with lower burstiness.

These scenarios happens mostly memory deployment is twice as number of input (or output) which is m=2xN. Because Control Operations for Full utilization of Output

Ports scheme [32], called OSV restriction, has been implemented using parameters j, k and depending on the growth of output queues inside the shared memory switch. This scheme provides full utilization of output ports by allowing packets of output ports to always find memory slots even when backlog occurs during burstiness. This control operations force some packets to be dropped if those packets are not eligible for maximum utilization purposes.

When 6 memory modules have been deployed to sliding window switch, OSV restriction algorithm drops so many packets because of lack of enough memory space in the shared memory, causes almost same average memory bandwidth in both smaller burstiness and higher burstiness.

When 8 memory modules (twice as number of the input lines) have been deployed to sliding window switch, OSV restriction algorithm will not drop any packet because there will be enough memory space in the shared memory switch and provide full utilization of output ports by allowing all packets of output ports to always find memory slots even when backlog occurs during burstiness.

Simulation of average memory-bandwidth required versus offered load using algorithm 2 is shown in figure 4.4. The effect of OSV restriction algorithm is same with the algorithm1 when 6 memory modules are deployed to sliding-window switch system. However, average memory bandwidth of the sliding-window switch has decreased from1.03 to 1.0045 for different burstinesses using algorithm 2. That is the impact of keeping pre-assigned (pre-occupied) memory modules and using them to minimize the average memory bandwidth.

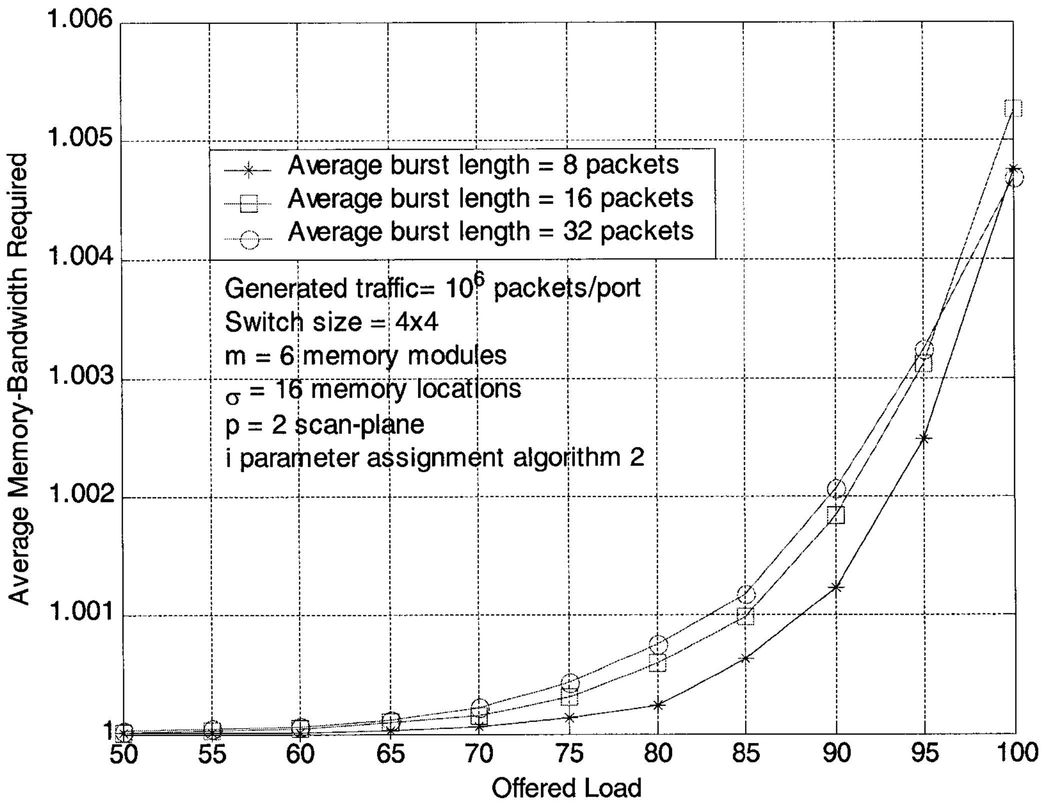


Fig. 4.4 Average Memory-Bandwidth Required vs. Offered Load at 4x4 switch with 6 memory modules using i parameter assignment algorithm 2

Table 4.3 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.4

Average Burst Length	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
8 packets	1.2×10^{-3}	0	0	0
16 packets	1.3×10^{-3}	0	0	0
32 packets	1.3×10^{-3}	0	0	0

Algorithm 2 minimizes not only average memory-bandwidth requirement, but also memory-write speed compared to algorithm 1. It is experienced that 2 packets can be assigned in one memory module in one input cycle shown in table 4.3. Assignment of 2packets in one memory means memory-write speed has to be 2 times faster than the line speed.

Higher average memory-bandwidth for traffic with lower burstiness is depicted in figure 4.5 when the number of memory modules is 8. Very similar phenomena regarding the traffic with smaller burstiness has higher average memory bandwidth is observed in figure 4.3 and 4.5 [39].

Table 4.4 also shows frequencies of multiple packets stored in one memory module at 100 % load traffic using algorithm 2.

Simulation of average memory-bandwidth required versus offered load using algorithm 3 (Queue-based) is shown in figure 4.6. The effect of different burstinesses to sliding-window switch system using algorithm 3 is the same as other algorithms. However, the pattern of average memory bandwidth requirement is completely different than the other algorithms. It is also observed that average memory-bandwidth requirement is much more algorithm 3 than algorithm 1 and 2. Table 4.5 shows that frequencies of having multiple packets assigned in one memory module using algorithm 3 is much more than the other algorithms.

Algorithm 3 increases not only average memory-bandwidth requirement, but also memory-write speed compared to algorithm 1 and 2. It is experienced that 4 packets can be assigned in one memory module in one input cycle shown in table 4.5. Assignment of 4 packets in one memory requires 4 times faster memory-write speed than the line speed.

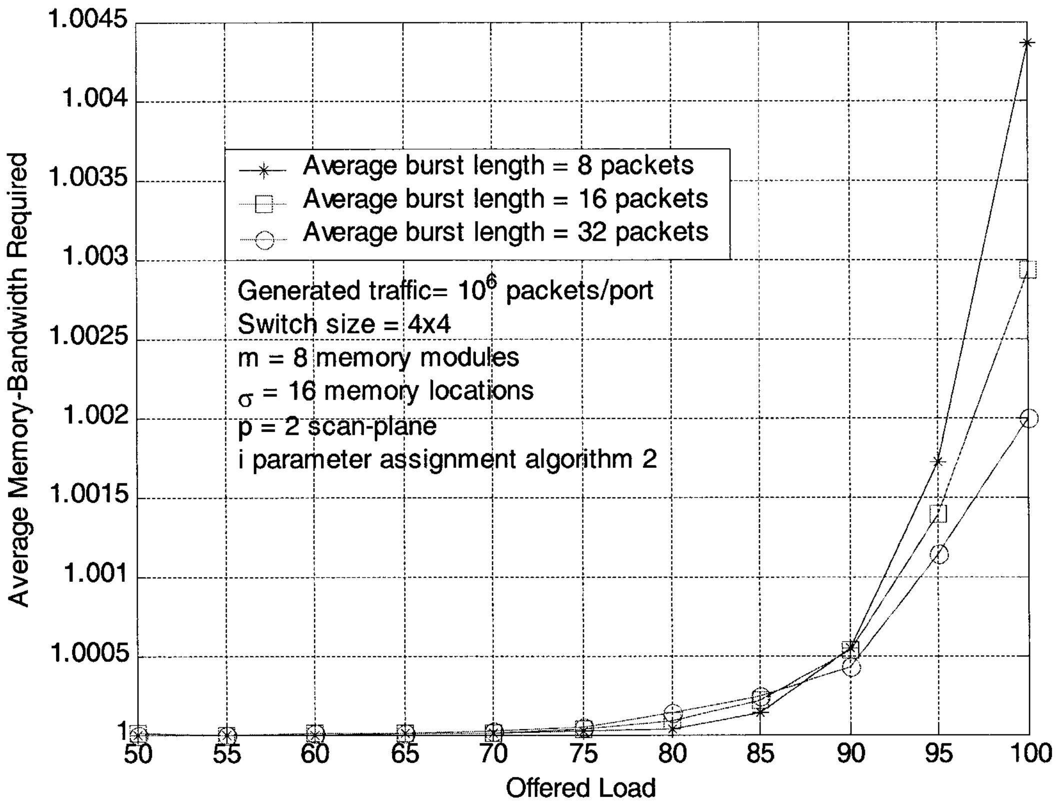


Fig. 4.5 Average Memory-Bandwidth Required vs. Offered Load at 4x4 switch with 8 memory modules using i parameter assignment algorithm 2

Table 4.4 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.5

Average Burst Length	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
8 packets	1.1×10^{-3}	0	0	0
16 packets	7.3×10^{-4}	0	0	0
32 packets	5×10^{-4}	0	0	0

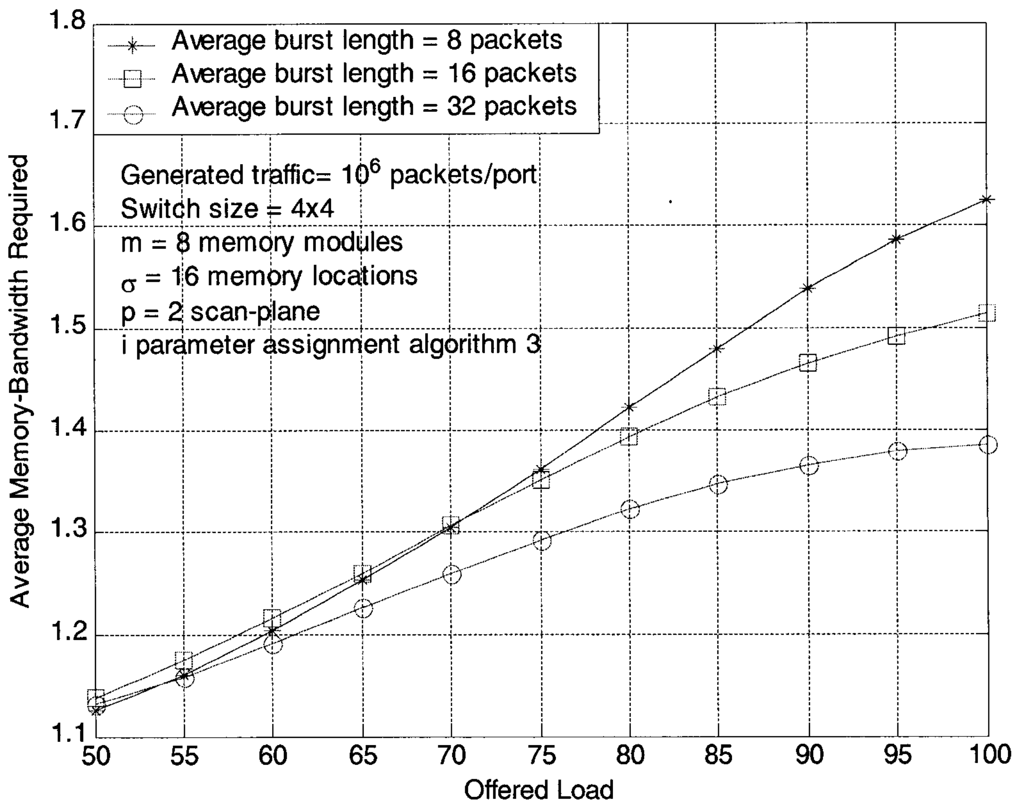


Fig. 4.6 Average Memory-Bandwidth Required vs. Offered Load at 4x4 switch with 8 memory modules using i parameter assignment algorithm 3

Table 4.5 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.6

Average Burst Length	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
8 packets	1.3×10^{-1}	1.6×10^{-2}	9.4×10^{-4}	0
16 packets	1.1×10^{-1}	1.2×10^{-2}	6.1×10^{-4}	0
32 packets	8.4×10^{-2}	7.9×10^{-3}	2.9×10^{-4}	0

The reason behind of this increased average memory-bandwidth requirement and memory-write speed using algorithm 3 can be explained as follows. Assignment of i parameter to an incoming packet using algorithm 1 and 2 is basically consecutive fashion in memory modules. However, assignment of i parameter to an incoming packet using algorithm 3 is random fashion. Because, available i values, which indicates an empty memory modules, for an incoming packet are kept in the one of the Queue for a given OSV.j. It is possible that first value (first element of queue) of each 4 Queues have the same i value (same memory module) for 4 incoming packets. Probability of having 4 packets in one memory module is smaller than the probability of having 3 or 2 packets in one memory module.

4.4. The Effect of Different Switch Size on Memory Bandwidth Requirement

In this section, different i parameter assignment algorithms are used to measure the effects of switch size on memory bandwidth performance of Parameter Assignment Circuit. First, algorithm 1 is simulated for 4x4, 8x8, 16x16, and 32x32 switch size, where 8, 16, 32, and 64 memory modules (m = 2N) are deployed respectively to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory conflicts (8 packets of average burst length has more memory conflicts than 16 and 32 packets of average burst length).

Fig. 4.7 shows that increase in switch size results increase of the memory bandwidth requirement at higher load traffic. Table 4.6 shows that increase in switch size

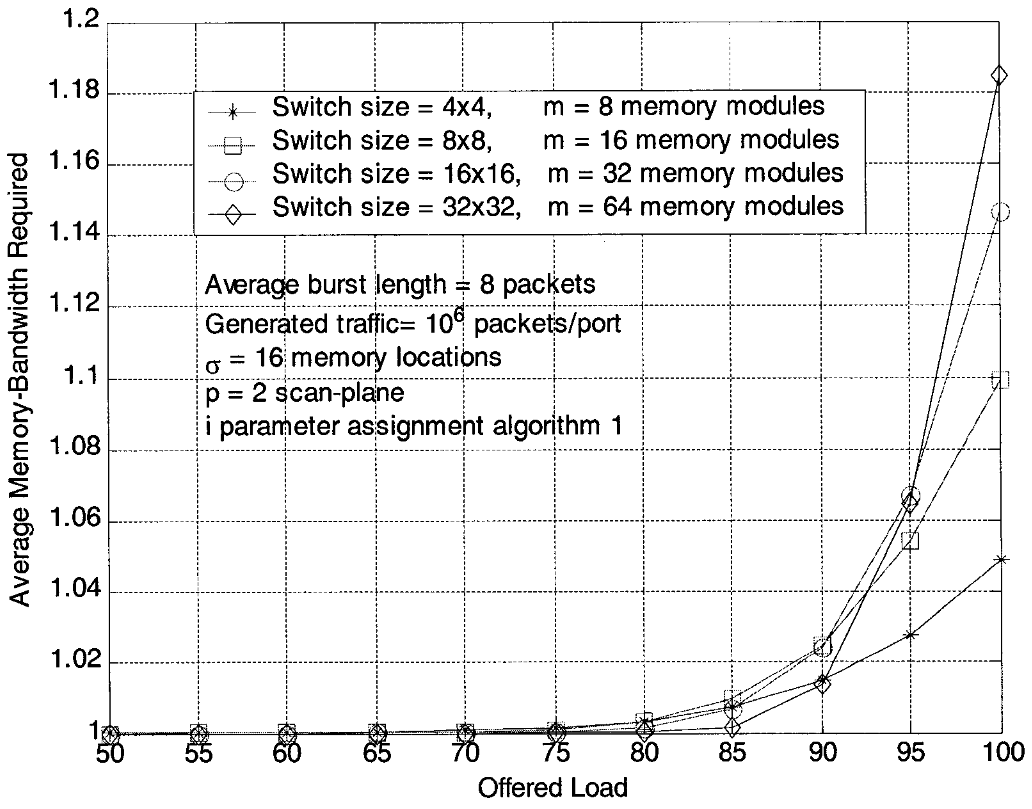


Fig. 4.7 Average Memory-Bandwidth Required vs. Offered Load at 8 packets average burst length traffic using i parameter assignment algorithm 1

Table 4.6 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.7

Switch Size	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
4x4	1.2×10^{-2}	1.3×10^{-6}	0	0
8x8	1.5×10^{-2}	8×10^{-6}	0	0
16x16	1.4×10^{-2}	6.8×10^{-6}	0	0
32x32	1.2×10^{-2}	1.9×10^{-6}	0	0

result increase of frequency of assigning multiple packets in one memory module. However, increase in switch size does not affect memory-write speed. According to the table 4.6, memory-write speed must be 3 times faster than the line speed in order to write 3 packets in one memory module.

Algorithm 2 is simulated for 4x4, 8x8, 16x16, and 32x32 switch size, where 8, 16, 32, and 64 memory modules (m = 2N) are deployed respectively to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory conflicts (8 packets of average burst length has more memory conflicts than 16 and 32 packets of average burst length).

Figure 4.8 shows that increase in switch size results reduce of the memory bandwidth requirement at higher load traffic. Table 4.7 shows that increase in switch size result reduce of frequency of assigning multiple packets in one memory module. Increasing the switch size to 32x32 with 64 memory modules has completely eliminated all memory conflicts. The average memory bandwidth requirement becomes 1. That means all incoming packets in each input cycle are stored in one memory-write cycle.

Algorithm 3 is simulated for 4x4, 8x8, 16x16, and 32x32 switch size, where 8, 16, 32, and 64 memory modules (m = 2N) are deployed respectively to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory conflicts (8 packets of average burst length has more memory conflicts than 16and 32 packets of average burst length).

Figure 4.9 shows that increase in switch size results increase of the memory bandwidth requirement at higher load traffic. Table 4.8 shows that increase in switch size

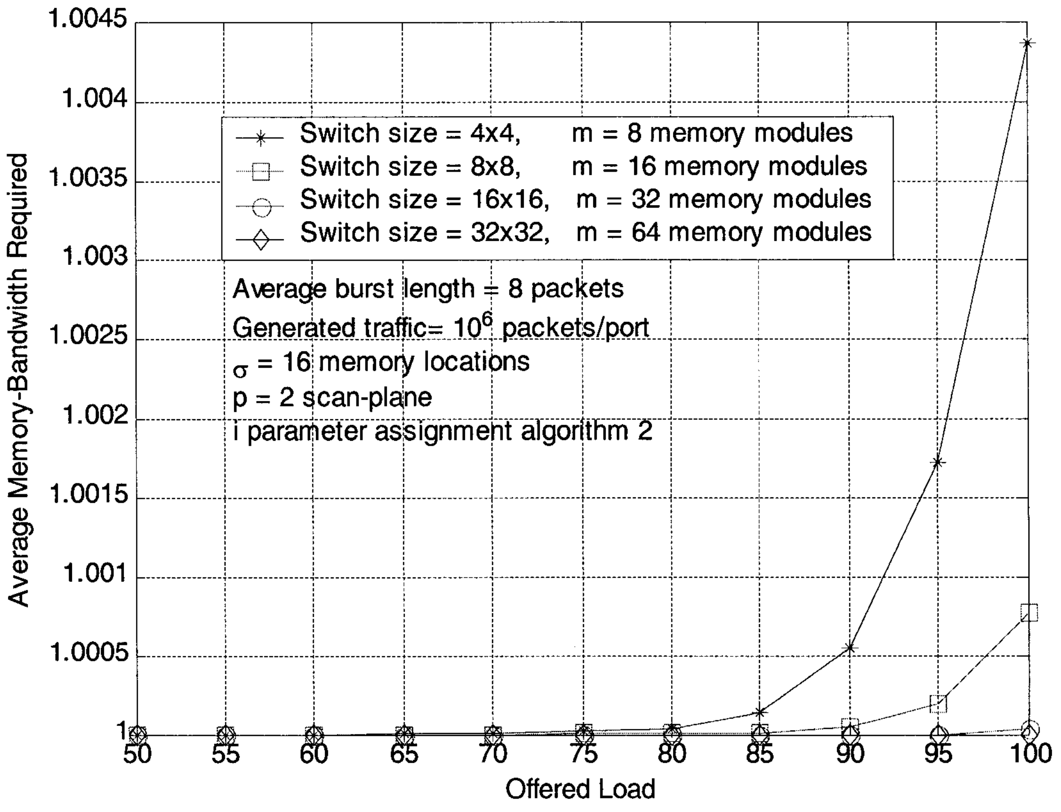


Fig. 4.8 Average Memory-Bandwidth Required vs. Offered Load at 8 packets average burst length traffic using i parameter assignment algorithm 2

Table 4.7 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.8

Switch Size	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
4x4	1.1×10^{-3}	0	0	0
8x8	8.7×10^{-5}	0	0	0
16x16	2.7×10^{-6}	0	0	0
32x32	0	0	0	0

result not only increase of frequency of assigning multiple packets in one memory module but also increase in memory-write speed. According to the table 4.8, memory-write speed must be 4 times faster than the line speed in order to write 4 packets in one memory module when switch size is 4x4, 6 times faster than the line speed in order to write 6 packets in one memory module when switch size is 8x8, 7 times faster than the line speed in order to write 7 packets in one memory module when switch size is 16x16, 8 times faster than the line speed in order to write 8 packets in one memory module when switch size is 32x32.

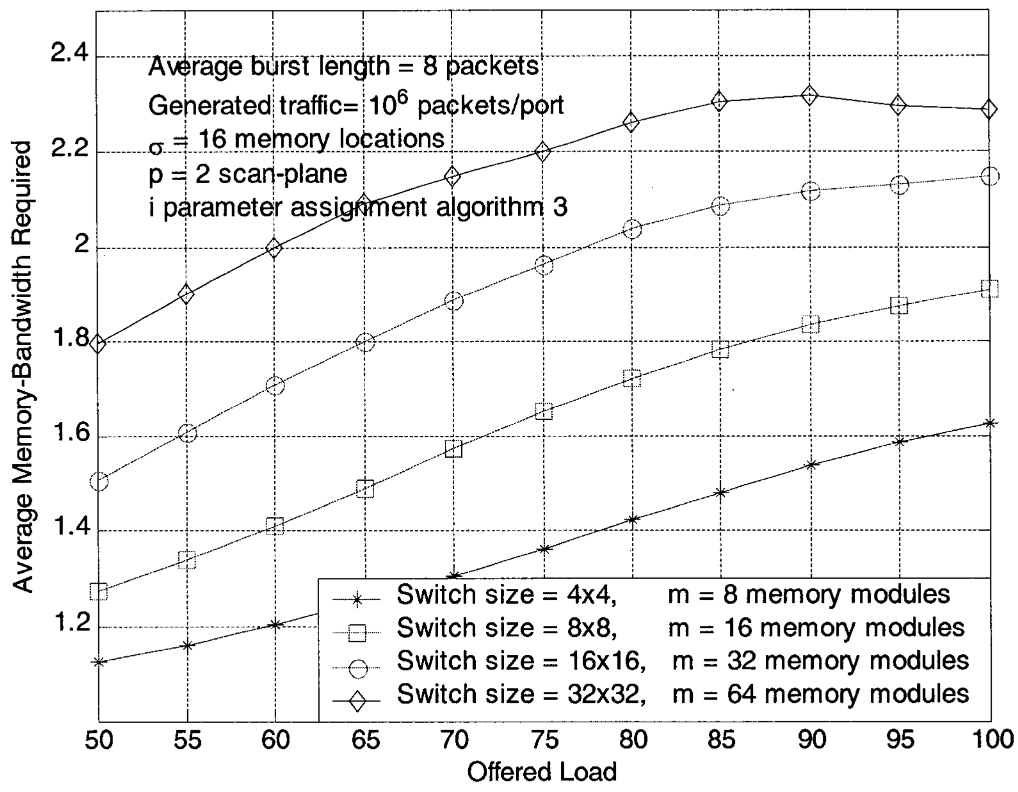


Fig. 4.9 Average Memory-Bandwidth Required vs. Offered Load at 8 packets average burst length traffic using i parameter assignment algorithm 3

Table 4.8 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.9

Ratio of multiple packets in one memory module	Switch Size 4x4	Switch Size 8x8	Switch Size 16x16	Switch Size 32x32
2 packets	1.3×10^{-1}	1.2×10^{-1}	1.1×10^{-1}	9.9×10^{-2}
3 packets	1.6×10^{-2}	1.6×10^{-2}	1.3×10^{-2}	9.5×10^{-3}
4 packets	9.3×10^{-4}	1.5×10^{-3}	1.2×10^{-3}	7.5×10^{-4}
5 packets	0	9.8×10^{-5}	8.4×10^{-5}	4.5×10^{-5}
6 packets	0	4.1×10^{-6}	5.3×10^{-6}	2.9×10^{-6}
7 packets	0	1.3×10^{-7}	5×10^{-7}	1.9×10^{-7}
8 packets	0	0	0	1.9×10^{-7}
9 packets	0	0	0	3.1×10^{-8}

The reason behind increase in both frequency of assigning multiple packets in one memory module and memory-write speed requirement for i assignment algorithm 3 while increasing the switch size is because there is no control (the i value resides in front of each Queue(j) are assigned randomly) to assign i parameter for incoming packets in algorithm 3. Hence, the larger switch size generates more packets, which increases the probability of assigning multiple packets in one memory module.

4.5.1 The Effect of σ on Switch Performance using Algorithm 1

In this section, different i parameter assignment algorithms are used to measure the effects of number of memory locations (σ) and different offered load on average memory bandwidth performance of Parameter Assignment Circuit.

First, Algorithm 1 is simulated, which switch size is set to 4x4, where 8 memory modules (m = 2N) are deployed to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory conflicts (8 packets of average burst length has more memory conflicts than 16 and 32 packets of average burst length), load offered were 90 %, 95 %, and 100 % because these load values give us the worst 3 cases. Simulation started with σ = 16 and doubled each time up to 4096 locations in each memory modules. Figure 4.10 shows that average memory bandwidth requirement is maximum when the traffic load = 100 % and σ = 256. Table 4.9 shows that memory-write speed must be maximum 3 times faster than the line speed in order to write 3 packets in one memory module when the traffic load = 100 % and σ = 256. The reason of lower frequencies of multiple packets assignment in one memory module at 90 %, and 95 % traffic load can be explained as follows. 90 % and 95 % traffic load generates some amount of empty slots (no incoming packet). When there is no packet in one of the incoming line, that will cause empty space in the global shared memory, will minimize the memory conflict. However, the average memory bandwidth requirement of the sliding window switch is increasing until σ increases up to 256 memory locations at 100 % load differently. After σ is 256, average memory bandwidth requirement decreases close to 1 when σ = 4096. This behavior can be explained with figure 4.11 that shows packet loss ratio versus number of memory Locations (σ) per memory module at 90 %,

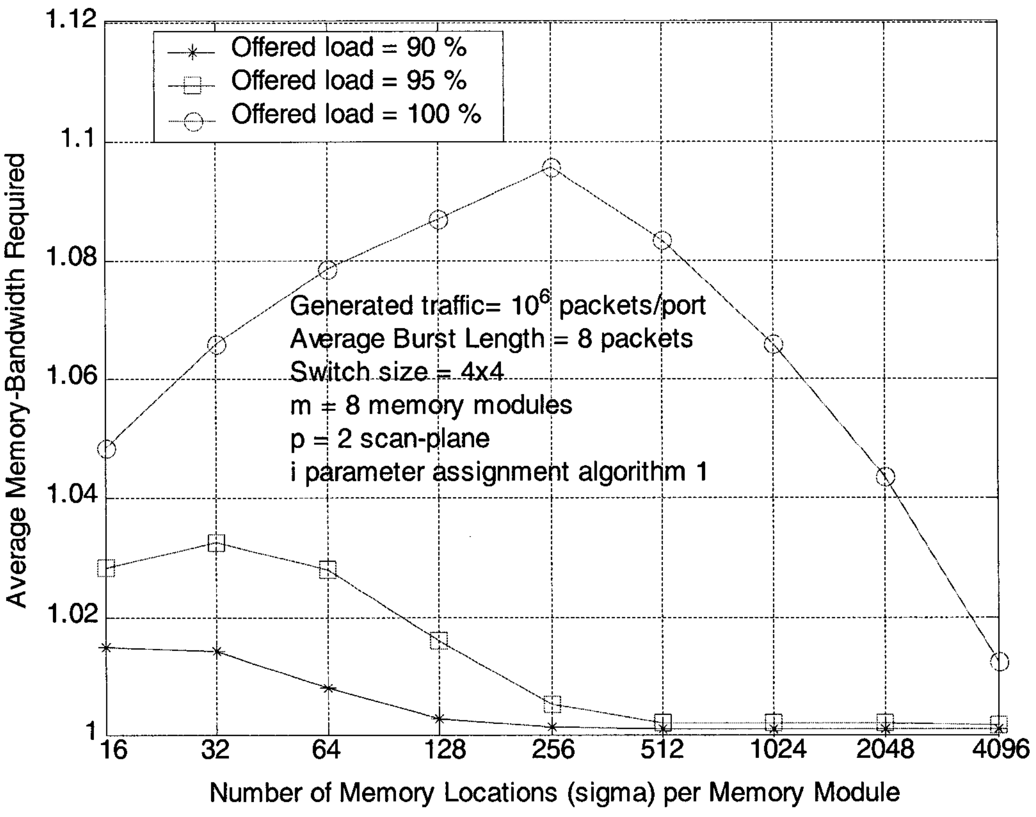


Fig. 4.10 Average Memory-Bandwidth Required vs. σ per Memory Module at 4x4 switch with 8 memory modules using i parameter assignment algorithm 1

Table 4.9 Worst-case ratio of multiple packets destined to same memory module at σ = 256 in figure 4.10

Offered Load at σ = 256	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
90 %	3.3×10^{-4}	0	0	0
95 %	1.3×10^{-3}	0	0	0
100 %	2.2×10^{-2}	1×10^{-6}	0	0

95 %, and 100 % load traffic.

Number of memory locations (σ) basically defines the Queue length for each output-port. The maximum number of packets for each output queue is given by,

$$Q_{max} = p \times \sigma \tag{7}$$

where p is number of scan-plane (p = 2 for this experiment), where σ is variable. The larger σ, the larger queue size for each output-port. Memory conflicts occur mostly when all the output queues are greater than σ. Because when the number of packets in the each queue exceeds the σ, 2 packets will be stored in the current OSV.j whose queues are greater than σ. Therefore, number of available slot will decrease depending on the switch size. Memory conflicts will occur when there is small number of available slot in the current OSV.j (if the number of available slot is one, incoming packet will be stored in that available slot in that OSV.j leaving no deciding mechanism). The number packets at each queue can reach up to 256 rapidly when sigma is 256, which has the maximum average memory bandwidth requirement, in figure 4.10 at 100 % load having average 8 packets burst length. Having greater σ will increase the queue for each output- port. The probability of having more than the number of σ ($\sigma > 256$) packets on the each queue is going to be declined until $\sigma = 4096$. When $\sigma = 2048$, there is still small amount of packet loss indicating some of the output queues can reach the 4096 packets at 100 % load having average 8 packets burst length shown in figure 4.11. Packet loss ratio is zero, and the average memory bandwidth requirement is close to 1 when $\sigma = 4096$.

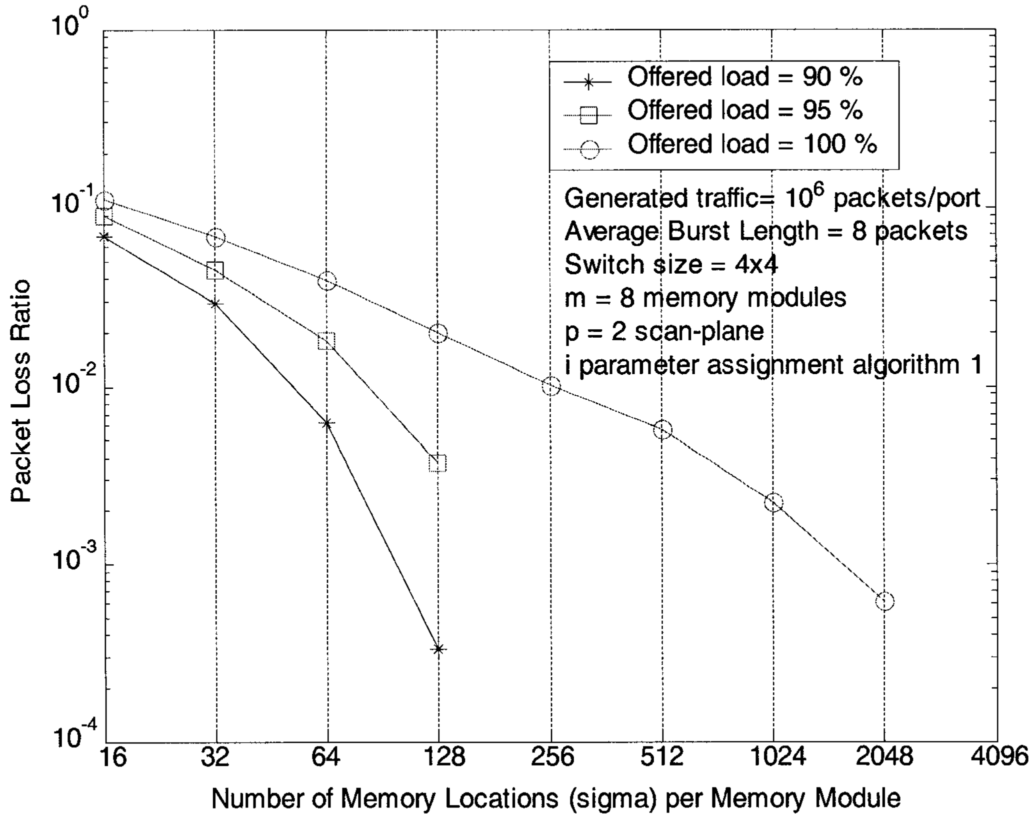


Fig. 4.11 Packet Loss Ratio vs. σ per Memory Module at 4x4 switch with 8 memory modules using i parameter assignment algorithm 1

Algorithm 1 is simulated in order to see the effect of different switch size 8x8, where 16 memory modules (m = 2N) are deployed to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory conflicts. Figure 4.12 shows us that the average memory bandwidth requirement is maximum when the traffic load = 100 % and σ = 512. The average memory bandwidth requirement is

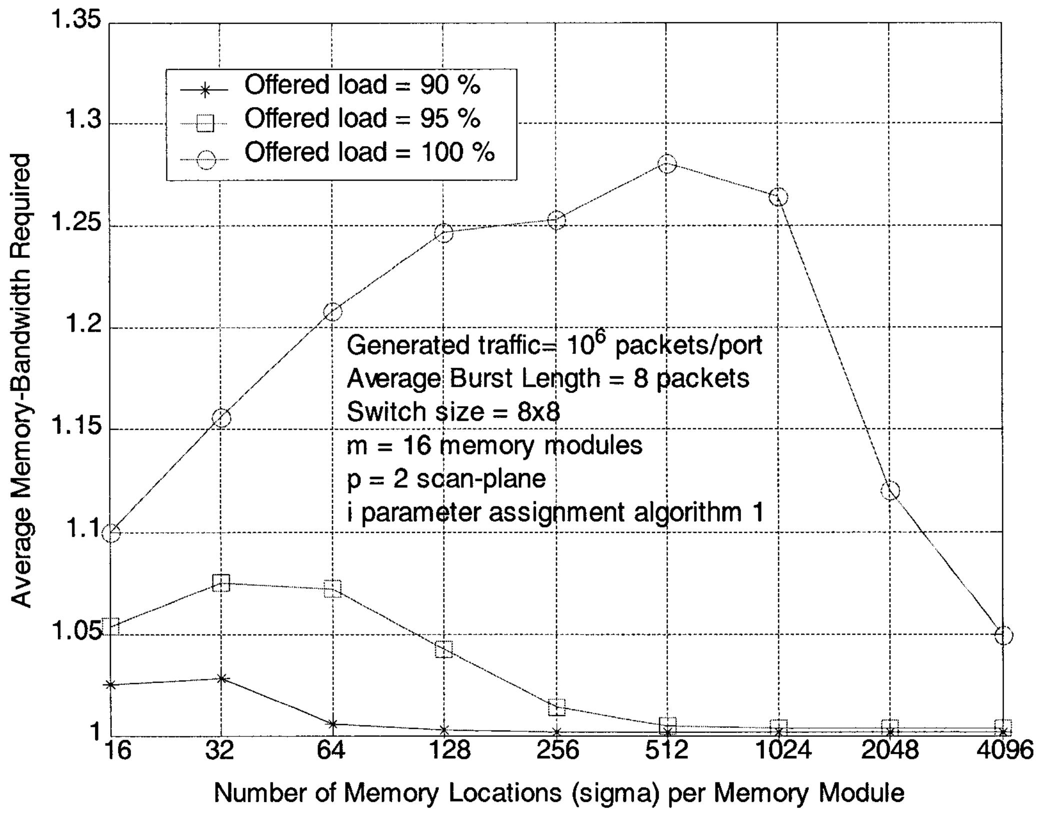


Fig. 4.12 Average Memory-Bandwidth Required vs. σ per Memory Module at 8x8 switch with 16 memory modules using i parameter assignment algorithm 1

Table 4.10 Worst-case ratio of multiple packets destined to same memory module at $\sigma = 512$ in figure 4.12

Offered Load at $\sigma = 512$	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
90 %	2.6×10^{-4}	0	0	0
95 %	6.9×10^{-4}	0	0	0
100 %	4.1×10^{-2}	6×10^{-5}	0	0

larger at each σ value when 8x8 switch size is deployed compared to 4x4 switch size. Table 4.10 shows that memory-write speed must be maximum 3 times faster than the line speed in order to write 3 packets in one memory module when the traffic load = 100 % and σ = 512.

Figure 4.13 shows packet loss ratio versus number of memory Locations (σ) per memory module at 90 %, 95 %, and 100 % load traffic. Having greater σ will increase the queue for each output-port. The probability of having more than the number of σ (σ > 512) packets on the each queue is going to be declined until σ = 4096. When σ = 2048,

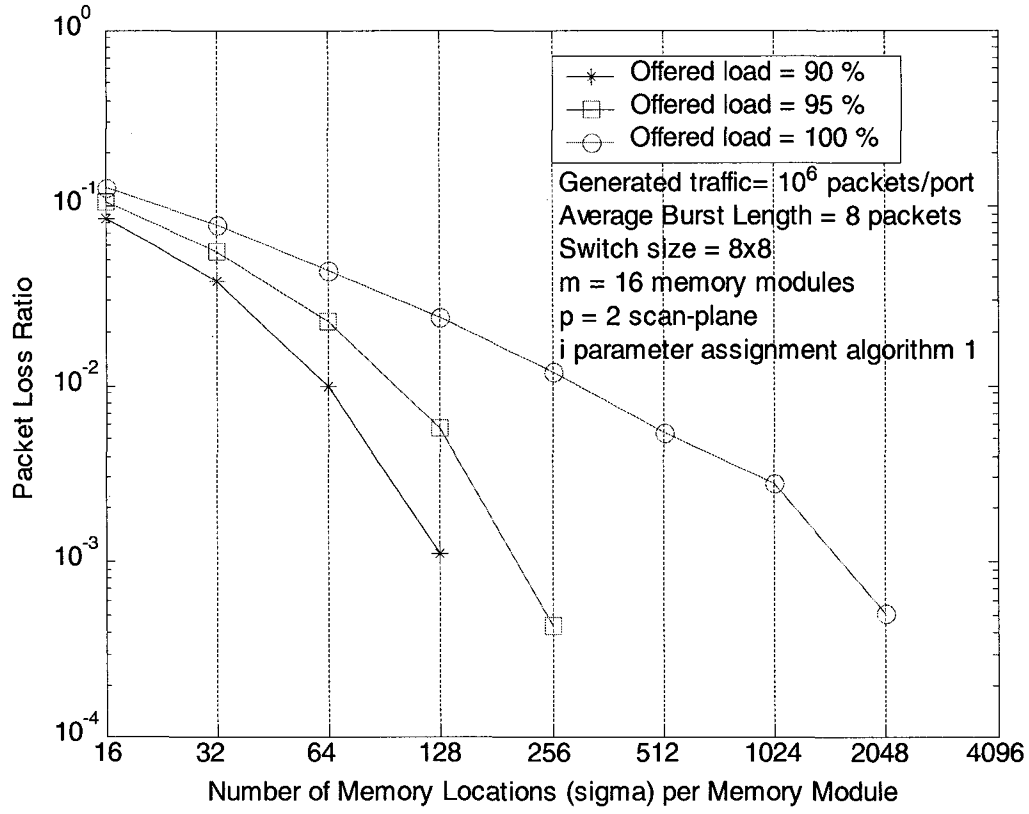


Fig. 4.13 Packet Loss Ratio vs. σ per Memory Module at 8x8 switch with 16 memory modules using i parameter assignment algorithm 1

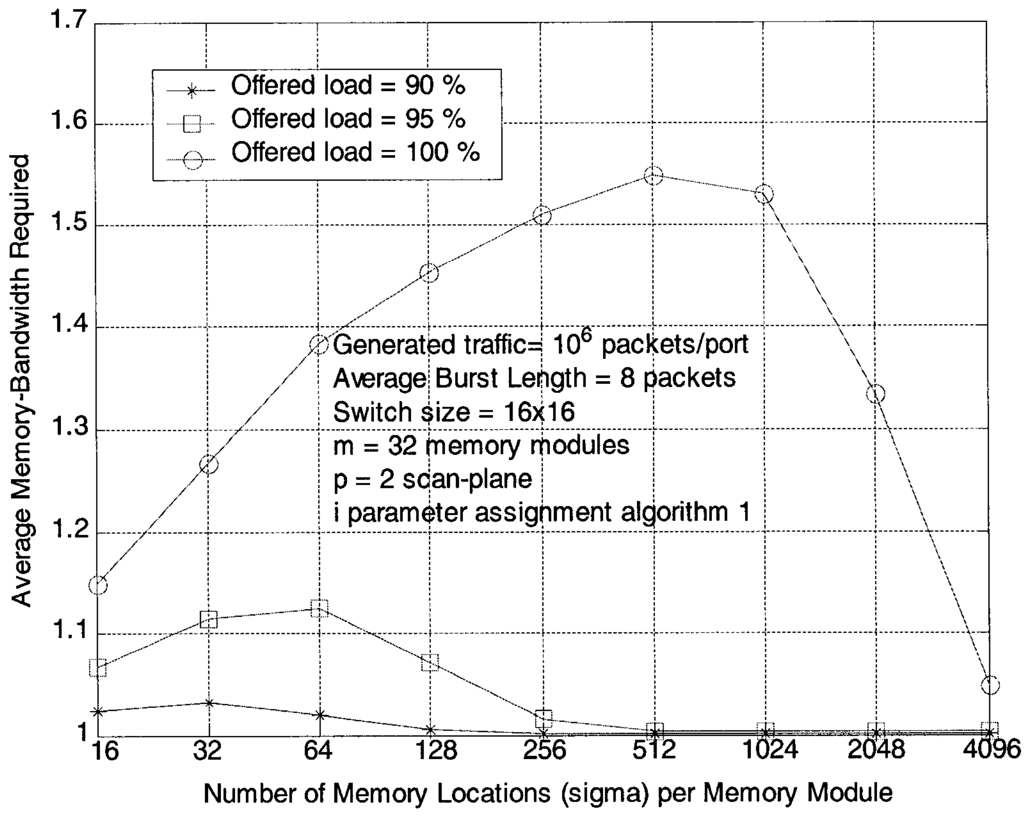


Fig. 4.14 Average Memory-Bandwidth Required vs. σ per Memory Module at 16x16 switch with 32 memory modules using i parameter assignment algorithm 1

Table 4.11 Worst-case ratio of multiple packets destined to same memory module at σ = 512 in figure 4.14

Offered Load at σ = 512	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
90 %	6.3×10^{-5}	0	0	0
95 %	3.4×10^{-4}	0	0	0
100 %	5.7×10^{-2}	1.1×10^{-4}	6.3×10^{-8}	0

there is still small amount of packet loss indicating some of the output queues can reach the 4096 packets at 100 % traffic load having average 8 packets burst length shown in figure 4.13. Packet loss ratio is zero, and the average memory bandwidth requirement is close to 1 when $\sigma = 4096$.

Algorithm 1 is simulated in order to see the effect of different switch size 16x16, where 32 memory modules (m = 2N) are deployed to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory conflicts. Figure 4.14 shows us that the average memory bandwidth requirement is maximum when traffic load = 100 % and $\sigma = 512$. The average memory bandwidth requirement is larger

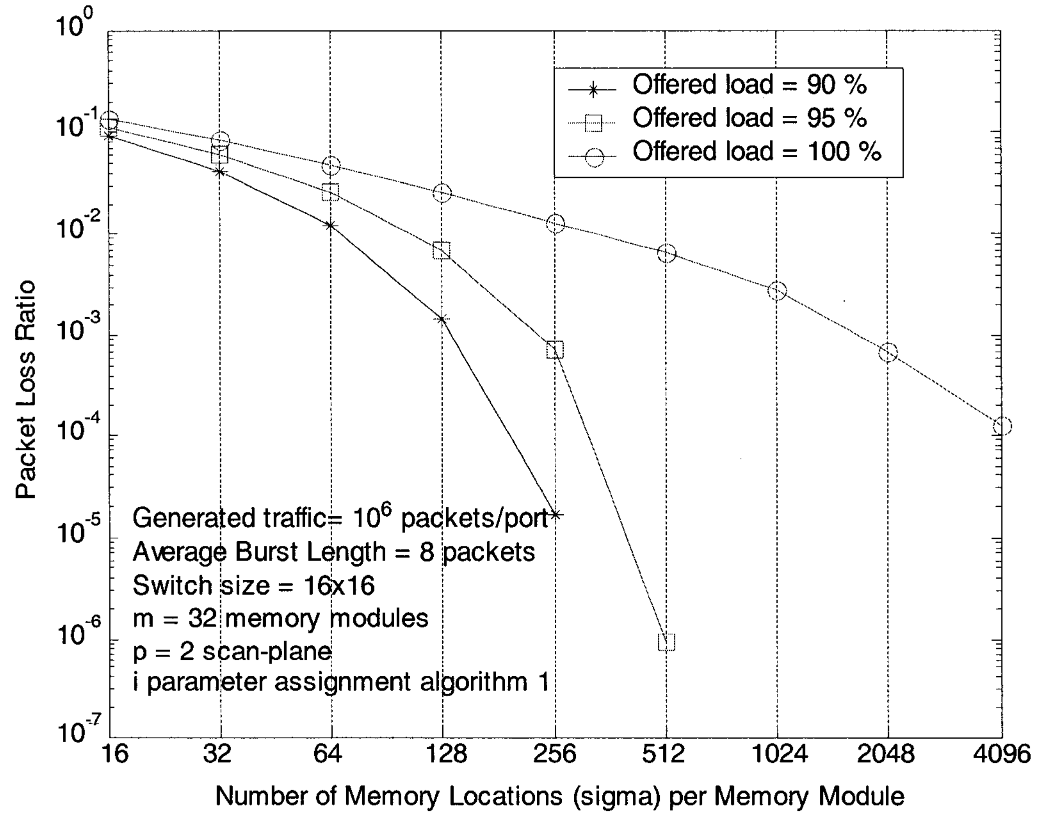


Fig. 4.15 Packet Loss Ratio vs. σ per Memory Module at 16x16 switch with 32 memory modules using i parameter assignment algorithm 1

at each σ value when 16x16 switch size is deployed compared to 8x8 and 4x4 switch size. Table 4.11 shows that memory-write speed must be maximum 4 times faster than the line speed in order to write 4 packets in one memory module when traffic load = 100 % and σ = 512.

Figure 4.15 shows packet loss ratio versus number of memory Locations (σ) per memory module at 90 %, 95 %, and 100 % load traffic. Having greater σ will increase the queue for each output-port. The probability of having more than the number of σ (σ > 512) packets on the each queue is going to be declined until σ = 4096. When σ = 2048, there is still small amount of packet loss indicating some of the output queues can reach the 4096 packets at 100 % load.

4.5.2 The Effect of σ on Switch Performance using Algorithm 2

Algorithm 2 is simulated, which switch size is set to 4x4, where 8 memory modules (m = 2N) are deployed to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory conflicts (8 packets of average burst length has more memory conflicts than 16 and 32 packets of average burst length), load offered were 90 %, 95 %, and 100 % because these load values give us the worst 3 cases. Simulation started with σ = 16 and doubled each time up to 4096 locations in each memory modules. Figure 4.16 shows that average memory bandwidth requirement is maximum when the traffic load = 100 % and σ = 256. Average memory bandwidth requirement is 1 when traffic load = 90 % and σ = 256, when traffic load = 95 % and σ = 512, when traffic load = 100 % and σ = 4096. The pattern of memory bandwidth requirement versus σ per memory module is same as algorithm 1.

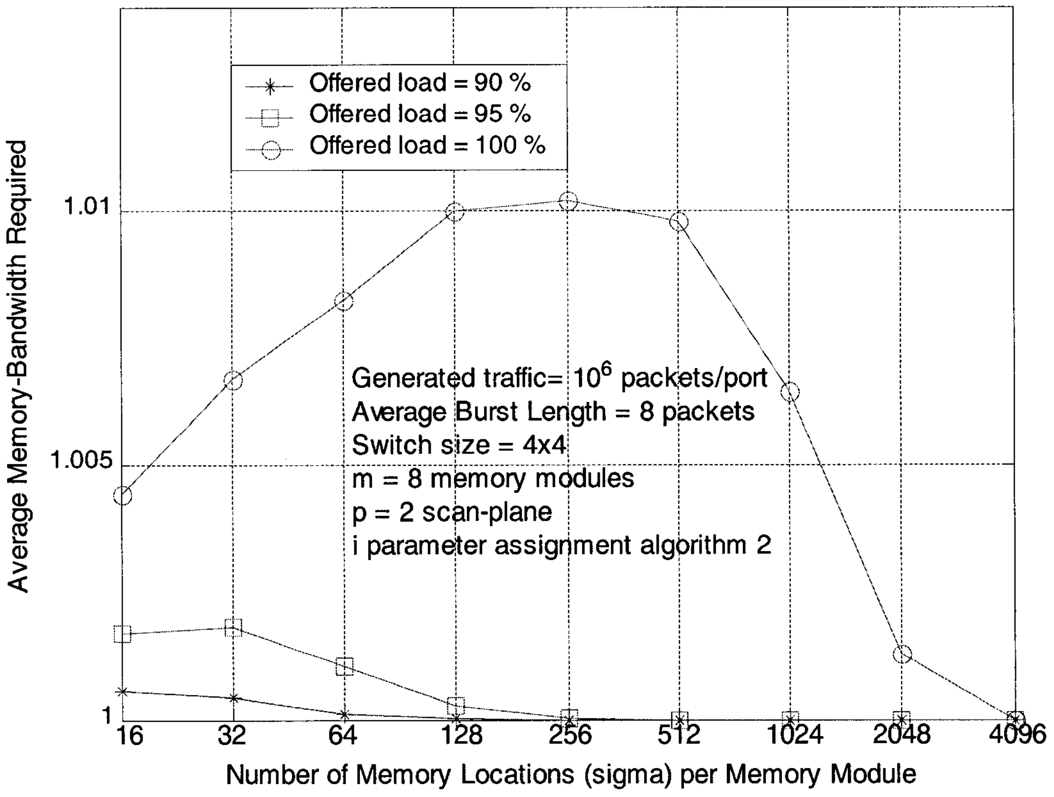


Fig. 4.16 Average Memory-Bandwidth Required vs. σ per Memory Module at 4x4 switch with 8 memory modules using i parameter assignment algorithm 2

Table 4.12 Worst-case ratio of multiple packets destined to same memory module at σ = 256 in figure 4.16

Offered Load at σ = 256	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
90 %	0	0	0	0
95 %	1.3×10^{-6}	0	0	0
100 %	2.8×10^{-3}	0	0	0

Table 4.12 shows that memory-write speed must be maximum 2 times faster than the line speed in order to write 2 packets in one memory module when the traffic load = 100 % and σ = 256. However, if there is load balancer, which adjusts traffic load such as 90 %, memory-write speed could be the same as line speed when the traffic = 90 % and σ = 256 because there is no memory conflict at 90 % traffic load and σ = 256, shown first row in table 4.12.

The larger σ, the larger queue size for each output-port. Memory conflicts occur mostly when all the output queues are greater than σ. Because when the number of packets in the each queue exceeds the σ, 2 packets will be stored in the current OSV.j

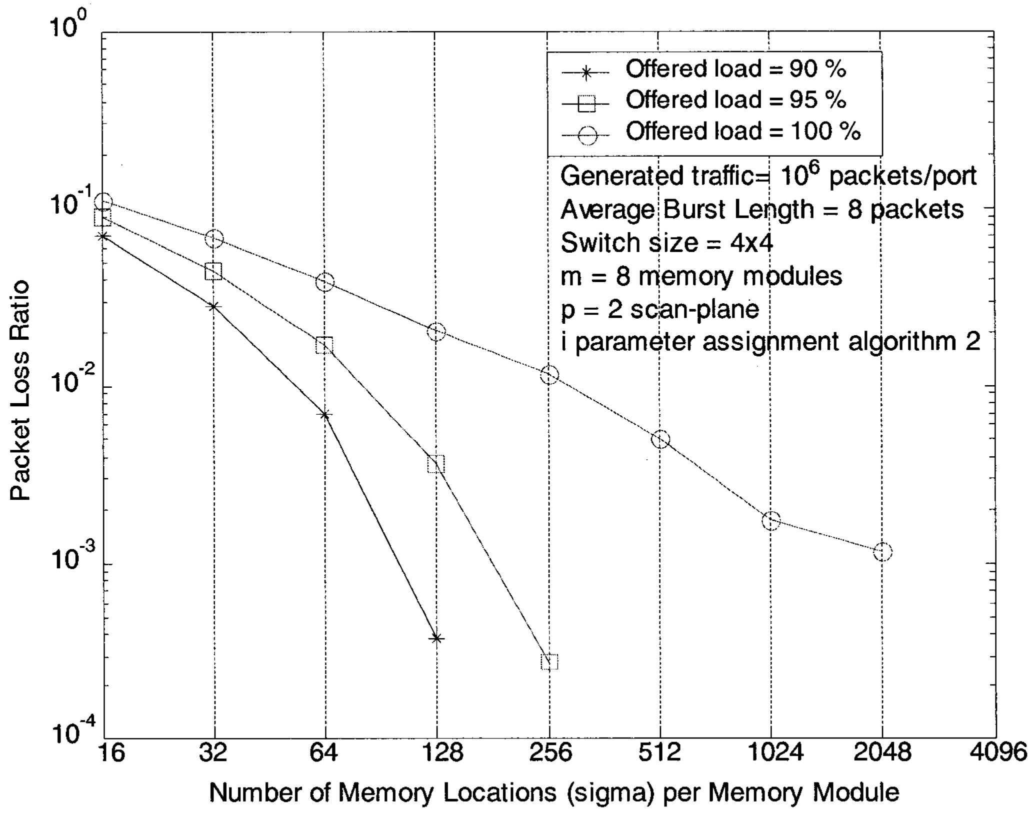


Fig. 4.17 Packet Loss Ratio vs. σ per Memory Module at 4x4 switch with 8 memory modules using i parameter assignment algorithm 2

whose queues are greater than σ. Therefore, number of available slot will decrease one or two depending on the switch size. Memory conflicts will occur when there is small number of available slot in the current OSV.j (if the number of available slot is one, incoming packet will be stored in that available slot in that OSV.j leaving no deciding mechanism). The number packets at each queue can reach up to 256 rapidly when sigma is 256, which has the maximum average memory bandwidth requirement, in figure 4.16 at 100 % traffic load. Having greater σ will increase the queue for each output-port. The probability of having more than the number of σ ($\sigma > 256$) packets on the each queue is going to be declined until $\sigma = 4096$. When $\sigma = 2048$, there is still small amount of packet loss indicating some of the output queues can reach the 4096 packets at 100 % load having average 8 packets burst length shown in figure 4.17. Packet loss ratio is zero, and the average memory bandwidth requirement is 1 when $\sigma = 4096$. Algorithm 2 is simulated, which switch size is set to 8x8, where 16 memory modules are deployed to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory, load offered were 90 %, 95 %, and 100 % because these load values give us the worst 3 cases. Simulation started with $\sigma = 16$ and doubled each time up to 4096 locations in each memory modules. Figure 4.18 shows that average memory bandwidth requirement is maximum when the traffic load = 100 % and $\sigma = 256$. Average memory bandwidth requirement is 1 when traffic load = 90 % and $\sigma = 256$, when traffic load = 95 % and $\sigma = 512$, when traffic load = 100 % and $\sigma = 4096$.

Table 4.13 shows that memory-write speed must be maximum 2 times faster than the line speed in order to write 2 packets in one memory module when the traffic load = 100 % and $\sigma = 256$. Memory-write speed could be the same as line speed when the traffic

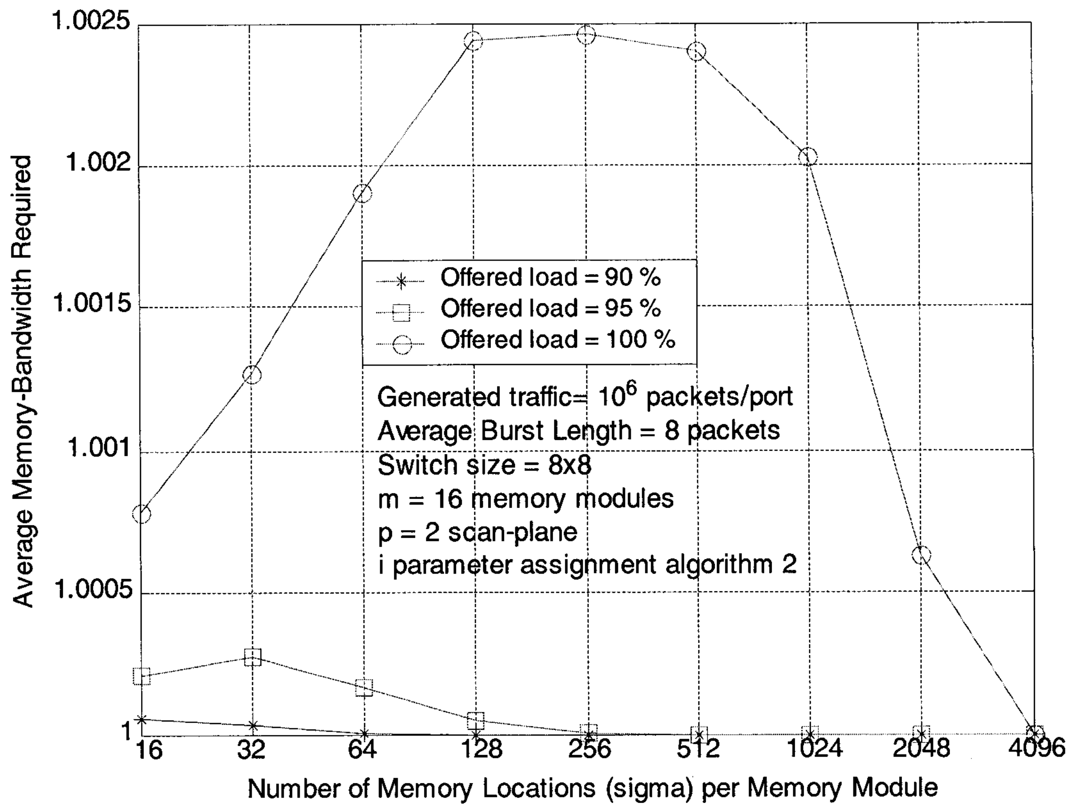


Fig. 4.18 Average Memory-Bandwidth Required vs. σ per Memory Module at 8x8 switch with 16 memory modules using i parameter assignment algorithm 2

Table 4.13 Worst-case ratio of multiple packets destined to same memory module at σ = 256 in figure 4.18

Offered Load at σ = 256	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
90 %	0	0	0	0
95 %	1.3×10^{-7}	0	0	0
100 %	3.7×10^{-4}	0	0	0

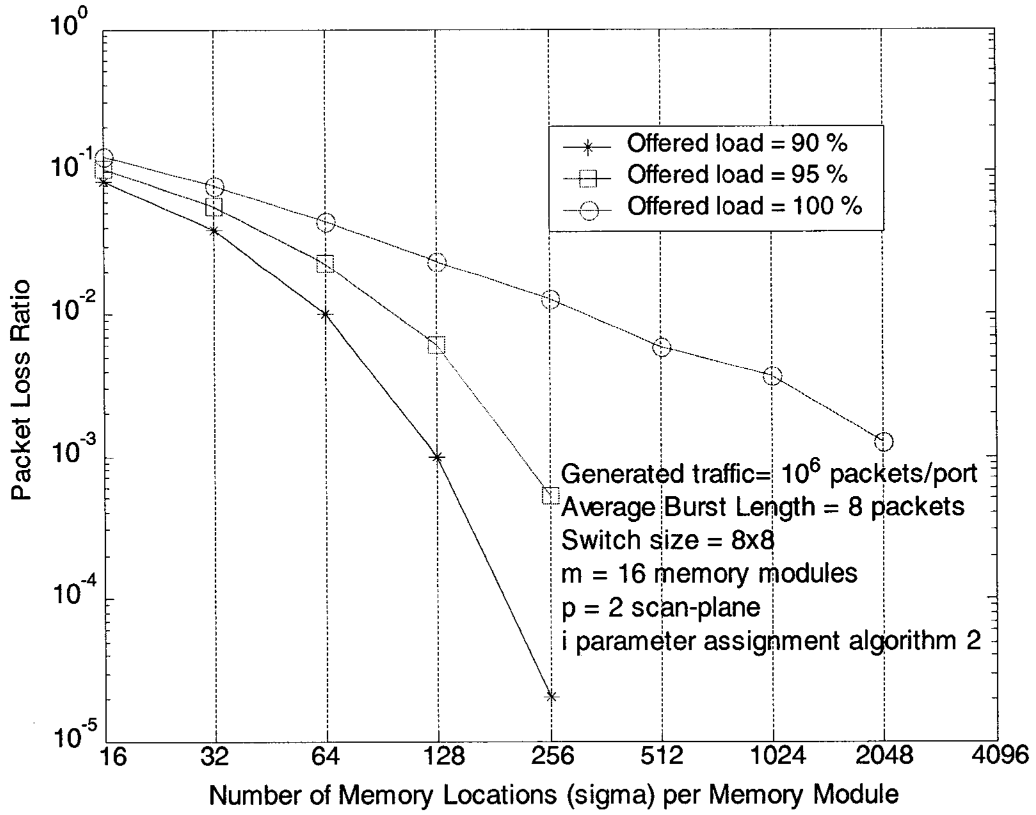


Fig. 4.19 Packet Loss Ratio vs. σ per Memory Module at 8x8 switch with 16 memory modules using i parameter assignment algorithm 2

90 % and σ = 256 because there is no memory conflict at 90 % traffic load and σ = 256.

Figure 4.19 shows packet loss ratio vs. σ per memory module at 8x8 switch with 16 memory modules using i parameter assignment algorithm 2. There are packet losses when σ = 256 at all traffic load values. There is no packet losses when σ = 512 for 90 %

and 95 % traffic load, σ = 4096 for 100 % traffic load value. Packet loss ratio pattern of 8x8 switch is almost same as packet loss ratio pattern of 4x4 switch. Packet loss ratio is getting higher when switch size is increasing because of increase in number of packets correlated with switch size. Therefore, σ value increases in order to keep certain packet loss ratio.

Algorithm 2 is simulated, which switch size is set to 16x16, where 32 memory modules are deployed to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory, load offered were 90 %, 95 %, and 100 % because these load values give us the worst 3 cases. Simulation started with σ = 16 and doubled each time up to 4096 locations in each memory modules. Figure 4.20 shows that average memory bandwidth requirement is maximum when the traffic load = 100 % and σ = 256. Average memory bandwidth requirement is 1 when traffic load = 90 % and all σ values, when traffic load = 95 % and σ = 256, when traffic load = 100 % and σ = 4096.

Table 4.14 shows that memory-write speed must be maximum 2 times faster than the line speed in order to write 2 packets in one memory module when the traffic load = 100 % and σ = 256. Memory-write speed could be the same as line speed when the traffic load = 90 %, 95 % and σ = 256 because there is no memory conflict at 90 %, 95 % traffic load and σ = 256.

It is observed that very similar phenomena occurs regarding the average memory bandwidth requirement stabilizes to 1 at 100 % load when σ is 4096, shown figure 4.20. The only difference between 8x8 and 16x16 switch sizes is that the average memory bandwidth is lower at each σ value when 16x16 switch size is deployed. The reason

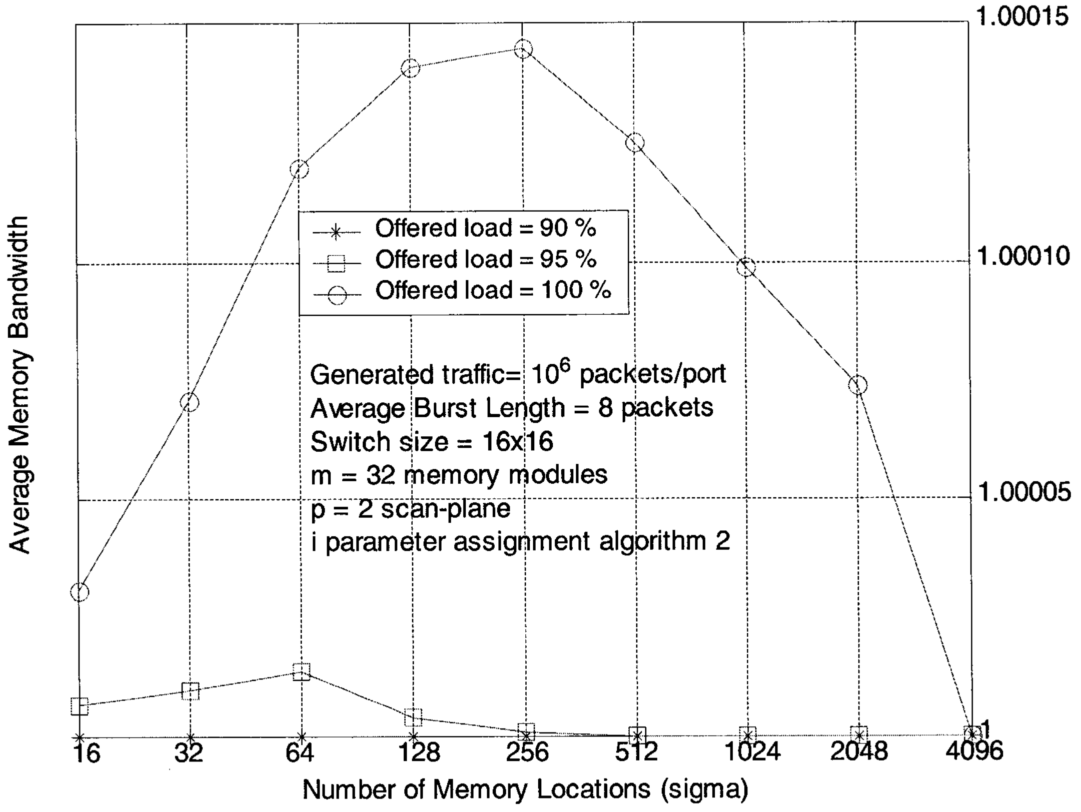


Fig. 4.20 Average Memory-Bandwidth Required vs. σ per Memory Module at 16x16 switch with 32 memory modules using i parameter assignment algorithm 2

Table 4.14 Worst-case ratio of multiple packets destined to same memory module at σ = 256 in figure 4.20

Offered Load at σ = 256	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Frequency of 5 packets stored in 1 memory module
90 %	0	0	0	0
95 %	0	0	0	0
100 %	1.2 x 10^{-5}	0	0	0

behind this can be explained that the more memory space will give more possibilities to store all incoming packets in parallel, requiring only one memory write cycle. However, the packet loss ratio is increasing while the switch size is increasing (shown figure 4.21). Because the bigger switch size processes more packets than the smaller switch size, causes queues at each output-port build up rapidly. There is no packet losses when σ = 512 for 90 % traffic load, σ = 1024 for 95 % traffic load value. It is observed that there is small number of packet losses when the σ is 4096 (figure 4.21). However, the memory bandwidth requirement stabilizes to 1 (all incoming packets are stored in different memory modules at the given memory cycle). Because very small number of output

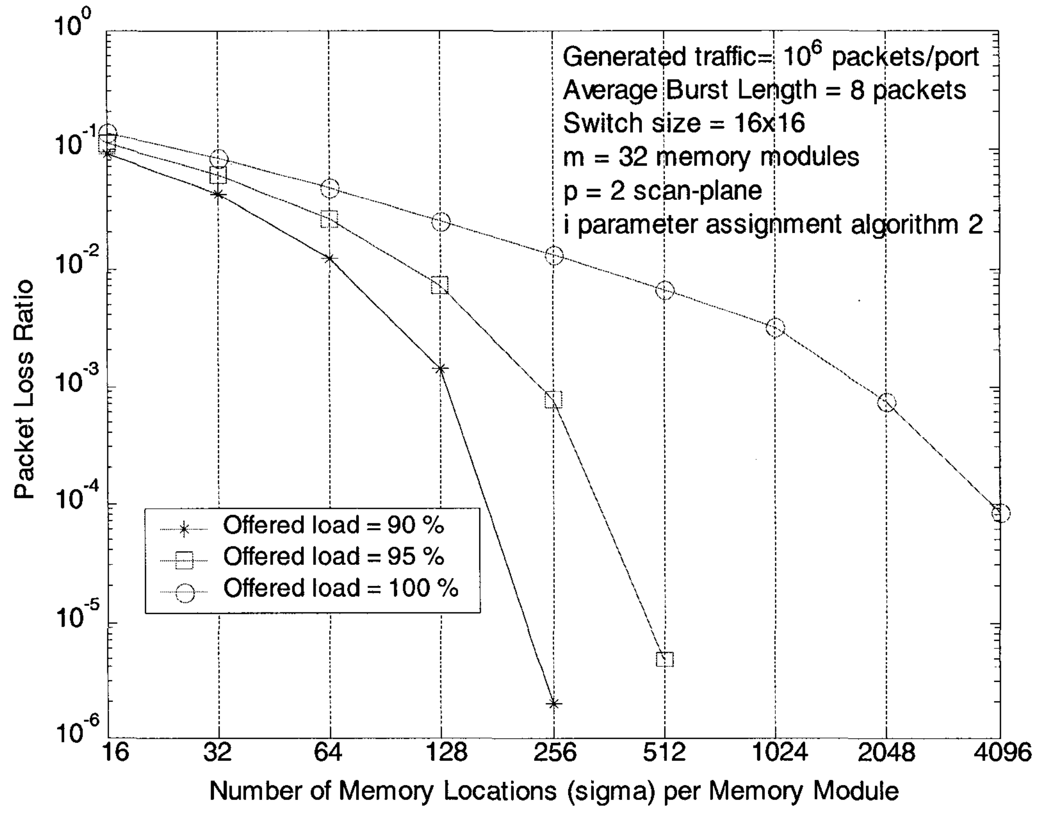


Fig. 4.21 Packet Loss Ratio vs. σ per Memory Module at 16x16 switch with 32 memory modules using i parameter assignment algorithm 2

queue can exceed the σ (4096). There will be enough memory slots to store all the incoming packets in different memory modules will decreases the memory bandwidth requirement to 1.

4.5.3 The Effect of σ on Switch Performance using Algorithm 3

Algorithm 3 is simulated, which switch size is set to 4x4, where 8 memory modules ($m = 2N$) are deployed to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory conflicts (8 packets of average burst length has more memory conflicts than 16 and 32 packets of average burst length), load offered were 90 %, 95 %, and 100 % because these load values give us the worst 3 cases. Simulation started with $\sigma = 16$ and doubled each time up to 4096 locations in each memory modules. Figure 4.22 shows that average memory bandwidth requirement is maximum when traffic load = 90 % and $\sigma = 128$, when traffic load = 95 % and $\sigma = 128$, when traffic load = 100 % and $\sigma = 1024$. The pattern of memory bandwidth requirement versus σ per memory module is completely different from algorithm 1 and algorithm 2. Because assignment of i parameter to an incoming packet using algorithm 1 and 2 is basically consecutive fashion in memory modules. However, assignment of i parameter to an incoming packet using algorithm 3 is random fashion. Because, available i values, which indicates an empty memory modules, for an incoming packet are kept in the one of the Queue for a given OSV.j. It is possible that first value (first element of queue) of each 4 Queues have the same i value (same memory module) for 4 incoming packets. Probability of having 4 packets in one memory module is smaller than the probability of having 3 or 2 packets in one memory module.

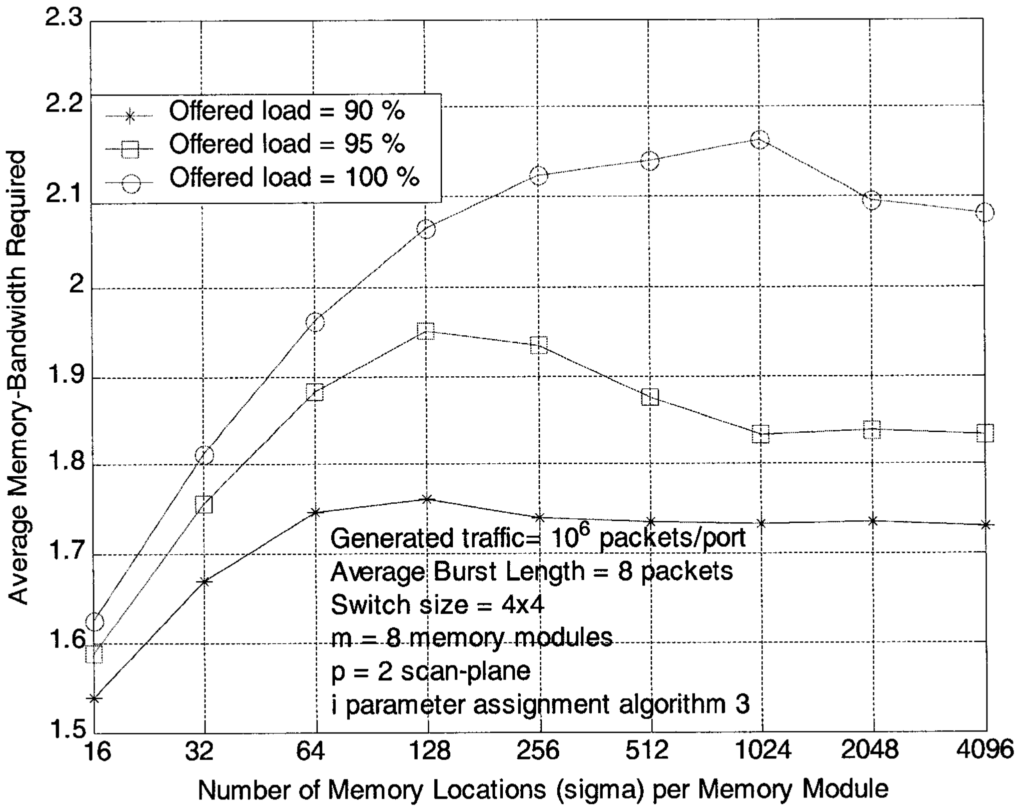


Fig. 4.22 Average Memory-Bandwidth Required vs. σ per Memory Module at 4x4 switch with 8 memory modules using i parameter assignment algorithm 3

Table 4.15 Worst-case ratio of multiple packets destined to same memory module at σ = 1024 in figure 4.22

Offered Load at σ = 1024	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module	Ratio of 5 packets stored in 1 memory module
90 %	1.6×10^{-1}	2.6×10^{-2}	2.1×10^{-3}	0
95 %	1.7×10^{-1}	3.1×10^{-2}	2.7×10^{-3}	0
100 %	2×10^{-1}	5.2×10^{-2}	6.9×10^{-3}	0

Table 4.15 shows that memory-write speed must be maximum 4 times faster than the line speed in order to write 4 packets in one memory module when the traffic load = 100 %, σ = 256.

Figure 4.23 shows packet loss ratio versus number of memory Locations (σ) per memory module at 90 %, 95 %, and 100 % load traffic. Packet loss ratio pattern for 4x4 switch using algorithm 3 is same as packet loss ratio pattern for 4x4 switch using algorithm 1 and algorithm 2. Because packet loss ratio depends on σ and average burst length of incoming traffic.

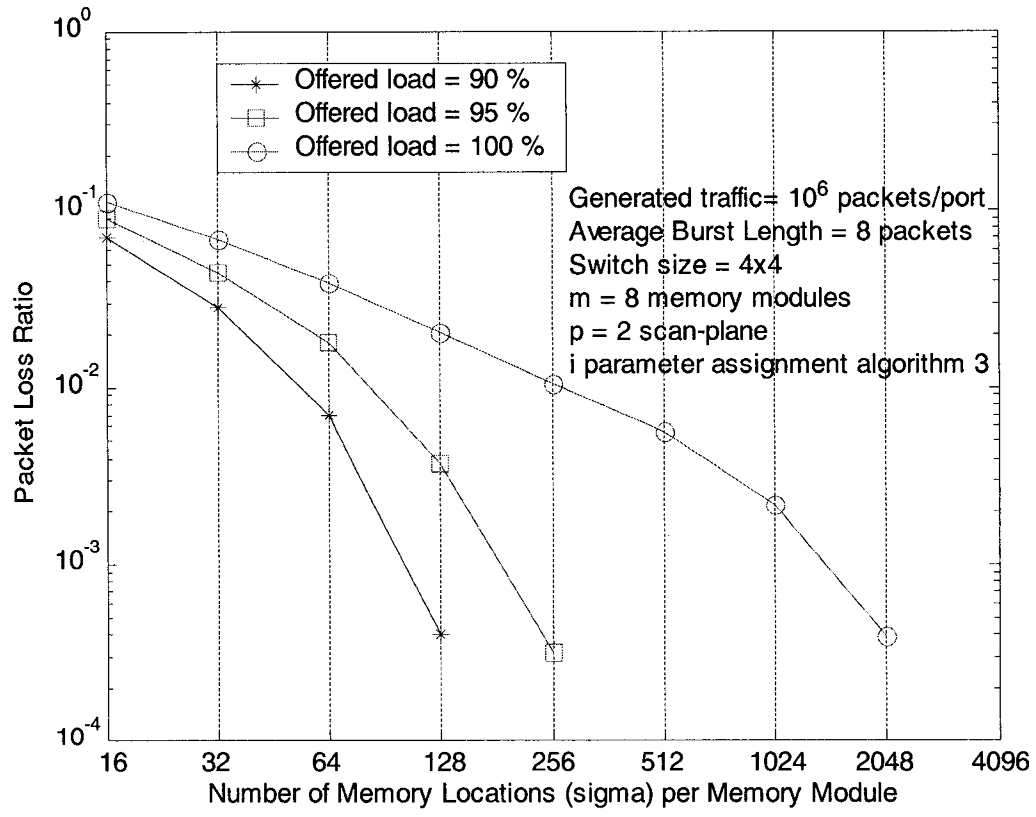


Fig. 4.23 Packet Loss Ratio vs. σ per Memory Module at 4x4 switch with 8 memory modules using i parameter assignment algorithm 3

Algorithm 3 is simulated, which switch size is set to 8x8, where 16 memory modules are deployed to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory conflicts, load offered were 90 %, 95 %, and 100 % because these load values give us the worst 3 cases. Simulation started with σ = 16 and doubled each time up to 4096 locations in each memory modules. Figure 4.24 shows that average memory bandwidth requirement is maximum when traffic load = 100 % and σ = 1024. Table 4.16 shows that memory-write speed must be maximum 8 times faster than the line speed in order to write 8 packets in one memory module when the traffic load = 100 % and σ = 1024.

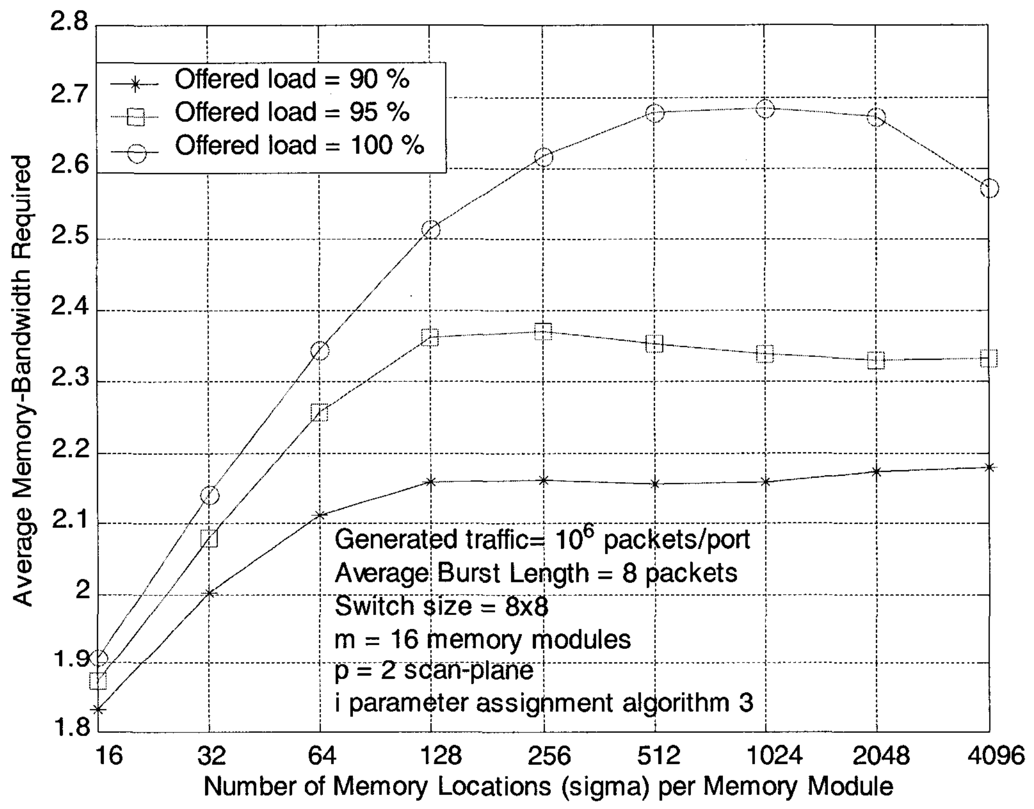


Fig. 4.24 Average Memory-Bandwidth Required vs. σ per Memory Module at 8x8 switch with 16 memory modules using i parameter assignment algorithm 3

Table 4.16 Worst-case ratio of multiple packets destined to same memory module at σ = 1024 in figure 4.24

	Ratio of multiple packets in one memory module		
Number of packets in one memory module	90 % Offered Load at σ = 1024	95 % Offered Load at σ = 1024	100 % Offered Load at σ = 1024
2 packets	1.6×10^{-1}	1.6×10^{-1}	1.9×10^{-1}
3 packets	3.1×10^{-2}	3.8×10^{-2}	5.8×10^{-2}
4 packets	4.5×10^{-3}	6.6×10^{-3}	1.3×10^{-2}
5 packets	5.3×10^{-4}	8.8×10^{-4}	2.2×10^{-3}
6 packets	4.7×10^{-5}	8.3×10^{-5}	3×10^{-4}
7 packets	2.5×10^{-6}	6.5×10^{-6}	2.3×10^{-5}
8 packets	4.2×10^{-7}	5.3×10^{-7}	6.3×10^{-7}
9 packets	0	0	0

Figure 4.25 shows packet loss ratio versus number of memory Locations (σ) per memory module at 90 %, 95 %, and 100 % load traffic. Even though packet loss ratio pattern using algorithm 3 is same as packet loss ratio pattern using algorithm 1 and algorithm 2, average memory bandwidth requirement is quite different. Algorithm 1 and algorithm 2 reduce number of memory conflicts while σ is increasing. Because those algorithms assign i values in a consecutive fashion to avoid assigning same i value to an incoming packet in the same input cycle. However, algorithm 3 (Queue-based) assigns i parameter in a random fashion for an incoming packets. Therefore, increase in σ using

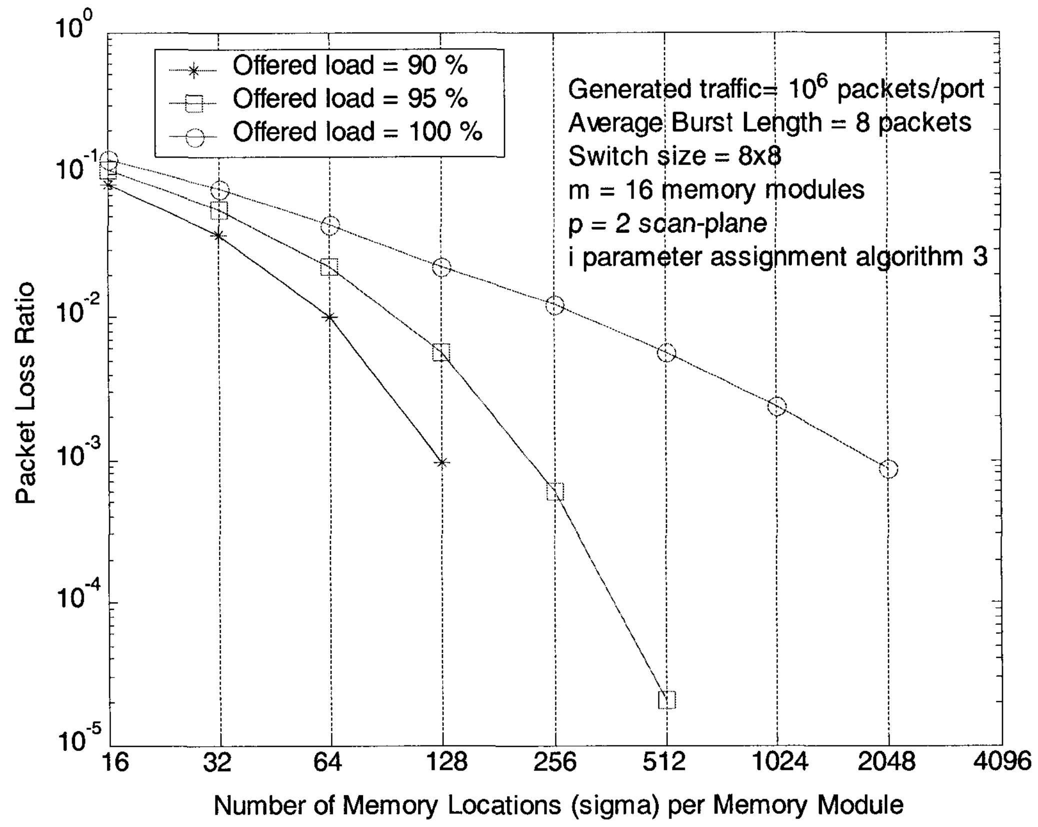


Fig. 4.25 Packet Loss Ratio vs. σ per Memory Module at 8x8 switch with 16 memory modules using i parameter assignment algorithm 3

algorithm 3 is not going to reduce average memory bandwidth requirement as much as using algorithm 1 and 2 for σ > 256 or 512 values which are explained in previous sections.

Algorithm 3 is simulated, which switch size is set to 16x16, where 32 memory modules are deployed to maximize full utilization. Average burst length is set to 8 packets to observe the maximum number of memory conflicts, load offered were 90 %, 95 %, and 100 % because these load values give us the worst 3 cases. Simulation started with σ = 16 and doubled each time up to 4096 locations in each memory modules. Figure 4.26 shows that average memory bandwidth requirement is maximum when traffic load = 100 % and σ = 2048.

It is observed that increase in σ does not reduce average memory bandwidth requirement at 16x16 switch using i parameter assignment algorithm 3. Table 4.17 shows that memory-write speed must be maximum 10 times faster than the line speed in order to write 10 packets in one memory module when the traffic load = 100 % and σ = 2048, 9 times faster than the line speed in order to write 9 packets in one memory module when

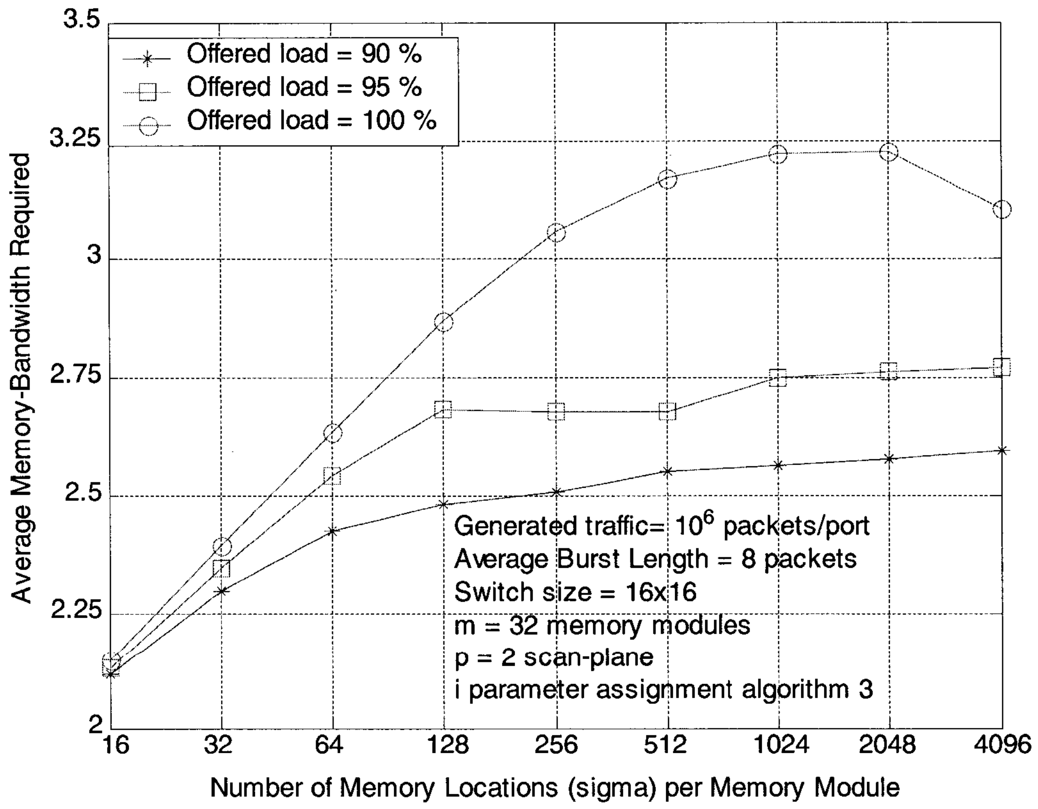


Fig. 4.26 Average Memory-Bandwidth Required vs. σ per Memory Module at 16x16 switch with 32 memory modules using i parameter assignment algorithm 3

Table 4.17 Worst-case ratio of multiple packets destined to same memory module at σ = 2048 in figure 4.26

	Ratio of multiple packets in one memory module		
Number of packets in one memory module	90 % Offered Load at σ = 1024	95 % Offered Load at σ = 1024	100 % Offered Load at σ = 1024
2 packets	1.5×10^{-1}	1.6×10^{-1}	1.8×10^{-1}
3 packets	3.2×10^{-2}	3.9×10^{-2}	5.9×10^{-2}
4 packets	5.7×10^{-3}	8.2×10^{-3}	1.6×10^{-2}
5 packets	8.9×10^{-4}	1.5×10^{-3}	3.8×10^{-3}
6 packets	1.2×10^{-4}	2.4×10^{-4}	7.8×10^{-4}
7 packets	1.5×10^{-5}	3.4×10^{-5}	1.5×10^{-6}
8 packets	1.3×10^{-6}	4.2×10^{-6}	2.5×10^{-5}
9 packets	0	4×10^{-7}	3.5×10^{-6}
10 packets	0	6.6×10^{-8}	6.9×10^{-7}
11 packets	0	0	1.3×10^{-7}

traffic load = 95 % and σ = 2048,8 times faster than the line speed in order to write 8 packets in one memory module when traffic load is 90 % and σ = 2048.

Figure 4.27 shows packet loss ratio versus number of memory Locations (σ) per memory for 16x16 switch with 32 memory modules. Packet loss ratio pattern using algorithm 3 is same as packet loss ratio pattern using algorithm 1 and algorithm 2. Because packet loss ratio is independent of i parameter assignment algorithm. It is also observed that increase in switch size results increase in memory-write speed requirement

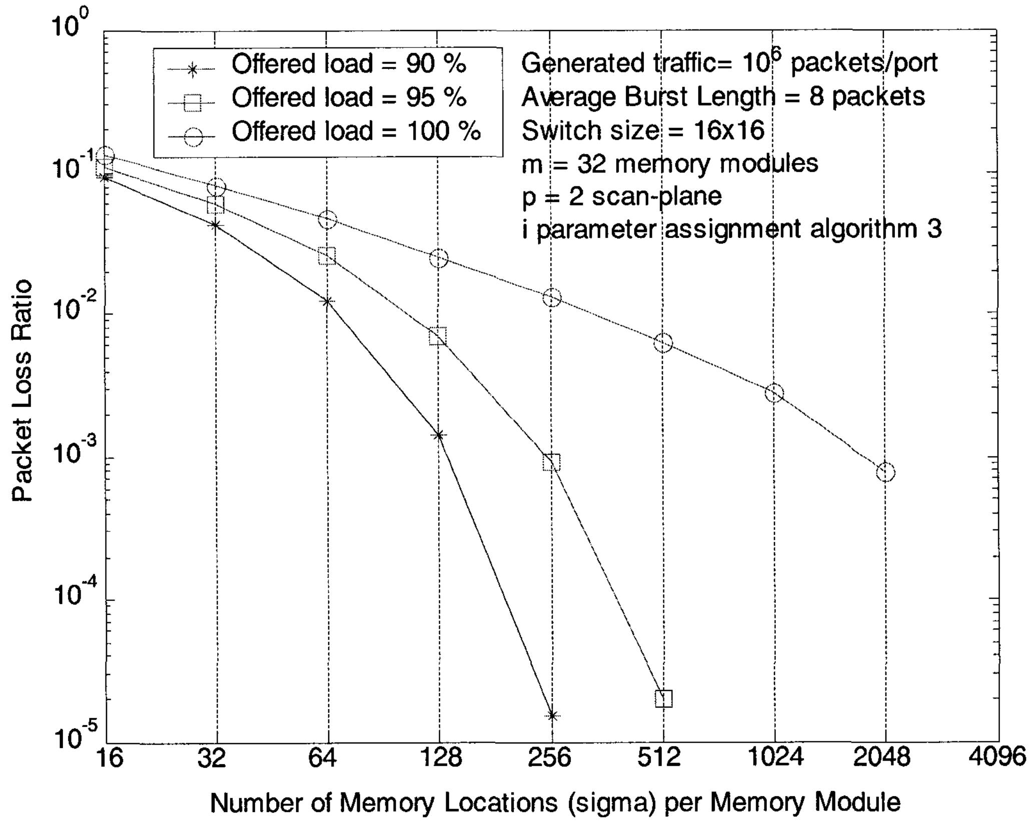


Fig. 4.27 Packet Loss Ratio vs. σ per Memory Module at 16x16 switch with 32 memory modules using i parameter assignment algorithm 3

such as, 4 times faster memory-write speed for 4x4 switch size, 8 times faster memory-write speed for 8x8 switch size, 10 times faster memory-write speed for 16x16 switch size. There is a way to reduce that memory-write speed and frequency of having multiple packets in one memory module. If we increase number of memory modules, the probability of having multiple packets in one memory would be reduced. This phenomena has simulated in a same switch architecture but increasing the number of memory modules of 8 and average burst length has chosen 8 packets, σ = 16 and 100 %

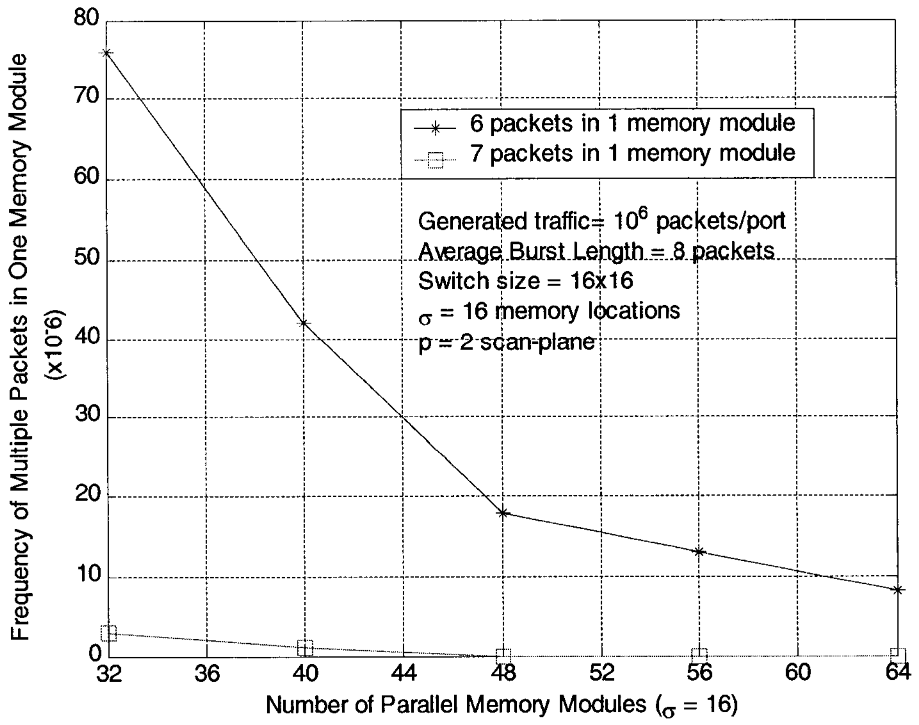


Fig. 4.28 Frequency of Multiple Packets in One Memory Module vs. Number of Memory Modules at 16x16 switch with σ = 16 using i parameter assignment algorithm 3

Table 4.18 Worst-case ratio of multiple packets destined to same memory module at σ = 16 in figure 4.28

	Ratio of multiple packets in one memory module				
Number of packets in one memory module	32 parallel memory modules deployed	40 parallel memory modules deployed	48 parallel memory modules deployed	56 parallel memory modules deployed	64 parallel memory modules deployed
2 packets	1.1×10^{-1}	9.9×10^{-2}	8.7×10^{-2}	7.9×10^{-2}	7.2×10^{-2}
3 packets	1.3×10^{-2}	1×10^{-2}	7.6×10^{-3}	6.2×10^{-3}	4.9×10^{-3}
4 packets	1.2×10^{-3}	8.6×10^{-4}	5×10^{-4}	3.6×10^{-4}	2.5×10^{-4}
5 packets	8.4×10^{-5}	5.9×10^{-5}	2.8×10^{-5}	1.7×10^{-5}	1×10^{-5}
6 packets	5.2×10^{-6}	3.3×10^{-6}	1.5×10^{-6}	7.5×10^{-7}	5×10^{-7}
7 packets	1.3×10^{-7}	6.3×10^{-8}	0	0	0

traffic load. Simulation results are shown in figure 4.28.Table 4.18 shows the frequency of having multiple packets (worst case) in one memory module at 32, 40, 48, 56, and 64 memory modules. Memory-write speed has reduced from 7 times to 6 times when the switch has 48 memory modules instead of 32 memory modules.

4.6. The Effect of the Number of Scan-Planes on Switch Performance

In this section, different i parameter assignment algorithms are used to measure the effects of number of scan-planes on memory bandwidth performance of parameter assignment circuit. In order to measure effects of number of scan-planes on memory-bandwidth requirement of parameter assignment circuit, total capacity of global shared memory has to be same amount using all 3 algorithms. Algorithm 1 is simulated on 16x16 switch where scan-plane = 2, number of memory modules m = 32, number of memory locations per memory module σ = 128, average burst length = 8 packets. In simulation, number of memory locations (σ) per memory module has to be half of previous σ size from 128 to 64, from 64 to 32, from 32 to 16, while number of scan-planes is doubling from 2 to 4, from 4 to 8, from 8 to 16 respectfully.

Figure 4.29 shows that average memory bandwidth requirement is declined from 1.45 to 1 while number of scan-planes are increasing. Because increase in number of memory modules due to increase in number of scan-planes will give enough number of consecutive parallel memory modules that reduce frequency of multiple packets stored in a one memory module.

Table 4.19 shows that memory-write speed must be maximum 3 times faster than the line speed in order to write 3 packets in one memory module when scan-planes = 2,

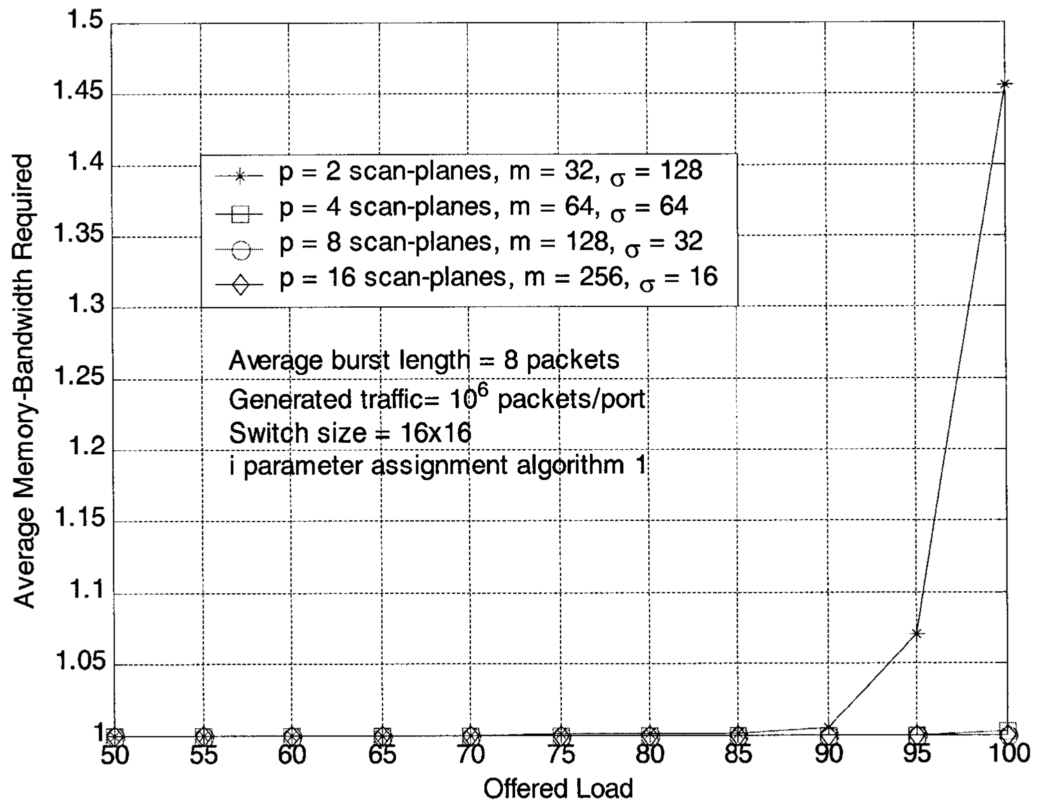


Fig. 4.29 Average Memory-Bandwidth Required vs. Offered Load on 16x16 switch at different number of scan-planes using i parameter assignment algorithm 1

Table 4.19 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.29

Number of SP at 100 % Load	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module
2	4.9×10^{-2}	7.1×10^{-5}	0
4	2.1×10^{-4}	0	0
8	0	0	0
16	0	0	0

number of memory module m = 32, σ = 128 at traffic load = 100 %. Memory-write speed must be maximum 2 times faster than the line speed in order to write 2 packets in one memory module when scan-planes = 4, number of memory module m = 64, σ = 64 at traffic load = 100 %. Memory-write speed can be the same line speed for scan-planes number of 8 and 16.

Algorithm 2 is simulated on 16x16 switch where scan-plane = 2, number of memory modules m = 32, number of memory locations per memory module σ = 128, average burst length= 8 packets. In simulation, number of memory locations (σ) per memory module has to be half of previous σ size, while number of scan-planes is doubling.

Figure 4.30 shows that average memory bandwidth requirement is declined from 1.00014 to 1 while number of scan-planes are increasing. Because increase in number of memory modules due to increase in number of scan-planes will give enough number of parallel memory modules comparing with pre-assigned memory modules that reduces frequency of multiple packets stored in a one memory module.

Table 4.20 shows that memory-write speed must be maximum 2 times faster than the line speed in order to write 2 packets in one memory module when scan-planes = 2, number of memory module m = 32, σ = 128 at traffic load = 100 %. Memory-write speed can be the same line speed for scan-planes number of 4, 8, and 16

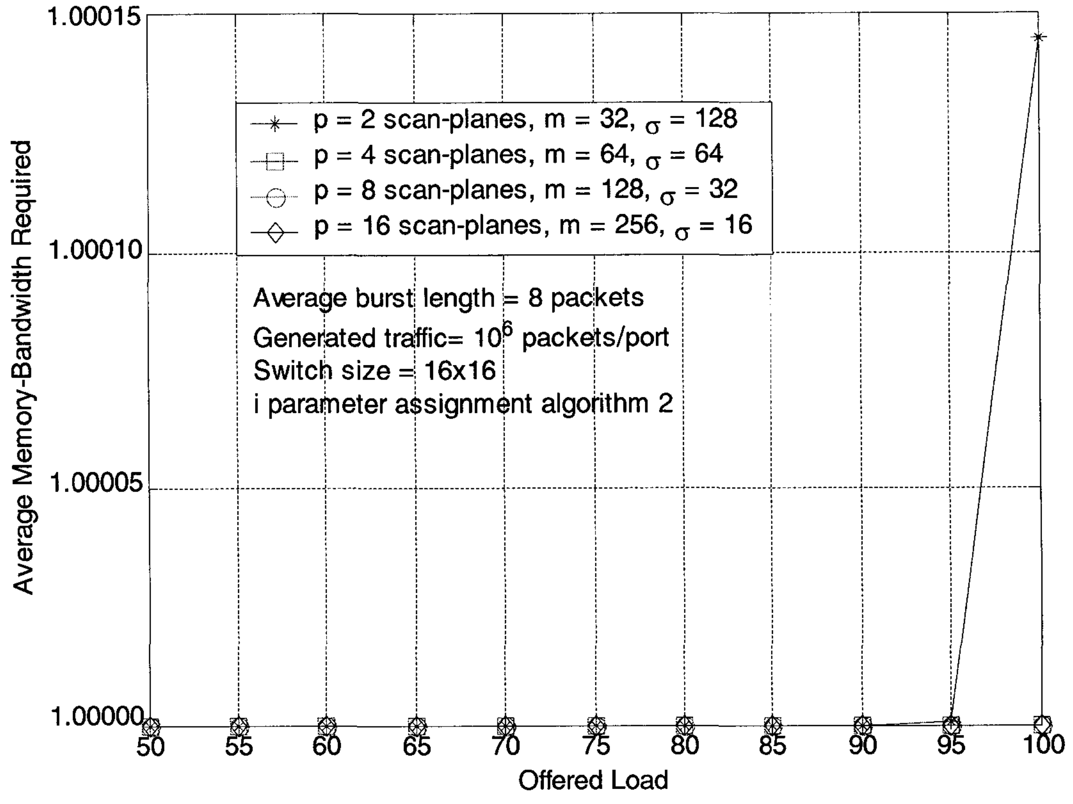


Fig. 4.30 Average Memory-Bandwidth Required vs. Offered Load on 16x16 switch at different number of scan-planes using i parameter assignment algorithm 2

Table 4.20 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.30

Number of SP at 100 % Load	Ratio of 2 packets stored in 1 memory module	Ratio of 3 packets stored in 1 memory module	Ratio of 4 packets stored in 1 memory module
2	8.5×10^{-6}	0	0
4	0	0	0
8	0	0	0
16	0	0	0

Algorithm 3 is simulated on 16x16 switch where scan-plane = 2, number of memory modules m = 32, number of memory locations per memory module σ = 128, average burst length= 8 packets.

Figure 4.31 shows that average memory bandwidth requirement is decreasing significantly, while number of scan-planes are increasing. Because increase in number of scan-planes results lower probability (due to increase in number of memory modules) that multiple packets stored in one memory module using Queue-based algorithm 3.

Table 4.21 shows that memory-write speed must be maximum 9 times faster than the line speed in order to write 9 packets in one memory module when scan-planes = 2.

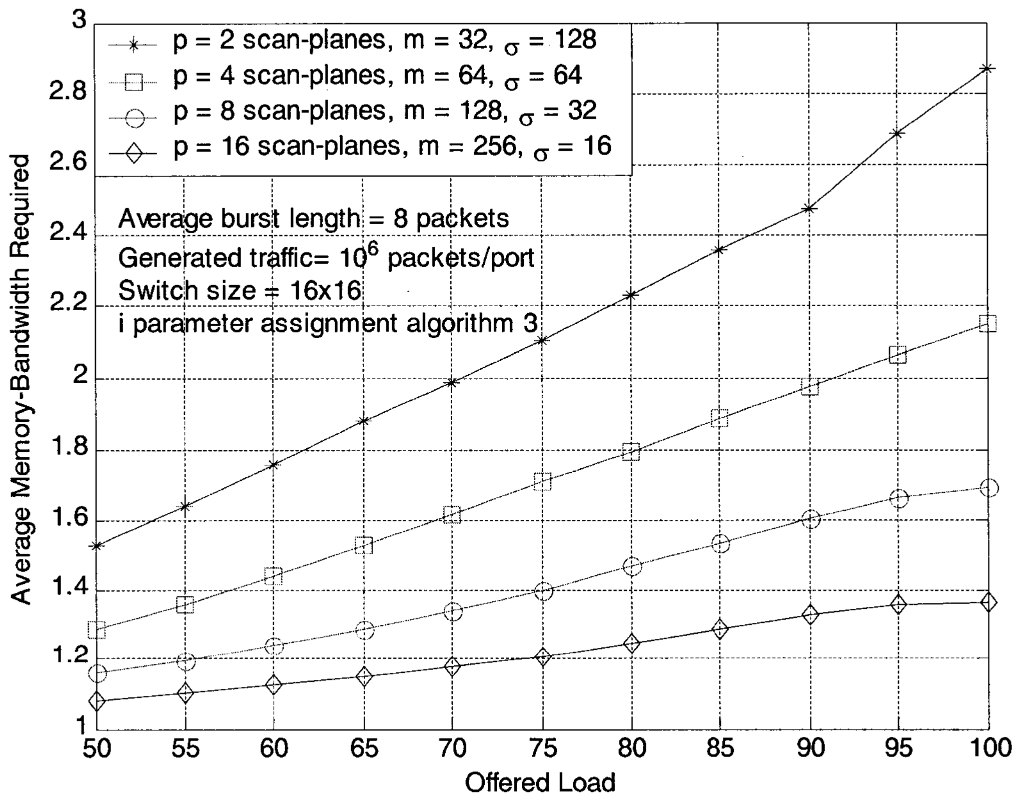


Fig. 4.31 Average Memory-Bandwidth Required vs. Offered Load on 16x16 switch at different number of scan-planes using i parameter assignment algorithm 3

Table 4.21 Worst-case ratio of multiple packets destined to same memory module at 100 % load in figure 4.31

	Ratio of multiple packets in one memory module			
Number of packets in one memory module	p=2 scan-planes at 100 % Load	p=4 scan-planes at 100 % Load	p=8 scan-planes at 100 % Load	p=16 scan-planes at 100 % Load
2 packets	1.6×10^{-1}	1.1×10^{-1}	5.6×10^{-2}	2.6×10^{-2}
3 packets	4.2×10^{-2}	1.2×10^{-2}	2.6×10^{-3}	4.8×10^{-4}
4 packets	9.3×10^{-3}	1.2×10^{-3}	1×10^{-4}	1.3×10^{-5}
5 packets	1.8×10^{-3}	9.6×10^{-5}	8.1×10^{-7}	1.9×10^{-6}
6 packets	3.1×10^{-4}	7.7×10^{-6}	1.9×10^{-7}	0
7 packets	4.6×10^{-5}	3.8×10^{-7}	0	0
8 packets	6×10^{-6}	6.3×10^{-8}	0	0
9 packets	1.2×10^{-6}	0	0	0
10 packets	0	0	0	0

Doubling number of scan-planes from 2 (number of memory module m = 32, σ = 128) to 4 (number of memory module m = 64, σ = 64) results reduce in memory-write speed requirement from 9 times to 8 times faster than line speed to write 8 packets in one memory module. Increasing number of scan-planes from 4 (number of memory module m = 64, σ = 64) to 8 (number of memory module m = 128, σ = 32) reduces memory-write speed requirement to 6 times faster than line speed in order to write6 packets in one memory module. Finally, increase in scan-planes from 8 (number of memory module m =

128, σ = 32) to 16 (number of memory module m = 256, σ = 16) decreases in memory-write speed to 5 times faster than line speed. Total memory spaces used for all different scan-planes are equal (4096-packet) for these simulations. As a result, increase in scan-planes for an equal memory spaces significantly reduces both average memory bandwidth requirement and memory-write speed requirement for all algorithms.

CHAPTER 5

CONCLUSION

In this thesis, we have evaluated the performance of self-routing parameter assignment circuit in sliding-window switch by applying 3 different assignment algorithms. Memory bandwidth requirements for these schemes have been simulated at different traffic burstiness, switch size, number of memory locations per memory module (σ), and the number of scan-planes using different assignment algorithms.

Our simulation results showed that the lowest memory bandwidth requirement can be achieved when algorithm 2 is applied to PAC especially for larger switch size. Algorithm 1 has rather lower overhead compared to algorithm 2. On the other hand, proposed algorithm 3 requires the highest memory bandwidth compared with other algorithms.

Several methods, such as increasing the number of scan-planes, increasing the number of parallel memory modules, etc., are proposed to reduce memory bandwidth requirements. Packet loss ratio for different assignment schemes has also been evaluated. Specialized hardware implementation leads to design of very high capacity sliding-window switching devices.

REFERENCES

[1] A. S. Acampora, An Introduction to Broadband Networks: Lans, Mans, Atm, B-Isdn, and Optical Networks for Integrated Multimedia Telecommunications, Plenum Press, 1994

[2] D. Comer, Computer Networks And Internets, Prentice Hall, Fourth Edition, 2003

[3] H. J. Chao, C. H. Lam, E. Oki, Broadband Packet Switching Technologies: A Practical Guide to ATM Switches and IP Routers, John Wiley & Sons, New York, 2001

[4] P. Newman, "Fast packet switching for broadband ISDN," Telecommunications, Second IEE National Conference on , pp: 391-396, 2-5 April 1989

[5] J. P. Coudreuse and M. Servel, "Prelude: an Asynchronous Time-Division Switch Network," ICC '87 Conf. Rec., paper 22.2, Seattle, WA, June 1987

[6] R. J. McMillen, " A survey of interconnection networks," In Proc. IEEE Globecom, pages: 105-113, Nov. 1984

[7] S. E. Minzer, "Broadband user-network interfaces to ISDN," In Proc. IEEE Int. Conf. Commun. (ICC '87), pages: 11.2.1-6, Seattle, June 1987

[8] J. F. Mollenauer, " Standards for Metropolitan area networks," IEEE Commun. Mag., 26(4), 15-19, Apr. 1988.

[9] M. J. Narasimha, "The Batcher-banyan self-routing network: universality and simplification," IEEE Trans. Commun., 36(10), 1175-1178, Oct. 1988.

[10] H. Ahmadi, W. E. Denzel, C. A. Murphy, E. Port, “A high-performance switch fabric for integrated circuit and packet switching,” In Proc. IEEE Infocom, pages: 9-18, N. Orleans, 1988

[11] M. Ajmone, A. Bianco, E. Leonardo, “RPA: a simple efficient and flexible policy for input buffer ATM switches,” IEEE Lett., vol. 1 no.3, pp. 83-86, May 1997

[12] H. J. Chao, J. S. Park, “Centralized contention resolution schemes for a large-capacity optical ATM switch,” Proc. IEEE ATM Workshop, Fairfax, VA, May 1998.

[13] H. J. Chao, “Satrun: a terabit packet switch using dual round-robin,” IEEE Commun. Mag., vol. 38, no. 12, pp. 78-79, Dec 2000.

[14] A. Charny, P. Krishna, N. Patel, R. Simcoe, “Algorithm for providing bandwidth and delay guarantees in input-buffered crossbar with speedup,” Proc. IEEE IWQoS ’98, pp 235-244, May 1998.

[15] R. Guering, K. N. Sivarajan, “Delay and throughput performance of speeded-up input queueing packet switches,” IBM Research Report RC 20892, Jun. 1997.

[16] M. G. Hluchyj, M. J. Karol, “Queueing in high-performance packet switching,” IEEE J. Select Areas Commun., vol. 6, no.9, pp. 1587-1597, Dec. 1998.

[17] S. C. Liew, “Performance of a various input-buffered and output-buffered ATM switch design principles under bursty traffic: simulation study,” IEEE Trans. Commun., vol. 42, no. 2/3/4, pp. 1371-1379, Feb./Mar./Apr. 1994.

[18] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri, “Packet scheduling in input-queued cell-based switches,” IEEE J. Select. Areas Commun., 1998.

[19] N. McKeown, “Scheduling algorithms for input-queued cell switches,” Ph.D.

thesis, University of California at Berkeley, 1995.

[20] A. Mekkitikul, N. McKeown, “A starvation-free algorithm for achieving 100% throughput in an input-queued switch,” in Proc. ICCCN’96, 1996.

[21] A. Huang, S. Knauer, “Starlite: A wideband digital switch,” In Proc. IEEE Globecom, pp. 121-125, Nov. 1984.

[22] M. G. Hluchyj, M. J. Karol, “Queueing in space division packet switching,” In Proc. IEEE Infocom, pp. 334-343, New Orleans, March 1988.

[23] Y. S. Yeh, M. G. Hluchyj, A. S. Acampora, “The Knock-out switch: A simple modular architecture for high performance packet switching,” IEEE J. Select. Areas Commun., SAC-5 (8), 1274-1283, Oct. 1987.

[24] R.Y. Awdeh, “Why fast packet switching ?,” Potentials, IEEE ,volume 12, issue: 2, pp. 10-12, April 1993.

[25] M. De Prycker, Asynchronous Transfer Mode: Solution for Broadband ISDN, Second Edition (Chichester, England: Ellis Horwood, 1993)

[26] J. P. Coudreuse and M. Servel, “Prelude: an Asynchronous Time-Division Switch Network,” ICC ’87 Conf. Rec., paper 22.2, Seattle, WA, June 1987

[27] J. Garcia-Haro, A. Jajszczyk, “ATM shared-memory switching architectures” Network, IEEE, Volume:8, Issue:4, Pages:18-26, July-Aug.1994

[28] H. Lee, K. H. Kook, C. S. Rim, K. P. Jun, an S. K. Lim, “A Limited Shared Output Buffer Switch for ATM,” Fourth International Conf. On Data Communications Systems and Their Performance, Barcelona, pp. 163-179, June 1990.

[29] H. Kitamura, "A Study on Shared Buffer-Type ATM Switch," Electronics and Communications in Japan, Part 1, vol. 73, no. 11, pp. 58-64, 1990

[30] K. Oshima, H. Yamanaka, H. Saito, H. Yamada, S. Kohama, H. Kondoh, and Y.Matsuda, "A new ATM switch architecture based on STS-type shared buffering and its LSI implementation," Proc IEICE, pp.359-363, 1992.

[31] K. Yamanaka et al., "Scalable shared-buffering ATM switch with a versatile searchable queue," IEEE J. Select. Areas Commun., vol. 15, pp. 773–784, June 1997.

[32] S. Kumar, "The Sliding-Window Packet Switch: A New Class of Packet Switch Architecture With Plural Memory Modules and Decentralized Control," IEEE Journal on Selected Areas in Communications, vol. 21, no. 4, May 2003

[33] S. Kumar and D. P. Agrawal, "The sliding-window approach to high performance ATM switching for broadband networks," in Proc. IEEE GLOBECOM, London, U.K., Dec. 1996, pp. 772-777.

[34] "IA-32 Intel Architecture Optimization Reference Manuel," Intel Corporation, 2003, world-wide-web: http://developer.intel.com

[35] Stephan D. Brown, Svonko G. Vranesic, Fundamentals of Digital Logic with VHDL Design with CD-ROM, Mc Graw-Hill, 1999

[36] Delgado-Frias, J.G., Nyathi,J., "A VLSI high-performance encoder with priority lookahead VLSI," Proceedings of the 8th Great Lakes Symposium on, 19-21 Feb. 1998 Pages: 59 - 64

[37] S. Kumar, T. Doganer, and A. Munoz, "Effect of traffic burstiness on memory-bandwidth of the sliding-window switch architecture," 3rd International

Conference on Networking (ICN'2004), Gosier, Guadeloupe, French Caribbean, vol. I, pp. 147-151, Feb-Mar 2004

[38] S. Kumar, A. Munoz, and T. Doganer, "Performance Comparison of memory-sharing schemes for Internet switching architecture," 3rd International Conference on Networking (ICN'2004), Gosier, Guadeloupe, French Caribbean, vol. I, pp. 160-163, Feb-Mar 2004

[39] S. Kumar, T. Doganer, "Memory-Bandwidth Performance of the Sliding-Window based Internet Routers/Switches," Accepted in IEEE Workshop on Local and Metropolitan Area Networks, San Francisco, CA, April 2004.

VITA

Taner Doganer was born in Mersin, Turkey on December 18, 1975 as the son of Meryem Doganer and Yasar Doganer. He graduated from Tevfik Sirri Gur High School, Mersin, Turkey in 1992. Later he was enrolled in Collage of Engineering, Karadeniz Technical University for Electrical & Electronics degree and graduated as electronic engineer in 1997. He worked Mavili Electronics Co. in Istanbul as an engineer in Istanbul, Turkey in 1998. Later he joined to Turkish Army in Ankara for military obligation in 1999, and completed as a second lieutenant in 2000. In September 2001 he entered the Graduate school of the University of Texas-Pan American in Electrical Engineering. Currently he is working as network administrator with responsibility to manage local area networks in a Rio Grande Valley area company.