DESIGN, MODELLING AND ANALYSIS OF THE BALANCED GAMMA MULTICAST SWITCH FOR BROADBAND COMMUNICATIONS

CHENG LI

Design, Modelling and Analysis of the Balanced Gamma Multicast Switch for Broadband Communications

by

©Cheng Li, B.Eng., M.Eng.

A thesis submitted to the School of Graduate
Studies in conformity with the requirements for the
Degree of Doctor of Philosophy

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

April 2004

St. John's Newfoundland Canada

Abstract

High-speed networks have become more and more popular worldwide driven by the Internet and its applications. Multicast has become a necessary feature for any switch designed for future broadband communication networks. In this dissertation, a multicast switch architecture called the Balanced Gamma (BG) multicast switch is proposed, analyzed and implemented. A comprehensive study of this promising multicast switch architecture has demonstrated its superiority in terms of loss performance, delay performance, and buffer requirement performance under various uniform and nonuniform traffic. At the same time, it is scalable, reliable, and fault-tolerant, and its hardware complexity is reasonably low which makes it feasible to build as a practical switch.

The new multicast BG switch fabric is characterized by its space-division architecture in which the control of cell routing is distributed over all switch elements. The key characteristic of a multicast switch, the cell replication function, is integrated into the routing function of the switch element. Two new algorithms are designed to support implicit cell routing and replication, namely the dynamic-length routing and replication algorithm and the dynamic-length backpressure algorithm. Topological equivalence to the unicast BG switch ensures that the new architecture inherits many attractive features such as reliability and fault tolerance from the latter.

A multicast traffic model is developed for the analysis of the multicast BG switch. The performance of the switch is examined under various traffic conditions, random and bursty, uniform and nonuniform. Numerous simulation trials are performed to obtain the loss, delay, and buffer requirement performance of the switch. An analytical model is derived under the multicast random traffic model to verify our simulation results. The discrepancy between the analytical model and simulation is justified and

further improvement of the model is suggested. Performance results are also compared to that of the ideal multicast switch to demonstrate how close the performance of the BG multicast switch is to the optimum result. It is determined through the analysis that the multicast BG switch is a high performance switch in handling unicast, multicast, and mixed traffic. At the same time, it is scalable in terms of architecture, performance, and implementation.

Following the digital IC design methodology recommended by the Canadian Microelectronics Corporation (CMC) and using the VLSI CAD tools they have provided, a 16×16 switch fabric has been implemented using $0.18\mu m$ CMOS technology and the Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL). It is demonstrated that the whole switching fabric module can be easily fit into a single IC chip with the current fabrication technology. Finally, the methodology of functional verification of the hardware design is presented and hardware complexity of larger switches is explored.

Acknowledgments

For their continuing guidance, support and encouragement throughout my study at Memorial, I would like to thank my supervisors Dr. *R. Venkatesan* and Dr. *H. Heys*. I greatly appreciate the freedom and collegial respect they have given me. Numerous long discussions with them led to my comprehensive understanding of the multicast switching that have provided the main body of this thesis.

I am grateful to my former master's program supervisor, Dr. *Naitong Zhang*, who educated me most with his abundant engineering knowledge and encouraged me to pursue a Ph.D. abroad. Also I would like to express my sincere gratitude to Dr. *Cecilia Moloney*, who is on my supervisory committee, and Dr. *Paul Gillard* for their kindly help and support.

I like to thank the School of Graduate Studies at the Memorial University of Newfoundland for their financial support during my Ph.D. program.

I will always remember the suggestions and help from Mr. *Nolan White* in the Department of Computer Science who offered me the pool of computational facilities for my simulations and fixed the problems related to the VLSI CAD tools. Without his help, I would probably still be struggling in the darkness of running numerous simulation trials.

I would also like to thank members of the Computer Engineering Research Laboratories (CERL) and the Center for Digital Hardware Applications Research (CDHAR) which provided a friendly and resourceful environment for conducting my research. I am grateful to *Reza Shahidi* and Dr. *Charles Robertson* who helped me to solve many computer problems I came across. Thanks especially to Dr. *Prashant Mehrotra*

and Dr. *Yaser El-Sayed* who encouraged me to work on the multicast switching for my Ph.D. research.

Far too many friends to mention individually have assisted in so many ways during my work and life. They all have my sincere gratitude. In particular, I would like to thank Dr. *Eric Gill*, Dr. *Benjamin Jeyasurya* and their families. I would also like to thank *Bo Liang, Wei Song, Dongmei Wu, Lu Xiao, Janaka Deepakumara, Qiyao Yu, Yingzi Wang, Zhiwei An, Padmini Vellore, P. Sainath, Andrew Cook, Jason Rhinelander, Atiq Awan, Yassir Nawaz*, and many friends whose names are not mentioned.

Last, but definitely the most, I would like to thank my family. Words cannot express my thanks to my dear wife *Fang Yang* for all her love, care and encouragement. I also owe a lot to my parents and sisters for their continuous support and great sacrifices in making this work a reality. I dedicate my thesis to them.

Contents

List of Figures

List of Tables

Notation and List of Abbreviations

α	: Switching Parameter Used by the Original BG Network
λ	: Optical Wavelength
ρ	: Switch Input Traffic Load
ρ_{in}	: Switch Fabric Input Link Traffic Load
ρ_{out}	: Switch Fabric Output Link Traffic Load
a_i	: Probability of i Cells Request the Same Output Link
k_i	: Fanout Factor for Multicast Cells in Stage i
n	: Number of Stages in a MIN
$\overline{n}_o$	: Average Output Queue Occupancy
$\overline{n}_{in}$	: Average Input Queue Occupancy
$p_a(i)$	: Load on the Upper Regular Output Link of SEs at Stage i
$p_b(i)$	: Load on the Upper Alternative Output Link of SEs at Stage i
$p_c(i)$	: Load on the Lower Regular Output Link of SEs at Stage i
$p_d(i)$	: Load on the Lower Alternative Output Link of SEs at Stage i
P_{blk_i}	: Blocking Probability of SEs in Stage i
$P_{blk_{OQ}}$	: Blocking Probability of Output Queue
$P_{blk_{SF}}$	: Blocking Probability of Switch Fabric
p_{ij}	: State Transition Probability From State i to State j
π_i	: Probability of Being in State i
t_{ij}	: Probability of Cell Arrived at i is Destined to Output j
A_m	: Number of Cell Arrivals During the mth Slot
B_o	: Output Queue Size
B_{in}	: Input Queue Size
D_m	: Number of Cells Departs During the mth Slot
$\overline{F}$	: Mean Multicast Fanout
$\overline{L}$	: Average Bursty Length
N	: Number of Inputs/Outputs in a Rectangular Switching Network
$\overline{T}_o$	: Average Output Queueing Delay
$\overline{T}_{in}$	: Average Input Queueing Delay
$\overline{T}$	: Average Cell Queueing Delay
Q_m	: Number of Cells in Output Queue at the End of the mth Slot

ACU	: Admission Control Unit
AF	: Address Filter
AOG	: Acknowledgement Output Generation Circuitry
ATM	: Asynchronous Transfer Mode
BCU	: Backward-path Control Unit
BIG	: Blocking Information Generation Unit
B-ISDN	: Broadband Integrated Services Digital Network
BIST	: Built-In Self-Test
CAC	: Connection Admission Control
CAD	: Computer Aided Design
CDPD	: Cellular Digital Packet Data
CCU	: Central Control Unit
CMOS	: Complementary Metal Oxide Semiconductor
CSIX	: Common Switch Interface
DFT	: Design For Testability
DF	: Design Framework
DWDM	: Dense Wavelength Division Multiplexing
DRC	: Design Rule Checking
DSM	: Deep Sub-Micron
EFCI	: Explicit Forward Congestion Indication
FCFS	: First-Come-First-Served
FCU	: Forward-path Control Unit
FIFO	: First-In First-Out
FRBS	: Frame Relay Bearer Services
GFC	: Generic Flow Control
GGSN	: Gateway GPRS Support Node
GPRS	: General Packet Radio Service
GPS	: Growable Packet Switch
HOL	: Head Of Line
IETF	: Internet Engineering Task Force
IN	: Interconnection Network
IP	: Internet Protocol
IPC	: Input Port Controllers
IPv4	: Internet Protocol version 4
IPv6	: Internet Protocol version 6
ITU-T	: International Telecommunications Unit-Telecommunications Standardization Sector
LAN	: Local Area Network
LANE	: LAN Emulation
LVS	: Layout-Versus-Schematic
MAN	: Metropolitan Area Network

MBone : Multicast Backbone
MEMS : Micro-Electro-Mechanical System
MGN : Multicast Grouping Network
MIN : Multistage Interconnection Network
MOBAS : Multicast Output Buffered ATM Switch
MoD : Music On Demand
MPLS : Multi-Protocol Label Switching
MPOA : MultiProtocol Over ATM
MTTF : Mean Time To Failure
NPF : Network Processing Forum
NMC : Network Main Controller
OEM : Original Equipment Manufacturer
OEO : Optical-Electrical-Optical
OOO : Optical-Optical-Optical
OPC : Input Port Controllers
OXC : Optical Cross Connect
PDP : Physical Design Planner
QoS : Quality of Service
RM : Routing Module
RTL : Register Transfer Level
SCCQ : Shared Concentration and Output Queueing Switch
SE : Switch Element
SF : Switch Fabric
SGSN : Serving GPRS Support Node
SN : Single-Stage Network
SM : System Management
SONET : Synchronous Optical Network
SWE : Switch Elements Array
TNT : Trunk Number Translation table
UPC : Usage Parameter Control
VCI : Virtual Channel Identifier
VHDL : VHSIC Hardware Description Language
VHSIC : Very High Speed Integrated Circuit
VLSI : Very Large Scale Integrated Circuit
VoD : Video On Demand
VPI : Virtual Path Identifier
WAN : Wide Area Network

Chapter 1

Introduction

Communication network applications are changing at an enormous speed. As shown in Figure 1.1, user traffic on the Internet has been doubling every year for the past 20 years and the trend is still going up [1]. However, the capacity of switching devices, such as switches or routers, grows at a speed that lags behind, and this trend makes the deployment of next generation high-speed networks necessary and urgent.

Packet switching technologies have been intensively investigated over the past two decades. As the Internet traffic and applications grow exponentially, there is a great need to build switching devices of large capacity and high speed, such as Internet Protocol (IP) routers, Asynchronous Transfer Mode (ATM) switches, and Multi-Protocol Label Switching (MPLS) switches [2].

With recent advances in fiber optics and optical transmission technologies, network operators can deploy huge capacity on transmission links. For example, 128 OC-192 channels (10 Gigabit/s) can be multiplexed over a single fiber to achieve a total link capacity of 1.2 Terabit/s using Dense Wavelength Division Multiplexing (DWDM) [2]. At the same time, the advances in Micro-Electro-Mechanical System (MEMS) technology, in which an array of hundreds of electrically configured microscopic mirrors is fabricated into a single substrate to direct light, has resulted in the deployment of the Optical Cross Connect (OXC) system which can achieve terabit

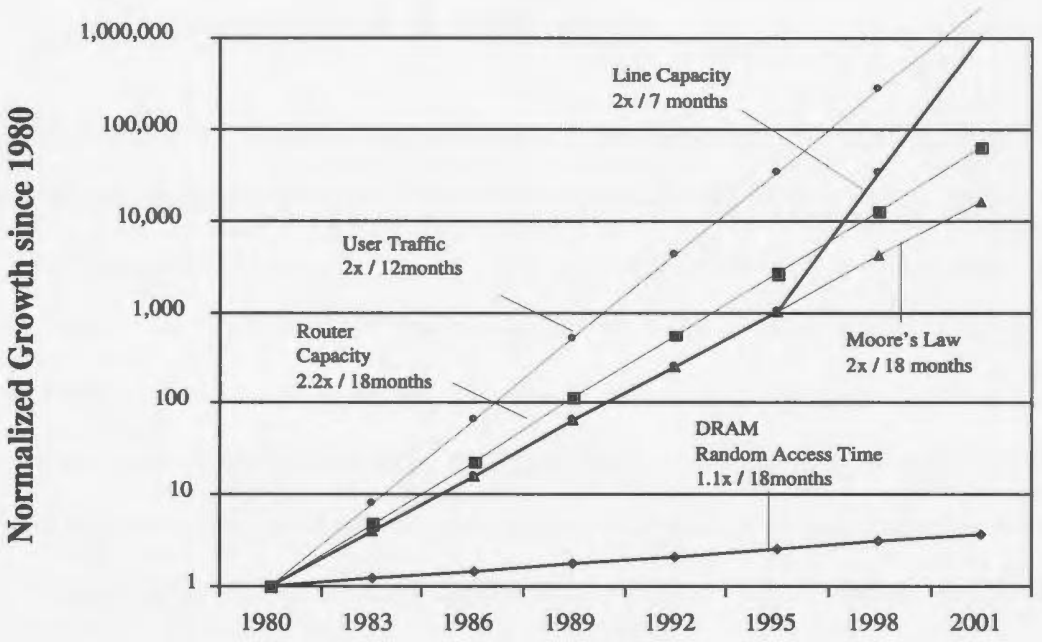


Figure 1.1: Trend of growth of network traffic and devices (modified from [1], pp. 1)

switching.

Although optical switching can achieve very high speed and huge switching capacity, the limitation in the current optical technology determines that they can only work at around the wavelength level, i.e., around 10 to 40 Gigabit/s level. That means the granularity of the switching is coarse [2]. As well, optical switching dose not possess as much intelligence as in the electronic switching. As a result, high-speed and large-capacity electronic packet switches/routers are required to provide switching as well as aggregating lower-bit-rate traffic to the high-speed Gigabit/s links.

This dissertation addresses the architecture, design, performance analysis, and VLSI implementation of a high-speed, large-capacity, and highly scalable multicast switching fabric. The proposed architecture can be used in ATM switches, MPLS switches, or high-end IP routers.

1.1 Background of ATM and IP

ATM technology has emerged in recent years as the most promising switching and transport technology, which combines the advantages of both circuit-switching and packet-switching techniques. It can carry a mixture of traffic including audio, video, and data, and can provide network resource, such as bandwidth, on demand [13]. ATM networks are based on some important concepts: virtual circuits, fixed-size packets, small packet size, statistical multiplexing, and integrated service [14].

ATM is a connection-oriented switched-path protocol. A path must be established for each connection before the real data transmission. It is fundamentally a point-to-point protocol. Packets transported over ATM networks are called cells which are characterized by their fixed-length. Each cell is comprised of 5 bytes of header and 48 bytes of payload, which makes a total of 53 bytes. The fixed-size cell structure helps to reduce the delay and jitter performance for audio and video traffic, and also helps to simplify the design of a switch. Broadcasting or multicasting in ATM networks has to be achieved by using point-to-multipoint or multipoint-to-multipoint connections. Resource reservation, traffic control and the ability to guarantee the Quality of Service (QoS) are emphasized in ATM. The maximum cell rate, maximum cell delay, average cell delay, delay jitter and cell loss probability are parameters used to characterize QoS. Because of the diverse service classes that ATM is designed to handle and the different network domains that ATM will be applied to, complicated congestion and flow control are included in ATM technology.

In ATM, congestion can be reduced or mitigated by either impeding the traffic entering the network until the resources become available or by dropping low priority traffic and allowing the higher priority traffic to get through. Nine mechanisms are used by ATM switches to handle congestion and flow control, and they are: Connec-

Attribute	IP	ATM
Orientation	Connectionless	Connection-Oriented
Packet Size	Variable	Fixed (53 bytes)
QoS	No	Yes
Information	Data	Data, voice, and video
Path Determination	Per Packet	Connection setup
Forwarding State	All possible networks	Local active transit connections
Forward Based	Longest match address prefix	Fixed-length label
Signalling	No	Yes

Table 1.1: Comparison of fundamental issues of IP and ATM (from [12], pp. 10)

tion Admission Control (CAC), Usage Parameter Control (UPC), Selective Cell Discarding, Traffic Shaping, Explicit Forward Congestion Indication (EFCI), Resource Management using Virtual Paths, Frame Discard, Generic Flow Control (GFC) and ABR Flow Control [14, 15].

Finally, ATM is flexible because it is not designed for any specific type of physical transport media, such as twisted pair, coaxial cable, or fiber optics [13].

As a routing protocol, the IP protocol can scale up easily due to the broadcast nature. It is initially designed for a broadcast media such as a shared bus, therefore, multicasting is a natural part of IP. Along with the growth of the Internet, IP has become the dominant protocol in data traffic. The IP packet length is variable, varying from 12 bytes to 64k bytes. It is a connectionless datagram protocol in which best-effort service is normally considered. IP is normally paired with TCP protocol to ensure reliability. Two well-known problems, IP address exhaustion and lack of QoS, exist in the Internet Protocol version 4 (IPv4), which is currently used by many network routers and applications. A new IP standard, IPv6, has been proposed to tackle these problems along with many other enhancements [13]. Table 1.1 points out several fundamental differences between IP and ATM [12].

Because of the wide existence of IP in data traffic, the effort from ATM proponents

is towards integrating the IP routing protocol into ATM switching networks. The typical examples include classical IP, LAN emulation (LANE) and Multi-protocol over ATM (MPOA) from the Internet Engineering Task Force (IETF), the Internet standards organization. At the same time, the improvement effort from IP proponents never stalls. In 1996, an revolutionary ideal called IP switching was introduced by a small startup company named Ipsilon from California [12]. Its IP switch binds an IP router processor with an ATM switch, removes all of the switch's ATM signalling and routing protocols, which enables the IP router processor to control the attached ATM switch and allows the whole switch to run IP routing protocols and perform normal hop-by-hop IP forwarding [12]. This concept was startling in its simplicity and elegance. Later on, the crucial influence of ATM leads to the emergence of MPLS, which combines the advantages of the ATM network, i.e., short labels and explicit routing, and the connectionless datagram of the IP network. A label is attached to each IP frame and this label guides the frame through the network. Each intermediate node performs both layer 2 and layer 3 switching. Therefore, with MPLS, circuit type switching is merged into IP packet forwarding, thus allowing traffic control, service segregation and QoS control while retaining the fundamental advantages of IP. The MPLS technology was derived in part from Cisco's Tag switching.

Although there were once heated debates about the pros and cons of IP versus ATM as the next-generation networking technology, it is now clear that IP will be present in all networks for the foreseeable future [12, 16]. The transport efficiency of optical networking has led people to start thinking about running MPLS and IP directly over DWDM, instead of first layering over ATM. The success of gigabit ethernet switches further makes some people argue that as technology moves on, ATM could be left behind ultimately [16]. However, the fact is that the current backbone network mainly consists of ATM switches and IP routers, where the ATM cells and

IP packets are carried over an optical physical layer network such as the Synchronous Optical Network (SONET). In the foreseeable future, ATM based networks will likely coexist with other wide area networking (WAN) technologies such as the TCP/IP Internet and frame relay bearer services (FRBS) [17]. However, the method of cell switching, which was initially introduced by ATM, has been widely accepted and is now used in IP routers and switches.

Low-end and middle-size IP routers use central CPU and shared bus structures while high-end routers are normally comprised of route controller, forwarding engine, switch fabric, and output port scheduler to achieve the speed and capacity requirement [2]. Switching operation is done through the switch fabric. The structure of the high-end IP router is very much like that of the ATM switch, which consists of input port controllers, switch fabric, and output port controllers. The main difference between ATM switches and IP routers is in their line cards. Hence, both ATM switches and IP routers can be constructed using a common switch fabric with suitable line cards [2]. The general switch model will be discussed in the next section.

To achieve high capacity, cell-based switch fabric, although not necessary the 53-byte ATM cell, is highly desired because it is easier for arbitration, data transmission, and other functions when switching operation is organized in synchronized time slot called switching cycle. Hardware implementation and buffer management will be simpler as well [2]. Variable-length packets in IP routers, Ethernet switches, and frame relay switches can be segmented into cell first at the inputs, switched through the fabric and then reassembled back to the original packets at the output [2].

It seems that the trend in the networking industry is toward switching rather than routing. Being the core of the whole routing process, the architecture design and performance of switches will directly or indirectly affect the performance of the whole network.

Figure 1.2: $N \times N$ packet switch reference model

1.2 Switch Model

Figure 1.2 provides the general model of an $N \times N$ packet switch [13, 18]. This reference switch includes three major components: N input port controllers (IPCs), N output port controllers (OPCs), and an $N \times N$ switch fabric (SF) formed by an interconnection network (IN). Hardware implementation for these components is normally required to meet the high-speed requirement. Two important blocks that are not shown in the figure are connection admission control (CAC) and system management (SM), which are normally implemented in software.

To allow the transfer of more than one packet from the source to the same destination within one switching cycle, the destination component should run at a higher speed than each of the incoming links and the ratio is called the speed-up. Many SFs adopt a multistage arrangement of simple switching element (SE) which provides the path for the incoming packets to their requested outputs. Queueing of the switch can be realized at the IPC, SF, OPC, or any combination of these. Further, queueing in-

side SF is normally achieved by buffers in the SEs, which can be at the input, output or shared by both input and output.

Two types of blocking might occur during the switching operation, internal blocking and output blocking [18]. The former one occurs when two cells compete for the same internal link, whereas the latter takes place when more packets are switched to the OPC than it can accommodate. Two schemes can be used to handle the blocking situation: backpressure and queue loss [18]. In backpressure, blocked cells will be kept at the sender. This might cause the congestion problem in the downstream network to spread to upstream network segments. While in the queue loss strategy, blocked cells will be simply dropped. It relies on the integrity features in networks to retransmit the cell automatically. Retransmissions will inevitably increase network traffic, which is not optimal. So switch vendors use large buffers and advise network managers to design switch network topologies properly to eliminate the source of the problem – congested segments.

The main functions of the two port controllers (IPC and OPC) include [13, 18]:

1. Rate matching between the input/output links and switch fabric.
2. Aligning cells for switching (IPC) and transmission (OPC).
3. Processing received cells according to the supported protocols.
4. Attaching (IPC) and stripping (OPC) internal routing tag to each cell.
5. For a bufferless SF, IPC stores the packets and probes the availability of an internal path through the SF. It also checks the availability of the output resource and makes the corresponding decision.

It is desired that a switch architecture has low complexity, large capacity, high throughput, small packet delay and low packet loss. In the real world, it is always a balance and tradeoff among these factors for a practical switch design.

1.3 Unicast and Multicast

In group communications, various types of comınunication methods can be differentiated based on the number of senders and receivers involved. The basic types of communications include Unicast $(1:1)$, Multicast $(1:n)$, Concast $(m:1)$ and Multipeer/Multipoint $(m:n)$ [19]. Unicast is equivalent to the traditional point-to-point communication in which there is exactly one source and one receiver. Bi-directional data exchange between two parties is also considered to be unicast. In multicast communication, a single source transmits data to a group of receivers. Multicast constitutes an extension of unicast and is referred to as a $1:n$ communication. Both unicast and broadcast can be viewed as an extreme case of multicast. Concast is a method that has several sources sending data to a single receiver unidirectionally. Multipeer communication takes place when several sources send data to the same set of receivers, which corresponds to an $m:n$ communication [19].

Multicasting is necessary for applications in which packets need to be sent to more than one destination. Multicasting in communications network can be achieved by replicating the original message at the source node and sending the same unicast message to each of the destinations. However, that is not desired because it will inevitably increase the network traffic load. Ideally with multicast, a single packet needs to be transmitted to the network and the packet is replicated by intermediate nodes only when necessary. In this case, much network bandwidth can be conserved through multicast transmission and it is much more efficient for information distribution.

This efficiency becomes extremely important as multimedia communications gets more and more popular. The network control information, such as network signalling, needs to be distributed to all network nodes and it happens all the time during the normal network operation. Multicast switch would be ideal to handle this kind of data

traffic. Also, typical application examples include e-mail services, video on demand (VoD), music on demand (MoD), remote diagnostic, and teleconferencing services, in which the audio and video signals are captured, compressed and transmitted to a group of receivers throughout the network. Another reason for multicast is the flexibility in joining and leaving a group provided by multicast can make the membership change much easier to handle at the local switch instead of imposing the management task upon the source node [20].

Multicasting in the Internet is currently handled at a high level. In 1992, a set of interconnected networks with routers capable of forwarding multicast packets were selected for experiment. This multicast experimental network was called Multicast Backbone (MBone), and provided a method of deploying multicast applications. The MBone is essentially a virtual network implemented on top of some portions of the Internet. In the MBone, islands of multicast-capable networks are connected to each other by virtual links called "tunnels". It is through these tunnels that multicast messages are forwarded through non-multicast-capable portions of the Internet [20]. Complicated software functions are required to handle multicast packet forwarding through these tunnels, and the packets have to be encapsulated and decapsulated as well. It is clear that multicasting support from all network nodes is essential to realize network wide point-to-multipoint communication. For high-speed network applications, it is much more efficient to implement multicast function at the switch level in hardware. Multicast has become a necessary feature for any switch designed for future broadband communication networks.

1.4 Motivation For The Research

Service differentiation and dynamic information provisioning require the multicast function be incorporated into the current switching networks. Although some multicast switches are proposed, they all more or less suffer from some problems, such as the overflow, output link contention, and high hardware complexity problem in the dedicated copy network solution [7, 21, 22], or the performance degradation problem in the broadcasting tree/bus based solution [4, 5, 23].

The unicast Balanced Gamma (BG) switch has demonstrated the high-performance, high-reliability and high-scalability through the previous research [14, 24, 25, 26, 27]. Inspired by its high performance, a new switch architecture which supports full implicit multicast cell routing and replication is proposed, analyzed and implemented in this dissertation. For consistency, the new switch is named the multicast BG switch. Topological equivalence between the two switch architectures ensures that the multicast BG switch inherits most of the important features in reliability, fault tolerance, and scalability from the unicast BG switch.

The multicast BG switch follows the space division architecture in which cell routing is distributed across all SEs. The key characteristic of a multicast switch, cell replication, is integrated into the routing function of each SE. Two algorithms are designed to cope with the three-phase switching operation in a multicasting environment. A comprehensive study of this multicast switch architecture is carried out to investigate the performance of the switch under various multicast traffic conditions, random and bursty, uniform and nonuniform. Numerous simulation trials are performed to obtain loss, delay, and buffer requirement performance. To verify our simulation results, an analytical model is developed. As well, performance results are compared to that of the ideal multicast switch under bursty and nonuniform traffic

conditions to demonstrate how close its performance is to the optimum results.

Throughout the analysis, the BG multicast switch demonstrates high performance under various unicast, multicast, and mixed traffic conditions. At the same time, it is scalable in terms of architecture and performance. Therefore, research extends to the VLSI design to investigate the feasibility of building a practical switch using this architecture. The $0.18\mu m$ CMOS technology is used. Modularity and implementation scalability are emphasized during this stage. A testing method and software are developed for functional verification of this complicated switching system.

1.5 Thesis Organization

Having described the motivation for the thesis and the main requirements for high-speed multicast switching, we will present the design, evaluation and implementation of the proposed multicast switch in the next four chapters.

Chapter 2 presents a classification and survey of high-speed switch architectures. The many proposed multicast switches are studied and categorized based on the method used for multicast cell replication. Typical examples are selected from each category to describe the advantage and disadvantage of each design approach. The historic background of the BG network and recent developments in switch fabric benchmarks are also described.

Chapter 3 presents the architecture design of the multicast switch. The architecture is justified by design choices to combat blocking and reduce hardware complexity. The self-routing and replication algorithm and the tag encoding scheme, which must satisfy a distributed control and high-speed requirement, are described in detail. A dynamic-length backpressure algorithm is proposed to cope with the three phase switching operation and to achieve an efficient multicast cell acknowledgement. Other

design features such as service discipline, fault tolerance, reliability and scalability are also investigated.

Chapter 4 studies the performance of the multicast BG switch. A multicast traffic model, which comprises three random processes (arrival, fanout and destination selection), is studied and used for the performance analysis and simulation. A multicast switch simulator is developed to obtain results for loss, delay, and buffer requirement performance. An analytical model is developed under the multicast random traffic. Simulation results are verified and confirmed by using the analytical results. Finally, performance under bursty and non-uniform traffic conditions is investigated and compared with that of the ideal multicast switch.

Chapter 5 describes the VLSI design and implementation of the multicast BG switch fabric using 0.18 μm CMOS technology. A general design methodology recommended by CMC is presented followed by the detailed design of the 16×16 BG switch fabric. Simplified IPCs and OPCs are used for testing and verification purposes. The design process is focused on the basic building component, the 4×4 SE, which is the key element to build a large switching fabric. Synopsys CAD tools are used for the front-end design process and Cadence CAD tools are used for the back-end design process. The hardware description language VHDL is used for describing the design and the C++ language is used to used to generate test vectors and verify functional simulation results.

Finally, the dissertation is concluded in chapter 6. Possible future research opportunities are provided as an indicator for continuing work.

Chapter 2

High-Speed Packet Switches and Multicast Switching Technology

In this chapter, the classification of high-speed packet switches is investigated. The similarities and differences amongst various architectures are highlighted. Two common approaches to construct a multicast switch are studied and compared with an extensive survey of various proposed multicast switches. The historical background of the BG switch is provided for a better understanding of a new multicast BG switch architecture. Recent developments in switch fabric benchmarking are introduced with a focus on how the benchmark standards are related to this research.

2.1 High-Speed Switch Architecture Classification

In the literature, a number of classifications of high-speed packet switches have been discussed [2, 3, 14, 28, 29, 30]. Generally speaking, packet switch architectures can be classified into two major categories based on the switching techniques used: time-division packet switches and space-division packet switches. The time-division switch architecture can be divided into shared-medium type and shared-memory type. The space-division switch architecture can be divided into single-stage network (SN) based switch architectures and multistage interconnection network (MIN) based switch ar-

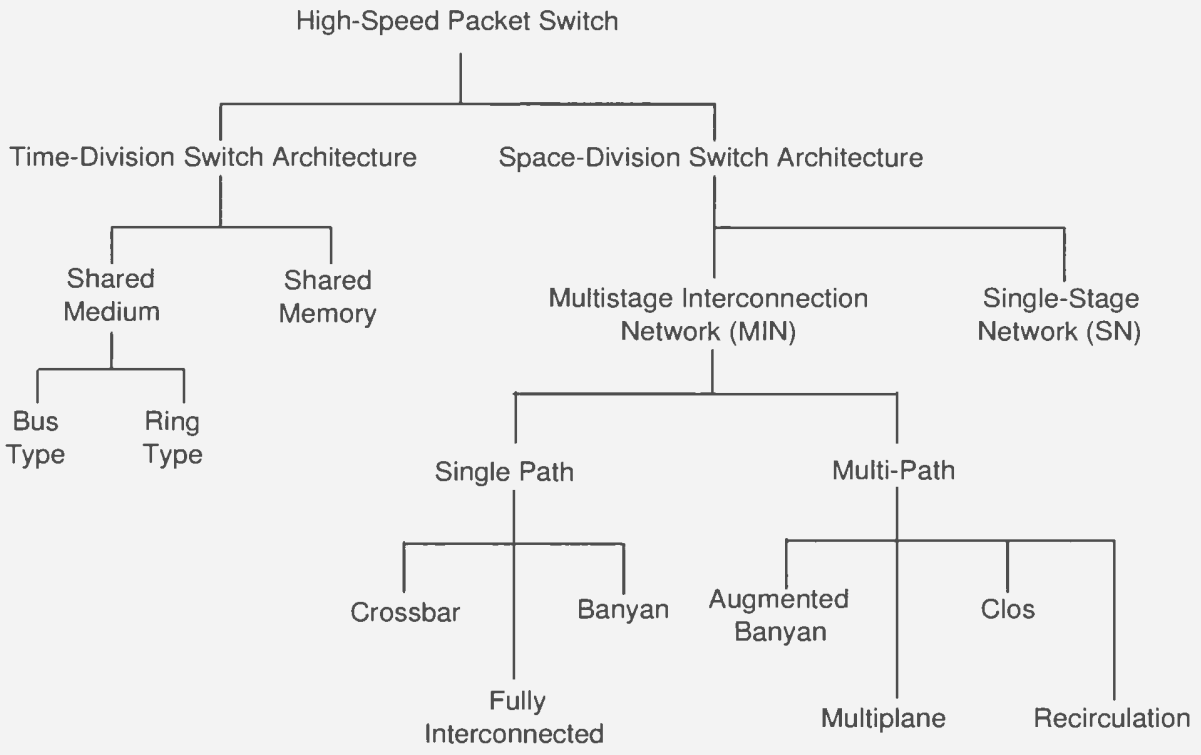


Figure 2.1: Classification of high-speed packet switch architecture (modified from [2], pp. 23)

chitectures. MINs can be further divided into single-path switches and multi-path switches. The single-path switches include crossbar based switches, banyan based switches, and full-interconnection based switches. The multi-path switches include augmented banyan based switches, multiplane based switches, Clos network based switches, and recirculation based switches [2, 3, 30]. Different switch types present different features and attributes. Figure 2.1 depicts the general architecture classification of various types of switches.

2.1.1 Time-Division Switch Architecture

In a time-division switch architecture, all cells flow through a single source shared by all inputs and outputs. This source may be either a common memory or a shared medium such as a ring or a bus. The capacity of the entire switch is usually limited

by the bandwidth of the shared resource. Thus, in general, switches belong to this category do not scale well. However, the architecture is simple and it can be easily extended to support multicast and broadcast.

2.1.1.1 Shared-Medium Type

In shared-medium type switches, all cells arriving on the input links are synchronously multiplexed into a common high-speed medium of bandwidth equal to N times the rate of a single link rate, where N is the size of the switch. Each output is connected to the bus through an interface that consists of an address filter (AF) and an output FIFO buffer [30]. The AF examines a cell header and decides whether to accept it or not. Conceptually, this approach is similar to the TDM-based circuit switch. The difference is that in a circuit switch, the destination for each slot is predetermined during the circuit setup, while in a packet switch, each cell must be processed on the fly to determine where it should be switched to. Each output interface must run at a maximum aggregate rate of N times a single link rate so that all possible cells to the output will be received.

The most important task in the shared-medium architecture design is how to implement the high-speed bus/ring and output buffers, all of which must operate at the speed of NV, where V is the link rate and N is the switch size [30]. Considering the typical limitations on memory access speed and chip size, parallel organization is normally required to implement such switch module. Also, lack of memory sharing among the FIFO buffers is another disadvantage. Examples of the shared-medium switch include NEC's ATOM (ATM Output Buffer Modular) switch [31] and Fore System's ForeRunner ASX-100 switch [2, 32].

2.1.1.2 Shared-Memory Type

In the shared-memory type architecture, the switch is constructed by using a single dual-ported memory shared by all input and output ports. Similar to TDM, cells arriving on all input ports are multiplexed into a stream and stored in the shared memory. Cells in the memory are organized into separate queues, one for each output link. The head-of-line cell of each queue is retrieved and used to form an output stream, which is then demultiplexed and transmitted through the output links [30].

Since the memory is shared by all input/output ports, this switch type has the best memory utilization [2]. The memory size can be properly adjusted so that the cell loss ratio can be controlled below a chosen level. However, there are two main constraints for this architecture. First is the processing time required to determine where to enqueue the cells and to issue proper control signals. It must be fast enough to keep up with the flow of incoming cells. More importantly, the shared memory's bandwidth should be sufficiently large to accommodate all input and output traffic simultaneously. Given N as the switch size and V as the link speed, the memory bandwidth must be at least $2NV$ [30]. Examples of this type of switch are Toshiba's 8×8 single-chip module [2, 33] and Hitachi's 32×32 module [2, 34].

2.1.2 Space-Division Switch Architecture

In the space-division switch architecture, different concurrent paths can be established for all non-conflict input-output pairs, each with the same data rate as an individual link [30]. The total capacity of the switch is therefore the product of the bandwidth of each path and the number of concurrent paths [2]. A space-division switch architecture can usually be built using small building blocks, also known as switch elements (SEs). Interconnection of these small SEs forms larger switches. High-speed memory or bus and centralized switch control are normally not required, hence, scalability is

easier to achieve using this architecture. Because all SEs can have the same structure and function, it eases system design and hardware implementation. Due to all these benefits, the space-division switch architecture plays an important role in the next-generation high-speed switch/router. Based on the number of stages presented in the switch, space-division switches are divided into multistage interconnection networks (MIN) and single stage networks [14]. As their names indicate, single stage network architectures contain only one stage. A well-known switch example using this architecture is the Knockout switch [35]. For an MIN architecture, based on the number of available paths between any input/output pair, switches are classified into single-path switches, in which only one path exists for any input/output pair, and multiple-path switches, where there is more than one path existing [2, 30].

2.1.2.1 Single-Path Switches

Single-path switches include crossbar based switches, fully interconnected based switches, and banyan based switches [2].

2.1.2.1.1 Crossbar Based Switches

The crossbar architecture is very popular network architecture. The term 'crossbar' derives from a particular design of a single-stage single-path non-blocking switching fabric, originally introduced and developed for circuit switching [3]. Figure 2.2 shows an $N \times N$ crossbar switch, in which horizontal lines represent switch inputs, and vertical lines represent the outputs.

Basically, a crossbar fabric consists of a square array of N^2 crosspoint switches, one for each input-output pair. Each crosspoint has two possible states: cross (default) and bar. A SE is located at each cross point to make a routing decision. A connection from input link i to output link j is established by setting the (i, j)th crosspoint switch

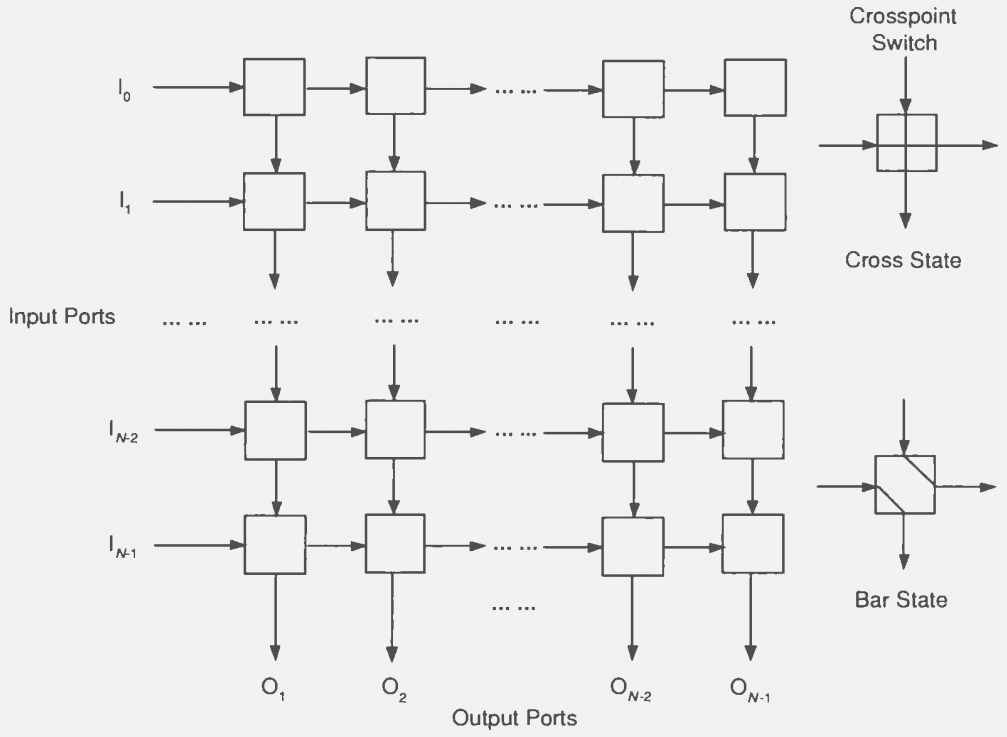


Figure 2.2: An $N \times N$ crossbar switch (modified from [3], pp. 1580)

to bar state while keeping other crosspoints along the connection in cross state. The advantages of a crossbar switch are its non-internal-blocking property, simplicity in architecture, and modularity. However, it suffers from two major problems: The first one is its high hardware complexity, which is $O(N^2)$. This means that the switch fabric does not scale efficiently. The arbitration of winner cells in each time slot can also become the system bottleneck as the switch size grows [2, 30]. The second problem is its lack of fairness. Different cells destined to different outputs may traverse different numbers of SEs and take different times. The cell with the shortest path will always arrive at the output first and thus obtain a higher priority over others. These two flaws limit the use of the crossbar switch. Although the crossbar network can be used as basic modules and interconnected to build a larger switch, such as the method used in the Clos network [36], this only solves the scalability problem but

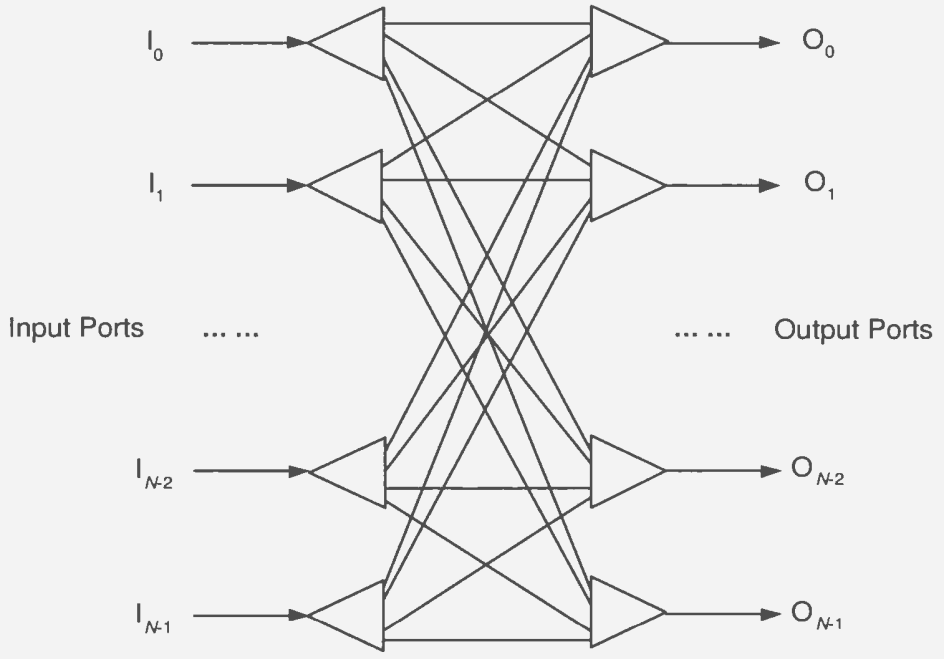


Figure 2.3: An $N \times N$ fully interconnected switch (modified from [2], pp. 28)

cannot remove the problems of the high complexity and lack of fairness.

2.1.2.1.2 Fully Interconnected Based Switches

In a fully interconnected architecture, the complete connectivity between all inputs and outputs is usually achieved by broadcasting input cells to all output ports using N separate buses [2], as shown in Figure 2.3.

The switching operation of the fully interconnected switch is similar to that of the shared-medium switch. Cells from several inputs can reach the same output at the same time. Therefore, address filters and dedicated buffers, one for each output port, are required. However, they are different in that the speedup requirement for sequential transmission over the shared medium is now replaced by the N separate broadcast buses [2]. This is considered a disadvantage of the switch. An example switch using this structure is the knockout switch [35].

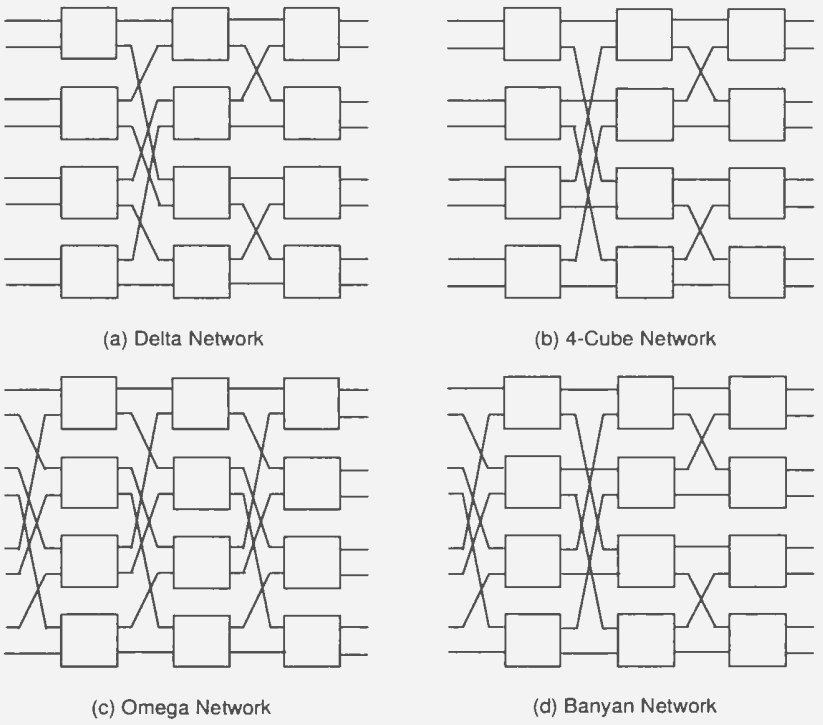


Figure 2.4: Different topologies of Banyan based switches (modified from [2], pp. 29)

2.1.2.1.3 Banyan Switches

Banyan switches are a family of switches constructed from 2×2 SEs with a single path between any input-output pair [2]. Figure 2.4 demonstrates four isomorphic topologies of the banyan switches, namely the delta, 4-cube, omega, and banyan networks [2, 18]. All switches of the family have the same performance due to their topological equivalence.

The banyan switches are suitable for the construction of large switches due to the lower hardware complexity of switching elements ($O(N \log_2 N)$), compared with $O(N^2)$ of the crossbar and fully-interconnected architecture. Self-routing is another important feature in which the control mechanism can be fully distributed. The parallel structure ensures that cells from different paths can be processed simultaneously. The modular and recursive structure provides easier VLSI implementation. However,

the biggest disadvantage is their internal blocking, which significantly degrades its performance as the switch size grows.

2.1.2.1.4 Batcher-Banyan Switch

It has been reported in [37] that banyan switches become nonblocking when permutation traffic is applied and when all active cells are sorted on their destinations. This property leads to the Batcher banyan network (BBN), which includes a Batcher sorting network [38], followed by a banyan network, as shown in Figure 2.5 for an 8×8 BBN. The Batcher sorter has $\frac{\log_2 N+1}{2} \log_2 N$ stages while the banyan network has $\log_2 N$ stages, each with $N/2$ sorting/switching elements in every stage. Therefore, the BBN has a total of $\frac{N}{2}(\frac{\log_2 N+1}{2} \log_2 N)$ sorting elements and $\frac{N}{2} \log_2 N$ SEs, which is less in hardware complexity than that of the crossbar network. The BBN has perfect performance under permutation traffic, in which the requested destination for every active cell is distinct. However, its performance degrades significantly when traffic more like real traffic is applied, including the random traffic, which still represents an ideal situation. It should be noticed that permutation traffic, which is used as the basis to characterize switch blocking properties, dose not represent real network traffic. Therefore, non-blocking architectures defined under permutation traffic do not necessarily have higher performance over some blocking switch architectures under other traffic conditions.

2.1.2.2 Multiple-Path Switches

Multiple-path switches can be further classified as augmented banyan switches, Clos switches, multiplane switches, and recirculation switches [2], as shown in Figure 2.6.

2.1.2.2.1 Augmented Banyan Switches

In the augmented banyan switch, extra stages are added in addition to the regular

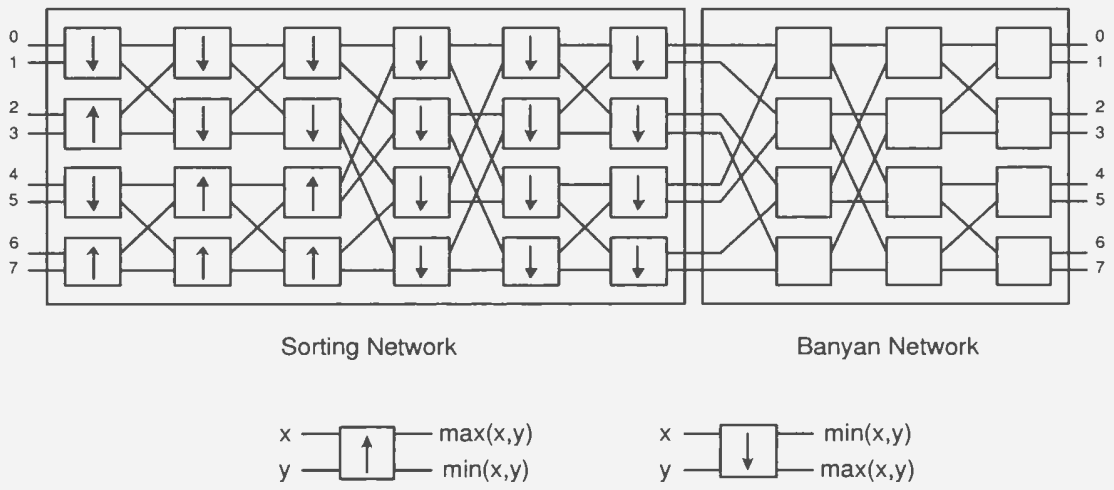


Figure 2.5: An 8 × 8 Batcher Banyan network (modified from [3], pp. 1595)

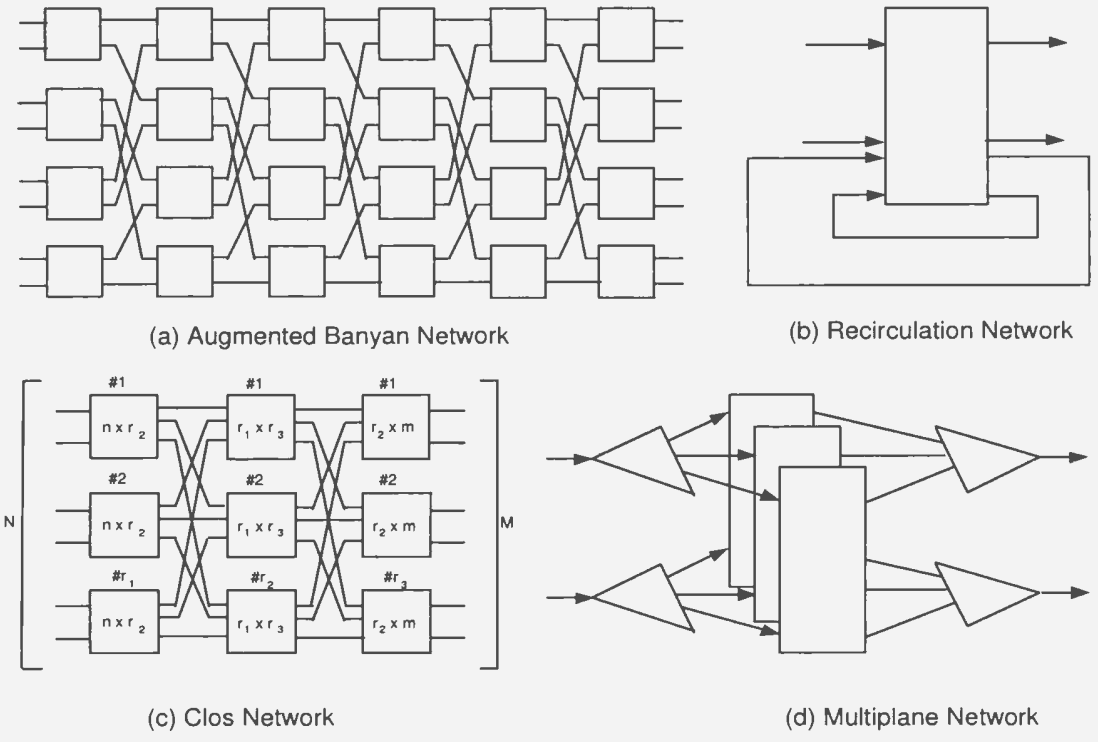


Figure 2.6: Multiple-path based switches (modified from [2], pp. 30)

banyan network. Hence, more paths are available per input-output pair, and therefore, deflected cells are provided more chances to reach their destinations through the augmented stages. Cell loss rate is reduced. However, additional stages mean more hardware is required and the routing scheme becomes more complicated. For every augmented stage, cells must be checked to decide whether they have arrived at the requested output ports or not. If so, they will be sent to the OPC directly, otherwise, they will be sent to the next augmented stage for examination. Examples of switches using such architecture include the Benes interconnection network [30, 39], tandem banyan switch [2, 40] and dual shuffle exchange switch [2, 41].

2.1.2.2.2 Recirculation Based Switches

To reduce the cell loss due to output port contention and to improve system throughput, recirculation switches are designed. For these switches, cells that do not make it to their destinations during the current time slot are recirculated back to the input ports via the recirculation paths [2]. However, to accommodate input ports for the recirculated cells, a larger switch is normally required. Further, the switch might also have the out-of-sequence problem. The examples of this type of switches are the Starlite switch [4] and the Sunshine switch [23].

2.1.2.2.3 Three-Stage Clos Switches

A general scheme of the three-stage network is given in Figure 2.6(c) for an $N \times M$ network. At the first stage, N input lines are broken into r_1 groups of n lines, where $r_1 = N/n$. Similarly, at the third stage, the M output lines are divided into r_3 groups of m lines, where $r_3 = M/m$. To interconnect the first and third stage, $r_1 \times r_3$ modules are used in the middle stage. Therefore, to construct a $N \times M$ switch, r_1 $n \times r_2$ switch modules are used in the first stage, r_2 $r_1 \times r_3$ switch modules are used in the second

stage, and r_3 $r_2 \times m$ switch modules are used in the third stage.

The Clos network is a three-stage network in which $r_1 = n$ and $r_3 = m$. It has been proved by C. Clos [18, 36] that a three-stage network is strict-sense non-blocking if and only if $r_2 \geq n + m - 1$, which is also known as the Clos theorem [18]. This indicates that by increasing the number of outputs from each first-stage module, the blocking probability of the switch is reduced. This enlightens us that by introducing extra links in the middle stages of the banyan based switches, their performance can be improved.

The advantage of the Clos switch is that the hardware complexity is reduced from $O(N^2)$ in crossbar and fully interconnected switches to $O(N^{\frac{3}{2}})$ and the switch can be designed to be non-blocking. Furthermore, it is more reliable because multiple paths exist for each input-output pair. However, a fast and intelligent mechanism is needed to rearrange the connections during every time slot to avoid the internal blocking. This would become a system bottleneck for large switches [2].

2.1.2.2.4 Multiple-Plane Switches

Multiple-plane based switches have more than one, usually identical switch planes. System throughput and reliability can be significantly improved by distributing traffic into different parallel switch planes. Cell collisions inside each switch plane are reduced. However, hardware complexity multiplies with every new plane introduced and cell sequencing may be affected unless cells belonging to the same connection are transferred through the same plane [2]. Otherwise, mechanisms for re-sequencing arrived cells at the output ports should be considered. Typical examples using this switch type include the parallel banyan switch and the Sunshine switch [23].

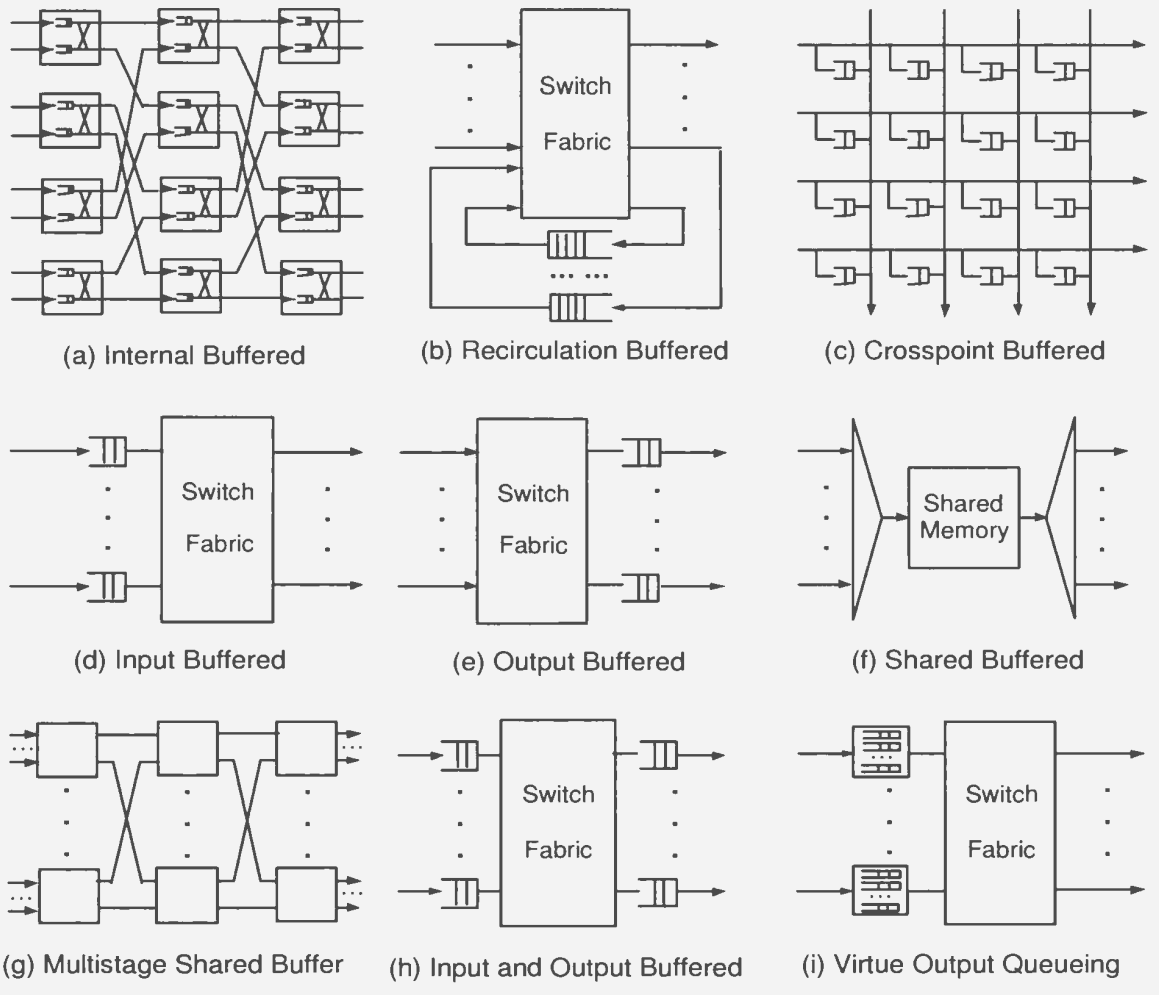


Figure 2.7: Buffering strategies for cell switches (from [2], pp. 34)

2.2 Buffering Strategies

Buffering strategy is an extremely important aspect to be addressed during switch design. Various buffering strategies are shown in Figure 2.7 [2]. In this section, various buffer placement strategies are briefly reviewed and evaluated.

2.2.1 Internal Buffering

Using internal buffering strategy, buffers are placed within the SEs. Blocked cells can be buffered at the SEs so that cell loss can be reduced. Larger switches can be

realized easily by replicating the SEs. However, low throughput and high transfer delay are the two major problems of switches following this buffering strategy [2]. Additional implementation cost will be required to meet QoS requirements.

2.2.2 Recirculation Buffering

This buffering strategy is used for recirculation based switches to resolve the output port contention problem. When output port contention occurs, cells that have lost the contention are stored in the buffer on the recirculation path and switched again during the next switching cycle. To deal with the out-of-sequence problem, different priority levels are suggested to be associated with each cell. By properly choosing the ratio of recirculation ports to input ports or/and by allowing more than one cell to arrive at the output port in each cycle, a desired cell loss rate can be achieved.

2.2.3 Crosspoint Buffering

This buffering strategy is used in the crossbar based switches such as the Bus-Matrix Switch (BMX) proposed by Fujitsu [42]. Each crosspoint has an AF and a buffer. AF accepts cells to that output and stores them in the buffer. Then the arbiter decides which cell is to be sent to the output port for that switching cycle. However, because the buffers are distributed to all crosspoints without being shared, the total buffer requirement is high. The chip size is normally limited by the amount of memory instead of the number of crosspoints and their control logic [2, 18].

2.2.4 Pure Input Buffering

Pure input-buffered switches suffer from the well-known HOL blocking, which limits switch throughput to 58.6% under uniform random traffic [30, 3]. A windowing technique, in which multiple cells from each input link are examined for switching, can be used to improve its throughput. However, this increases the implementation

complexity of the input buffers and arbitration mechanism.

2.2.5 Pure Output Buffering

Compared with the pure input buffering strategy, pure output buffering has no HOL blocking problem and can achieve 100% throughput. The performance seems optimum. However, to avoid any cell loss, the output buffer needs to accept up to N cells in one switching cycle. As the switch size becomes larger and link speed goes higher, memory speed normally becomes the system bottleneck.

2.2.6 Shared-Buffer Strategy

In this strategy, a common buffer is shared by all input and output ports. During each switching cycle, cells from all input ports can be stored in the buffer and retrieved by the corresponding output ports. Optimum throughput and delay performance and more efficient buffer utilization can be achieved using strategy, which is similar to that of the output-buffered switch. However, the switch size is limited by the memory read/write access time [2], which must be shorter than $1/2N$ of one switching cycle. As shown in Figure 1.1, the increase of memory access speed lags far behind the growth rate predicted by Moore's law. Also considering the fact that large amounts of such high-speed memory will be very costly, the shared-buffer strategy is not suitable to build large-scale high-speed switches.

2.2.7 Multistage Shared-Buffer Strategy

It is clear from the previous subsection that the shared-buffer strategy is not suitable for a large switch due to the limit of memory access speed. But its high throughput, low delay, and high memory utilization make it widely used in small scale switches. In the multistage shared-buffer solution, small shared-buffer switch modules are interconnected to form a larger switch. However, internal blocking of the multistage

network might degrade its performance. Also, buffering at several places and different queue lengths for modules used in different stages cause an out-of-sequence problem, which is complex and costly to handle [2].

2.2.8 Input-Output Buffering

This buffering strategy combines the advantages of pure input-buffering and output-buffering strategies. The input buffer runs at the input link speed while the speed of the output buffer is L times the output link rate, where $1 < L < N$. This means up to L cells can be received for each output port during the same switching cycle. If more than L cells are destined to the same output port, the excessive cells are stored in the input buffer. A large-scale switch can be achieved using this buffering strategy. The challenge of a switch using this strategy is to decide which cells (out of the N possible HOL cells) can get through and which cells need to be kept in the input buffer. A backpressure mechanism can be used for this purpose.

2.2.9 Virtual Output Queueing Buffering

To overcome the HOL blocking in purely input-buffered switches, virtual output queueing is proposed [43, 44]. Each input buffer is logically divided into N logical queues, one for each output port. All these N logical queues of the input buffer share the same piece of physical memory [2]. This buffering strategy can improve system throughput and reduce cell loss rate. However, because a total of N^2 HOL cells from all logical queues in the input buffers need to be arbitrated during each time slot, a very fast and intelligent arbitration mechanism is required [2], such as the maximum matching algorithm described in [44].

2.3 Multicast Switches and Switching Techniques

Multicasting in a packet switch means that an incoming cell will be delivered to more than one destination. In the study of multicast switches, cell replication is the most important operation which needs special attention. In this dissertation, the multicast switches using the space-division architecture are studied in depth.

Generally speaking, the methods to construct a multicast switch can be categorized into two approaches: the dedicated copy network solution (cascade approach) and the implicit replication and routing solution (integrated approach).

2.3.1 Multicast Switches Using the Cascade Approach

The intuitive approach to make a multicast switch is to employ a copy network in tandem with a point-to-point unicast routing network, as shown in Figure 2.8. The copy network is a special kind of a functional network which can replicate cells according to the fanout number specified in the header. The routing network uses the output of the copy network as its input and routes each copy to its destination. In this subsection, the most typical multicast switches using this approach are reviewed, which include: the Starlite switch [4], which is the first switch architecture dealing with multicast traffic; the Knockout switch [5], which is one of the most well-known switch architectures; Turner's broadcast packet switch [21] and Lee's non-blocking copy network for multicast switching [7]. More recent switches within this category are briefly reviewed and summarized as well [6].

2.3.1.1 Starlite Switch

The Starlite switch, proposed in 1984, is the earliest architecture which considers multicast traffic [6]. The basic switch architecture is comprised of the crossbar network and Batcher-Banyan network. The multicasting is achieved by using a two-stage copy

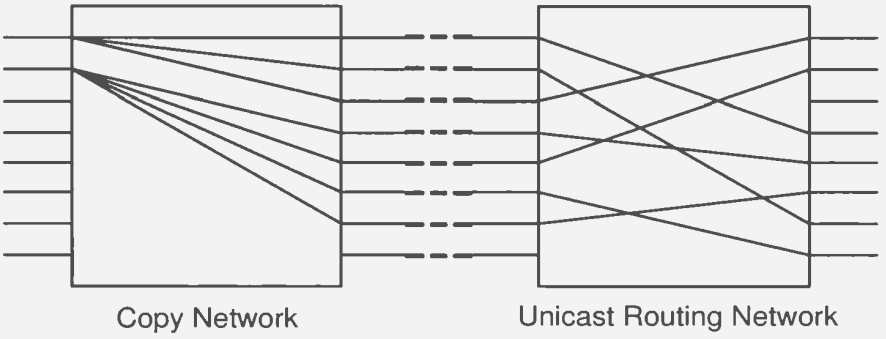


Figure 2.8: Multicast switch architecture using the cascade approach

network placed above the concentrator of the unicast structure, which is shown in Figure 2.9 (a). Multicasting cells are treated as special cells which contain the input channel number and the destination port where each copy of a cell should be sent to. The sorting network sorts the cells on their channel number and then the copy network replicates the copies and sends to the concentrator [4]. During the replication process, synchronization of the source and destinations, and an empty packet setup procedure are required by the Starlite switch.

2.3.1.2 Knockout Switch

There are two possible ways to implement multicasting in the original Knockout switch [35] either by adding cell replication function into the multicast modules [5] or by adding a fast cell filter. The second solution, in which cell replication is integrated into each output module, will be discussed in the next subsection. In the first solution, to handle multicast cells, the original Knockout switch is modified by adding M multicast modules and M multicast buses, as shown in Figure 2.9 (b). Each multicast module has N inputs from the input interface modules and one output which drives one of the M multicast buses. Multicast cells selected through the multicast cell filters experience the $N : L$ knockout process and the winner cells are stored in the

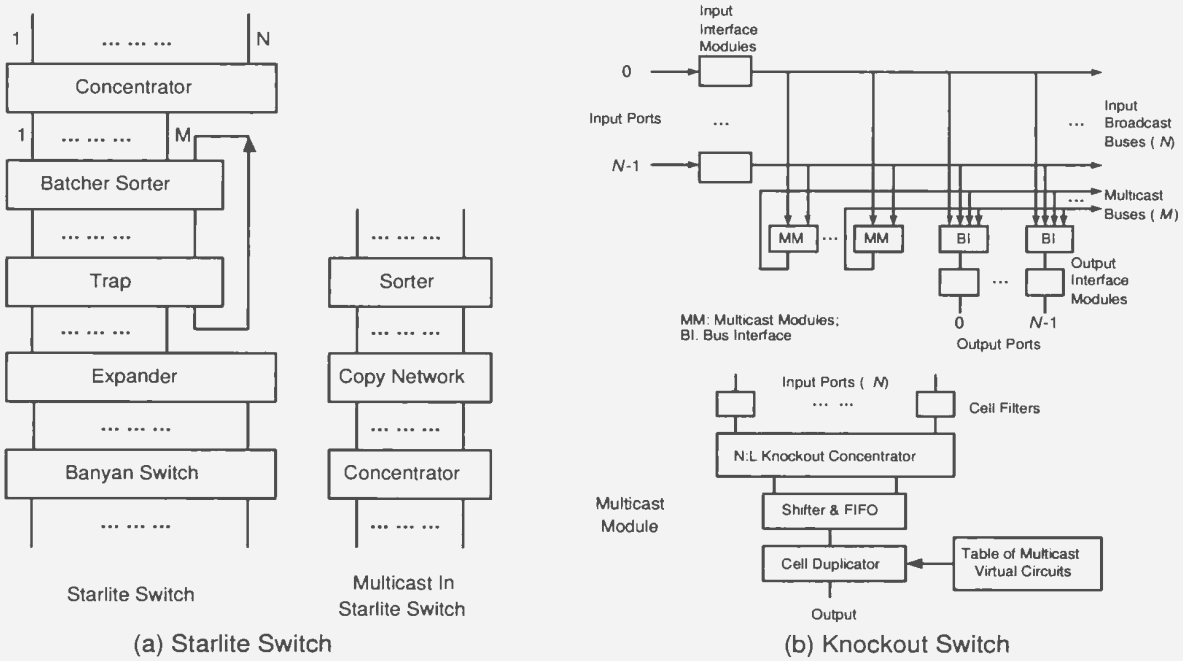


Figure 2.9: Starlite switch (modified from [4], pp. 122) and knockout multicast switch (modified from [5], pp. 30)

FIFO buffer. Each cell is replicated based on the different destination addresses in the cell header. The replicated cells are then sent out along the broadcast bus [6].

2.3.1.3 Turner's Broadcast Packet Switch

In [21], Turner proposed a broadcast switch which utilizes the copy network to replicate a multicast cell, as shown in Figure 2.10. After multicast cells are replicated, they are translated by the broadcast and group translator and then routed to their destination through the distribution and routing networks. This switch's contribution is the flexible broadcast capability. However, blocking inside the routing network requires that buffers are used for every internal node in the routing network [6].

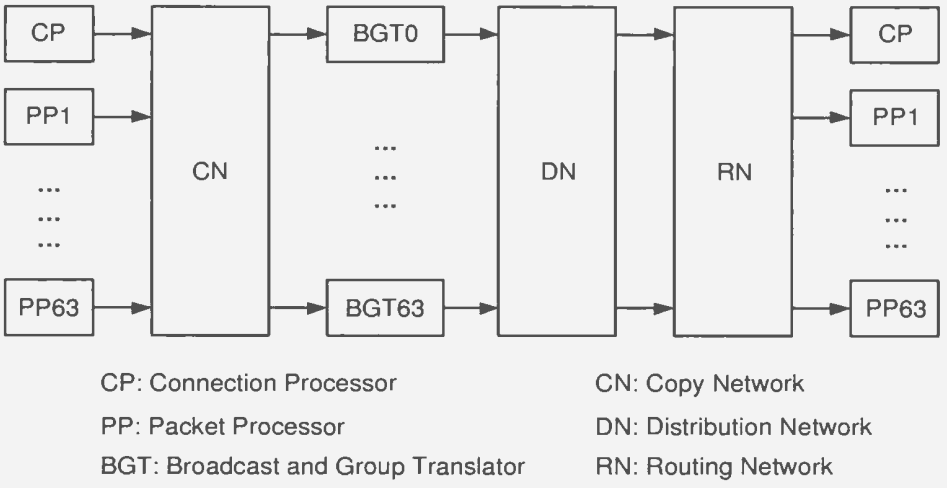


Figure 2.10: Turner's broadcast packet switch (modified from [6], pp. 105)

2.3.1.4 Lee's Multicast Switch

Lee's multicast switch [7] is probably the most important and the foundation of the copy network based multicast switch. As shown in Figure 2.11, the most noticeable part of Lee's switch is the copy network design, which consists of a running adder network, dummy address encoders, a broadcast banyan network and trunk number translators. Theoretically, there is no special requirement for the point-to-point routing network in that any routing network can be used to route cells from output port of the copy network to output port of the multicast switch. Cell replication is done inside the broadcast banyan network according to a Boolean interval splitting algorithm based on the address interval in the new header. The details on how this algorithm works can be found in [7, 45].

Lee's multicast switch fabric design suffers from two problems. One is overflow, i.e., the total requested number of copies can exceed the available number of output ports of the copy network. In this situation, any cell whose fanout is larger than the remaining free output ports will be dropped [6, 7]. This will eventually decrease system performance and throughput. The other is the output port conflict problem in

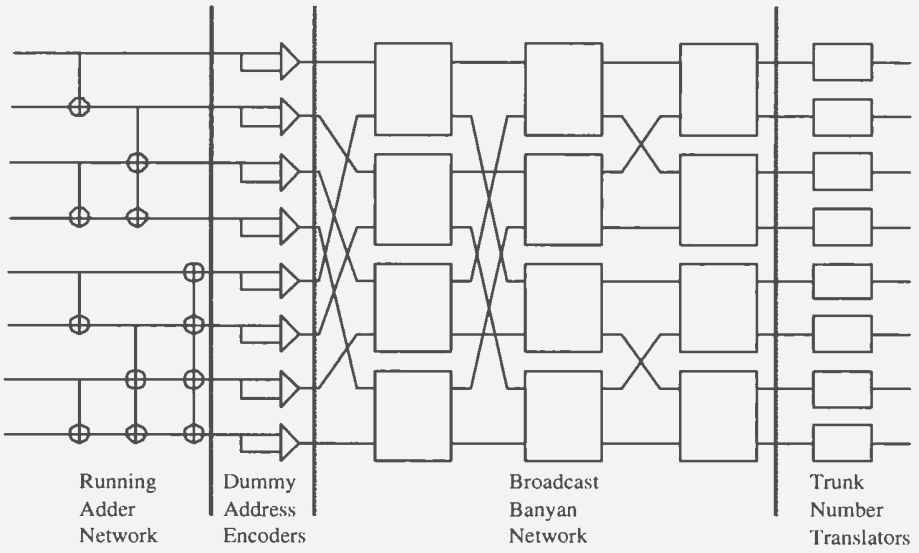


Figure 2.11: Copy network of Lee's multicast switch (modified from [7], pp. 1457)

the routing network when multiple packets request the same output port concurrently. Some modifications have been proposed to improve the design [46, 47], such as adding a cyclic running adder network to solve the overflows of the copy network and the input fairness problem in the broadcast banyan network. However, those modifications only mitigate the situation, while at the same time, they increase the design complexity. Besides these two problems, the memory size of the trunk number translation tables will increase significantly as the fanout and the switch size increase. Further, the internal blocking of the broadcast banyan network will not only degrade the switch performance, but also increases the cell loss probability [6].

2.3.1.5 Other Multicast Switches

Following Turner and Lee's multicast switch design, many multicast switches with a copy network design are proposed. These multicast switch architectures include:

(1) The Shared Concentration and Output Queueing (SCCQ) Multicast Switch [48], which is comprised of a sorting network, L switching modules, a copy network,

and N input buffers. A recycling technique is used for cell replication when transmitting the multicast and broadcast cells. The cells are sent back through the copy network to the switch input through which the copies are made.

(2) The Multicast Switches based on Link-Grouped MIN (LGMIN switches) consists of multiple shared-buffer copy network modules and small memory switch modules [49]. Different architectures of the LGMIN switch are presented to deal with different traffic situations. Two cell replication mechanisms can be used: replication through recycling technique and replication via a broadcast banyan network.

(3) The self-routing multistage Multinet switch consists of virtual FIFO buffers located between stages and in the output ports [50]. There are two ways for the Multinet switch to handle multicast traffic, with an explicit copy network, or without. The latter one will be discussed in the next subsection.

(4) Many other switches can be found in the literature as in [22, 23, 51, 52, 53, 54, 55, 56, 57]. They are essentially modified and improved from the architectures that have been discussed above.

2.3.2 Multicast Switches Following the Integrated Approach

In the integrated approach, there is no dedicated copy network presented in the switch. Often in this kind of switch, cell replication is a subfunction in some components such as the switch elements. For MIN-based switches, by properly integrating the replication function into the routing function, we can benefit most from a MIN design and can efficiently turn a high-performance unicast switch into a multicast switch.

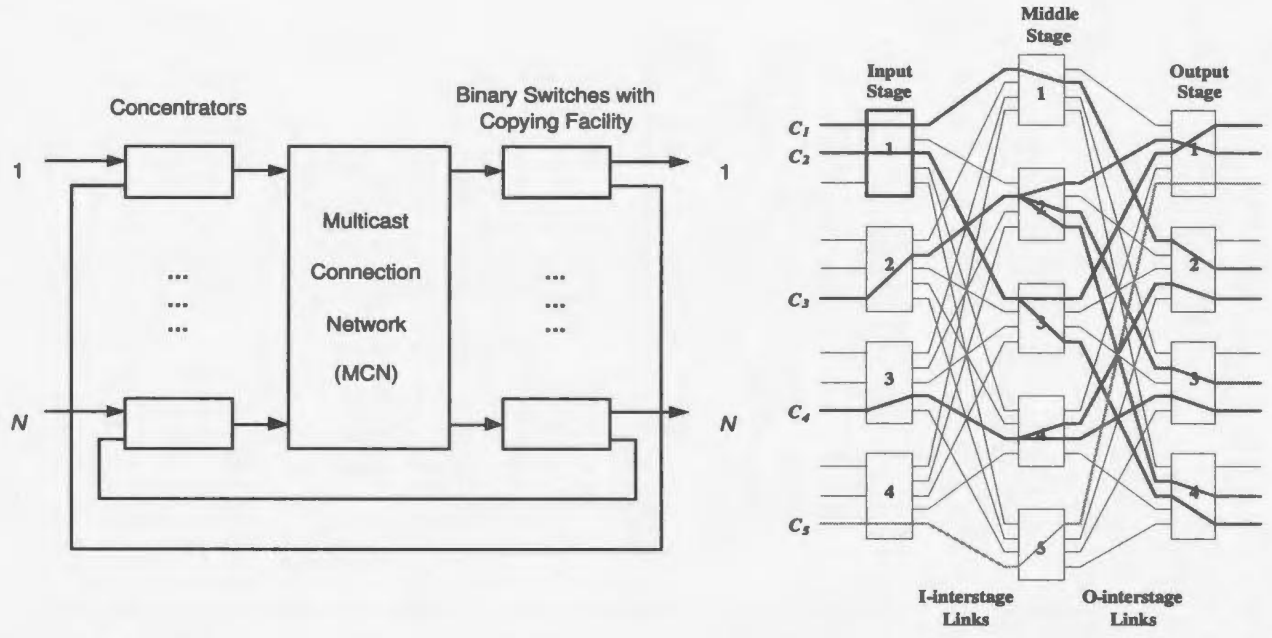
2.3.2.1 Knockout Switch

Here, the second approach to implement multicasting in the Knockout switch is investigated [35]. It is based on the use of a fast cell filter. The cell filter quickly

decides whether an incoming multicast cell is destined toward its output or not [6]. Therefore, the multicast module is similar to that shown in Figure 2.9 (b) except that the cell duplicator has been removed. Once the multicast cell becomes the winner from the knockout concentrator, it is buffered and broadcast to all output interface modules directly. The multicast virtual circuit number in the incoming cell header is compared to a list of multicast virtual circuits stored in the bus interface to decide whether the cell should be accepted or discarded. The Knockout switch can be implemented in a modular fashion. It is simple to maintain and can be made fault tolerant [6]. However, since it is based on the crossbar network, the hardware complexity is very high. Also, output memory access time may become a problem as switch size gets large and link speed becomes high.

2.3.2.2 Recursive Multistage Structured Multicast Switch

The recursive multistage structured multicast switch [58] utilizes self-routing switching nodes with multicast facility in a buffered $N \times N$ MIN with external links connecting switch outlets to inlets [6, 58]. A block diagram of the switch is shown in Figure 2.12 (a). The switch is able to generate up to M copies of the same cell, where $M << N$, and transmit through M pre-defined adjacent output ports. M is defined as the multiplication factor of the multicast connection network. For multicast cells with more than M destination requests, some of the M copies are transmitted in the first crossing of the switch, some are fed back to the input and used to generate the remaining number of copies. Although the recursive mechanism is simple, the function of the switch elements are complex, which makes the hardware complexity high. Cell sequencing is another problem for this switch.



(a) Recursive Multistage Structured Multicast Switch

(b) Three-Stage Clos Multicast Switch

Figure 2.12: Multicast switch using recycling technique (from [6], pp. 115) and Clos network (from [8], pp. 526)

2.3.2.3 Three-Stage Clos Multicast Switch

The Three-Stage Clos Multicast Switch [8] is an example of implicit routing and replication switch fabric. As shown in Figure 2.12 (b), it is built on the Clos network with each SE formed by a small crossbar network. In the example, incoming cells C_1 and C_5, which arrive at the first and the fourth SE of the input stage, are unicast cells and should be delivered to the second and first SE of the output stage respectively. C_2, C_3 and C_4 are all multicast cells and have multiple output port requests, as shown in the figure. Replication work is performed when necessary and is done within the switch fabric as the cell is being routed through the switch. [6].

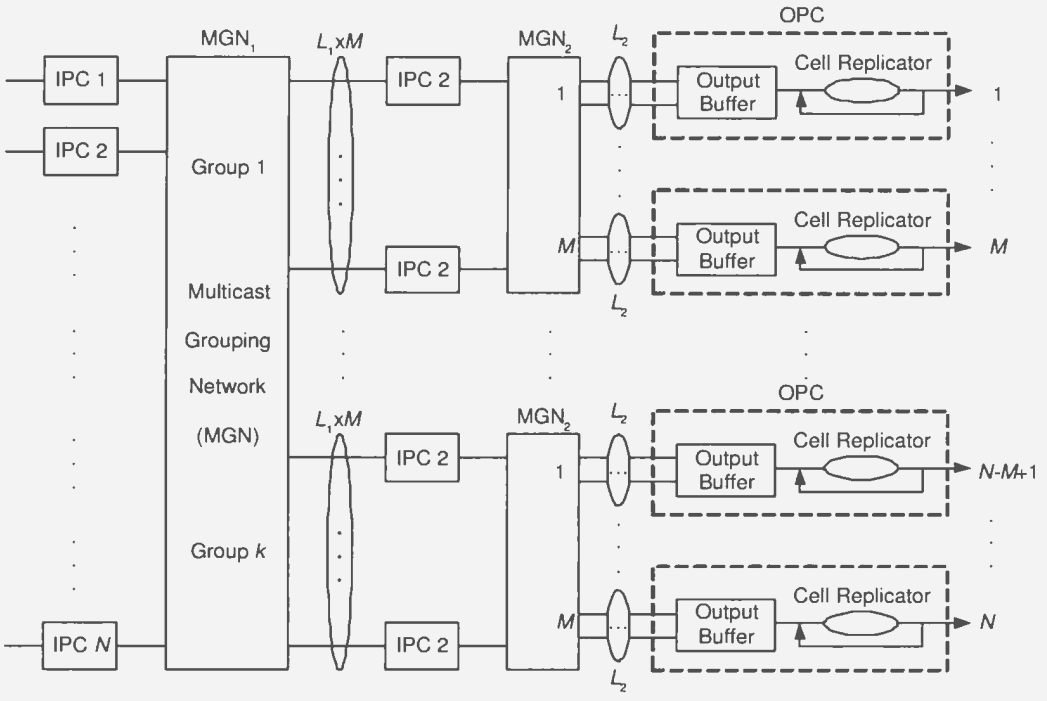


Figure 2.13: Architecture of the multicast output-buffered ATM switch (MOBAS) (modified from [2], pp. 155)

2.3.2.4 Multicast Output Buffered ATM Switch (MOBAS)

MOBAS is an output-buffered multicast switch based on the multicast grouping network (MGN) concept proposed by Chao [59]. The grouping network idea is inspired by the Knockout switch in which L is used instead of N as the number of internal links toward each output port so that hardware complexity is reduced from $O(N^2)$ to $O(LN)$, while the performance still maintains an acceptable level. Its architecture is shown in Figure 2.13. Cell routing and replication are handled inside the MOBAS switch by combining the multicasting tree and broadcast buses [6], i.e., the routing links, which interconnect the MGNs and OPCs, form a multicasting tree, while inside each MGN, the cells are sent through broadcast buses. MOBAS has several problems such as the internal blocking problems of the routing links which form the multicasting tree and high complexity of the MGNs because of the use of broadcast buses.

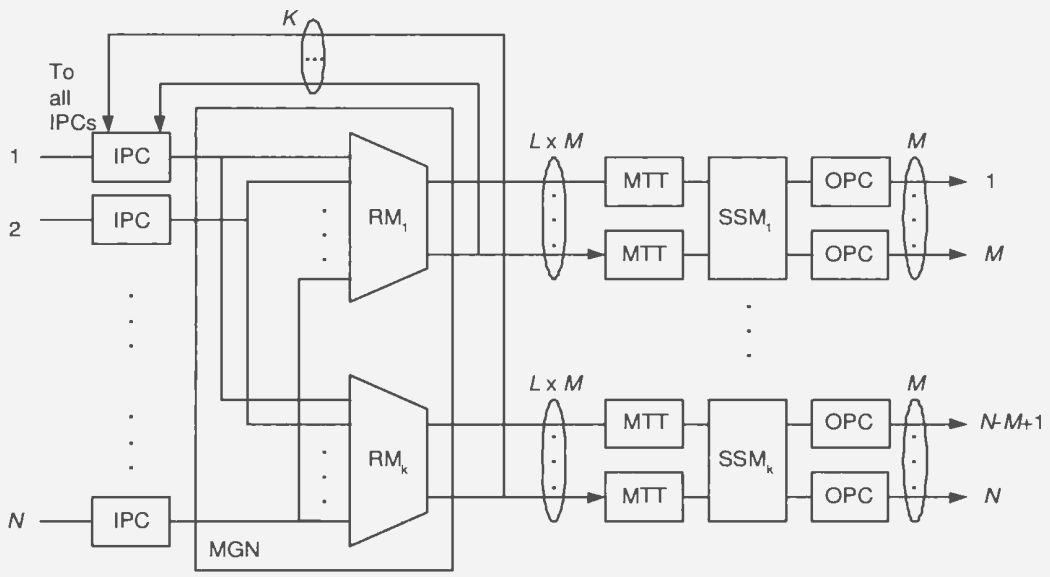


Figure 2.14: Abacus switch (from [2], pp. 191)

The delay caused by the Knockout concentrator used in each SE is considered to be another problem [6].

2.3.2.5 Abacus Switch

To decrease the possibility of cell loss due to the internal link contention, Chao proposed another switch architecture called the Abacus switch [60]. Figure 2.14 depicts its architecture. The Abacus switch is composed of a non-blocking switching fabric followed by small switch modules at the output, in which cell replication and cell routing are performed simultaneously [6]. Similar to MOBAS, cell replication is performed through the broadcast buses to all routing modules (RMs). Cell copies are then routed to the output module via switch elements array (SWE). The enhancement in the Abacus switch is that there are K additional lines to pass the acknowledgements back to the IPCs. With that information, it can be determined whether the cell should be buffered in IPC or not. Similar to the problems for MOBAS, internal routing link blocking, hardware complexity of the switch elements, and delay caused

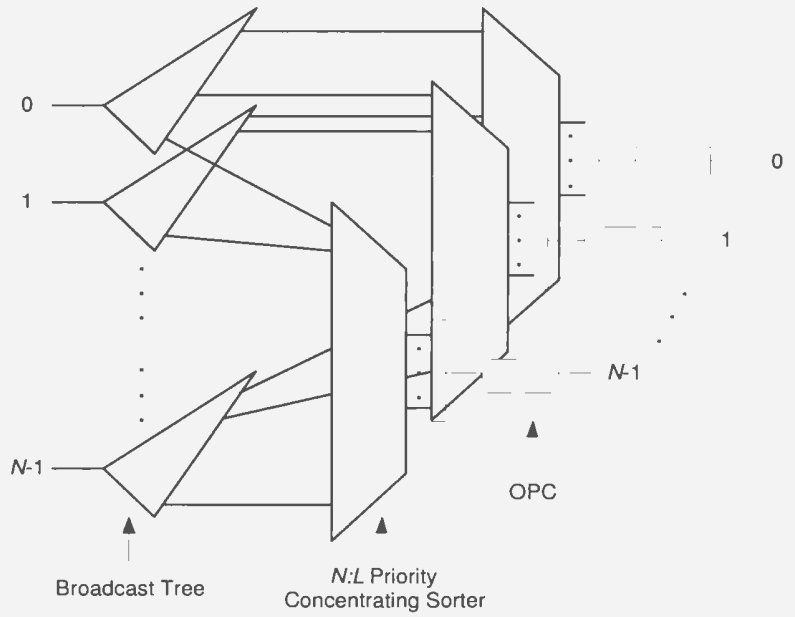


Figure 2.15: The architecture of PINIUM switch (from [9], pp. 844)

by the Knockout concentrator are also problems for the Abacus switch

2.3.2.6 Prioritized Services, Internal Nonblocking, Internally Unbuffered Multicast Switch (PINIUM Switch)

From the previous discussion on the MOBAS and Abacus switches, it has been demonstrated that for networks using broadcast trees for cell replication, the blocking of internal links connecting routing modules always exists. In [9], a switch called prioritized services, internal nonblocking, internally unbuffered multicast switch (PINIUM switch) is proposed to tackle this problem, as shown in Figure 2.15. The PINIUM switch is constructed by using multiple routing planes with each plane supporting only one broadcast tree [6, 9]. It does eliminate the internal link blocking between the routing modules, but unfortunately, the price is high. If there are less than N multicast trees within the same switching cycle, it will not fully utilize the switch capacity. Besides that, since more than one cell can arrive at the same output port within one slot, OPC must have enough capability to handle the output contention.

With more cells that each output can receive, more buffering space and high-speed buffers are required, which brings up the cost of the entire switch.

2.3.2.7 Other Switches Following the Integrated Approach

Other multicast switches following the integrated approach include:

(1) Gauss ATM Switching Element [61], which has the same basic structure as the Knockout switch but uses different output modules.

(2) Growable Packet Switch (GPS) [62], which uses a three-stage structure. The first stage is used just for cell routing. The second stage replicates cell and sends one copy to each necessary third stage module. The third stage replicates cell again if needed and sends to the requested output port.

(3) Multicast BG switch, which will be extensively studied in this dissertation.

(4) Other proposed switches belonging to this category include those in [5, 50, 63, 64, 65]. Again, the architecture and design methodology of those switches are similar or modified from the switches that have been discussed above.

2.3.3 Comparison of the Two Approaches of Constructing Multicast Switches

The cascade approach was popular initially because it requires the least modification to an existing router/switch design. Providing the copy network as an add-on module will easily convert a unicast switch to support multicast. Theoretically, there is no restriction on the types of routing networks that will be used. Hence, it is attractive because there is no need to design a brand new switch architecture from scratch. Further, it is a quick-to-market solution when the unicast switches and routers are dominant and the demand for multicast support is less significant.

As discussed in Section 2.3.1, many proposed multicast switch architectures follow this approach after the foundation work of Lee's multicast switch [7]. However, this

family of SF designs suffers from two major problems. One is overflow at the copy network, i.e., the total number of copies requested exceeds the available number of output ports of the copy network. In this situation, any cell whose fanout is larger than the remaining free output ports will be dropped [7, 46]. This problem will eventually lead to the degradation of system performance and throughput if not handled properly. The other problem is the output port contention in the routing network when multiple packets request the same output port simultaneously. A proper mechanism must be devised to handle such situation, which will normally complicate the control function and generate higher demand for buffering resources. Besides these two problems, the memory size of the translation table in between the copy network and the routing network will also increase significantly as the fanout and the switch size increase. This will increase the hardware complexity for the whole switch. Although some modifications are proposed to improve the design [7, 46], such as adding feedback loops from the copy network output port to the switch input for the excessive copies, and adding buffering resources between the copy network and the routing networks for cells that lose their output contention, they only mitigate the situation. At the same time, they have an increased design complexity.

The integrated approach combines the routing and multicast cell copy functions into a single unified network. The problems encountered by the cascade approach no longer exist in the integrated approach. Even though each individual SE must be enhanced to handle both functions, which will increase its hardware complexity slightly, the overall complexity of the switch fabric is normally less than the sum of the copy and routing network because many resources originally required by both networks are now shared, such as the memory components which are used to store the routing and replication tags. Besides the advantage of reduced hardware complexity, the characteristics of reliability, scalability, and fault tolerance in the single unified

network solution are also easier to improve. All these benefits make the integrated solution attractive for new architectures of the next-generation multicast switches.

2.4 Historical Background of the BG Network

In this section, previous research on the BG network is introduced. The evolution of the network topology, the corresponding routing algorithm, and some important results and features are also presented.

2.4.1 Topology and Routing Algorithm

The unicast BG network structure was first proposed in [24]. It follows the multi-path MIN architecture design. Due to its multi-path property, packets can be routed through the network even in the presence of failures of some of the SEs in the network [25]. The BG network can offer an outstanding performance compared with other well-known networks that have similar or even higher hardware complexity, such as the Crossbar network [30], the Batcher-Banyan network [3], and the 2-replicated and 2-dilated Banyan network [14, 25].

2.4.1.1 Topology

The original BG network has the same structure as the Kappa network [66], except that in the last stage, the 4:1 concentrator at each output port of the Kappa network is replaced by buffers which can receive up to four cells in one switching cycle [25]. An $N \times N$ BG network consists of $n + 1$ stages, where $n = \log_2 N$. The first stage has N 1×4 SEs and each of the following n - 1 stages will be comprised of N 4×4 SEs. The last stage is the buffer stage as mentioned earlier. Figure 2.16 depicts the structure of an 8×8 BG switch.

Let us denote each SE in the BG network as $SE_{i,j}$ $(0 \leq i < N, 0 \leq j < n)$, where

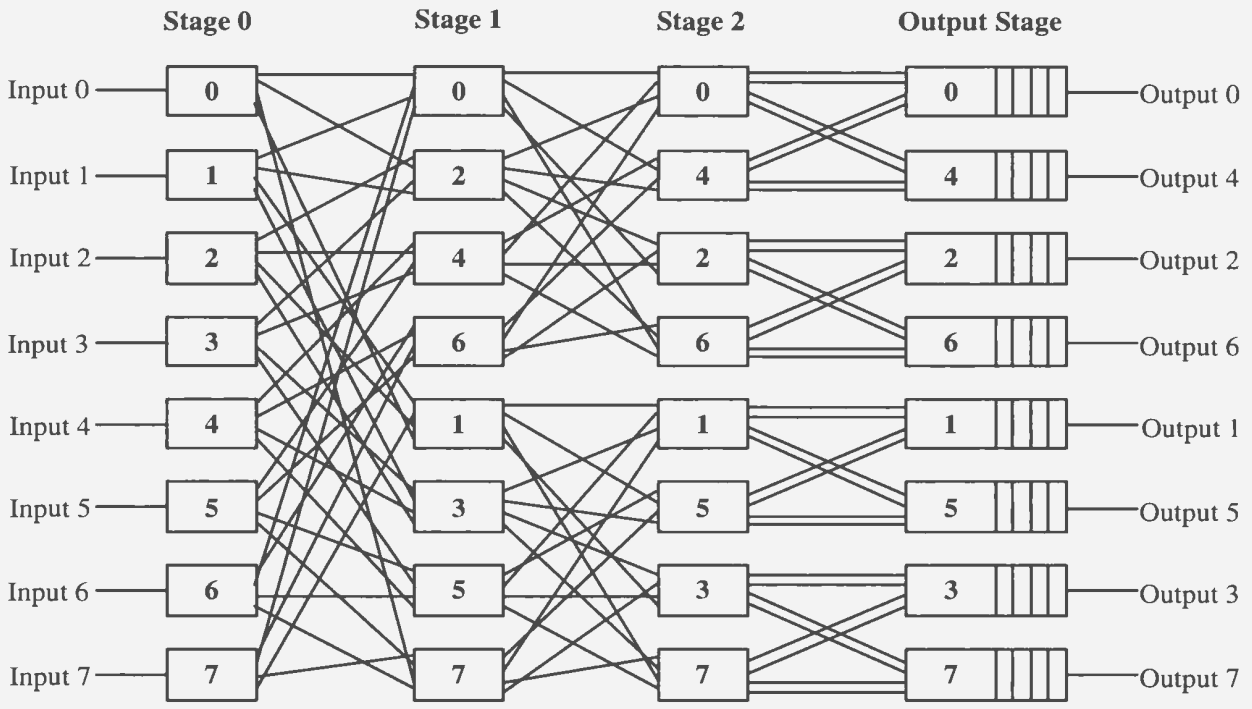


Figure 2.16: The structure of the 8×8 original BG switch

i represents the position of the SE within a stage and j represents the stage number. Each SE has four output links connected to the SEs in the adjacent downstream stage. The output links from each SE are numbered from 0 to 3 from top to bottom, and are divided into the upper links and lower links. Among them, links 0 and 2 are called primary links and links 1 and 3 are called alternative links. For $SE_{i,j}$, output link 0 will be connected to one of the inputs of $SE_{i-2^j,j+1}$, output link 1 will be connected to $SE_{i,j+1}$, output link 2 will be connected to $SE_{i+2^j,j+1}$, and output link 3 will be connected to $SE_{i-2^{j+1},j+1}$.

2.4.1.2 Routing Algorithm

An efficient routing algorithm is a key design task of a network architecture. In the original BG network, a distance tag algorithm was used [24, 25]. In the algorithm, the IPCs calculate the routing tag by $D - S \bmod N$, where S and D denotes the source

and destination number respectively. The SEs interpret the routing tag in a reversed order, i.e., the SEs of stage 0 use the least significant bit for their routing decision. If the routing bit is 0, the cell is routed to the upper links, otherwise, it will be routed to lower links. During the routing process, the primary links are always used first before the alternative links are used. However, the routing tag has to be modified for cells directed to the alternative links in the algorithm. In any given switching cycle, any SE can take up to two cells to either upper links or lower links. If there are more than two cells requesting the same output port, the excessive cells will be dropped by the SE, thus cause cell blocking.

2.4.1.3 Modified BG Network Structure

Another routing algorithm called the reversed destination tag was introduced to reduce the complexity of the routing tag generation and modification [14]. In this algorithm, the routing tag for a cell from source S to destination D is the binary representation of D, i.e., $d_{n-1}d_{n-2}d_{n-3}...d_0$. Each SE interprets the tag in the reverse order: SEs in stage 0 switch a cell based on bit d_0, SEs in stage 1 switch a cell based on bit d_1, and SEs in stage $n-1$ make their routing decision based on bit d_{n-1}. Each SE is associated with a value α given by the formula $\alpha = \left\lfloor \frac{i}{2^j} \right\rfloor \bmod 2$ to make the routing decision. If $d_j \oplus \alpha = 0$ for an incoming cell, the SEs route this cell through output link 0 or link 2, else it will route the cell to link 1 or link 3. The difference between this routing algorithm and the previous one is that there is no need to modify the routing tag if the cell is sent to the alternative links.

In the study of the pipelined architecture using the BG network [14], further modification was made to the reverse destination routing algorithm. This modification reduced the complexity of making routing decision in the SEs. The SE now checks only the routing bit, instead of both the routing bit and parameter α. If the routing

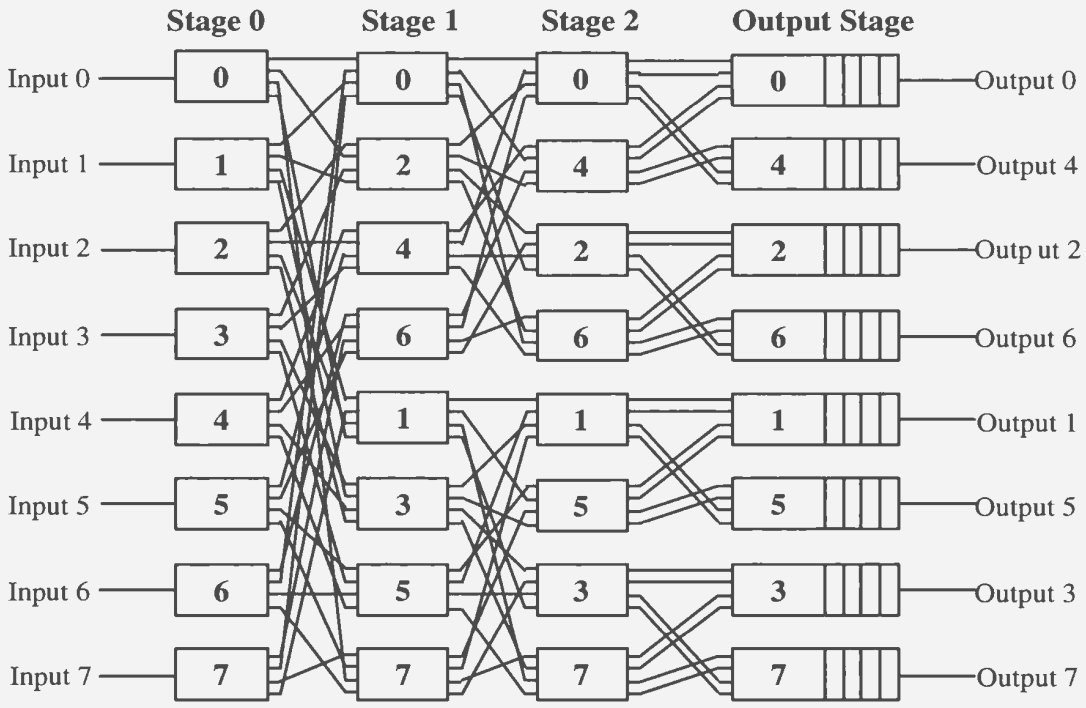


Figure 2.17: The modified structure of the 8×8 BG switch

tag bit is 0, output link 0 will be selected. Output link 1 will be used only when link 0 is occupied. The same approach applies to the down links when the routing bit is 1. The network topology was revised accordingly to accommodate this change, as shown in Figure 2.17.

2.4.2 Features of the BG Network

Previous research on different aspects of the unicast BG switch architecture has been reported in [24, 25, 67, 27, 26, 14, 68], which include its hardware complexity, fault tolerance property, reliability, scalability, and performance. It has been demonstrated that the BG network architecture is single fault-tolerant and robust when multiple faults exist. It is highly reliable and scalable. Its performance is very close to that of an ideal non-blocking switch, and is better than for other switches which have similar or higher hardware complexity. It is a promising MIN candidate to handle unicast

traffic in broadband packet switching networks.

2.5 Switch Fabric Benchmarking Framework

The performance comparisons between different architectures are not straightforward due to the lack of a common criteria of defining adequate network traffic models. Currently in industry, switch fabric selection follows complex performance comparisons using the data that the switch fabric vendors provide [10]. However, the data are very limited and commonly based on idealistic traffic patterns, rather than those of the real-world, application-oriented traffic that would stress fabric implementation. To facilitate the work of evaluating different architectures, the Network Processing Forum (NPF) [69] was founded in February 2001 by the merge of the Common Switch Interface Consortium (CSIX) and the Common Programming Interface Forum (CPIX). Their recent effort is to develop the benchmark framework for switch design, testing and comparison. The benchmarks that have been proposed include traffic modelling, performance metrics, testing methodology, and test benches.

2.5.1 Benchmark Traffic Models

Modelling data traffic in a real network is extremely difficult because traffic patterns vary significantly with network type, topology, time, and other parameters [10]. In recent years, some statistical models have been used by academic and industrial researchers to analyze traffic patterns [70, 71]. The following aspects are the key points that must be properly addressed in a traffic model [10, 69]:

(1) Traffic load behavior;

(2) Bursty nature and correlation in traffic arrivals;

(3) Traffic destination selection distribution;

(4) The distribution of traffic quality-of-service (QoS) classes;

(5) Proportion of unicast and multicast traffic and their characteristics.

By strictly defining the parameters to the above points, NPF characterizes network traffic using the following processes:

1. Packet size distribution

The distribution is used to describe variable-length packet, such as in the case of IP traffic in the Internet, or fixed-size packet, such as in the case of ATM.

2. Packet Arrival Process

The Bernoulli arrival process is used to define a memoryless random arrival pattern. Bursty arrival is used when packet arrivals are correlated. There are several bursty traffic models defined by NPF, which include the constant burst size arrivals, the ON-OFF Markov-Modulated arrivals and its modification, and the ON-OFF arrivals model with minimal burst size.

3. Packet Destination Distribution

In some applications, traffic may tend to be uniformly distributed among all destinations. However, in other applications, traffic destinations may be unevenly distributed. In the NPF benchmark, traffic destination distributions range from uniform, through to hot spot, and indeed to the special case in which each port has traffic requesting only a single destination [69]. The Zipf's law [72] is used for the destination distribution in NPF proposal.

4. Multicast Traffic

In multicast traffic, the term fanout is defined as the number of destinations to which a packet will be replicated. The fanout is a configurable parameter which can be a constant but also can vary.

5. QoS Modelling

A QoS distribution defines the probability of a generated packet's association with a given QoS class. The QoS can be uniformly distributed, or arbitrarily distributed

among the classes. The QoS distribution is independent of the arrival process and the destination distribution. However, in bursty traffic, packets within the same burst belong to the same QoS class.

2.5.2 Benchmark Performance Metrics and Test Suites

The initial purpose in creating the benchmark was to determine common criteria for testing third party's switch fabrics. For performance metrics, the goal of NPF is to establish an unambiguous specification that defines latency and jitter as well as to delineate the way vendors measure them. The testing environment follows a common switch interface (CSIX) reference model between a traffic manager and a switch fabric for ATM, IP, MPLS, and other data applications, as shown in Figure 2.18. The traffic models unit generates packets based on the specified traffic type and passes them sequentially to the segmentation unit (SAR). In SAR, packets are segmented into cell frames, which are then queued into the pseudo traffic manager (pTM) unit. From pTM, cell frames are passed onto the device under test (DUT), normally a switch fabric, for switching. A number of points shown in the figure are used for performance measurements, which include latency, accepted-versus-offered bandwidth, and jitter. Latency and jitter have two sets of measurement points, one for the fabric (point 2 and 3) and the other for the total system (point 1 and 3).

The test suites benchmark includes a wide variety of testing scenarios under different traffic conditions. For unicast traffic, it includes the uniform destination distribution and non-bursty traffic pattern, the uniform destination distribution and bursty traffic pattern, the non-uniform destination distribution and non-bursty traffic pattern, the non-uniform destination distribution and bursty traffic pattern, and the variable non-uniformity factor port level test. Test scenarios in multicast condition include the test for pure and combined multicast traffic under overload and

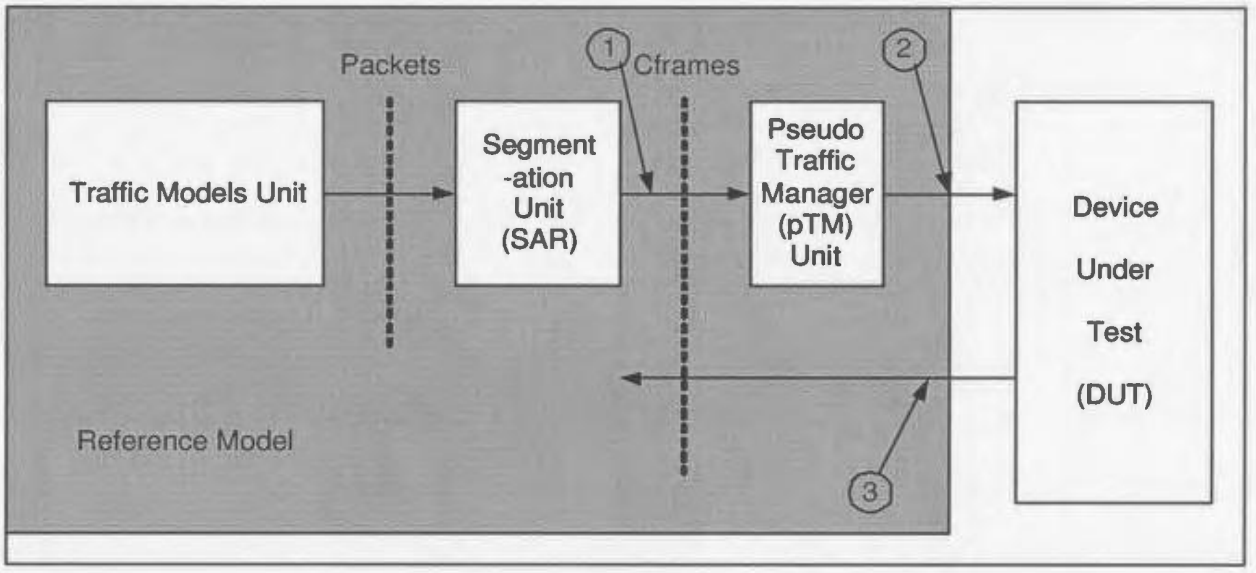


Figure 2.18: The CSIX reference model (from [10], pp. 109)

no-overload bursty and non-bursty traffic. Currently, the destination distribution for multicast traffic is considered to be uniformly distributed only.

2.5.3 Relation to this Research

As stated earlier, the main purpose of the benchmark system is to obtain the performance of third-party SFs. Therefore, the SF is treated as a black box in the test and what is normally considered is its port-level behavior. However, this is not sufficient for study of an architecture because there are other important aspects to be covered. Therefore, in this research, besides the performance analysis, which is no doubt the most important aspect, other practical concerns such as hardware complexity, fault tolerance, scalability, reliability, and implementation feasibility are investigated.

The traffic models outlined in the benchmark significantly facilitate the process of identifying standard traffic, which makes the performance analysis across different architectures comparable. Such traffic models are used in this research. For example, the uniform model and the ON-OFF bursty model for traffic arrivals, and the uniform

and non-uniform models for destination selection. However, for multicast traffic, the fanout distribution is not very clearly described in the benchmark framework. Therefore, the truncated geometric distribution, which is commonly used in the literature [9, 60] to model and generate multicast traffic fanout, is used. As well, the multicast test scenarios from the benchmark test suites only deal with the uniform destination distribution. In the dissertation, it is extended to the non-uniform situation. A model modified from the original non-uniform model presented in [73] is used. The fixed-size packets, or cells, are considered for switching inside the SF. However, QoS control is beyond the scope of this research, hence, QoS modelling and the jitter performance of the single switch are not covered.

2.6 Summary

In this chapter, we discussed the classification of various architectures that have been proposed for high-speed packet switches. The pros and cons of different categories are briefly analyzed. It is hard to single out any one among the architectures as the best. Multistage Interconnection Networks have attracted more research interest because of the desirable properties of the structure, such as self-routing, equal latency, modularity and scalability. As multicasting capability is becoming a general requirement for future high-speed networks, the two common approaches of constructing a multicast switch fabric are studied with an extensive survey of the existing multicast switches. The integrated approach is promising because it inherits the advantages of the MIN design. At the same time, it avoids the overflow problem in the copy network solution and can be designed to handle output contention efficiently. Also, in this chapter, we briefly reviewed the history of the BG switch and present the recent development in switch fabric benchmarking.

Chapter 3

Multicast Balanced Gamma (BG) Switch

3.1 Introduction

The Balanced Gamma network is a fault-tolerant and reliable MIN that was first reported in [24] as a broadband switch architecture. The performance, fault tolerance and reliability properties of the BG network for unicast traffic have been extensively studied and it has been shown that the BG network is superior to other well-known MINs that have similar hardware complexity, such as the 2-dilated 2-replicated (2D2R) Banyan network [26], and that it performs much better than the crossbar network, which has a higher hardware complexity [14]. To support multicast traffic, a new switch architecture has been developed to support efficient implicit multicasting, and at the same time, preserving other attractive features of the BG network.

3.2 Switch Architecture

Unlike the Banyan network, which utilizes a 2×2 SE, the BG multicast switch utilizes basically the 4×4 SE. The basic architecture of an $N \times N$ BG multicast switch consists of N IPCs, an $N \times N$ multistage interconnected SF that supports self-routing, copy

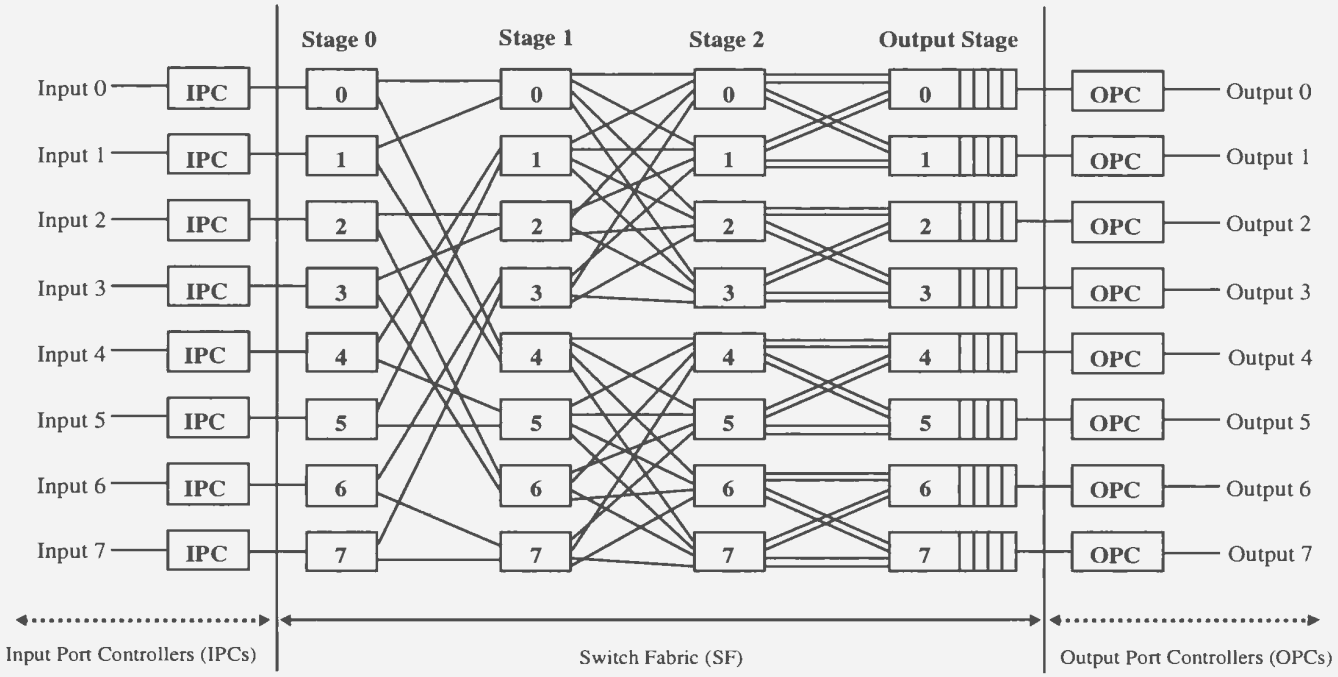


Figure 3.1: The architecture of an 8×8 multicast BG switch.

replication and delivery acknowledgement, and N OPCs. No dedicated copy network is required. Routing and replication are performed in a distributed fashion inside the SF. Figure 3.1 shows the architecture of an 8×8 multicast BG switch. The difference between this architecture and the original BG architecture that has been shown in Figure 2.17 is that the alternative links between Stage 0 and Stage 1 are removed. However, these links will be discussed again in Section 3.7 to demonstrate how they will help to improve the fault tolerance and reliability properties of the switch.

The IPC terminates the input signals from the network, strips the information contained in the cell header, and uses a lookup table to determine the destinations. In the dissertation, a simplified IPC is assumed, in which the destination requests contained in the cell header have been translated into a bitmap string, with a bit corresponding to each output port. An internal tag generation circuit within the IPC uses the bitmap string to generate the self-routing and self-replication tag. The tag

is used internally for path reservation inside the SF. Different choices of tag encoding schemes will be studied in Section 3.4.3. A fanout splitting scheme [74] is used in the BG multicast switch to ensure efficiency when dealing with multicast traffic, i.e., copies of a multicast cell may be delivered to output ports over a number of cycles. Only those undelivered copies due to output or internal link contention are kept in the IPC and contend for the output in the next cycle.

The SF is the core of the multicast BG switch because most of the routing and replication functions are completed inside the SF. An $N \times N$ BG switch fabric consists of $n+1$ stages, where $n = log_2 N$, with each stage consisting of N SEs numbered from 0 to $N-1$. 1×2 SEs are used for stage 0 and 2×4 SEs are used for stage 1. Each of the following $n-2$ stages is comprised of 4×4 SEs. The last stage is the output buffer stage, which can accept up to 4 cells per output port in one switching cycle [24, 75]. The output buffers can be considered either as the last stage of the SF or as a part of an OPC along with other scheduling hardware. The OPC updates each arrived cell, unicast cell or copy of a multicast cell, with a new cell header and sends onto the output links.

Network bandwidth is expanded through the first two stages and then remains the same for all subsequent stages. Through internal bandwidth expansion, the multicast BG switch can achieve better performance while keeping the hardware complexity reasonable. The interconnection pattern between adjacent stages is regular. In the middle stages, the outlets of each SE are evenly distributed to the SEs in the following stage. Details about the algorithms which specify the interconnection pattern can be found in Appendix A for the forward data path and in Appendix B for the backward acknowledgement path.

3.3 Justification for the Architecture

Blocking is a problem with which every switch design must deal. Though some non-blocking switches are proposed, e.g., Batcher-Banyan network and crossbar network, they are typically non-blocking only in the unrealistic traffic pattern of permutation traffic, where the port requests for all incoming cells are different. When a more realistic traffic is applied, many of these switches inevitably become blocking with quick degradation of performance.

The multicast BG switch has slight blocking under permutation traffic. However, it keeps its high performance under many other traffic conditions due to the characteristics of the switch architecture design. Blocking occurs either internally when cells contend for the same internal link, or at the output port when the number of incoming cells exceeds the available queuing space. In the BG switch, cell loss due to the effect of blocking is minimized by providing input buffers and by incorporating a backpressure mechanism. The backpressure mechanism reports the blocking status to the IPC so that the blocked cell is kept at the input buffer. Output blocking is also mitigated by choosing a SF architecture that is capable of accepting multiple cells at each output line in one switching cycle [24, 75]. The input-buffered switch architecture may suffer from HOL blocking, in which the temporarily un-transmissible HOL cell impedes the transmission of cells behind it and thus reduces the switch throughput. However, due to the very high throughput of the BG switch, most HOL cells can be delivered immediately without being buffered and delayed [26]. Hence, the HOL blocking does not significantly degrade performance.

Output buffered switches have been shown to provide the best delay and throughput performance [60, 76]. It is difficult to achieve a pure output-buffered switch because the output buffers have to operate N times the link speed to avoid any cell

loss. As link speed becomes higher and for larger switch sizes, output buffer access time will eventually become the system bottleneck. With a suitable backpressure mechanism, the input-output buffering strategy will reduce the output buffer speed constraint and will store the cells that lose contention to internal links or output ports using the input buffer. Through this approach, we can not only achieve good performance and reduce the speed requirements of the output buffer, but also reduce the overall complexity to build a practical switch. With a high throughput SF, only small amounts of input buffering will be required.

An important observation which has enabled the architecture of the BG switch to produce near-optimal performance even under realistic traffic conditions is the design choice that the output stage of the SF is able to accept up to four cells per output line in one switching cycle. Under unicast uniform random traffic at full load, in any given switching cycle, the probability that there are i cells arriving at the input lines of a switch destined to a particular output line is given by

$$\Pr(i) = \binom{N}{i} \left(\frac{1}{N}\right)^{i} \left(1 - \frac{1}{N}\right)^{N-i}. \tag{3.1}$$

The result is plotted in Figure 3.2 for different switch sizes. A similar distribution is obtained through simulation for the multicast bursty traffic, as shown in Figure 3.3, in which an average burst length ($\overline{L}$) of 5 and a mean fanout ($\overline{F}$) of 2 are used. Details of the multicast bursty traffic model will be described in Section 4.2.

Figure 3.2 and 3.3 clearly demonstrate that there is only approximately a 1% probability that more than four cells arriving at the input lines in any given cycle would try to reach the same output line for a fully loaded switch. It should be noted that this is almost irrespective of the size of the switch. This observation justifies our architectural decision of the BG switch accepting up to four cells at each output line in one cycle. For the rare cases of more than four cells requesting the same output,

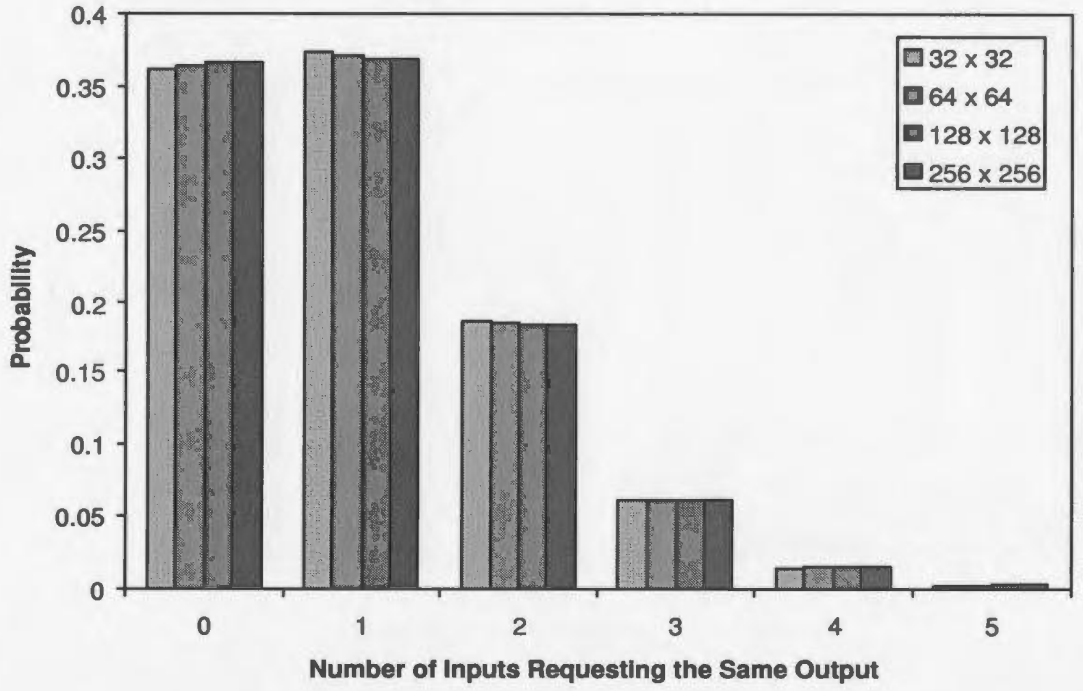


Figure 3.2: Output request probability under unicast uniform random traffic at 100% load.

the backpressure scheme would ensure that the excess cells are held in input queues. In addition, the BG network has a low probability of internal blocking, hence, input buffers should be used to hold the cells that would have otherwise been dropped. In Chapter 4, switch performances under nonuniform traffic are also studied and the high performance of the multicast BG switch architecture is demonstrated.

3.4 Dynamic-Length Self-Routing and Self-Replication Algorithm

3.4.1 Switching Operation

A three-phase switching operation is performed inside the multicast BG switch [18, 24, 77, 78]. First is the reservation phase during which the tag of the HOL cell is routed through the SF. Multicast cell replication is performed implicitly by the SEs

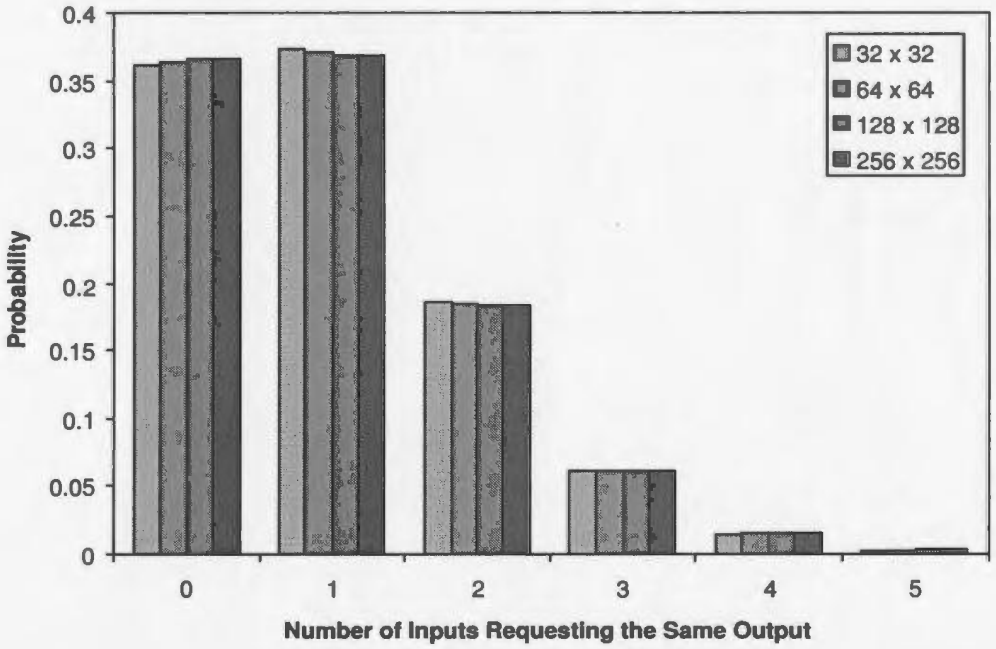


Figure 3.3: Output request probability under multicast bursty traffic at 100% load.

along with the routing operation only when necessary. Next comes the acknowledgement phase, during which cell delivery information is reported to the IPC by use of backpressure mechanism. Based on that information, the IPC decides whether the HOL cell should be transmitted or kept in the input buffer for the next cycle, or both when in the case of a multicast cell. In the third phase, the payload is transmitted via the established path just as in a circuit switch. Because of the memoryless design of the SF, a cell will either reach the desired output port(s) or be kept in the input buffer. Cell sequence can be easily maintained. Cell loss occurs only when the input buffer is full and a new cell arrives.

3.4.2 Self-Routing and Self-Replication Algorithm

In the multicast BG switch, there are three types of SEs inside the SF: the 1×2 and 2×4 SEs are used for the first two expanding stages while the 4×4 SEs are used for all subsequent stages. Functionally, the first two types of SEs can be treated as

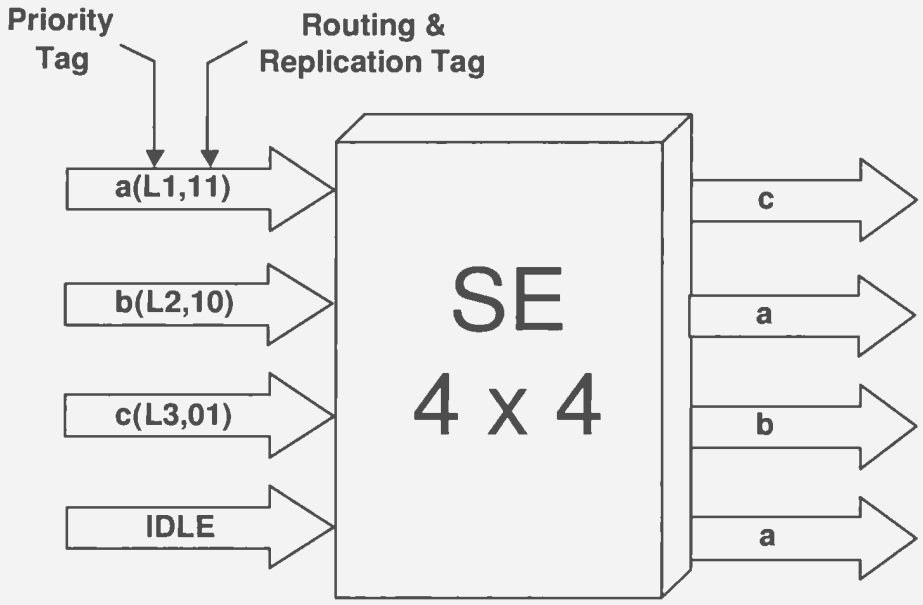


Figure 3.4: Self-routing and cell replication in the 4 × 4 SE.

a special (simpler) case of the 4 × 4 SE. Therefore, in the following discussion, the more general 4 × 4 SE is used, as shown in Figure 3.4. The four output links are numbered 0 to 3 from top to bottom. Among the four links, link 0 and link 2 are called upper and lower regular links, respectively, while link 1 and link 3 are called upper and lower alternate links, respectively. Both the regular link and its alternate link have the same capability of reaching the same destination. Upon switching, the regular links are always used first. The alternate link is used only when the regular link has been assigned to a connection.

In the multicast environment, tag design becomes more challenging because not only the routing information should be carried but also the cell replication information, and the tag length should be minimized to minimize the delay in the reservation phase. In the multicast BG switch, for each SE to make the right routing and replication decision, a 2-bit tag is used by each SE for each input link. Four different actions can be taken by SE and are summarized in Table 3.1.

Bit 1	Bit 0	Routing Action	Replication Action
0	0	Idle (no action)	Idle (no action)
0	1	Lower link	No replication
1	0	Upper link	No replication
1	1	Both links	Replication

Table 3.1: Routing and replication actions based on tag pair information.

The SE will make its decision in two steps. First, the SE decides the process order of incoming cells based on the priority level associated with each cell. For cells of the same priority level, they are selected based on the processing order. Priority switching is a feature considered in the multicast BG switch with up to 8 priority levels currently supported. The priority consideration will be further discussed in Section 3.6. Second, incoming cells are switched following the order determined in the first step. Cells with higher priority are always processed first until all the sources are used up. In that case, remaining low priority cells will be blocked. Pseudo-code for the self-routing and self-replication algorithm can be found in Appendix C.

Figure 3.4 provides an example in which cells are coming in from the top three input links. By sorting on the priority tag, the process order is $c \rightarrow b \rightarrow a$. Following the routing and replication table, cell c is a unicast cell which requests an upper output link, it is switched to output 0 and similarly cell b is switched to output 2. For cell a, tag pair '11' indicates that replication is required. The available outputs are checked and cell a is replicated and sent to both upper and lower alternative output links, links 1 and 3, respectively.

3.4.3 Dynamic length Bitmap Tag Encoding Scheme

To switch a cell from each input to the required destinations, an internal tag must be carefully and properly selected. First, it should contain enough information for each

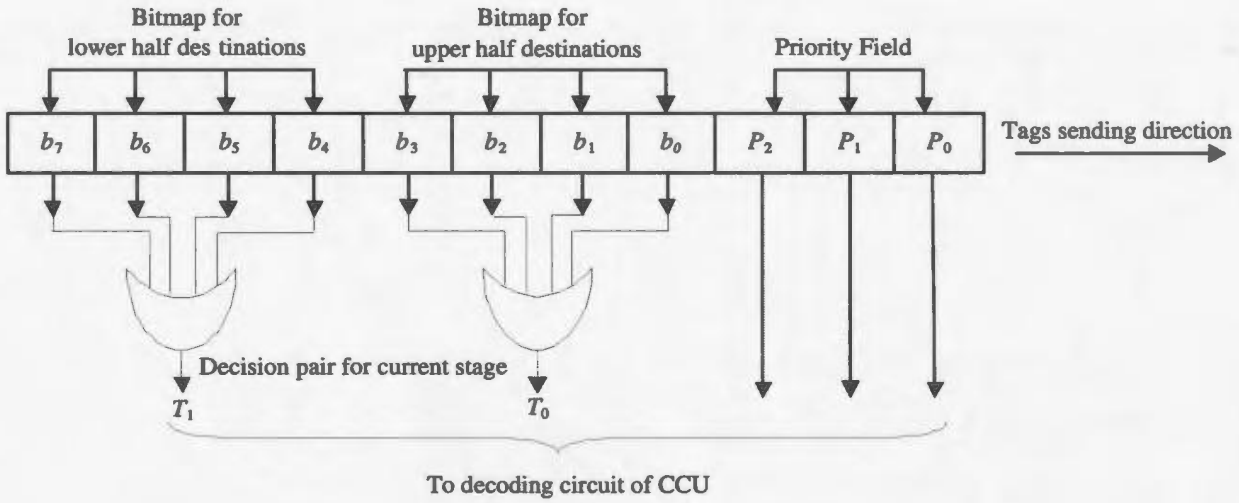


Figure 3.5: Decision pair generation in the variable bitmap tag scheme.

SE to make the routing and replication decision. However, the tag is an overhead to the system, thus it should be minimized so as to reduce communication latency and save system bandwidth. In [79], the authors studied different multi-address encoding schemes for multicast and pointed out that a good encoding scheme should consider minimizing the tag length, reducing tag processing time, and supporting cut-through switching.

The dynamic length bitmap tag encoding scheme is similar to the bit string encoding scheme described in [79]. The tag size changes as the cell is switched through the switch fabric. In this scheme, the destination requests are represented in the bitmap format, with each bit corresponding to an output port. Because fewer output ports are associated with each SE in later stages, the tag length is halved after passing through each stage. Based on the bitmap received, the SE generates the decision pair before sending it to the decoding circuit. The decision pair generation circuit is quite simple, the upper half of the destination ports are used to generate the first bit, and the lower half will be used for the second bit. As shown in Figure 3.5 for the SEs in $Stage_1$ of a 16×16 BG multicast switch, a simple ORing operation is sufficient.

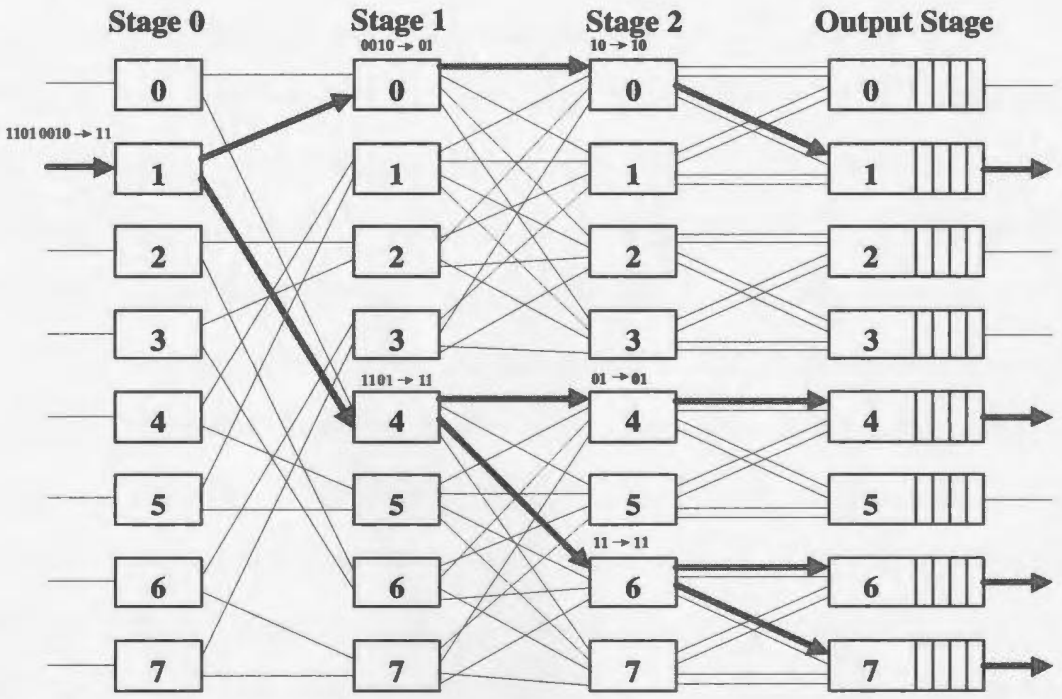


Figure 3.6: Dynamic length bitmap tag and self-routing and cell replication algorithm.

Excluding the priority bits, the sum of the tag length over all links is

$$\begin{aligned} S_{Tag} &= N + \frac{N}{2} + \frac{N}{4} + ... + 4 + 2 \\ &= 2^n + 2^{n-1} + 2^{n-2} + ... + 2^2 + 2 \\ &= \sum_{i=1}^{n} 2^i \\ &= 2N - 2. \end{aligned} \tag{3.2}$$

Figure 3.6 shows an example of a multicast cell from input 1 with output requests of 1, 4, 6 and 7. Using the dynamic length bitmap tag encoding scheme, the decision pair can be extracted from the bitmap string using the circuit depicted in Figure 3.5. The SE controller sorts the four input lines on their priority first, then interprets and processes the decision pair based on the resulting sequence, i.e., from the highest priority to the lowest. The bitmap tag is then divided into two groups, each attached with the priority bits, then sent to the next stage. The first group is used as the tag

for the cell to be sent to the upper link and second group is used as the tags for a cell to be sent to the lower link. In the example, at Stage 0, based on the incoming tag 11010010, the $SE_{1,0}$ generates a decision pair of (1,1). Using Table 3.1, the SE knows that both routing and replication should be done to this cell. So, one copy of the cell is sent to the upper link and the other copy is sent to the lower link. Therefore, the tag 11010010 will be split into two groups: 0010 and 1101. Each will be appended to the priority bits and then sent to the respective upper and lower links. A similar approach will be taken in the SEs of the following stages. In this example, priority bits are not shown. In general, when several cells appear at the inputs of an SE, the priority bits will be considered first before routing and replication decision are taken.

The strength of this scheme is its small tag transmission latency and low tag buffering requirement. Because fewer overhead bits are transmitted after passing each stage, switch throughput and bandwidth are better utilized.

3.5 Dynamic-length Backpressure Algorithm

An acknowledgement mechanism to report blocking status is needed so that cells can be kept in the input buffer when blocking occurs. As long as cells get queued in the IPC, they will not be lost, but might be delayed when blocking occurs. The backpressure unit in each SE forms a path to pass blocking information to the IPC. Compared to that of the unicast BG switch in [14], an efficient backpressure mechanism design in a multicast switch becomes more complex because the IPC should get the acknowledgement for all possible copies.

In the multicast BG switch, a dynamic-length backpressure algorithm is used. In the algorithm, blocking status is encoded in a bitmap format, with each bit representing delivery status to one output port. Each SE only sends the output information

that it is associated with to the previous stage. At the output stage (Stage n), the only needed information is whether the cell can be placed in the output buffer or not. Hence, 1 bit suffices. In the second last stage (Stage $n-1$), each SE is connected to two output queues in the last stage. Thus, two bits are used as the backpressure output. A similar approach applies to all the SEs at different stages. In Stage 0, SE receives the acknowledgement from its downstream stages and generates its backpressure information of N bits based on the connection type and the blocking condition. The N-bit information is sufficient to decide the blocking status for any cell, unicast or multicast. Upon receiving this information, the IPC makes the final decision on whether all copies of the multicast cell are successfully received at the output. If yes, the IPC removes the HOL cell and tries the next cell during the next switching cycle. Otherwise, this cell is retained in the input buffer and retried until all copies are delivered. Using a work conserving method as described in [74], for any multicast or broadcast cell, only blocked copies will be retransmitted.

The BG network architecture enables us to achieve an efficient design and an easy implementation of the dynamic-length backpressure algorithm. The backpressure unit located at each SE decides its backpressure information using two steps: the first step is to decide whether the cell is blocked at the current stage or not, and the second step is simple concatenation operation.

If there is no blocking, in the case of a unicast cell, the SE waits for the acknowledgement from the link where the tag is sent out and concatenates it with a string of zeros. Where the 0s string will be concatenated to the acknowledgement stream, i.e., head or tail, depends on whether the upper or lower output link is selected. In the case of a multicast cell, the SE simply waits for the acknowledgement string from downstream stages and concatenates that from the upper link with that from the lower link. The resulting string will be sent to SE in upstream stage. If

blocking occurs, the SE generates a string of ones as the negative acknowledgement for the copy/cell that loses the link contention immediately. Similar concatenation operation will be performed, just this time, the acknowledgement string received from downstream stages is replaced by the 1s string generated locally at the SE.

For a multicast cell, if one copy loses contention for one link while the other copy wins its contention for the other link, the control becomes more complex. In this case, in the backpressure information of this stage, half of the outputs, which are associated with the blocked link, are marked as blocked by using the 1's string. For the other half that is associated with the copy that got through, the SE waits for the downstream backpressure information to come back, and then concatenates them to generate the complete backpressure bits and sends to the upstream stage.

The only thing a SE needs to remember is its stage number so that it can decide the length of its acknowledgement information. For SEs at Stage i ($0 \leq i \leq n-1$), the incoming backpressure information length is 2^{n-i-1} bits and the outgoing backpressure information is 2^{n-i} bits. The detailed design and implementation of the backpressure unit inside the SE will be described in Chapter 5. Pseudo-code for the dynamic length backpressure algorithm can be found in Appendix D.

It might seem that more output ports than the real port requests would be marked since all output ports related to a SE are marked as blocked when blocking occurs on that link. In fact, when the IPC receives the acknowledgement from the SF, an ANDing operation is performed to determine which requests are actually blocked. For example in a 16×16 BG switch, assume that the IPC has a HOL cell with port requests 0, 4, 6, 7, 12 and 15. In IPC, the port request is represented as 1001000011010001. If the copy to the port 4 is blocked at stage 2 while other copies reached their destinations, then the IPC receives a backpressure string of 0000000000110000. After a logic operation (1001000011010001 AND 0000000000110000), the result is 0000000000010000. It

is then clear that the cell is correctly delivered to all other ports except 4. Therefore, the cell will still be kept in the input buffer. However, only the copy to port 4 will be retransmitted in the next switching cycle.

The dynamic-length backpressure algorithm is used in Figure 3.7 to show how self-routing, cell replication, acknowledgement and blocking are handled in an 8×8 multicast BG switch. Output requests of incoming cells are given in brackets.

In this example, blocking occurs at $SE_{2,2}$ for the unicast cell from input 5 and at $SE_{4,2}$ for a multicast cell from input 4. Cells from other input ports are all successfully transferred. Blocking information, as shown on each SE in the diagram, is sent along the backward path until it reaches the IPC. The IPC then knows that the unicast cell is blocked and the copy of the multicast cell heading towards output 4 is blocked as well. Since there is only one copy left for the master cell, in the next switching cycle, it becomes a unicast cell. So, it is re-sent along with the unicast cell left over in input 5 and new incoming cells in the next switching cycle.

3.6 Service Discipline and Priority

3.6.1 Service Discipline

In a multicast SF, two kinds of service disciplines can be used: the no-fanout splitting scheme and the fanout splitting scheme [74]. In the no fanout splitting, all copies of a cell must be sent in the same switching cycle. If any copy of the cell loses contention for an output port, the cell will not be transmitted but will be tried again in the next cycle. In the fanout splitting scheme, copies of a multicast cell may be delivered to output ports over a number of cycles. Only those copies that are not successfully transferred will contend for the output in the next cycle.

Using the no-fanout splitting scheme, if the traffic load is high and the priority

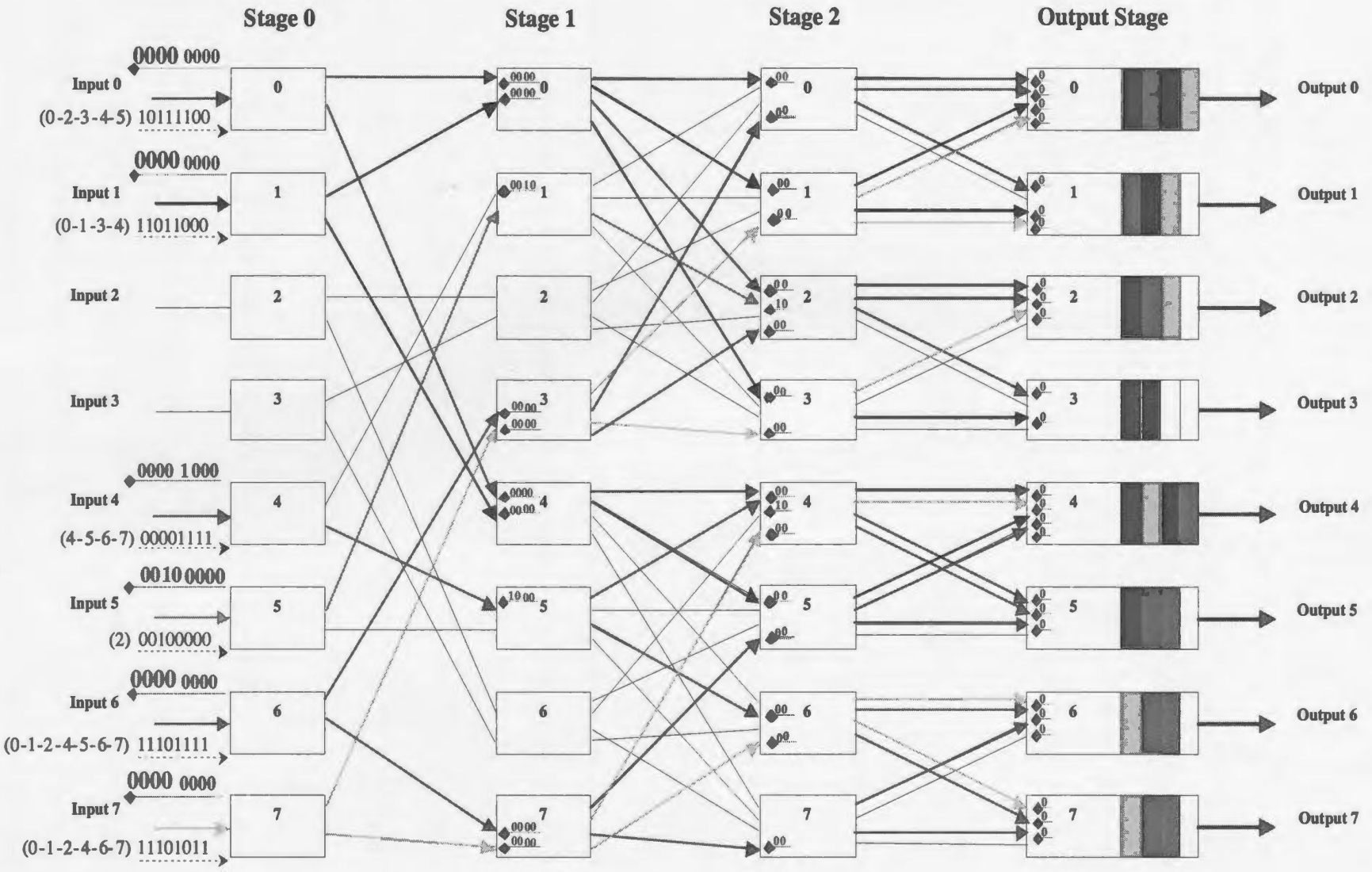


Figure 3.7: Dynamic length backpressure scheme for an 8×8 BG multicast switch.

associated with the multicast cell is low, a multicast cell may be blocked at the HOL for a long time, and impede successive cells from being switched. The fanout splitting scheme is more work conserving and it enables a higher switch throughput. Therefore, it is a more realistic solution and it is implemented in the multicast BG switch design. The dynamic-length backpressure algorithm fits well with the work conserving service discipline.

3.6.2 Priority Consideration

Priority bits are used by SEs to decide which cell(s) will be allowed to get through when there are more requests than the available resources. In practice, the number of priority levels depends on the number of service levels that the network operator wants to support. In an integrated services network, at least three priority levels are necessary: a high priority level for urgent messages, usually for network control; a medium priority level for guaranteed service traffic; and a low priority level for best-effort traffic. Considering the many different applications in future broadband networks, in the multicast BG switch, a priority hierarchy is designed to support up to eight levels, starting from level 0 (L_0) to level 7 (L_7). In the analysis and simulation, multicast traffic are treated equally to the unicast traffic. It is the operator's decision whether to give a higher priority to the multicast traffic or not.

3.7 Fault Tolerance and Reliability

In this section, some possible architectural modifications with the objective to improve fault tolerance and reliability of the switch are studied. All architectural improvements are compatible with the original functions and features discussed earlier.

Fault tolerance and reliability are desired features for any practical switch. A fault-tolerant SF is able to send a cell to its destination in the presence of link or

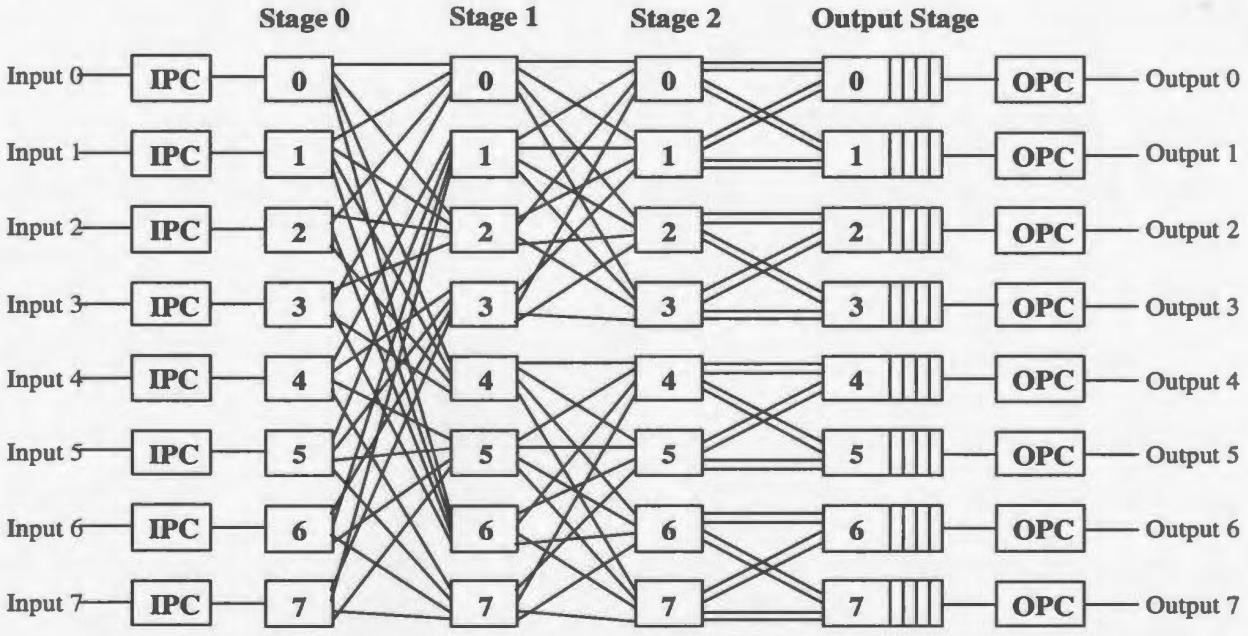


Figure 3.8: A single fault-tolerant and robust architecture.

SE failures. From Stage 1 and on, each SE has four output links, which are divided into two groups with two links in each group. One is used as the regular link and the other as the alternate link. When the regular link or the downstream SE connected to the link fails, the SE can always re-route the cell to the alternate link, which has the same capability of reaching its destination. However, since there is no such alternate links in between Stage 0 and Stage 1, failure of an inter-stage link or SE in Stage 1 would prevent the cell from reaching its destination. To keep a consistent fault-tolerant property for the whole SF, links between Stage 0 and Stage 1 are replicated and rearranged, as shown in Figure 3.8. The modified architecture can be designed to tolerate single fault and are robust in the presence of several faults. Only two types of SEs exist in the SF: the 1×4 SEs are used for Stage 0 and 4×4 SEs are used for all remaining stages.

3.8 Scalability

The scalability of the multicast BG network are studied from three different aspects, namely, the architectural scalability, the implementation scalability and the performance scalability.

3.8.1 Architectural Scalability

The multicast BG switch has a scalable architecture. A larger switch can be efficiently constructed using a smaller switch as the building block. For example, using the single fault-tolerant architecture shown in Figure 3.8, to make a 16×16 switch, we can simply replace Stage 0 of the 8×8 switch with 4×4 SEs, duplicate the 8×8 switch module, add one more stage which consists of 1×4 SEs in the front and connect them using the pattern specified in Appendix A.

The 1×4 SEs in Stage 0 can be replaced by the 4×4 SEs so as to achieve even better architectural scalability. Only one input link of each SE in Stage 0 will be used. Others can be either grounded or can be used as alternative links by the IPC to achieve better fault tolerance and reliability. Because only one type of SE is used for the whole switch, it makes the switch more scalable and easier to build. Using this structure, to achieve a bigger switch, the switch of smaller size is duplicated and one stage of 4×4 SEs is added in the front, thus the switch size is doubled, as shown in Figure 3.9.

3.8.2 Implementation Scalability

The main factor considered in the implementation scalability is the overhead length associated with the tag transmission from stage to stage. The overhead associated with the multicast BG switch is a bit different from that of the unicast BG switch despite one thing in common: the tag received by SEs in the first stage is the longest.

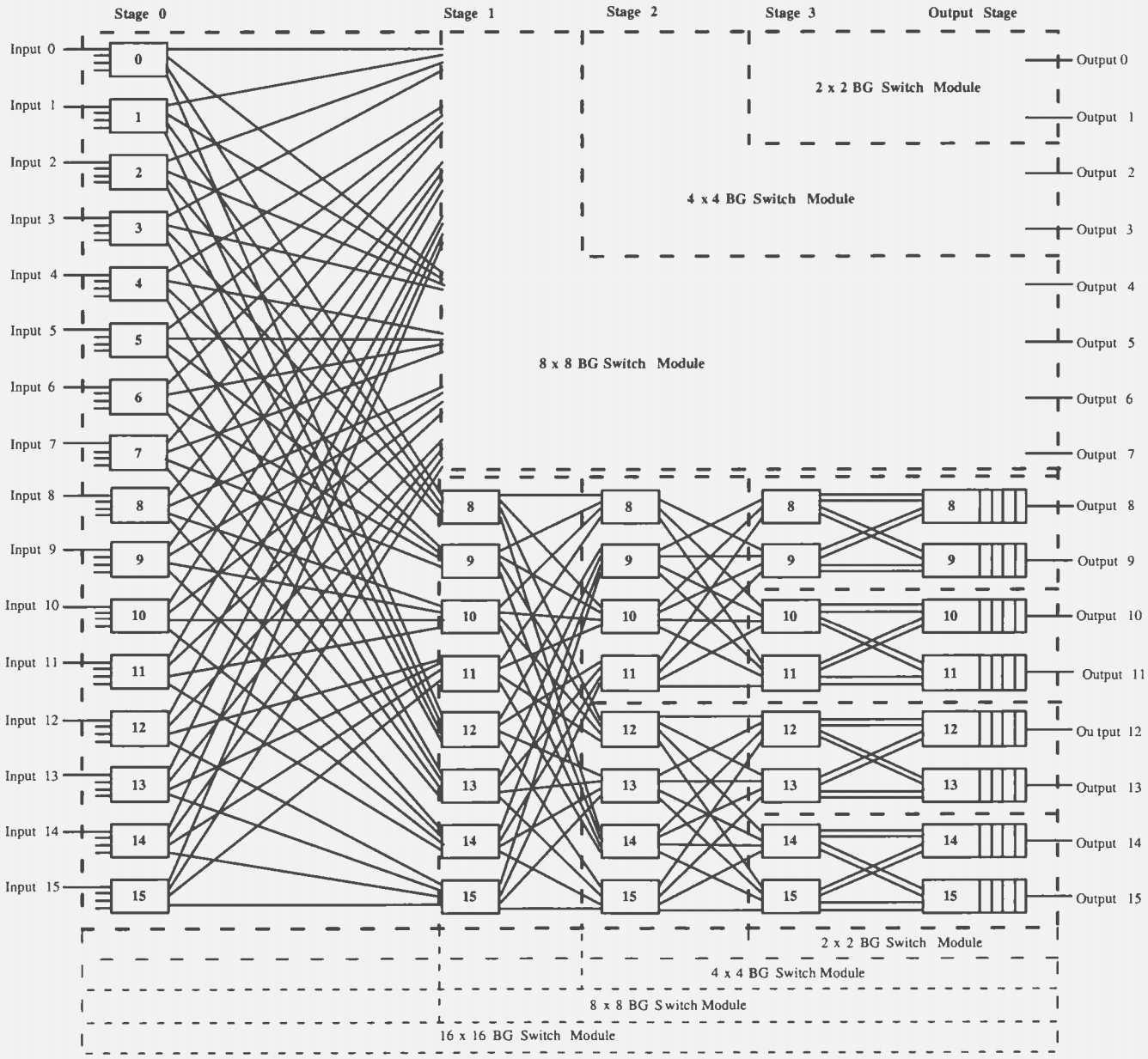


Figure 3.9: Architectural scalability of the BG multicast switch.

For the unicast BG switch, the longest tag is n bits, which is much less than the payload. For example, for a 256×256 switch, the tag sent to Stage 0 is 8 bits for routing plus the priority bits and an active bit (which is used to indicate whether it is an active cell or not). Compared to the payload, such as an ATM cell, which is $48 \times 8 = 384$ bits, the overhead can almost be neglected. While in the multicast BG switch, more information is carried by the tag. The maximum tag length now becomes N bits. As the switch size increases, the tag length grows linearly. Using the same example, for a multicast BG switch of 256×256, the maximum tag length now becomes 256 bits. In our research, we have attempted to improve the situation. The explicit active bit, which is required in the unicast switch, has been removed. As well, the dynamic-length algorithm has been designed to reduce the tag length by half for every stage along the path. For an incoming cell to traverse the whole SF, the sum of the tag bits to be passed is $2N - 2$. We conjecture that any integrated multicast MIN switch that supports implicit cell replication must have a tag of at least $O(N)$.

3.8.3 Performance Scalability

The performance scalability describes how the BG performance is maintained as the switch size grows. Here, performance scalability is demonstrated in terms of throughput and average cell delay. Throughput refers to the ratio of the number of delivered cells to the total number of incoming cells over a given period of time. As shown in Figure 3.10 and Figure 3.11, under a multicast bursty traffic with 80% offered load, the multicast BG switch preserves good performance scalability. The detailed performance study will be carried out in Chapter 4.

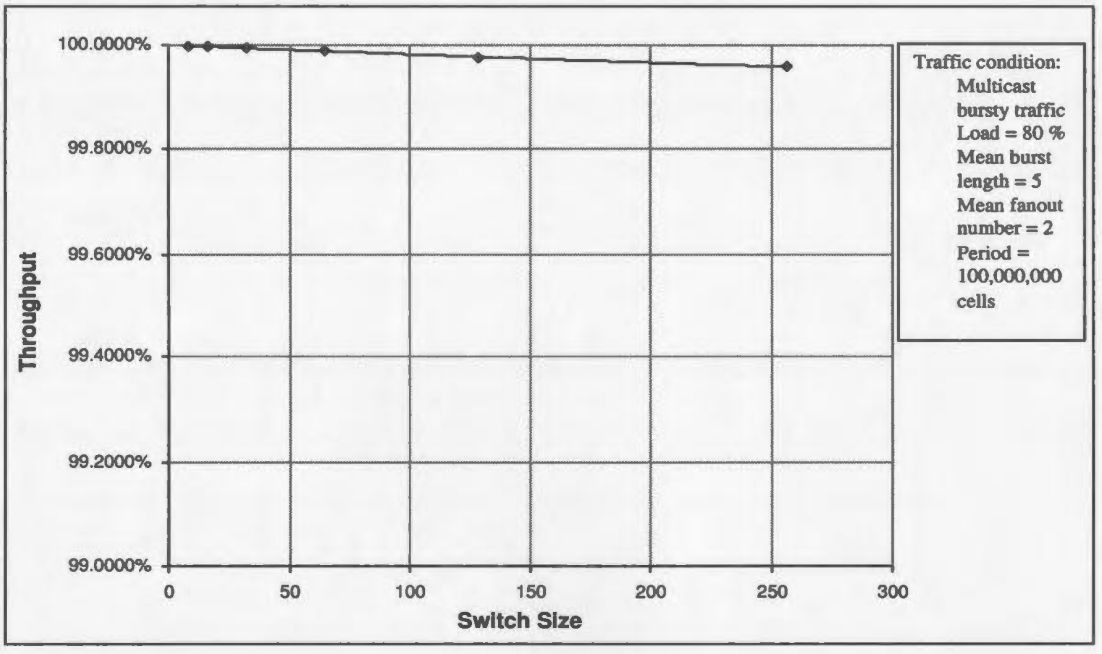


Figure 3.10: Performance scalability of the BG multicast switch – Throughput.

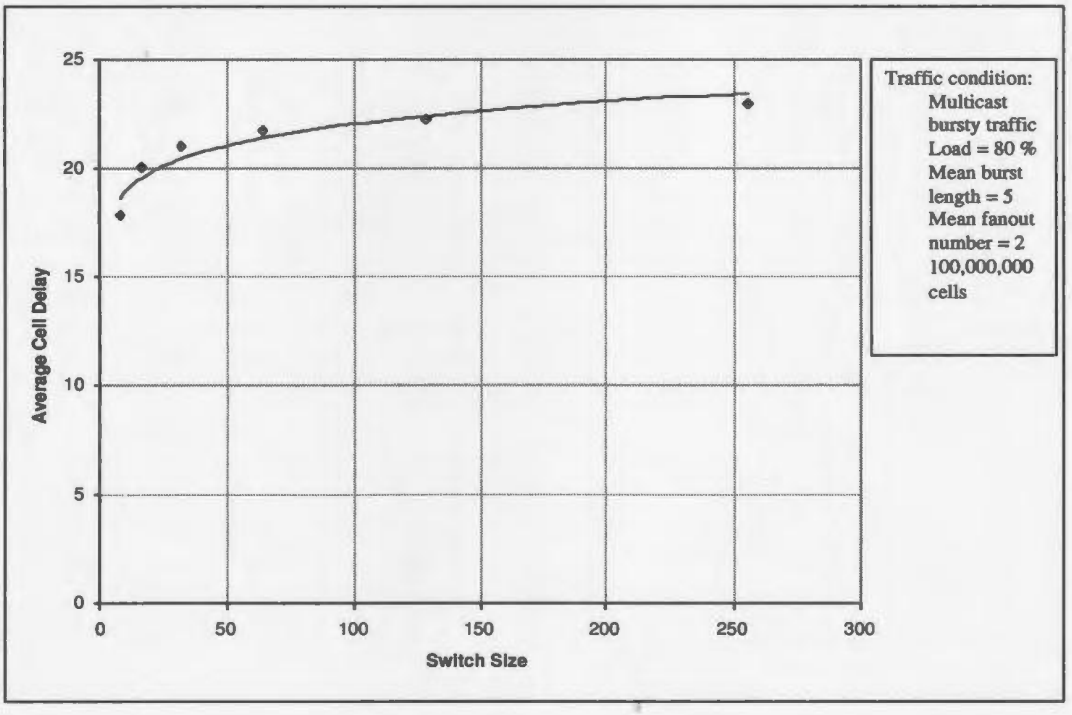


Figure 3.11: Performance scalability of the BG multicast switch – Average cell delay.

3.9 Summary

In this chapter, the key design aspects of the multicast BG switch are studied. A new modular BG network architecture is proposed to accommodate the multicasting operation. Cell routing and replication function is distributed into each SE to fulfill the high-speed requirement. The similarity of functionality and structure of all the SEs facilitates a modular design and ease of VLSI implementation. A dynamic-length backpressure algorithm is devised to report the blocking information. Besides that, other attractive features such as priority switching, fault tolerance and scalability, are briefly studied.

Chapter 4

Performance Analysis under Uniform and Non-Uniform Multicast Traffic

4.1 Introduction

In this chapter, the performance of the multicast BG switch is studied under uniform and non-uniform traffic. Performance results are obtained under random and bursty conditions. Buffers are used at the IPCs and the OPCs of the switch. Performance measures are given in terms of cell delay, cell loss ratio, and input and output buffer requirements. The performance of the BG switch is compared to that of the ABACUS switch [60], which has been described in subsection 2.3.2.5, the PINIUM switch [9], which has been introduced in subsection 2.3.2.6, and an ideal purely output-queued multicast switch, which is capable of switching all incoming cells, regardless of their traffic characteristic, to their destination output buffers in one switching cycle.

The number of cells used in each simulation is a balance of the simulation time and required data accuracy. To achieve more reliable and trustworthy simulation results, it is desired that more cells should be switched. However, it is at the cost of simulation time. As a rule of thumb, if the target cell loss rate is 10^{-8}, switching 10^9 cells through the switch fabric should suffice. In this research, the effort has been

taken to maximize this number while keeping the simulation duration reasonable. Also, critical simulation cases are repeated multiple times for accuracy. Therefore, unless otherwise specified, the simulation results provided are based on simulations that have run through a period of switching one billion cells.

A confidence interval gives an estimated range of values calculated from a given set of sample data. In the dissertation, it is not considered. However, instead of repeating the same trial multiple times, the duration is extended for one trial. Because the magnitude level is the major concern of the research, the confidence range will have little impact on the trustworthiness of the simulation results. In fact, the rule of thumb, viz., ten times the reciprocal of the desired cell loss ratio, which is one billion cells for a cell loss of 10^{-8}, is used for all simulations.

The remaining of this chapter is organized as following: firstly, the multicast traffic model is introduced. Then, an analytical model for the switch performance under multicast random traffic is provided, followed by a comparison with simulation results obtained under the same traffic conditions. Subsequently, the performance of the switch under various bursty traffic loads using simulation results is investigated. Finally, this chapter is concluded by providing the performance results under various non-uniform traffic conditions.

4.2 Multicast Traffic Model

An important part of any performance analysis is an accurate traffic model. A multicast traffic model [78] is used to generate traffic for both simulation and analytical purposes. The traffic model can be described by three random processes: the destination selection process, the arrival process and the fanout process. The destination selection process describes whether cell destination will be selected uniformly or non-

uniformly. The arrival process specifies the correlation among the successive cells. The fanout process determines whether unicast cell or multicast cell will be generated. Based on the general choices for the three processes, eight different traffic types are obtained, as shown in Figure 4.1.

The destination selection process describes the distribution of the cell destinations. Based on the two general choices, multicast traffic can be categorized into uniform traffic and non-uniform traffic. For uniform traffic, each output port has the same probability to be selected. While for non-uniform traffic, some output ports will have higher probability to be requested over others.

4.2.1 Uniform Traffic

4.2.1.1 Arrival Process

In the case of uniform traffic, two types of arrival patterns are considered: random and bursty. For random traffic, the cell arrival is randomly selected based on the link load and is independent of cell arrival during the previous switching cycle. For bursty traffic, the ON-OFF model is used [9, 10, 26, 69, 78, 80]. The ON-OFF model is the least complex and the most widely used model to simulate bursty sources. It can describe most of the existing sources with a reasonable accuracy [81]. The source generates cells in a bursty manner: one active period (ON period) followed by an idle period (OFF period). During the ON period, the traffic source continues sending cells in every switching cycle to the same destination. The duration of ON and OFF periods are independently evaluated from two geometric distributions with the period length L in cells derived from

$$L = 1 + \left\lceil \frac{\ln(1-R)}{\ln(1-p)} - 1 \right\rceil, \tag{4.1}$$

where R, $0 \leq R < 1$, is the random number generated, and p, $0 < p < 1$, is the reciprocal of the average period length in cells. The cells arriving at each input line

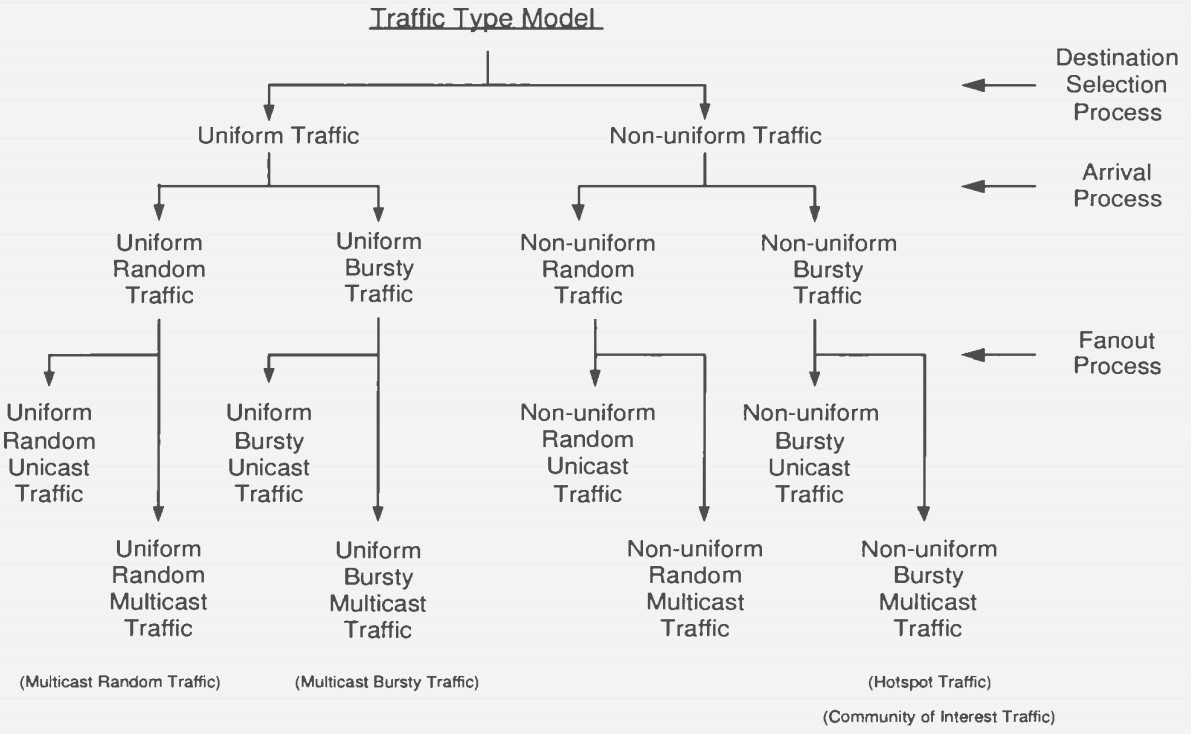


Figure 4.1: Traffic classification

in a burst have the same fanout number and are destined to the same output ports.

4.2.1.2 Fanout Distribution

The fanout process in Figure 4.1 describes the fanout of a multicast cell, i.e., the distribution of the number of copies of an incoming cell. A multicast cell with a fanout of one is a unicast cell. The traffic model provides a mix of unicast and multicast traffic with the level determined by the fanout distribution. The truncated geometric distribution is used to model the fanout distribution of the multicast cells [9, 80]. Given a switch size N, parameter q can be calculated numerically for any given mean fanout $\overline{F}$ following the equation which specifies their relation

$$\overline{F} = \sum_{i=1}^{N} i \times \frac{(1-q) \times q^{i-1}}{1-q^N} = \frac{1}{1-q} - \frac{N \times q^N}{1-q^N}. \tag{4.2}$$

With parameter q, the probability of having a fanout value f, denoted by $P_{tg}(f)$, can be calculated by using

$$P_{tg}(f) = \begin{cases} \frac{(1-q)\times q^{f-1}}{1-q^N} & \text{for } 1 \leq f \leq N \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

Under uniform destination selection, all output ports are equally likely to be requested, therefore, the input and output load of the switch can be represented by the load of each of the input and output links, denoted by ρ_{in} and ρ_{out}. Given an ideal strict-sense non-blocking switch fabric, for unicast traffic, ρ_{out} is given by

$$\rho_{out} = \rho_{in}. \tag{4.4}$$

However, for multicast traffic, when cell replication occurs inside the SF, the load at the input port is different from that at the output port. The offered load ρ_{out} can be associated with ρ_{in} via the mean traffic fanout $\overline{F}$ by using

$$\rho_{out} = \overline{F} \times \rho_{in}. \tag{4.5}$$

The intuitive approach for the performance analysis under multicast traffic is to use the same input load ρ_{in} to analyze different fanout conditions. From basic queuing theory, we know that a queue will become unstable when the data arrival rate is greater than the departure rate. For each output queue, the departure rate is assumed to be one cell per switching cycle. To avoid overflow, the offered load ρ_{out} should normally be kept below one. Even though the average load is kept below one, due to the statistical nature of the traffic, it is possible that the load momentarily exceeds one. To accommodate large fanout situations, for example a fanout of 8, the input load must be controlled below 12.5%. Using the approach mentioned above and with the same input load, the offered load at switch output would be just around 25% in the case of a mean fanout of 2, which is not realistic and far too low to be

considered. It is obvious that an output load of around 25%, which is the case for a mean fanout of 2, would produce much better results in any switch than an output load of around 100%, which is the case for a mean fanout of 8.

To have the performance analyzed under a reasonable traffic condition, the backward approach is used. The offered load to the switch is defined at the switch output. The offered load is converted to the input load via the mean fanout $\overline{F}$. As long as the load used does not cause the output queue to overflow, it is guaranteed that there is no overflow problem at the BG switch input. Unless otherwise stated, the traffic load reported in the dissertation refers to the offered output load.

4.2.2 Nonuniform Traffic

For nonuniform traffic, cell destination selection probability is not evenly distributed across all output ports. This will not affect the cell arrival process and the fanout process. In the dissertation, a modified form of the model presented in [14, 73] is used for nonuniform destination generation. The benefit of using this model is that, instead of providing only one or two hot destinations, it gives a substantial number of hot spots, which is more realistic when the switch size becomes very large. Details of the nonuniform traffic model are discussed in Section 4.6 before the performance of the multicast BG switch under the nonuniform multicast traffic is analyzed.

4.3 Multicast BG Switch Fabric Simulator

Before presenting the analytical modelling and evaluating the switch performance, we first briefly describe the BG switch fabric simulator developed for study.

Simulation techniques are used to obtain the performance of the switch fabric under various traffic conditions. A comprehensive simulation toolkit has been developed for the unicast BG switch by Dr. Venkatesan and his students during the past 10

years. In this research, another tool is developed for the new multicast BG switch. A multicast traffic generator is developed as well. Unless otherwise stated, simulation results are obtained through the simulation of switching one billion cells.

The multicast SF simulator is written in C/C++ and can be compiled and run under Unix, Linux, and Microsoft Windows environments. It follows a modular structure and object oriented design. The characteristic and operation of the switch are abstracted to five major component classes: input queue class (*InQueue*), packet class (*Packet*), source class (*Source*), switching element class (*SwitchElement*), and the output queue class (*OutQueue*). The switching operation is completed through the manipulation of those class objects. Program execution mimics the three-phase switching operation that occurs inside the SF. Pre-switching is used to load the switch with traffic, and it follows exactly the same switching operation except that no performance measurement is taken. Figure 4.2 shows the class information and the major modules of the simulator. Figure 4.3 provides a screen shot of the simulator user input interface.

The performance of the BG switch is measured using cell loss ratio, average cell delay, average input/output queueing delay, and average/maximum input/output buffer requirement. These metrics are calculated as follows:

$$Cell\ Loss\ Ratio = \frac{\text{Number of Lost Copies}}{\text{Number of Generated Copies}}, \tag{4.6}$$

$$Average\ Cell\ Delay = \frac{\sum \text{Delay of Each Arrived Copy}}{\text{Number of Arrived Copies}}, \tag{4.7}$$

$$Average\ Queue\ Occupancy = \frac{\text{Amount of Queue Space Used}}{\text{Switch Size} \cdot \text{Switching Cycles}}. \tag{4.8}$$

Note that for multicast traffic, an incoming master cell may contain several copies. Upon reaching the output queue, due to replication at one or more intermediate

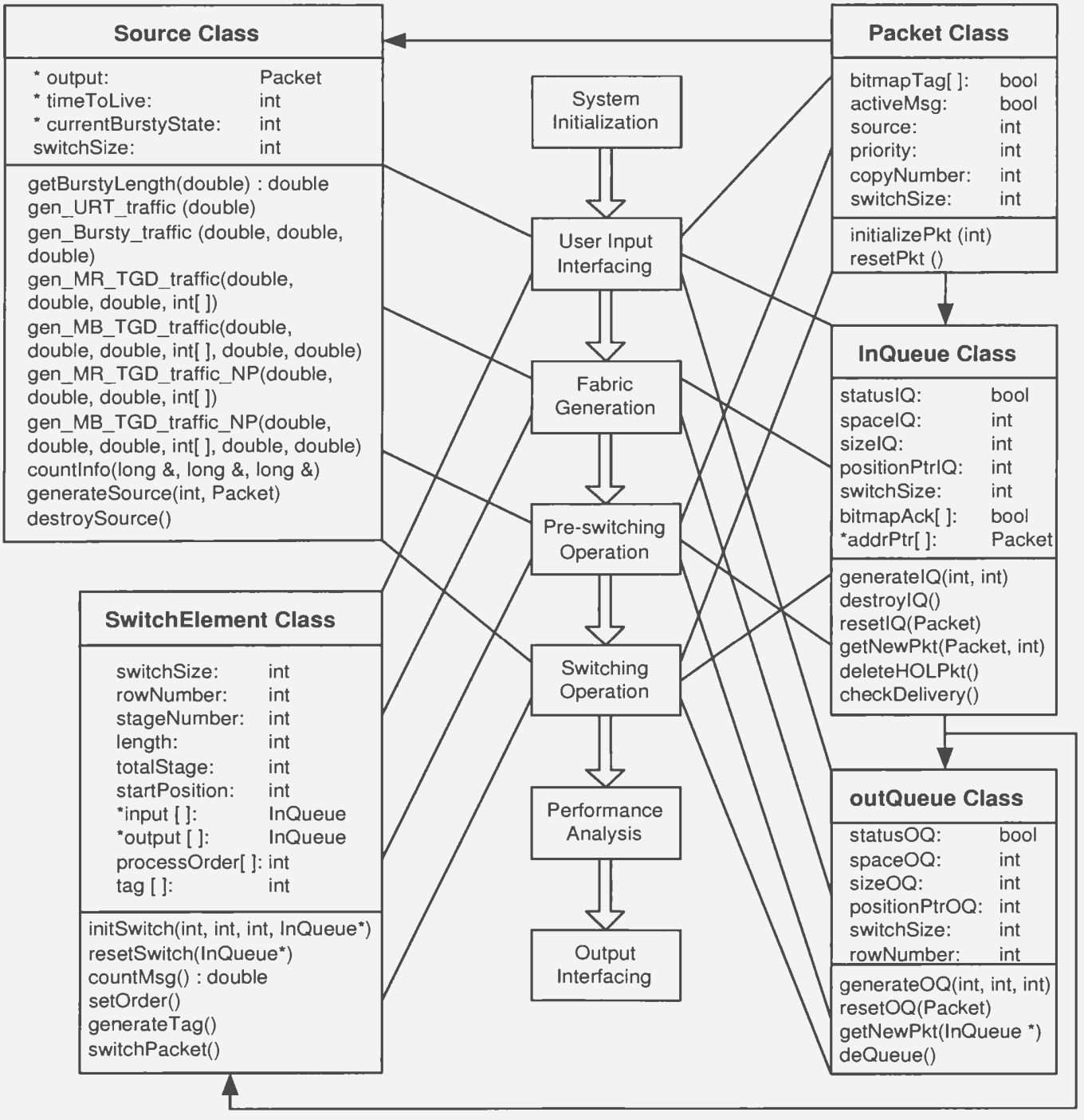


Figure 4.2: Class diagram and simulator flow chart

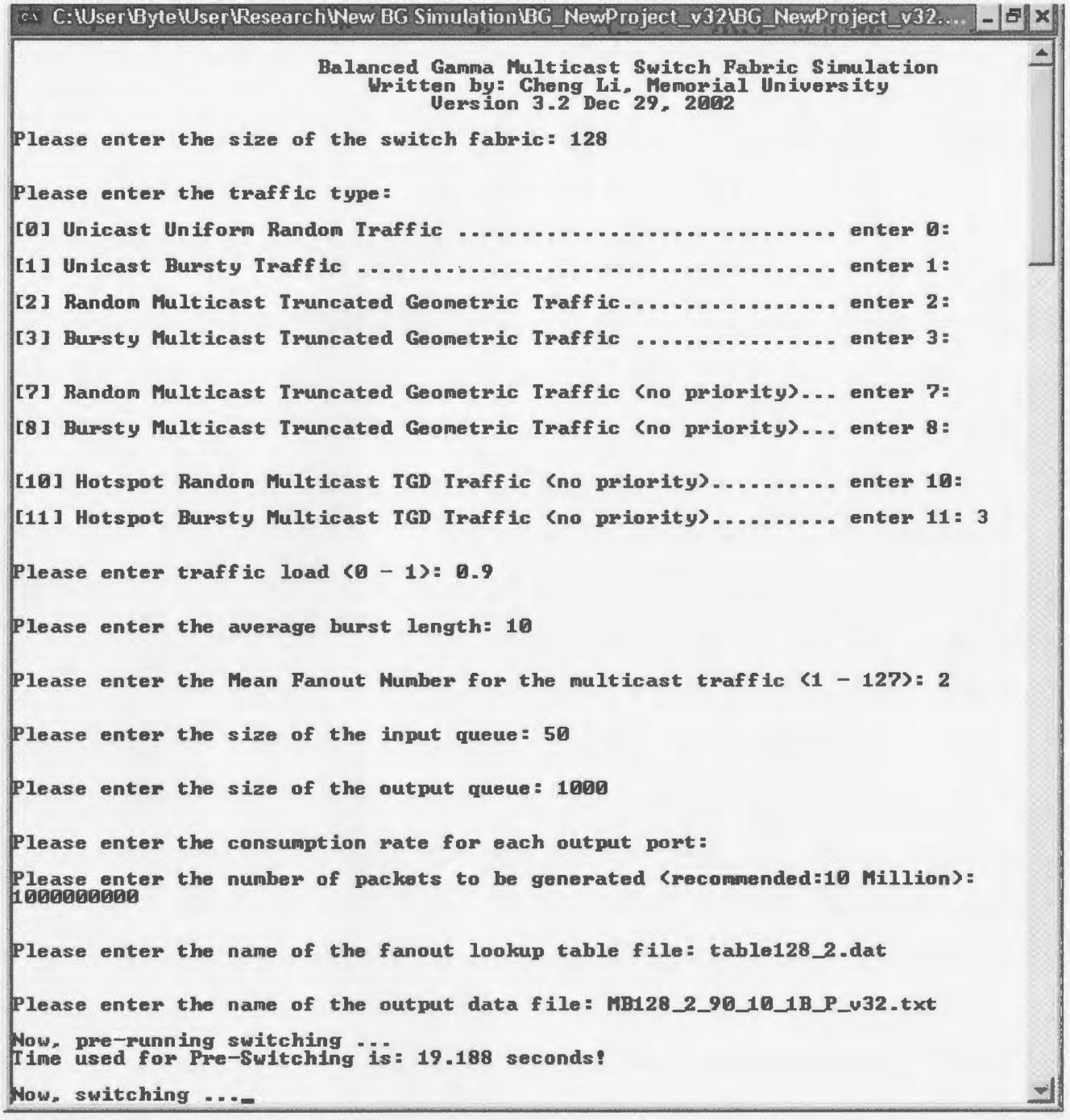


Figure 4.3: Multicast BG switch fabric simulator user input interface

stages, each copy becomes an independent cell. Therefore, the measures for the input queue are given on the master cell basis while the measures for the output queue and the overall switch performance are measured based on each copy.

4.4 Performance Analysis under Multicast Random Traffic

4.4.1 Analytical Modelling

Analytical modelling provides a good method to validate simulation results theoretically. Even though an exact analytical model is very hard to obtain for a complicated system like the BG switch, it is possible to perform an analysis under some loosened conditions and preserve enough accuracy. In the literature, many efforts have been put forth to model the internal working of switches under uniform and nonuniform, and unicast and multicast traffic [60, 9, 75, 76, 80, 82]. The analytical model that will be introduced is used to study the performance of the multicast BG network under uniform random multicast traffic conditions. Because unicast traffic can be seen as a special case of multicast traffic, our model can be used for the performance study of the switch under both unicast and multicast traffic conditions.

Generally speaking, the analysis follows the three-phase backpressure switching operation. Firstly, the cell blocking probabilities at SEs of different stages are analyzed. With this information, the cell blocking probability for the whole SF and the traffic arrival probability on the four links feeding each output queue are obtained. Secondly, the output queue is analyzed using a discrete-time Markov chain. The cell blocking probability, queue occupancy and queueing delay can be obtained through the output queue analysis. Then, the overall cell blocking probability for the combined SF and output queue is calculated, which is also the probability of cells being

kept in the HOL position of the input queue. Finally, input queueing analysis is performed to get the cell loss probability and other performance measures.

The following conditions are assumed for the switch and the traffic:

1. The input queue, SF, and output queue operate independently.
2. Incoming cells are multicast cells with a mean fanout of $\overline{F}$.
3. Cell arrival is identically and independently distributed (i.i.d) among all input links.
4. The arrivals of incoming cells follow a Bernoulli distribution with probability ρ on each input link.
5. Destinations of cells are uniformly distributed to all output ports.
6. All cells are of the same priority.
7. The operation of an SE of any stage is independent of any other SEs within that stage.
8. SEs of upstream stages and downstream stages are operated independently.
9. Cell arrival only occurs at the beginning of each switching cycle.
10. Cells are served on the first-come-first-served (FCFS) discipline in the input and output queues.

The first assumption breaks the switch into three independent components so that performance can be carried out for each of the components. The next five assumptions give a multicast random traffic with equal priority. The seventh and eighth assumptions provide that any SE located in a particular stage is indistinguishable from the

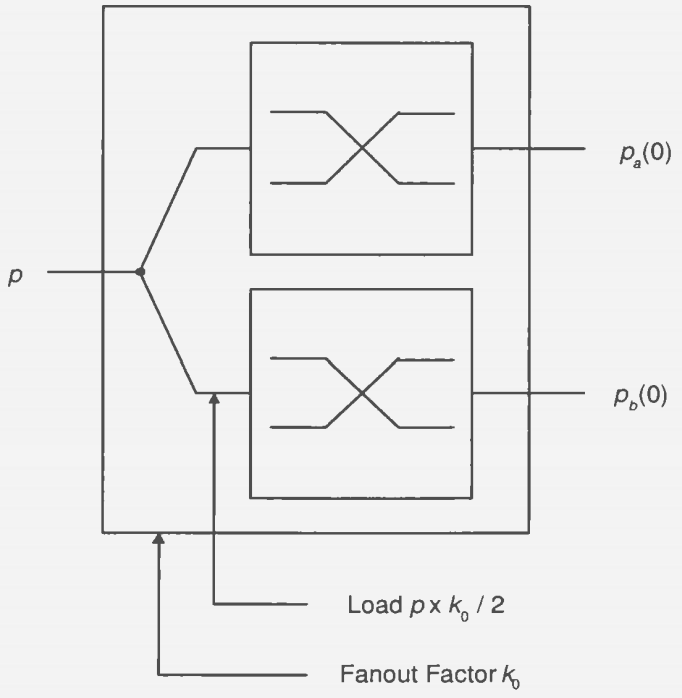


Figure 4.4: The 1×2 SE in the first stage (Stage 0).

other SEs belonging to the same stage. As a result, each stage can be characterized by any of its SEs. Finally, under the last two assumptions, the switch operates in slots, thus a discrete-time Markov chain can be used to depict the state of both the input and output queue.

4.4.1.1 Analysis of Stage 0 (1×2 SEs)

First, some notation are introduced for the analysis of this stage:

ρ = Offered traffic load at the switch input.

p = Pr{Input link of the 1×2 SE carries an active cell}.

$p_a(0)$ = Pr{Upper output link of the 1×2 SE carries an active cell}.

$p_b(0)$ = Pr{Lower output link of the 1×2 SE carries an active cell}.

k_0 = Fanout factor for multicast cells in the first stage (Stage 0).

P_{blk_0} =Pr{Cell being blocked in the first stage (Stage 0)} .

As shown in Figure 4.4, the inputs to the SF are connected directly to the input links of SEs in Stage 0. Hence, the load offered to the input link of each 1×2 SE is equal to the load offered to each input of the SF:

$$p = \rho. \tag{4.9}$$

The multicast effect is taken into consideration, as shown in Figure 4.4. It can be seen that bandwidth is expanded through this stage. Because of the random traffic assumption, both output links have the same probability of being requested by the incoming cell. Thus,

$$p_a(0) = p_b(0) = p \cdot k_0/2. \tag{4.10}$$

It is easy to prove that Stage 0 is a non-blocking stage [25]. Therefore,

$$P_{blk_0} = 0. \tag{4.11}$$

4.4.1.2 Analysis of Stage 1 (2×4 SEs)

Figure 4.5 shows the 2×4 SE used in Stage 1. Similar to the analysis of Stage 0, the following notation is defined for the analysis:

$p_a(0)$ = Pr{Upper input link of the 2×4 SE is active}.

$p_b(0)$ = Pr{Lower input link of the 2×4 SE is active}.

$p_a(1)$ = Pr{Upper regular output link of the 2×4 SE is active}.

$p_b(1)$ = Pr{Upper alternative output link of the 2×4 SE is active}.

$p_c(1)$ = Pr{Lower regular output link of the 2×4 SE is active}.

$p_d(1)$ = Pr{Lower alternative output link of the 2×4 SE is active}.

k_1 = Fanout factor for multicast cells in the second stage (Stage 1).

P_{blk_1} =Pr{Cell being blocked in the second stage (Stage 1)}.

Due to the random traffic assumption, all links between Stage 0 and Stage 1 have the same probability to be active. Therefore, instead of distinguishing the load on

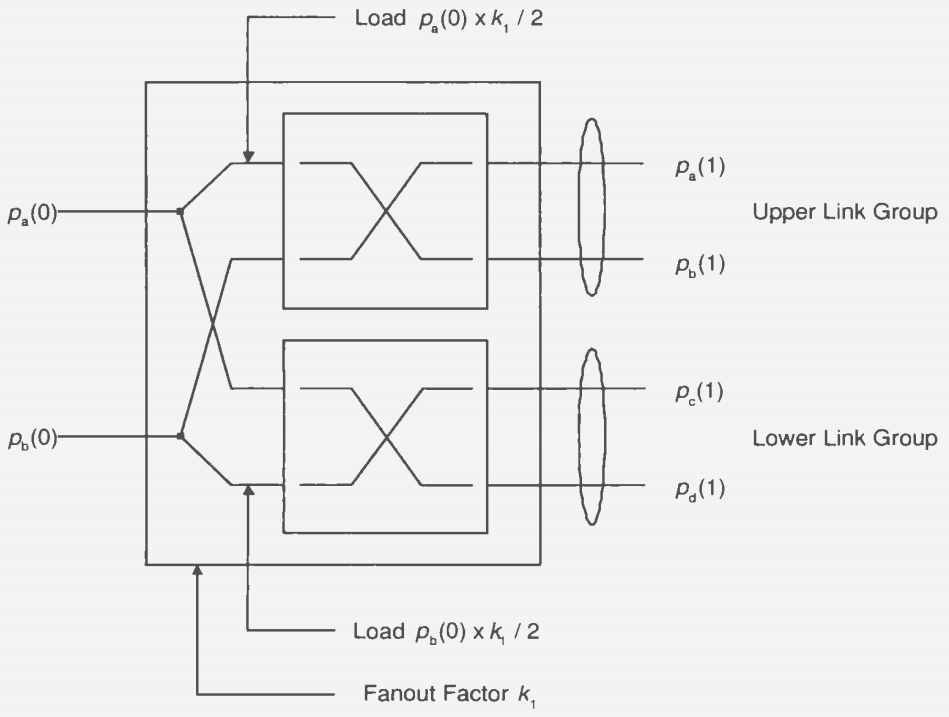


Figure 4.5: The 2×4 SE in the second stage (Stage 1).

the two input links to each SE in Stage 1 as $p_a(0)$ and $p_b(0)$, p_0 is used. The input link load of the 2×4 SE in Stage 1 is equal to the output link load of the 1×2 SE in Stage 0

$$p_0 = p_a(0) = p_b(0). \tag{4.12}$$

As described in Section 3.4.2, for all 2×4 and 4×4 SEs, because each regular link and its alternative link have the same capability of delivering cells to their destination, they are represented using the name "link group". Therefore, the four output links are divided into two link groups, the upper link group and the lower link group. The regular link in a link group is always used first when there is a cell requesting that link group. The alternative link is used only when the regular link is occupied and there is another cell request. Because any incoming active cell will not request both links within the same link group and the two link groups are equally likely to be

selected due to the random traffic condition, for the 2×4 SE, the regular link will be active when one or both of the two input links are active and alternative link will be active only when both of the input links are active. Therefore, the probability that two regular output links are active is given by

$$\begin{aligned} p_a(1) &= p_c(1) \\ &= \sum_{i=1}^{2} \binom{2}{i} \cdot \left(\frac{p_0 \cdot k_1}{2}\right)^i \cdot \left(1 - \frac{p_0 \cdot k_1}{2}\right)^{2-i} \\ &= p_0 \cdot k_1 \cdot \left(1 - \frac{p_0 \cdot k_1}{2}\right) + \left(\frac{p_0 \cdot k_1}{2}\right)^2. \end{aligned} \tag{4.13}$$

As well, the two alternative links $p_b(1)$ and $p_d(1)$ have the same probability to be active and the probability is:

$$\begin{aligned} p_b(1) &= p_d(1) \\ &= \binom{2}{2} \cdot \left(\frac{p_0 \cdot k_1}{2}\right)^2 \cdot \left(1 - \frac{p_0 \cdot k_1}{2}\right)^0 \\ &= \left(\frac{p_0 \cdot k_1}{2}\right)^2. \end{aligned} \tag{4.14}$$

Again, it is proved that Stage 1 is a non-blocking stage [25]. Therefore,

$$P_{blk_1} = 0. \tag{4.15}$$

4.4.1.3 Analysis of Stage 2 to Stage n - 1 (4×4 SEs)

For all remaining stages, 4×4 SEs are used, as shown in Figure 4.6. The following notation is used for the analysis when $2 \leq i \leq (n-1)$:

p_{ir} = Pr{Regular input to the 4×4 SE at Stage i is active}.

p_{ia} = Pr{Alternative input to the 4×4 SE at Stage i is active}.

$p_a(i)$ = Pr{Upper regular output link of the 4×4 SE is active}.

$p_b(i)$ = Pr{Upper alternative output link of the 4×4 SE is active}.

$p_c(i)$ = Pr{Lower regular output link of the 4×4 SE is active}.

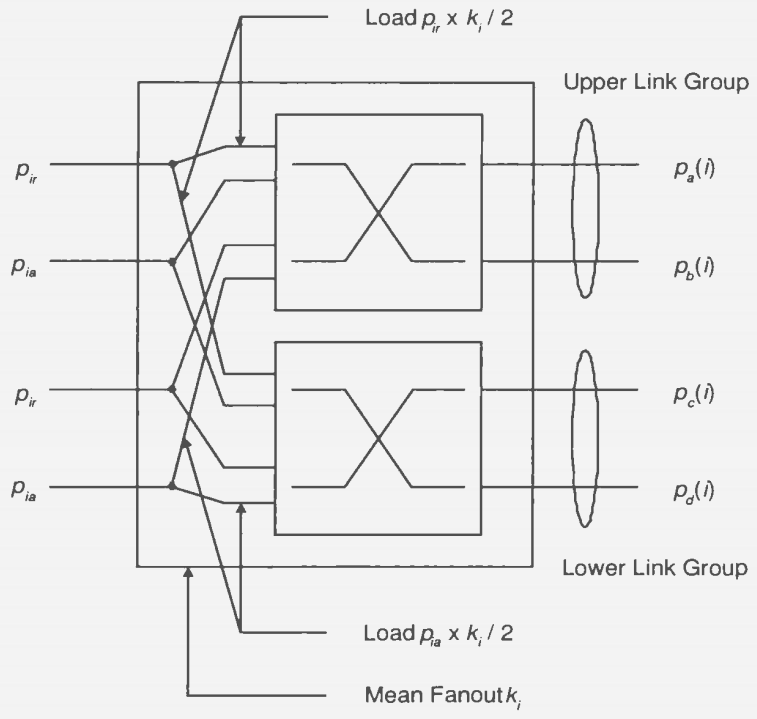


Figure 4.6: The 4×4 SE for all remaining stages (Stage 2 to Stage $n-1$)

$p_d(i) = \Pr\{\text{Lower alternative output link of the } 4 \times 4 \text{ SE is active}\}.$

$k_i = $ Fanout factor for multicast cells in Stage i.

$P_{blk_i} = \Pr\{\text{Cell being blocked in Stage } i\}.$

$a_j = \Pr\{j \text{ incoming cells request the output link group}\}.$

From the interconnection pattern between stages, it is not difficult to find that among the four input links of each 4×4 SE, two of them are from the regular output links of previous stage and the other two are from the alternative output links. The random traffic assumption ensures that the load for both regular links of the same stage is the same and that the load for all alternative links of the same stage is also the same, i.e.,

$$p_{ir} = p_a(i-1) = p_c(i-1), \tag{4.16}$$

$$p_{ia} = p_b(i-1) = p_d(i-1). \tag{4.17}$$

Similar to the analysis for the 2×4 SEs, the four output links can be divided into the upper link group and the lower link group. Under random traffic, the probability of the upper link group being requested by a cell is equal to that of the lower link group. Therefore, either of those two link groups can be chosen as the targeted link group for analysis. The probability that there are i cells requesting the targeted link group can be calculated as:

1. Probability of no cells requesting the targeted link group.

$$\begin{aligned} a_0 &= (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \\ &= (1 - \frac{p_{ia} \cdot k_i}{2})^2 \cdot (1 - \frac{p_{ir} \cdot k_i}{2})^2 \end{aligned} \tag{4.18}$$

2. Probability of one cell requesting the targeted link group.

$$\begin{aligned} a_1 &= (\frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \\ &+ (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (\frac{p_{ir} \cdot k_i}{2}) \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \\ &+ (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \cdot (\frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \\ &+ (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (\frac{p_{ir} \cdot k_i}{2}) \\ &= p_{ia} \cdot k_i \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2})^2 \\ &+ p_{ir} \cdot k_i \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \cdot (1 - \frac{p_{ia} \cdot k_i}{2})^2 \end{aligned} \tag{4.19}$$

3. Probability of two cells requesting the targeted link group.

$$\begin{aligned} a_2 &= (\frac{p_{ia} \cdot k_i}{2}) \cdot (\frac{p_{ir} \cdot k_i}{2}) \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \\ &+ (\frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \cdot (\frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \\ &+ (\frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (\frac{p_{ir} \cdot k_i}{2}) \\ &+ (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (\frac{p_{ir} \cdot k_i}{2}) \cdot (\frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \\ &+ (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (\frac{p_{ir} \cdot k_i}{2}) \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (\frac{p_{ir} \cdot k_i}{2}) \end{aligned}$$

$$
\begin{aligned}
&+ \quad (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \cdot (\frac{p_{ia} \cdot k_i}{2}) \cdot (\frac{p_{ir} \cdot k_i}{2}) \\
&= \quad p_{ia} \cdot p_{ir} \cdot k_i^2 \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \\
&+ \quad \frac{1}{4} \cdot p_{ia}^2 \cdot k_i^2 \cdot (1 - \frac{p_{ir} \cdot k_i}{2})^2 + \frac{1}{4} \cdot p_{ir}^2 \cdot k_i^2 \cdot (1 - \frac{p_{ia} \cdot k_i}{2})^2 \qquad (4.20)
\end{aligned}
$$

4. Probability of three cells requesting the targeted link group.

$$
\begin{aligned}
a_3 &= \frac{p_{ia} \cdot k_i}{2} \cdot \frac{p_{ir} \cdot k_i}{2} \cdot \frac{p_{ia} \cdot k_i}{2} \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \\
&+ \frac{p_{ia} \cdot k_i}{2} \cdot \frac{p_{ir} \cdot k_i}{2} \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot \frac{p_{ir} \cdot k_i}{2} \\
&+ \frac{p_{ia} \cdot k_i}{2} \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \cdot \frac{p_{ia} \cdot k_i}{2} \cdot \frac{p_{ir} \cdot k_i}{2} \\
&+ (1 - \frac{p_{ia} \cdot k_i}{2}) \cdot \frac{p_{ir} \cdot k_i}{2} \cdot \frac{p_{ia} \cdot k_i}{2} \cdot \frac{p_{ir} \cdot k_i}{2} \\
&= \frac{1}{4} \cdot p_{ia}^2 \cdot p_{ir} \cdot k_i^3 \cdot (1 - \frac{p_{ir} \cdot k_i}{2}) \\
&+ \frac{1}{4} \cdot p_{ia} \cdot p_{ir}^2 \cdot k_i^3 \cdot (1 - \frac{p_{ia} \cdot k_i}{2}) \qquad (4.21)
\end{aligned}
$$

5. Probability of four cells requesting the targeted link group.

$$
\begin{aligned}
a_4 &= \frac{p_{ia} \cdot k_i}{2} \cdot \frac{p_{ir} \cdot k_i}{2} \cdot \frac{p_{ia} \cdot k_i}{2} \cdot \frac{p_{ir} \cdot k_i}{2} \\
&= \frac{1}{16} \cdot p_{ia}^2 \cdot p_{ir}^2 \cdot k_i^4 \qquad (4.22)
\end{aligned}
$$

6. Probability of more than four cells requesting the targeted link group.

$$
a_j = 0 \qquad for \quad j \geq 5 \qquad (4.23)
$$

Therefore, the probability that any regular output link (upper/lower) carries an active cell is given by

$$
\begin{aligned}
p_a(i) &= p_c(i) \\
&= \Pr(\text{One or more cells request the targeted output group}) \\
&= \sum_{j=1}^{4} a_j. \qquad (4.24)
\end{aligned}
$$

The probability for the alternative output link is given by:

$$\begin{aligned} p_b(i) &= p_d(i) \\ &= \Pr(\text{More than one cell requests the targeted output group}) \\ &= \sum_{j=2}^{4} a_j. \end{aligned} \tag{4.25}$$

Blocking occurs inside the 4×4 SEs when there are more than two incoming cells requesting the same link group. The cell blocking probability for stage i, where $2 \leq i \leq n-1$, is given as:

$$\begin{aligned} P_{blk_i} &= \frac{\mathrm{E}\{\text{Number of blocked copies}\}}{\mathrm{E}\{\text{Number of copies}\}} \\ &= \frac{\sum_{j=3}^{4} (j-2) \cdot a_j}{\sum_{j=1}^{4} j \cdot a_j}. \end{aligned} \tag{4.26}$$

Therefore, for any unicast cell that is appearing at the IPC, the cell blocking probability inside the SF, $P_{blk_{SF}}$, is given as

$$P_{blk_{SF}} = 1 - \prod_{i=0}^{n-1} (1 - P_{blk_i}). \tag{4.27}$$

While for a multicast cell at the IPC, the probability that the master cell being blocked inside the SF is calculated by considering the number of copies available before each stage and the blocking probability for the stage. Let N_i denote the number of copies of the multicast cell from an IPC sent to SEs in Stage i, as shown in Figure 4.7. The blocking probability, that is, the probability that one or more cell copy is blocked inside the SF for a multicast cell is calculated using

$$\begin{aligned} P_{blk_{SF}} &= 1 - (1 - P_{blk_0})^{N_0} \cdot (1 - P_{blk_1})^{N_1} \cdot (1 - P_{blk_2})^{N_2} \ \ldots \ (1 - P_{blk_{n-1}})^{N_{n-1}} \\ &= 1 - \prod_{i=0}^{n-1} (1 - P_{blk_i})^{N_i}. \end{aligned} \tag{4.28}$$

It is not possible to obtain a general expression for N_i as cells arrive at the IPC with different fanout values and cell replication and blocking may occur anywhere inside the SF. However, by using the two extreme cases, i.e., the unicast case and the broadcast case, we form the two bounds correspond to the best and worst performance that the switch fabric can achieve. In the unicast case, N_i is fixed to one while in the broadcast case, N_i is equal to 2^i, where i is the stage number. Besides the two bounds, an analytical approximation is calculated for random traffic by using

$$N_i = \begin{cases} 1 & i = 0 \\ \lceil N_{i-1} \cdot k \rceil & i \geq 1, \end{cases} \tag{4.29}$$

where variable k is the fanout factor for each stage and k equals to $\sqrt[n]{F}$. Because roundup is used to obtain an integer value for the average copy number between stages, the actual traffic would be lighter than this and so the performance result should be better than what is predicted by the approximation.

4.4.1.4 Finite Output Queueing Analysis

Among the four links coming into each output buffer, two links are from the regular output of the SF while the other two are from the alternative output links. Figure 4.8 shows the traffic toward the OPC under investigation. From the previous analysis, all regular links or alternative links that connect the SF and the OPC have the same probability of carrying an active cell. Let $p_{(n-1)r}$ and $p_{(n-1)a}$ denote the loads on the regular and alternative output link from SEs in the last stage of the SF, respectively. The link probabilities p_a and p_c are from the regular output links of the SF and link probabilities p_b and p_d are from the alternative output links and they are given by

$$p_a = p_c = p_{(n-1)r}, \tag{4.30}$$

$$p_b = p_d = p_{(n-1)a}. \tag{4.31}$$

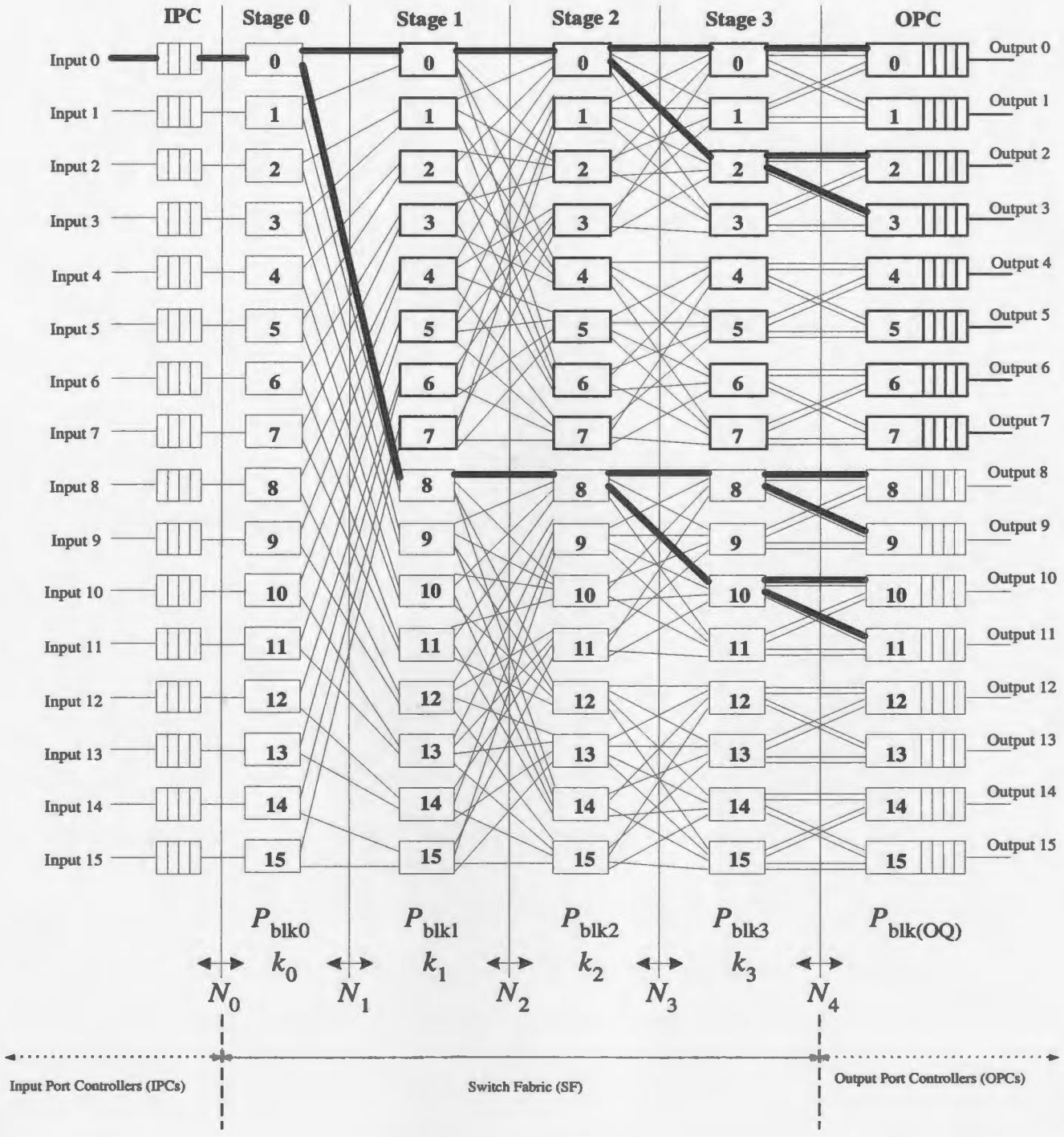


Figure 4.7: Multicast cell blocking analysis for an 16×16 BG multicast switch.

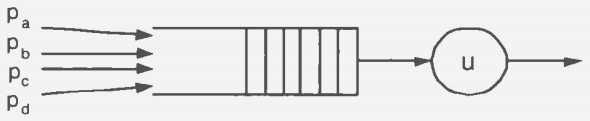


Figure 4.8: The output queue at the targeted output port controller

Similar to the approach that has been used in the analysis of the 4×4 SE, the probability of having i requests to the output queue, where $0 \leq i \leq 4$, is given as the following.

1. The probability of zero cell arrivals to the output queue:

$$\begin{aligned} a_0 &= (1-p_a)\cdot(1-p_b)\cdot(1-p_c)\cdot(1-p_d) \\ &= (1-p_{(n-1)a})^2\cdot(1-p_{(n-1)r})^2 \end{aligned} \tag{4.32}$$

2. The probability of one cell arrival to the output queue:

$$\begin{aligned} a_1 &= p_a\cdot(1-p_b)\cdot(1-p_c)\cdot(1-p_d) + (1-p_a)\cdot p_b\cdot(1-p_c)\cdot(1-p_d) \\ &+ (1-p_a)\cdot(1-p_b)\cdot p_c\cdot(1-p_d) + (1-p_a)\cdot(1-p_b)\cdot(1-p_c)\cdot p_d \\ &= 2p_{(n-1)a}\cdot(1-p_{(n-1)r})^2\cdot(1-p_{(n-1)a}) \\ &+ 2p_{(n-1)r}\cdot(1-p_{(n-1)r})\cdot(1-p_{(n-1)a})^2 \end{aligned} \tag{4.33}$$

3. The probability of two cell arrivals to the output queue:

$$\begin{aligned} a_2 &= p_a\cdot p_b\cdot(1-p_c)\cdot(1-p_d) + p_a\cdot(1-p_b)\cdot p_c\cdot(1-p_d) \\ &+ p_a\cdot(1-p_b)\cdot(1-p_c)\cdot p_d + (1-p_a)\cdot p_b\cdot p_c\cdot(1-p_d) \\ &+ (1-p_a)\cdot p_b\cdot(1-p_c)\cdot p_d + (1-p_a)\cdot(1-p_b)\cdot p_c\cdot p_d \\ &= 4p_{(n-1)a}\cdot p_{(n-1)r}\cdot(1-p_{(n-1)a})\cdot(1-p_{(n-1)r}) \\ &+ p_{(n-1)a}^2\cdot(1-p_{(n-1)r})^2 + (1-p_{(n-1)a})^2\cdot p_{(n-1)r}^2 \end{aligned} \tag{4.34}$$

4. The probability of three cell arrivals to the output queue:

$$a_3 = p_a\cdot p_b\cdot p_c\cdot(1-p_d) + p_a\cdot p_b\cdot(1-p_c)\cdot p_d$$

$$
\begin{aligned}
&+ \quad p_a \cdot (1 - p_b) \cdot p_c \cdot p_d + (1 - p_a) \cdot p_b \cdot p_c \cdot p_d \\
&= \quad 2p^2_{(n-1)a} \cdot p_{(n-1)r} \cdot (1 - p_{(n-1)r}) \\
&+ \quad 2p^2_{(n-1)r} \cdot p_{(n-1)a} \cdot (1 - p_{(n-1)a}) \qquad (4.35)
\end{aligned}
$$

5. The probability of four cell arrivals to the output queue:

$$
\begin{aligned}
a_4 &= p_a \cdot p_b \cdot p_c \cdot p_d \\
&= p^2_{(n-1)a} \cdot p^2_{(n-1)r} \qquad (4.36)
\end{aligned}
$$

6. The probability of more than four cell arrivals to the output queue is zero:

$$a_j = 0, \qquad \text{where } j \geq 5 \qquad (4.37)$$

The sum of the probabilities for all possible cell arrivals is given by:

$$\sum_{i=0}^{4} a_i = 1 \qquad (4.38)$$

The output queue analysis follows the approach described in Hluchyj and Karol's paper [76]. Let Q_m denote the number of cells in the output queue at the end of the mth switching cycle, A_m denote the number of cell arrivals during the mth switching cycle, and B_o denote the output queue size, then the control function of the output queue is [76, 83]:

$$Q_m = min\{max\{0, Q_{m-1} + A_m - 1\}, B_o\}. \qquad (4.39)$$

When $Q_{m-1} = 0$ and $A_m > 0$, one of the arriving cells is immediately transmitted to the switch's output link without experiencing any delay.

Similar to the infinite queue analysis described in [76], for finite queue size B_o, Q_m is modelled by a finite-state, discrete-time Markov chain, which is shown in Figure

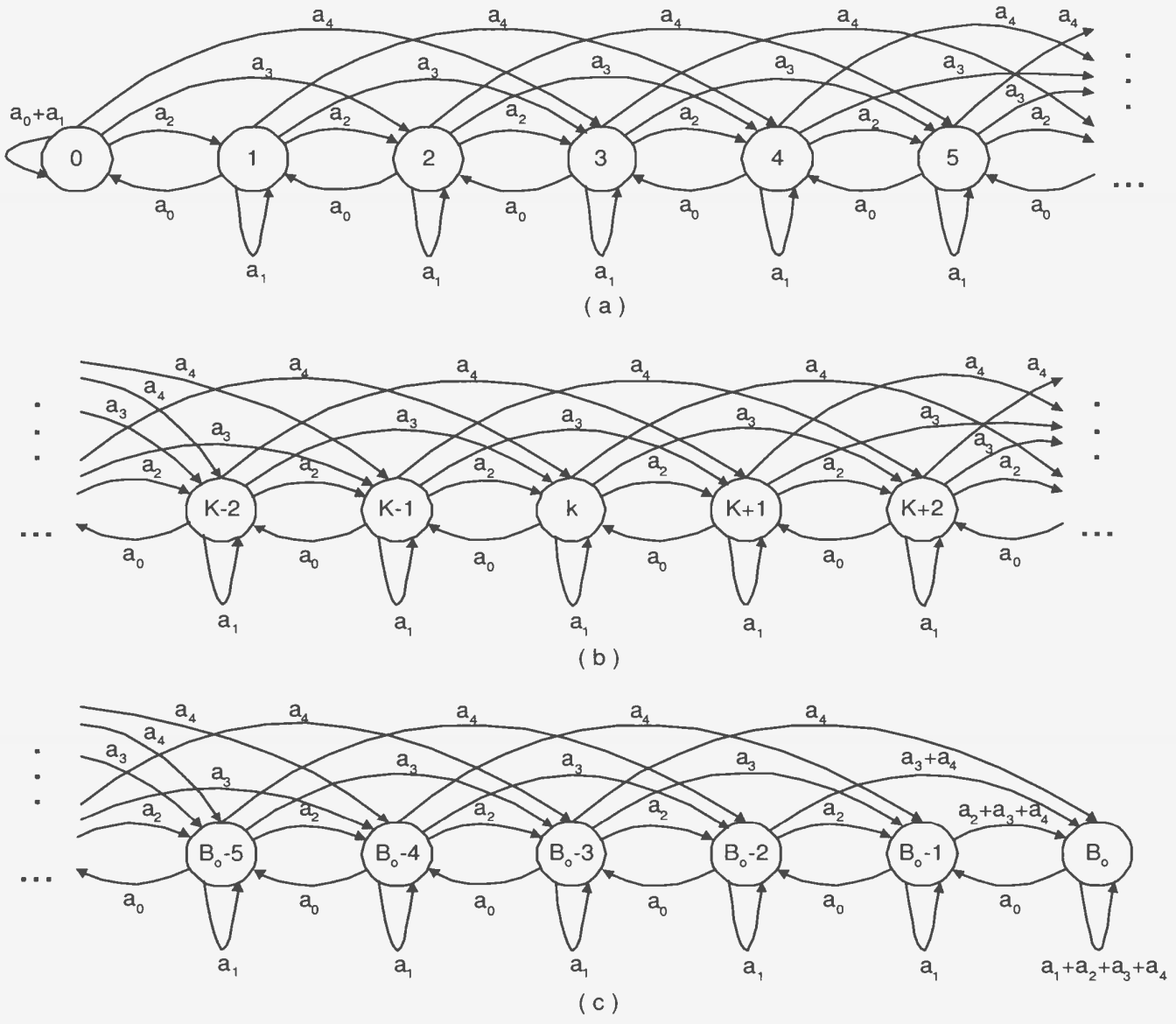


Figure 4.9: The output queue state transition diagram.

4.9. The state transition probabilities $p_{ij} \equiv Prob[Q_m = j | Q_{m-1} = i]$ are modified and given as:

$$p_{ij} = \begin{cases} a_0 + a_1 & i = 0,\ j = 0 \\ a_0 & 1 \le i \le B_o,\ j = i - 1 \\ a_{j-i+1} & 1 \le j \le B_o - 1,\ 0 \le i \le j. \\ \sum_{k=j-i+1}^{4} a_k & j = B_o,\ 0 \le i \le j \\ 0 & \text{otherwise} \end{cases} \tag{4.40}$$

The queue size can be obtained from the Markov chain balance equation. Let π_i denote the probability of the output queue being in state i, where $0 \le i \le B_o$, i.e.,

$$\pi_i \equiv Prob[Q_m = i]. \tag{4.41}$$

By using the balance equations, that is, the total unconditional rate of leaving any state equals the total unconditional rate of entering the state [15], the recursive relation between the queue state probabilities is obtained:

$$\pi_1 = \frac{1 - a_0 - a_1}{a_0} \cdot \pi_0 \tag{4.42}$$

$$\pi_2 = \frac{1 - a_1}{a_0} \cdot \pi_1 - \frac{a_2}{a_0} \cdot \pi_0 \tag{4.43}$$

$$\pi_3 = \frac{1 - a_1}{a_0} \cdot \pi_2 - \frac{a_2}{a_0} \cdot \pi_1 - \frac{a_3}{a_0} \cdot \pi_0 \tag{4.44}$$

... ...

$$\pi_i = \frac{1 - a_1}{a_0} \cdot \pi_{i-1} - \sum_{k=2}^{4} \frac{a_k}{a_0} \cdot \pi_{i-k}, \quad \text{for } 4 \le i \le B_o \tag{4.45}$$

$$\text{where } \pi_0 = \frac{1}{1 + \sum_{i=1}^{B_o} \pi_i/\pi_0}. \tag{4.46}$$

It is very difficult to get an analytical expression for the output queue state probability for an arbitrary queue size B_o. Therefore, a numerical analysis approach is used. For any given output queue size B_o, short programs using Maple [84] are developed to solve the linear equations, and to obtain the state probabilities of the output queue. With all state probabilities known, the analysis of the output queue becomes straightforward.

The average number of cells in the output queue $\overline{n}_{OQ}$ can be calculated through the sum of the product of all states of the output queue and the corresponding probability. That is,

$$\overline{n}_{OQ} = \sum_{i=0}^{B_o} i \cdot \pi_i. \tag{4.47}$$

The probability of cells being blocked at the output queue is equal to the overflow probability of the output queue, which is:

$$\begin{aligned} P_{blk_{OQ}} &= \pi_{B_o} \cdot \frac{\sum_{i=1}^{4} a_i}{\sum_{i=1}^{4} a_i} + \pi_{B_o-1} \cdot \frac{\sum_{i=2}^{4} a_i}{\sum_{i=1}^{4} a_i} + \pi_{B_o-2} \cdot \frac{\sum_{i=3}^{4} a_i}{\sum_{i=1}^{4} a_i} + \pi_{B_o-3} \cdot \frac{a_4}{\sum_{i=1}^{4} a_i} \\ &= \frac{\sum_{i=0}^{3} (\pi_{B_o-i} \cdot \sum_{j=i+1}^{4} a_j)}{\sum_{i=1}^{4} a_i}. \end{aligned} \tag{4.48}$$

By applying the well-known Little's formula, the average cell waiting time in the output queue, $\overline{T}_{OQ}$, is given by:

$$\overline{T}_{OQ} = \frac{\overline{n}_{OQ}}{\lambda_{OQ} \cdot (1 - P_{blk_{OQ}})}, \tag{4.49}$$

where the λ_{OQ} is the sum of the traffic on all the four incoming links to an output queue, which is in the unit of cells per switching cycle.

4.4.1.5 Finite Input Queueing Analysis

By considering the blocking effect inside the SF and at the output queue together, the probability of the HOL cell in the queue being retained at the head position is obtained. For a unicast cell, the blocking probability is given by

$$\begin{aligned} P_{blk_{SF\&OQ}} &= 1-(1-P_{blk_{SF}})\cdot(1-P_{blk_{OQ}}) \\ &= 1-(1-P_{blk_{OQ}})\cdot\prod_{i=0}^{n-1}(1-P_{blk_i}). \end{aligned} \tag{4.50}$$

For a multicast cell, the two bounds and the approximate result for the probability of the multicast cell being blocked at SF and OPC are considered. The lower bound is corresponding to the unicast case, which is exactly the same as in Equation 4.50. The upper bound corresponds to the broadcast case and it is given by

$$\begin{aligned} P_{blk_{SF\&OQ}} &= 1-(1-P_{blk_{SF}})\cdot(1-P_{blk_{OQ}})^N \\ &= 1-(1-P_{blk_{OQ}})^N\cdot\prod_{i=0}^{n-1}(1-P_{blk_i})^{2^i}. \end{aligned} \tag{4.51}$$

The approximate result is given by

$$\begin{aligned} P_{blk_{SF\&OQ}} = 1 \;-\; & (1-P_{blk_{OQ}})^{\bar{F}}\cdot(1-P_{blk_0})\cdot(1-P_{blk_1})^{\lceil k\rceil}\cdot \\ & (1-P_{blk_2})^{\lceil\lceil k\rceil\cdot k\rceil}\cdots(1-P_{blk_{n-1}})^{\lceil\lceil\lceil\lceil k\rceil\cdot k\rceil\cdot k\rceil\cdots k\rceil}, \end{aligned} \tag{4.52}$$

where $\overline{F}$ represents the mean fanout for the multicast traffic and k represents the fanout factor at different stages.

Let $P_{success}$ denote the probability that all copies of the master cell are successfully switched through the SF and queued in the output queue. $P_{success}$ is given by

$$P_{success} = 1 - P_{blk_{SF\&OQ}}. \tag{4.53}$$

This is also the probability that the HOL cell will be deleted and the following cell in the input queue will become the HOL cell.

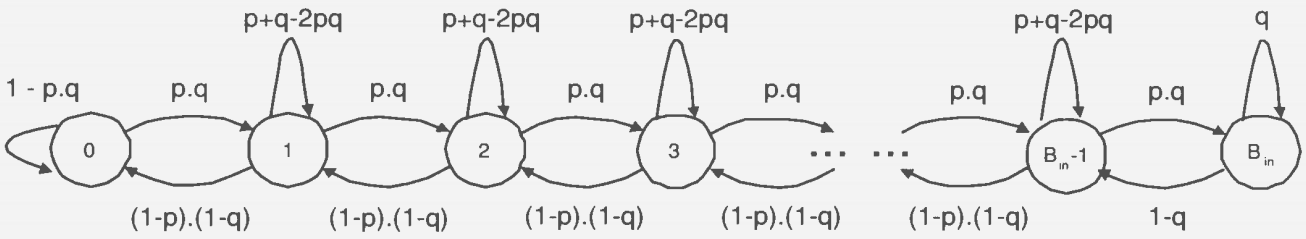


Figure 4.10: The input queue state transition diagram.

Let Q_m denote the number of cells in the input queue at the end of the mth switching cycle, A_m denote the number of cell arrivals during the mth switching cycle, D_m denote the number of cell departures during the mth switching cycle, and B_{in} denote the input queue size. Both A_m and D_m must be 0 or 1. The control function of the input queue is given as [76]:

$$Q_m = min\{max(0, Q_{m-1} + A_m - D_m), B_{in}\}. \tag{4.54}$$

Cell arrivals to all inputs are based on an independently and identically distributed (i.i.d) Bernoulli distribution with a load ρ, which gives the cell arrival probability in a switching cycle. The load to the input queue under study is defined as p, where

$$p = \rho. \tag{4.55}$$

Based on arrival load p and the resulting cell blocking probability $q = P_{blk_{SF\&OQ}}$ from the previous analysis, the input queue can also be modelled using the finite state discrete-time Markov chain, as shown in Figure 4.10. Similar to [76], the probability of different queue states π_i, which is defined by Equation 4.41, can be obtained from the Markov chain balance equations:

$$\pi_1 = \frac{p \cdot q}{(1-p) \cdot (1-q)} \cdot \pi_0 \tag{4.56}$$

$$\pi_2 = \frac{p \cdot q}{(1-p) \cdot (1-q)} \cdot \pi_1 \tag{4.57}$$

... ...

$$\pi_i = \frac{p \cdot q}{(1-p) \cdot (1-q)} \cdot \pi_{i-1} \qquad for\ 2 \leq i \leq B_{in} - 1 \tag{4.58}$$

$$\pi_{B_{in}} = \frac{p \cdot q}{1-q} \cdot \pi_{B_{in}-1}, \tag{4.59}$$

and the sum of the probabilities for all states is

$$\sum_{i=0}^{B_{in}} \pi_i = 1. \tag{4.60}$$

We define t as:

$$t = \frac{p \cdot q}{(1-p) \cdot (1-q)}. \tag{4.61}$$

Substituting all variables in Equation 4.60 with those defined in Equation 4.56 to 4.59, and by solving the recursive equations, the probability of the input queue having zero cells, π_0, is given by

$$\begin{aligned} \pi_0 &= \frac{1}{t^0 + t^1 + t^2 + ... + t^{B_{in}} - p \cdot t^{B_{in}}} \\ &= \frac{1}{\sum_{i=0}^{B_{in}} t^i - p \cdot t^{B_{in}}} \\ &= \frac{1}{\frac{1-t^{B_{in}+1}}{1-t} - p \cdot t^{B_{in}}} \\ &= \frac{1-t}{1 - p \cdot t^{B_{in}} - (1+p) \cdot t^{B_{in}+1}}, \end{aligned} \tag{4.62}$$

and the other input queue state probabilities as:

$$\pi_i = \begin{cases} t^i \cdot \pi_0 & for\ 1 \leq i \leq B_{in} - 1 \\ (1-p) \cdot t^{B_{in}} \cdot \pi_0 & for\ i = B_{in}. \end{cases} \tag{4.63}$$

Because the blocked cells are backpressured and kept in the HOL position of the input queue, there will be no cell loss either inside the SF or at the output queue.

Therefore, the overall cell loss probability is given by the overflow probability of the input queue, which is:

$$\begin{aligned} P_{loss} &= \pi_{B_{in}} \\ &= (1-p) \cdot t^{B_{in}} \cdot \pi_0. \end{aligned} \tag{4.64}$$

With the state probability π_i, where $0 \leq i \leq B_{in}$, the average number of cells in the input queue can be calculated by:

$$\overline{n}_{in} = \sum_{i=0}^{B_{in}} i \cdot \pi_i. \tag{4.65}$$

Dividing by the actual incoming traffic load $p \cdot (1 - P_{loss})$, the average cell delay in the input queue, $\overline{T}_{in}$, can be calculated by using Little's formula:

$$\overline{T}_{in} = \frac{\overline{n}_{in}}{p \cdot (1 - P_{loss})}. \tag{4.66}$$

Assuming delay through the SF is negligible, $\overline{T}$, the average cell delay in the switch is obtained using:

$$\overline{T} = \overline{T}_{in} + \overline{T}_o. \tag{4.67}$$

4.4.1.6 Constraints

To ensure a stable system for analysis, the system under study is constrained by the following conditions:

1. The mean fanout factor for each stage k_i, where $0 \leq i \leq n-1$, satisfies

$$1 \leq k_i \leq 2. \tag{4.68}$$

2. The mean fanout of the multicast traffic, $\overline{F}$, satisfies

$$\overline{F} = \prod_{i=0}^{n-1} k_i. \tag{4.69}$$

3. The effective offered output load λ_{OQ} in cells per switching cycle should be kept below 1 to make a stable output queueing system

$$\lambda_{OQ} = \rho \cdot \overline{F} = \rho \cdot \prod_{i=0}^{n-1} k_i < 1. \tag{4.70}$$

4.4.2 Performance Comparison

In this subsection, the analytical model and simulation results of the performance of the multicast BG switch under multicast random traffic are compared. Then, simulation results are used for performance comparison between the BG switch and the ideal switch. Because the ideal switch is a pure output-buffered switch, its performance analysis is only required for the output-queue.

4.4.2.1 Comparison Between the Analytical Model and Simulation

For comparison, the loss and delay performance are examined by using both simulation and analytical results for an 128×128 switch. Then, switch performance under different switch size, load and fanout conditions are compared and analyzed for random traffic.

4.4.2.1.1 Loss Performance

Because of the backpressure algorithm, a blocked cell is buffered in the input queue for further switching. Cell loss occurs only when the input queue is full and a new cell is arriving. In that case, all copies implicitly contained in the new cell will be dropped. Therefore, the cell loss performance of the multicast BG switch is tightly associated with the size of the input queueing space, measured in cells. During this analysis, the output queue is assumed to have enough capacity to receive any cell appearing at its input. Therefore, the only reason for the HOL cell to be kept in the input queue is the internal blocking of the switch fabric.

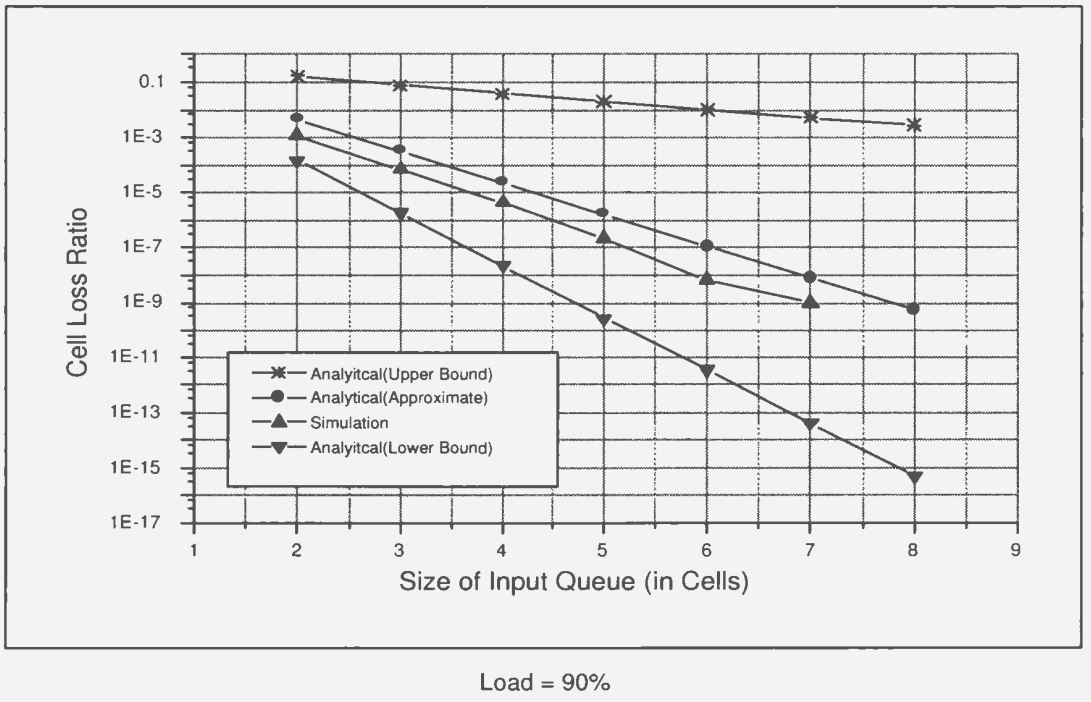


Figure 4.11: Cell loss performance comparison under 90% load for 128 × 128 BG switch under multicast random traffic with mean fanout 2

Table 4.1 shows the cell loss ratio versus the size of input queue for a 128 × 128 BG switch under various multicast random traffic loads with a mean fanout of 2. The results under 90% load are also plotted in Figure 4.11. It is observed that the input buffering requirement for the multicast BG switch is very low. With an input queue of six cell spaces, a cell loss ratio of around 10^{-8} can be achieved even under 90% offered load. The lower bound represents a best-case scenario which corresponds to the unicast traffic that has the same input load, while the upper bound represents the worst-case condition which is from the result of the broadcast traffic. The simulation results fit well between the two bounds from the analytical model and is very close to but slightly better than the approximate analysis result. As the traffic load gets lighter, even smaller queue sizes are sufficient to achieve the desired performance.

In Appendix E, the loss performance comparison for other switch sizes ranging

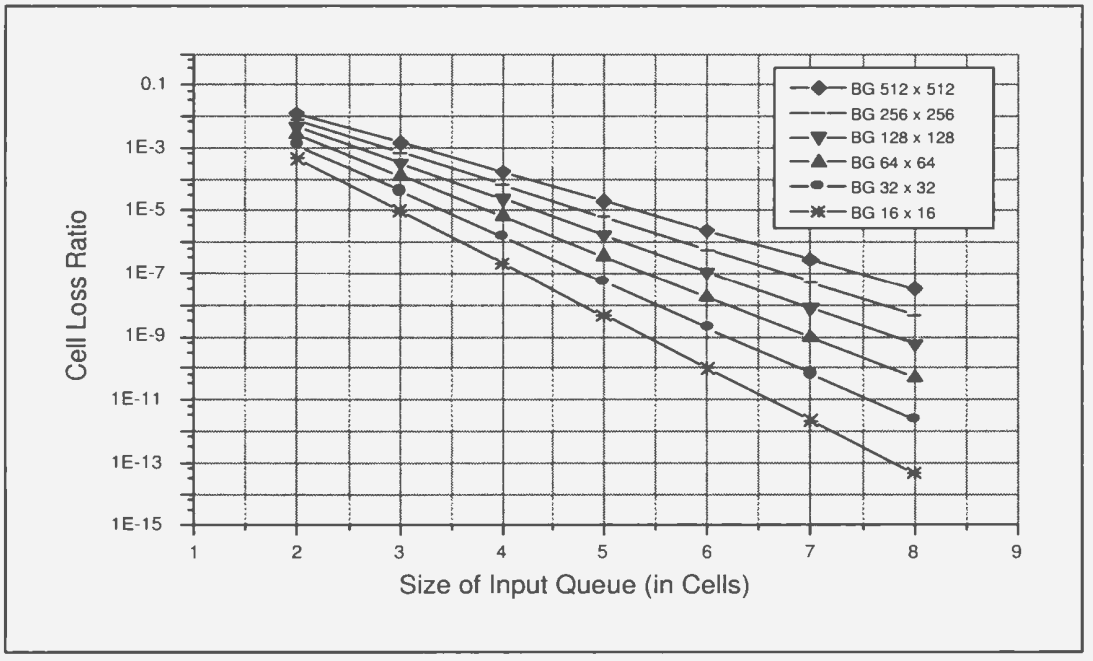
Effectively Offered Traffic Load	Size of Input Buffer	Cell Loss Probability			
		Analysis (Upper Bound)	Analysis (Approximate)	Simulation	Analysis (Lower Bound)
Load = 90%	2	1.6513E-01	4.6573E-03	1.2225E-03	1.4880E-04
	3	7.7484E-02	3.2855E-04	7.0212E-05	1.8261E-06
	4	3.8588E-02	2.3227E-05	4.4098E-06	2.2412E-08
	5	1.9755E-02	1.6423E-06	2.2102E-07	2.7507E-10
	6	1.0252E-02	1.1612E-07	7.0009E-09	3.3761E-12
	7	5.3575E-03	8.2102E-09	1.0001E-09	4.1436E-14
	8	2.8099E-03	5.8051E-10	< 1.0E-09	5.0855E-16
Load = 80%	2	6.0694E-02	1.5840E-03	4.6955E-04	5.0235E-05
	3	1.6686E-02	6.4295E-05	1.6201E-05	3.5731E-07
	4	4.7012E-03	2.6115E-06	6.4702E-07	2.5415E-09
	5	1.3336E-03	1.0608E-07	9.0012E-09	1.8078E-11
	6	3.7904E-04	4.3086E-09	< 1.0E-09	1.2859E-13
	7	1.0779E-04	1.7501E-10	< 1.0E-09	9.1463E-16
	8	3.0657E-05	7.1087E-12	< 1.0E-09	6.5057E-18
Load = 70%	2	1.8654E-02	4.7588E-04	1.5936E-04	1.5072E-05
	3	2.7228E-03	1.0495E-05	3.0695E-06	5.8628E-08
	4	4.0026E-04	2.3149E-07	5.6008E-08	2.2805E-10
	5	5.8899E-05	5.1060E-09	2.0004E-09	8.8709E-13
	6	8.6685E-06	1.1262E-10	< 1.0E-09	3.4507E-15
	7	1.2758E-06	2.4842E-12	< 1.0E-09	1.3423E-17
	8	1.8778E-07	5.4794E-14	< 1.0E-09	5.2212E-20
Load = 60%	2	4.7349E-03	1.2162E-04	4.5596E-05	3.8588E-06
	3	3.3723E-04	1.3486E-06	4.3705E-07	7.5877E-09
	4	2.4056E-05	1.4956E-08	< 1.0E-09	1.4920E-11
	5	1.7162E-06	1.6585E-10	< 1.0E-09	2.9337E-14
	6	1.2243E-07	1.8392E-12	< 1.0E-09	5.7687E-17
	7	8.7345E-09	2.0396E-14	< 1.0E-09	1.1343E-19
	8	6.2313E-10	2.2618E-16	< 1.0E-09	2.2304E-22
Load = 50%	2	9.5199E-04	2.4901E-05	1.0512E-05	7.9253E-07
	3	2.9833E-05	1.2457E-07	6.7008E-08	7.0586E-10
	4	9.3512E-07	6.2319E-10	< 1.0E-09	6.2866E-13
	5	2.9312E-08	3.1176E-12	< 1.0E-09	5.5991E-16
	6	9.1879E-10	1.5596E-14	< 1.0E-09	4.9868E-19
	7	2.8800E-11	7.8022E-17	< 1.0E-09	4.4414E-22
	8	9.0274E-13	3.9032E-19	< 1.0E-09	3.9557E-25

Table 4.1: Cell loss performance comparison between analytical model and simulation results for 128×128 BG switch

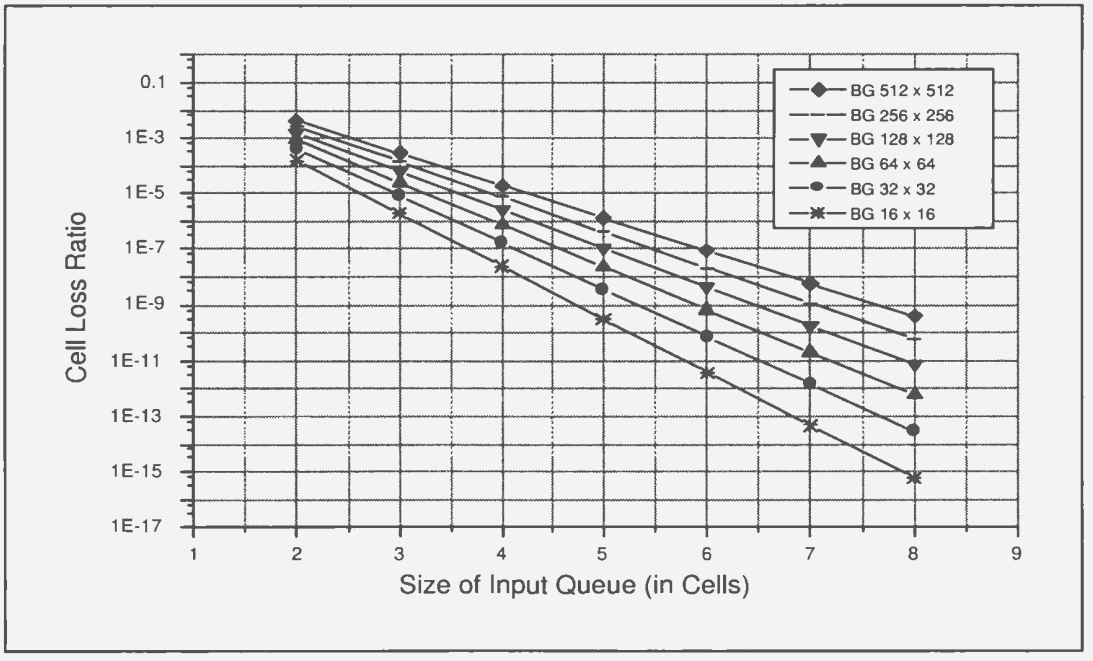
from 16×16 to 512×512 are provided. Multicast random traffic with a mean fanout of 2 is used. Traffic loads range from 50% to 90%. All figures demonstrate one thing in common: the input buffer requirement to achieve a desired loss performance is very low. For almost all cases, 8 cell buffers for each input queue are sufficient to achieve a cell loss rate better than 10^{-8} for all switch sizes. This is the result of the low internal blocking of the BG switch fabric: with enough output buffering, very few cells will be left behind at the input queue after each switching cycle, thus the input buffering requirement is low. For all cases, the simulation results fit well between the two bounds and are close to the approximation results derived from the analytical modelling.

It is also observed that with small amount of input buffering, such as 2, 3 and 4, the trend of the simulation results is smooth and very consistent because larger amounts of cell loss samples are collected. With a total number of 10^9 cells generated during the simulation, as the size of input buffers approaches the boundary condition, such as 7 or 8 in most cases, very few cell losses will occur. One more or one less cell loss makes the loss performance vary significantly. That is why in some boundary cases, the simulation results divert from the original trend. To obtain a more reliable simulation results in these cases, more cells ($>> 10^9$) will be required in the simulation. However, it would be at the cost of longer simulation time.

From the previous discussion, it has been noted that even though the two bounds set the best and the worst switch performance, they are either too high or too low to be useful when making a design choice in a practical switch. It is also observed that the analytical approximation provides a good indication on the switch performance, which is close to and consistent with our simulation results for different switch sizes and load conditions. Therefore, in the remaining discussion, the analytical approximation will be used as the analytical result to compare with the simulation result. In Figures

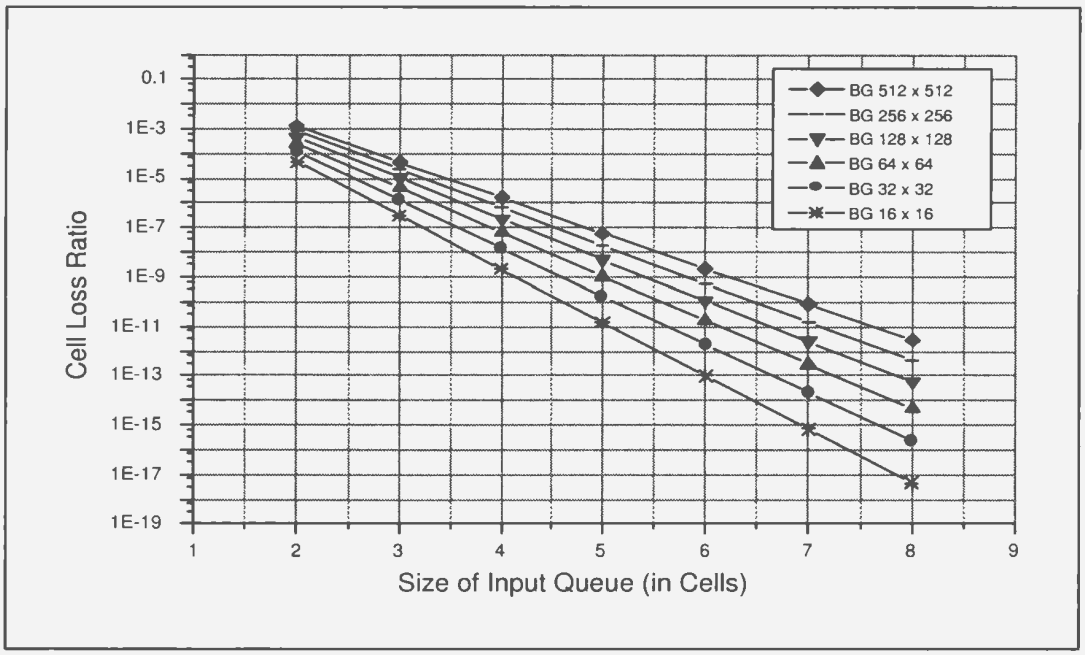


(a) Load = 90%

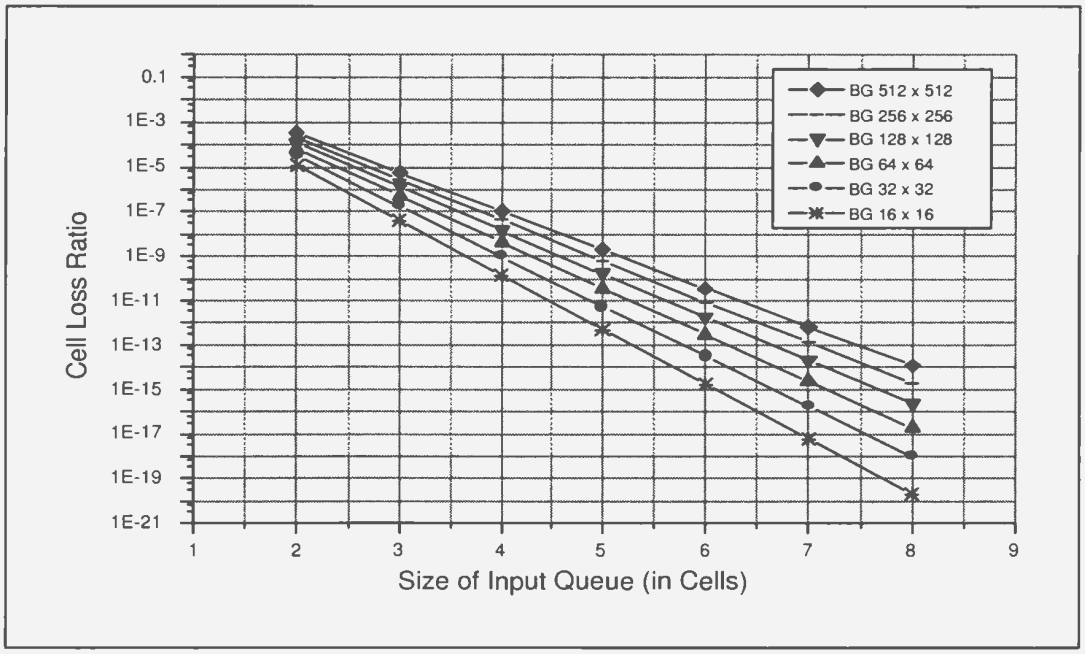


(b) Load = 80%

Figure 4.12: Loss performance comparison (analytical) for various switch sizes under multicast random traffic with mean fanout 2

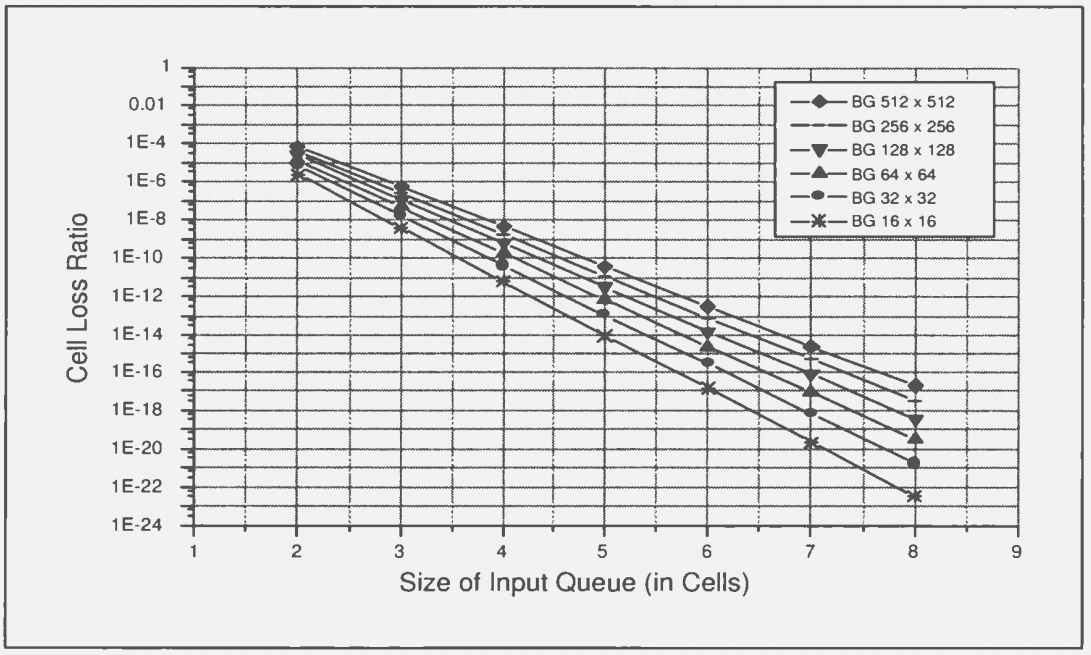


(a) Load = 70%



(b) Load = 60%

Figure 4.13: Loss performance comparison (analytical) for various switch sizes under multicast random traffic with mean fanout 2 (continued)



(a) Load = 50%

Figure 4.14: Loss performance comparison (analytical) for various switch sizes under multicast random traffic with mean fanout 2 (continued)

4.12 to 4.14, the analytical approximation for various switch sizes under various loads is provided. Multicast random traffic with a mean fanout of 2 is used.

Fanout is the most important characteristic for multicast traffic. Even under the same load, multicast traffic with different fanouts behave differently. Table 4.2 presents the loss performance comparison for a 128×128 BG switch under mean fanout values of 2, 4, and 8. The analytical approximation and simulation results are plotted in Figure 4.15 for comparison. The heavy load situation, i.e., 90% and 80% offered load, can be considered to demonstrate the high performance of the BG switch because that will generate larger demand on the resources. Once the performance under high load conditions are accepted, with similar or even less buffering resources, it is guaranteed that the performance requirements under the low traffic load will be satisfied. The simulation and analytical model results are consistent for different

Traffic Load	Mean Fanout	Input Buffer Size	Cell Loss Probability			
			Analysis (Upper bound)	Analysis (Approximate)	Simulation	Analysis (Lower bound)
Load = 90%	F = 2	2	1.6513E-01	4.6573E-03	1.2225E-03	1.4880E-04
		3	7.7484E-02	3.2855E-04	7.0212E-05	1.8261E-06
		4	3.8588E-02	2.3227E-05	4.4098E-06	2.2412E-08
		5	1.9755E-02	1.6423E-06	2.2102E-07	2.7507E-10
		6	1.0252E-02	1.1612E-07	7.0009E-09	3.3761E-11
		7	5.3575E-03	8.2102E-09	1.0001E-09	4.1436E-14
	F = 4	2	1.7377E-02	4.2648E-04	3.6248E-04	7.8483E-06
		3	2.4500E-03	8.8995E-06	8.1333E-06	2.2018E-08
		4	3.4717E-04	1.8572E-07	1.9530E-07	6.1769E-11
		5	4.9229E-05	3.8759E-09	5.4000E-09	1.7329E-13
		6	6.9813E-06	8.0888E-11	< 1.0E-09	4.8615E-16
		7	9.9006E-07	1.6881E-12	< 1.0E-09	1.3639E-18
	F = 8	2	2.3357E-03	1.1322E-04	1.0235E-04	7.7232E-07
		3	1.1572E-04	1.2112E-06	9.4410E-07	6.7903E-10
		4	5.7355E-06	1.2957E-08	9.0000E-09	5.9701E-13
		5	2.8427E-07	1.3861E-10	< 1.0E-09	5.2489E-16
		6	1.4089E-08	1.4828E-12	< 1.0E-09	4.6149E-19
		7	6.9831E-10	1.5863E-14	< 1.0E-09	4.0575E-22
Load = 80%	F = 2	2	6.0694E-02	1.5840E-03	4.6955E-04	5.0235E-05
		3	1.6686E-02	6.4295E-05	1.6201E-05	3.5731E-07
		4	4.7012E-03	2.6115E-06	6.4702E-07	2.5415E-09
		5	1.3336E-03	1.0608E-07	9.0012E-09	1.8078E-11
		6	3.7904E-04	4.3086E-09	< 1.0E-09	1.2859E-13
		7	1.0779E-04	1.7501E-10	< 1.0E-09	9.1463E-16
	F = 4	2	6.1513E-03	1.5779E-04	1.4481E-04	2.9270E-06
		3	5.0221E-04	1.9946E-06	1.9448E-06	5.0118E-09
		4	4.1063E-05	2.5215E-08	2.4800E-08	8.5818E-12
		5	3.3579E-06	3.1875E-10	< 1.0E-09	1.4695E-14
		6	2.7460E-07	4.0295E-12	< 1.0E-09	2.5162E-17
		7	2.2455E-08	5.0939E-14	< 1.0E-09	4.3084E-20
	F = 8	2	8.3429E-04	4.2881E-05	1.4979E-05	2.9726E-07
		3	2.4455E-05	2.8172E-07	2.5760E-07	1.6212E-10
		4	7.1691E-07	1.8509E-09	1.6000E-09	8.8412E-14
		5	2.1017E-08	1.2160E-11	< 1.0E-09	4.8217E-17
		6	6.1611E-10	7.9893E-14	< 1.0E-09	2.6296E-20
		7	1.8062E-11	5.2489E-16	< 1.0E-09	1.4341E-23

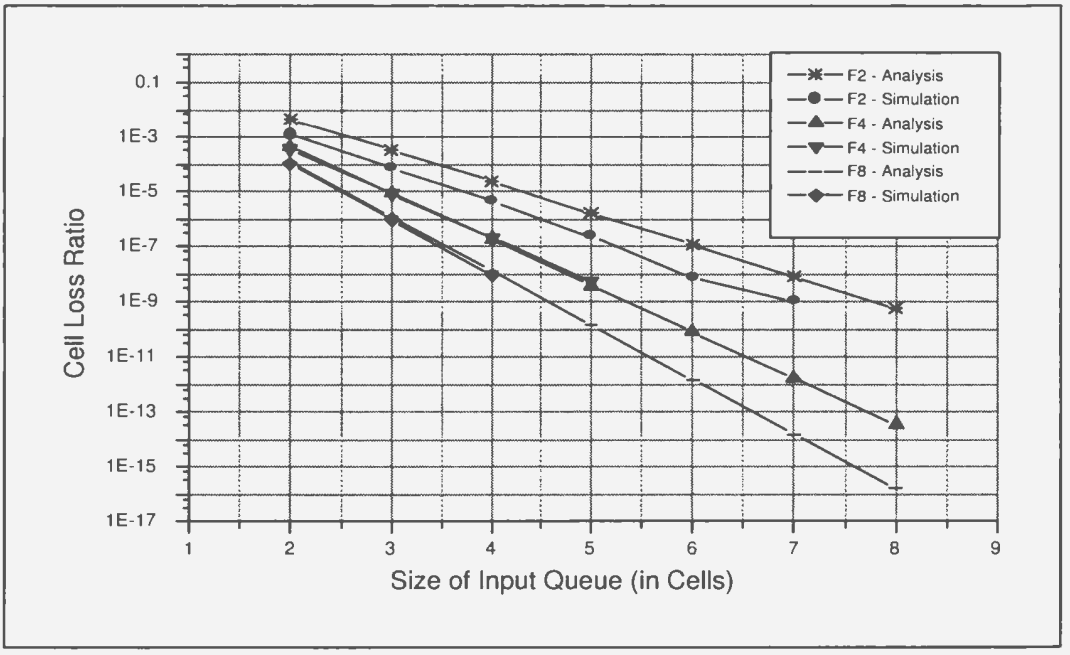
Table 4.2: Loss performance comparison between analytical model and simulation results for 128 × 128 BG switch for different fanout and load

load and fanout. The input buffering requirements are very low for different fanouts, even under very high traffic load. With the same offered load, the larger the mean fanout, the better the loss performance of the BG switch. This is because multicast cell replication is done within the switch as late as required, thereby reducing load and blocking at the earlier switch stages.

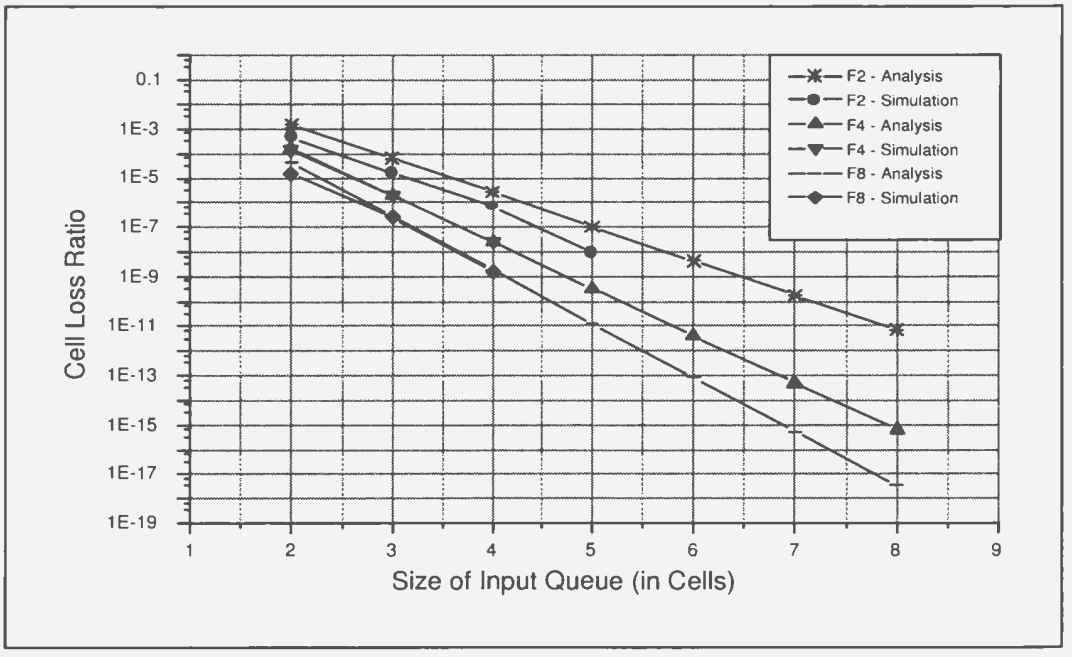
4.4.2.1.2 Delay Performance

Because of the bufferless SF design, cells are delayed either at the input queue or at the output queue. The delay associated with the overhead transfer during the reservation phase, which is a constant value and applies to every cell, is not included. At the input queue, only the master cell is stored. Multiple destination requests are contained in the cell header. When reaching the output queue, each copy becomes an independent cell. Therefore, in delay performance analysis, the input queueing delay is measured in terms of the master cell while the output queueing delay and total delay are calculated based on an individual copy. Figures 4.16 and 4.17 presents delay performance comparison between the analytical approximation and simulation for various switch sizes under 90% multicast random traffic with a mean fanout of 2. Enough buffering resources are provided at both input ports and output ports to ensure virtually no cell loss. The trends for different switch sizes are almost the same. The input queueing delay is much smaller than the output queueing delay. This implies that the input buffering requirement is very small when compared to that of the output buffering.

Table 4.3 presents the comparison of the total delay and its breakdown between the analytical model and the simulation results for an 128×128 multicast BG switch under various load conditions. Enough output buffering is provided so that any cell that manages to arrive at the output port is accepted. From the table, it is clear

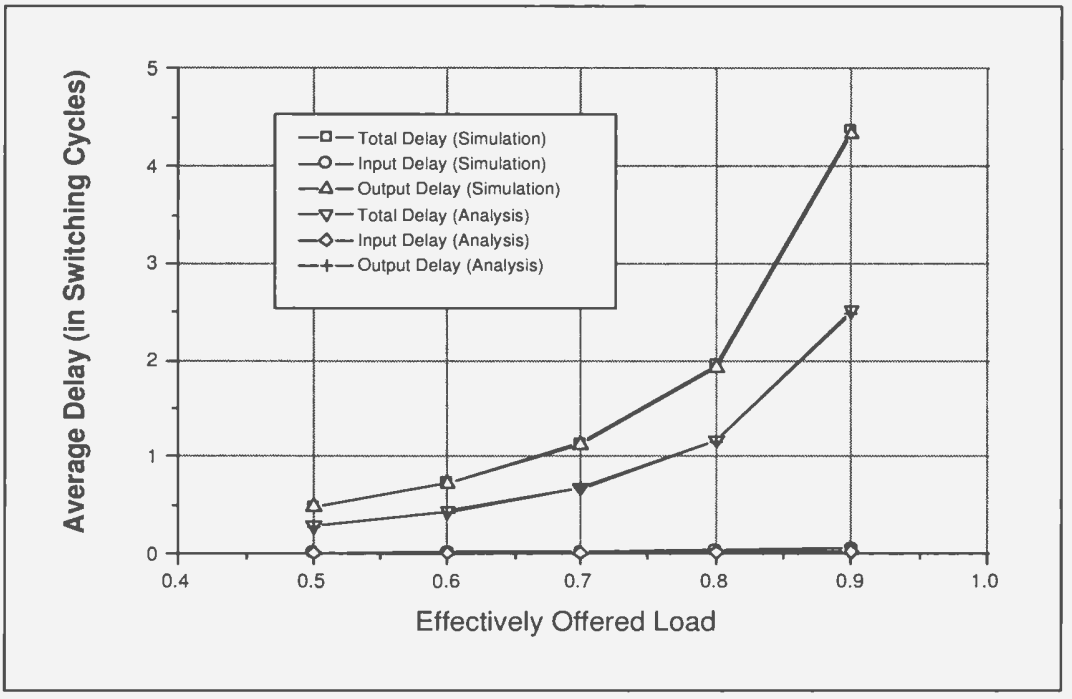


(a) Load = 90%

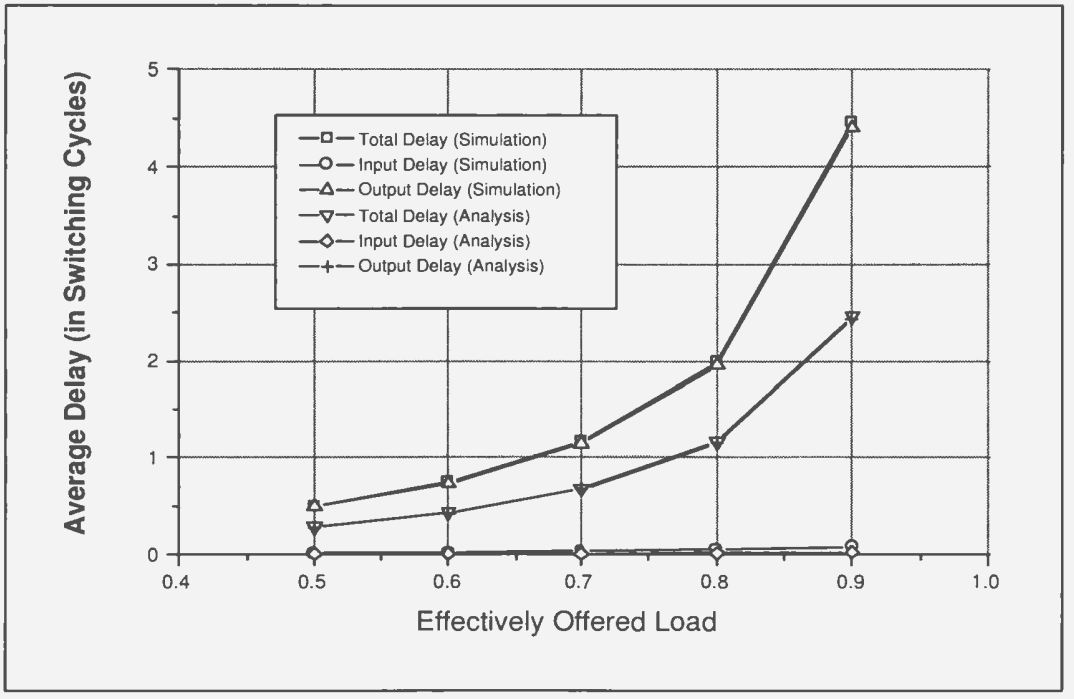


(b) Load = 80%

Figure 4.15: Loss performance comparison between analytical approximation and simulation results under different fanout and loads for 128×128 BG switch

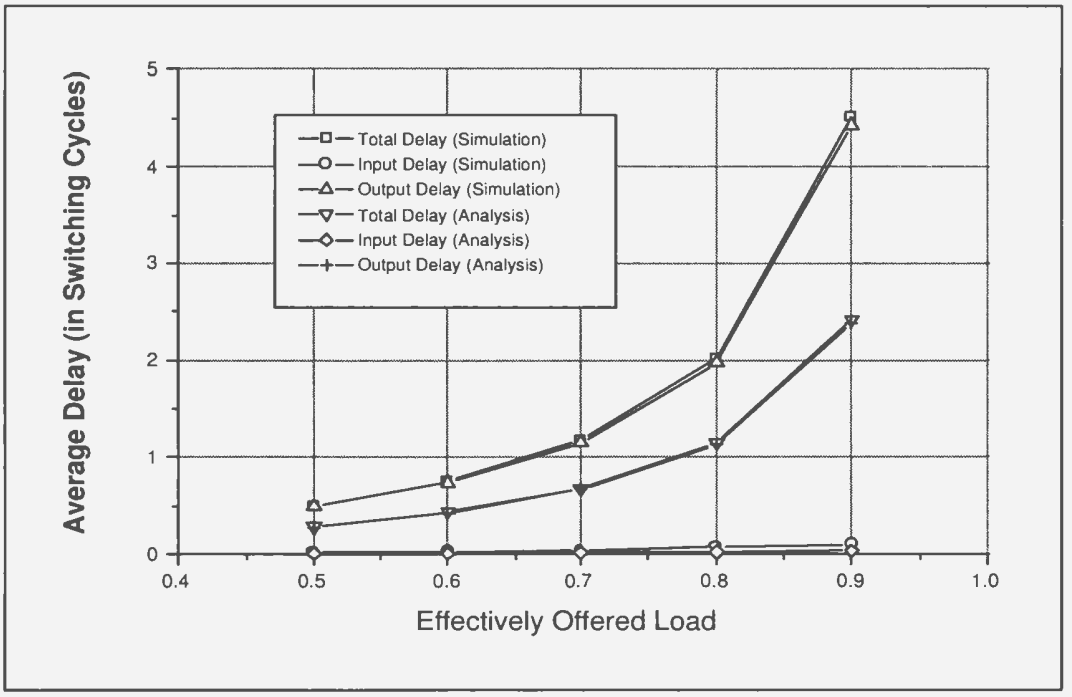


(a) 32 x 32 BG Switch

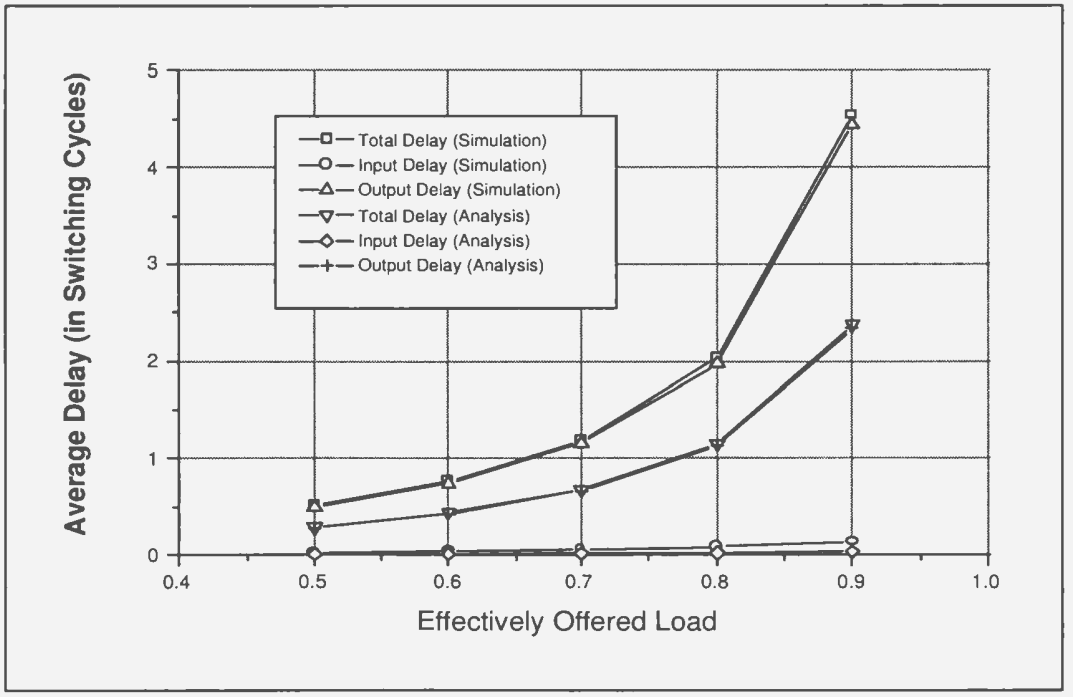


(b) 64 x 64 BG Switch

Figure 4.16: Delay performance breakdown for simulation and analytical model under 90% load multicast random traffic with mean fanout of 2: (a) 32×32 switch (b) 64×64 switch



(c) 128 x 128 BG Switch



(d) 256 x 256 BG Switch

Figure 4.17: Delay performance breakdown for simulation and analytical model under 90% load multicast random traffic with mean fanout of 2: (c) 128×128 switch (d) 256×256 switch

Effectively Offered Load	Input Buffer Size	Delay Performance					
		Simulation			Analysis		
		Total	Input	Output	Total	Input	Output
Load = 90%	2	4.4549	0.0970	4.3821	2.41356	0.03056	2.3830
	3	4.5109	0.1019	4.4333	2.41361	0.03061	2.3830
	4	4.5141	0.1023	4.4361	2.41361	0.03061	2.3830
	5	4.5186	0.1024	4.4406	2.41361	0.03061	2.3830
	6	4.5193	0.1023	4.4413	2.41361	0.03061	2.3830
Load = 80%	2	2.0103	0.0641	1.9639	1.15523	0.01923	1.1360
	3	2.0164	0.0658	1.9684	1.15524	0.01924	1.1360
	4	2.0181	0.0659	1.9700	1.15524	0.01924	1.1360
	5	2.0184	0.0659	1.9703	1.15524	0.01924	1.1360
	6	2.0166	0.0659	1.9695	1.15524	0.01924	1.1360
Load = 70%	2	1.1753	0.0407	1.1470	0.6805	0.01170	0.6688
	3	1.1762	0.0411	1.1474	0.6805	0.01170	0.6688
	4	1.1762	0.0411	1.1474	0.6805	0.01170	0.6688
	5	1.1762	0.0412	1.1473	0.6805	0.01170	0.6688
	6	1.1764	0.0412	1.1474	0.6805	0.01170	0.6688
Load = 60%	2	0.7538	0.0243	0.7375	0.4334	0.00680	0.4266
	3	0.7542	0.0245	0.7378	0.4334	0.00680	0.4266
	4	0.7540	0.0245	0.7375	0.4334	0.00680	0.4266
	5	0.7541	0.0245	0.7378	0.4334	0.00680	0.4266
	6	0.7541	0.0245	0.7376	0.4334	0.00680	0.4266
Load = 50%	2	0.5009	0.0135	0.4922	0.2837	0.00360	0.2801
	3	0.5011	0.0136	0.4924	0.2837	0.00360	0.2801
	4	0.5010	0.0136	0.4923	0.2837	0.00360	0.2801
	5	0.5011	0.0136	0.4924	0.2837	0.00360	0.2801
	6	0.5010	0.0136	0.4923	0.2837	0.00360	0.2801

Table 4.3: Average delay performance comparison for 128 × 128 BG switch under multicast random traffic with a mean fanout of 2

	Probability of Simultaneous Cell Arrival	Switch load				
		0.5	**0.6**	**0.7**	**0.8**	**0.9**
Results Based on Analytical Modelling	a_4	0.00002	0.00004	0.00009	0.00021	0.00041
	a_3	0.00178	0.00359	0.00646	0.01069	0.01656
	a_2	0.06441	0.09148	0.12209	0.15548	0.19079
	a_1	0.36579	0.40611	0.43603	0.45613	0.46708
	a_0	0.56801	0.49878	0.43531	0.37749	0.32515
Results Based on Binomial Distribution	a_4	0.00024	0.00051	0.00094	0.00160	0.00256
	a_3	0.00683	0.01147	0.01768	0.02560	0.03531
	a_2	0.07178	0.09754	0.12506	0.15360	0.18244
	a_1	0.33496	0.36848	0.39306	0.40960	0.41894
	a_0	0.58618	0.52201	0.46325	0.40960	0.36075

Table 4.4: Simultaneous cell arrival probability to output queue

that the delay caused by output queueing is the dominant contributor. The increase of the size of the input buffer only slightly affects the delay performance. It is also noted that there is always a gap on the output queueing delay between the analytical model and simulation, which constitutes the main reason for the difference in overall average cell delay. This is because when the average loads on the four input links feeding each output queue are used for the output queue analysis, the distribution of the simultaneous arrival probability is changed.

To demonstrate this, a 4×4 BG switch under unicast uniform random traffic is used. The 4×4 switch is guaranteed to be non-blocking under any traffic condition. Therefore, the possible influence due to switch fabric internal blocking does not exist. Using the analytical model, the probabilities of simultaneous cell arrival to the output queue can be calculated by using Equations 4.32 to 4.36 and they are provided in Table 4.4 for various load conditions. These calculations assume the links feeding the output queue have biased loads and that the links are independent. Although the regular and alternative link concept results in biased loads, the traffic on the links

is not independent. A more realistic view of traffic arrival to the output queue is to consider simply the number of arrivals based on the binomial distribution of Equation 3.1. The simultaneous cell arrival probabilities based on the binomial distribution are plotted in the same table for comparison.

From Table 4.4, for example, under 90% load, the probability of having 4 arrivals in a time slot to output queue is expected to be 0.256% based on the random traffic assumption at the switch input. But this probability is changed to 0.041% by using the analytical model. Similarly, the probability of having 3 arrivals is changed from 3.53% to 1.66%. The probability of having 2 arrivals remains largely unchanged. Even though the probability of having one cell arrival increases, the probability for larger simultaneous arrival events, that is, more than 2 cells arriving to the output queue in one switching cycle, has the most significant impact on the queue delay performance because the output can only consume one cell during each switching cycle. Hence, one cell arrival will not cause any queue back up. Therefore, the overall impact of simultaneous cell arrivals to the performance of the output queue becomes less significant in the analytical model, which makes the delay performance from the analytical model appear better than the results from the simulation. The same argument also applies to other load conditions and switch sizes.

From Equation 4.5, it is obvious that with the same offered load at the output port, traffic with larger mean fanout will have smaller input traffic load because more copies are contained in each master cell. That means fewer incoming cells will arrive at switch inputs during each switch cycle. Even though each copy will become an individual cell eventually, replication is performed stage by stage. Therefore, before coming into the last stage, the chance of cells being blocked is reduced because fewer cells are competing for the internal links. With more cells switched to the output queue and fewer cells left behind at the input side, it seems that the input delay

Mean Fanout	Input Buffer Size	Delay Performance					
		Simulation			Analysis		
		Total	Input	Output	Total	Input	Output
F = 2	2	4.45491	0.09706	4.38218	2.41362	0.03056	2.38306
	3	4.51090	0.10197	4.43333	2.41367	0.03061	2.38306
	4	4.51412	0.10239	4.43612	2.41367	0.03061	2.38306
	5	4.51867	0.10245	4.44064	2.41367	0.03061	2.38306
	6	4.51641	0.10239	4.44136	2.41367	0.03061	2.38306
F = 4	2	4.47206	0.10143	4.42430	2.51073	0.01371	2.49702
	3	4.49091	0.10251	4.44211	2.51074	0.01372	2.49702
	4	4.49170	0.10258	4.44283	2.51074	0.01372	2.49702
	5	4.49111	0.10253	4.44227	2.51074	0.01372	2.49702
	6	4.48644	0.10252	4.43762	2.51074	0.01372	2.49702
F = 8	2	4.48335	0.11378	4.45069	2.56670	0.00875	2.55795
	3	4.47406	0.11402	4.44117	2.56670	0.00875	2.55795
	4	4.47128	0.11394	4.43838	2.56670	0.00875	2.55795
	5	4.47722	0.11400	4.44430	2.56670	0.00875	2.55795
	6	4.47058	0.11389	4.43772	2.56670	0.00875	2.55795

Table 4.5: Delay performance comparison between analytical model and simulation results for 128×128 BG switch for different fanout under 90% load

should decrease as traffic fanout becomes larger, as given by the analytical result in Table 4.5 for the 128×128 BG switch under 90% multicast random traffic. However, it is interesting to note that the simulation results indicate an opposite trend. This can be explained as follows. The master cell has to stay in the input queue until all copies are delivered, and even though the lowered traffic load mitigates the situation, the larger fanout counteracts this effect and increases the waiting time. However, the correlation for HOL cells between consecutive switching cycles, which is caused by SF internal blocking, is not reflected in the analytical model. Therefore, it is observed from the simulation results in Table 4.5 that the input delay, which is measured on master cell, increases marginally as the fanout becomes larger.

4.5 Performance Analysis under Multicast Bursty Traffic

Due to congestion and bottleneck nodes in the network, traffic in high-speed networks tends to be bursty. Bursty traffic is a traffic type in which the switch inputs receive sudden bursts of packets destined to one output [85]. The bursty traffic model resembles the real traffic nature much better than uniform random traffic. In this section, the multicast BG switch performance under multicast bursty traffic is investigated. Simulation results are used for this discussion. All performance measures are obtained through a simulation period of switching 10^9 cells across the SF. The ON-OFF model [9, 26, 80] is used to describe the traffic arrival behavior. The truncated geometric distribution [9, 80] is used for cell fanout. Fanout remains the same throughout a burst. Cell destination is selected uniformly. Using this model, a traffic source generates cells in a bursty manner with alternating ON period and OFF period. During the same ON period, all cells from the same input go to the same destination. Therefore, traffic is highly correlated.

Similar to the analysis for the multicast random traffic, the loss performance and delay performance are studied. The results are compared to those of the random traffic to show the impact of bursty traffic on switch performance. Furthermore, the maximum/average input/output buffer requirements under the new traffic condition are studied. Performance results are compared to those of an ideal multicast switch, which is a purely output-buffered switch and represents the best performance that a SF can achieve. As described earlier, it would be too costly to build such a switch in terms of hardware complexity, especially when the switch size becomes very large.

4.5.1 Loss Performance

In the loss performance study under multicast bursty traffic, the impact from traffic load and switch size is examined first. Then the influence of the traffic fanout and burstiness is studied using the 128 × 128 BG switch. The results are compared to those of the multicast random traffic in which the burstiness index can be considered to have a value of one. In the analysis, adequate output buffering resource is assumed so that any cell that manages to reach any output queue will be accepted. Cell loss probability is measured over the size of the input queue. Because the ideal multicast switch can transfer all incoming cells to their requested outputs during the same switching cycle, zero cell loss will always be achieved as long as adequate output buffering is provided. Therefore, the loss performance study focus only on the BG switch.

First, the cell loss performance is studied with respect to traffic load and switch size. Similar to the observation from the multicast random traffic, given the same burst length and fanout, better the loss performance can be achieved under a lighter traffic load. The loss performance is only slightly affected by switch size. However, burstiness increases traffic correlation. Hence, compared to that of the multicast random traffic, the degradation of BG switch loss performance is obvious as incoming traffic gets burstier. In the following discussion, multicast bursty traffic with a mean fanout of 2 and an average burst length of 5 is used.

Figure 4.18 plots the cell loss probability versus the size of the input queue for an 128 × 128 multicast BG switch under 90% random traffic and bursty traffic. It is clear that to achieve negligible cell loss under multicast random traffic, space for 8 cells for each input queue suffices. However, this number increases to almost 50 when the multicast bursty traffic is used. Because of the assumption that the output queue

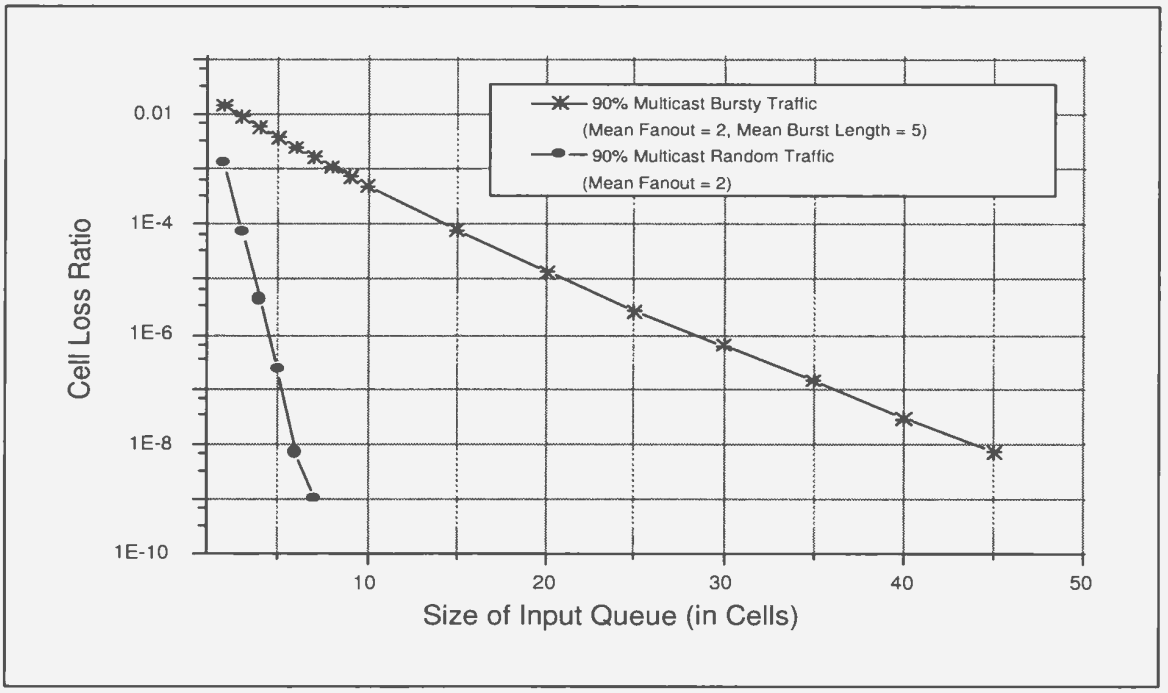
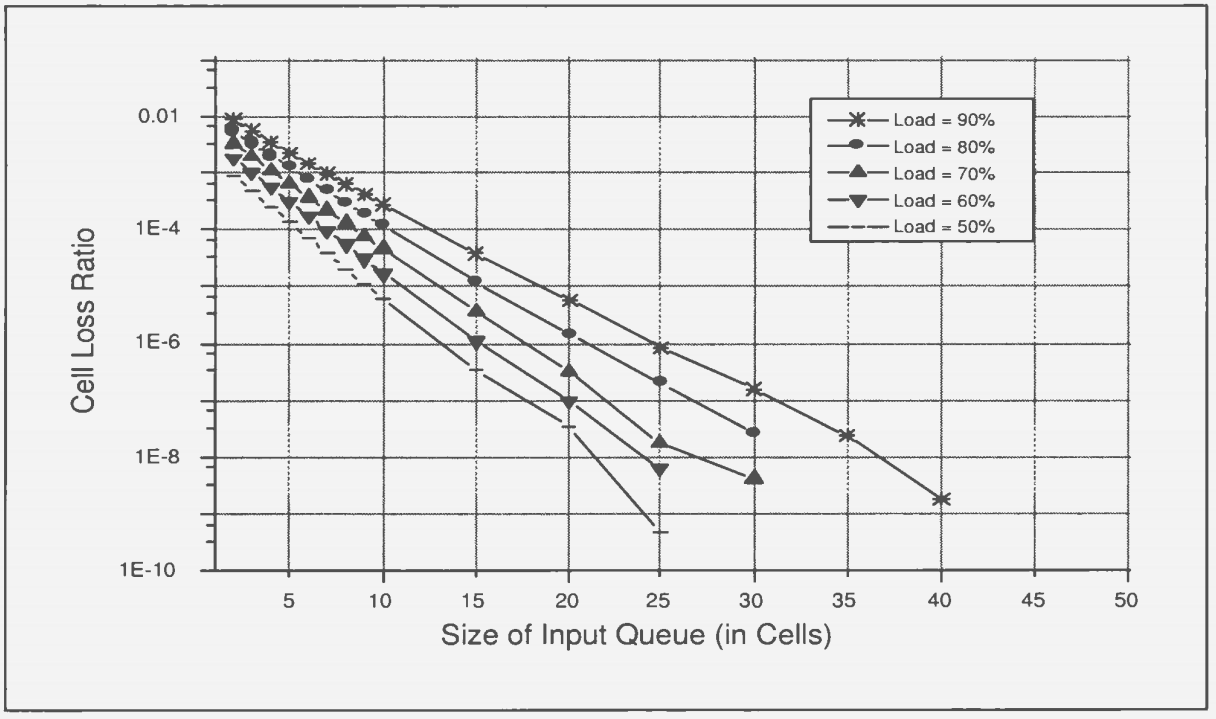


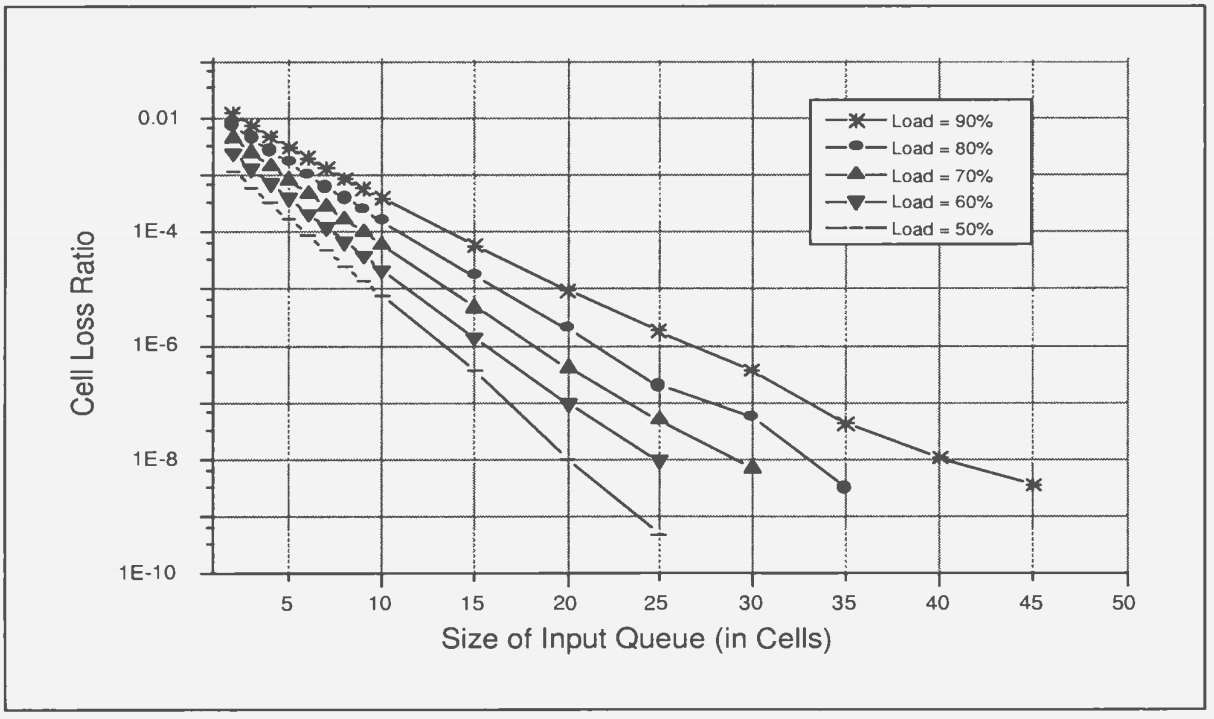
Figure 4.18: Loss performance comparison for the 128 × 128 BG switch under 90% multicast random and multicast bursty traffic with average burst length of 5

has the capability to accept any incoming cell, this difference implies that the switch fabric will experience more internal blocking as incoming traffic gets burstier. In the following discussion, it will be seen that the input buffer requirement will become even larger as traffic burstiness increases further. But no matter how much it grows, it is always a very small portion of the size of the required output queue.

In Figures 4.19 and 4.20, the loss performance under different load conditions are plotted for switch sizes ranging from 32 to 256. In all cases, as traffic load increases, the loss performance degrades. This implies the necessity to equip more input buffering resource to maintain the same level of loss performance. For all multicast BG switches in the figure, a cell loss probability better than 10^{-8} has been achieved with each input queue with space for 30 cells for the 70% bursty traffic. The number has to be increased to 45 when traffic load is increased to 90%.

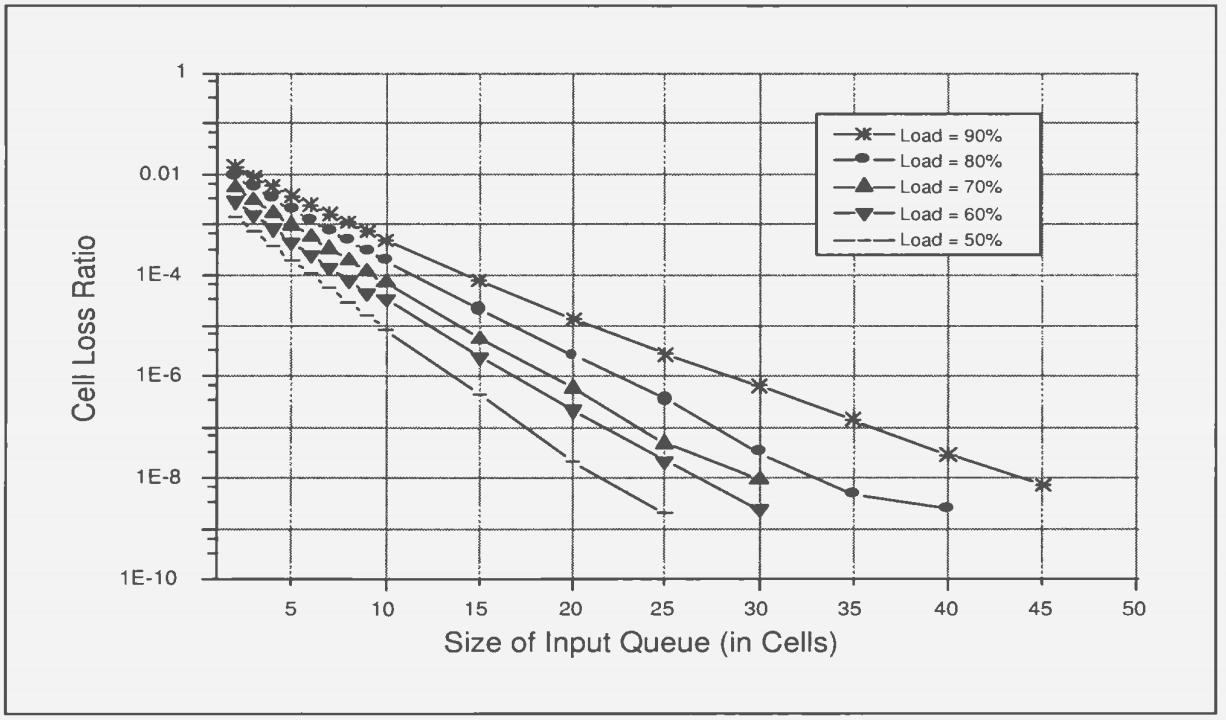


(a) 32 x 32 BG Switch

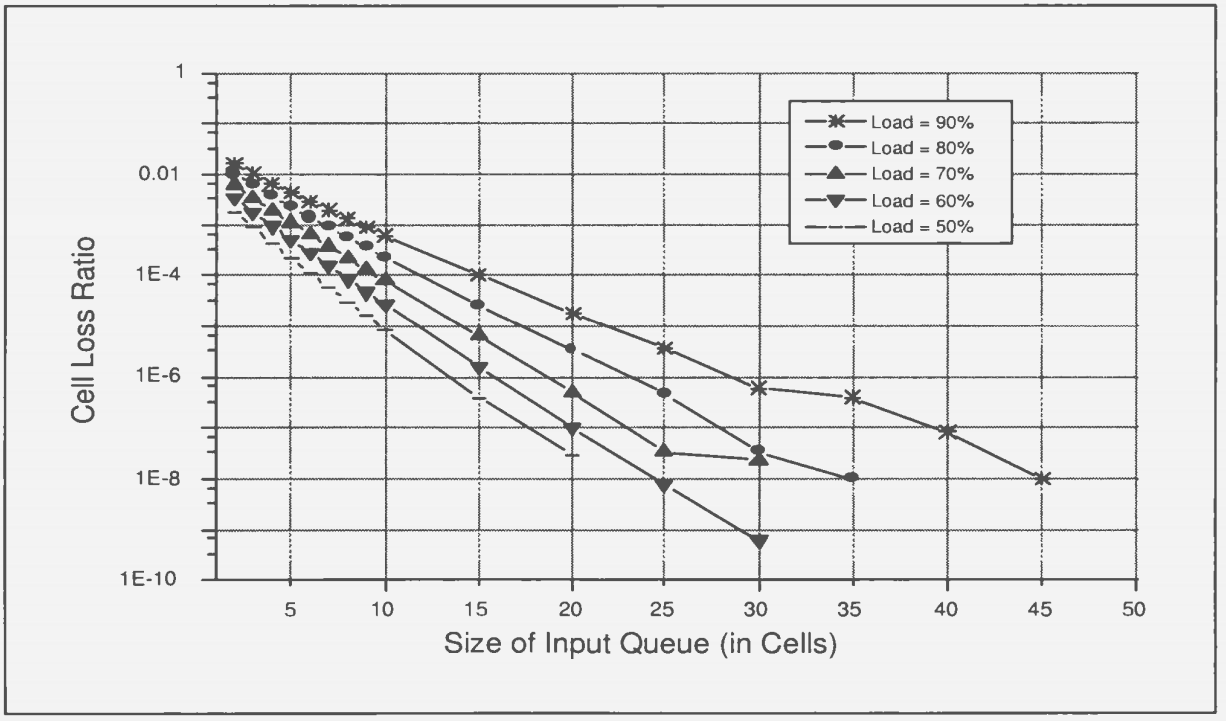


(b) 64 x 64 BG Switch

Figure 4.19: Loss performance of 32×32 and 64×64 BG switch under various load multicast bursty traffic with mean fanout of 2 and average burst length of 5



(c) 128 x 128 BG Switch



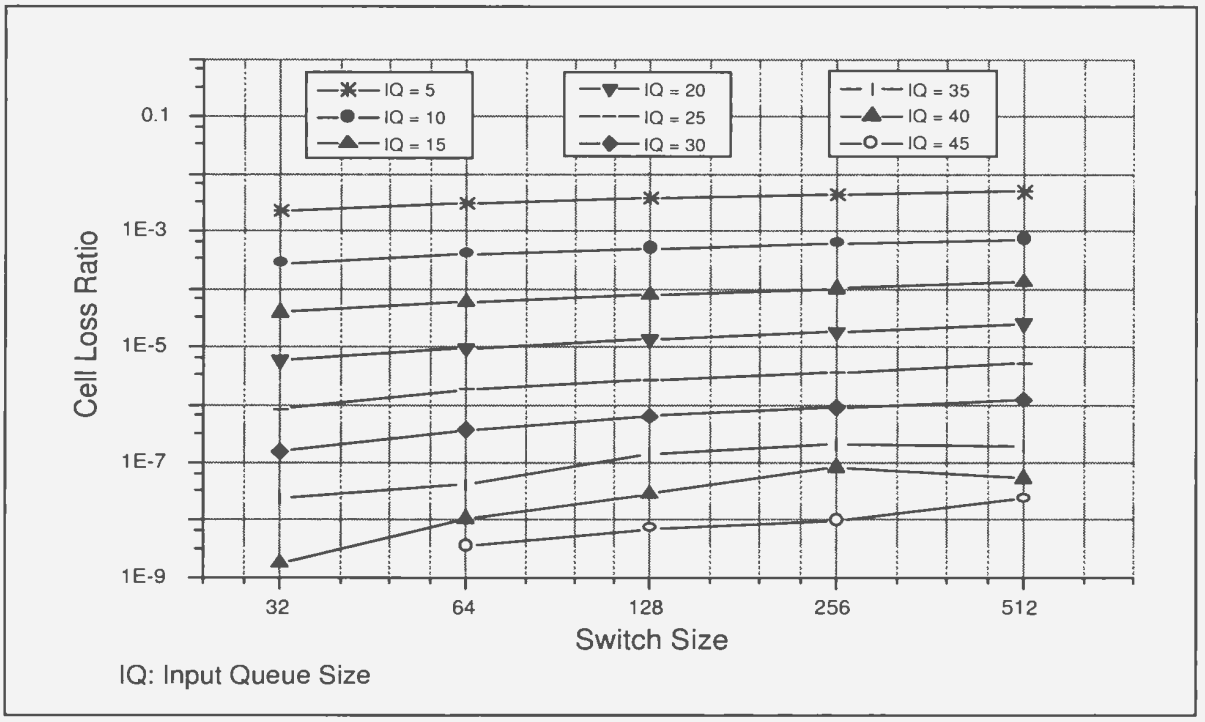
(d) 256 x 256 BG Switch

Figure 4.20: Loss performance of 128 × 128 and 256 × 256 BG switch under various load multicast bursty traffic with mean fanout of 2 and average burst length of 5

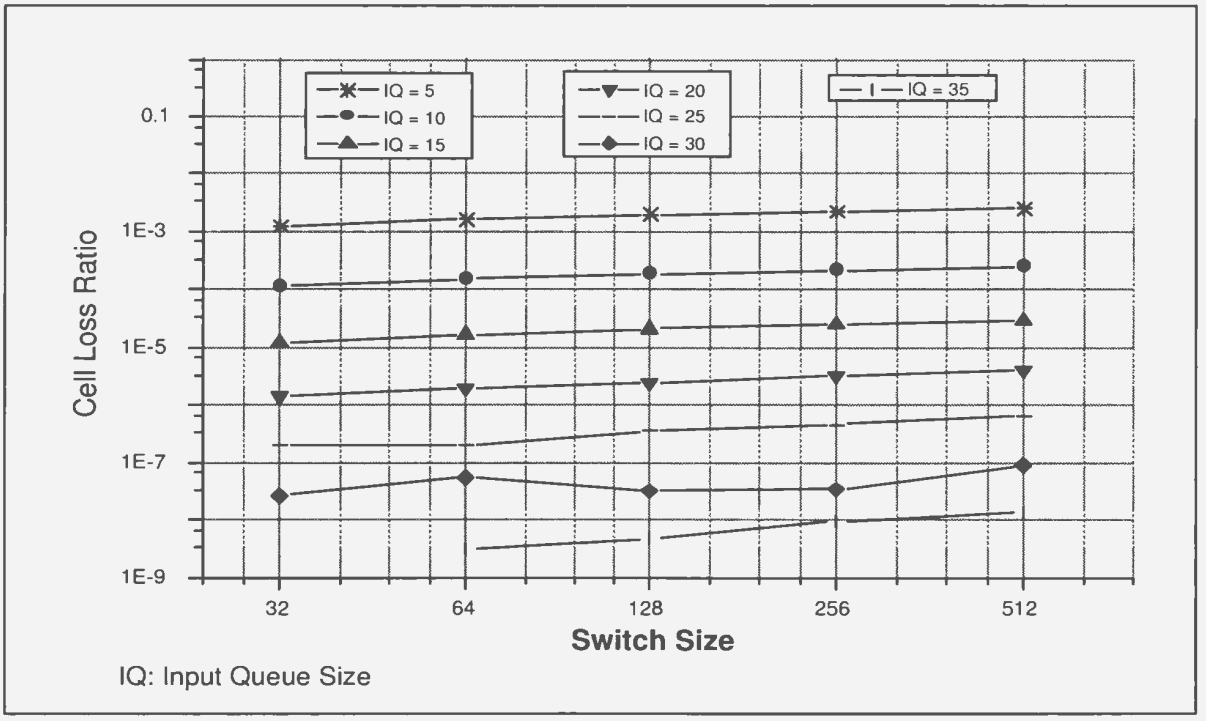
It is also observed that under the same traffic type and load, the required input buffering to achieve a desired loss performance increases only slightly as the switch size becomes larger. Figure 4.21 depicts the loss performance versus the switch size for 90% and 80% multicast bursty traffic respectively. Different choices of the size of the input queue are plotted in the same figure for comparison.

It is observed that when each input queue is equipped with small amount of buffering, such as 10 and 15, the trend across switch sizes is consistent. But when more buffers are added, the fluctuation becomes large, such as in the case of a 40-cell or 45-cell input queue for the 90% traffic load and 30-cell or 35-cell input queue for the 80% traffic load. This is because for every simulation experiment, the performance is measured over one billion cells which are switched through the SF. When the input buffering is small, more lost cells occur. Therefore, one more or one less cell loss will little change the result. As more buffers are added, in particular, for the low traffic load situation, very few loss cases will be experienced throughout the trial. Hence, even the change of one cell loss will affect the result significantly. As stated earlier to make the results more reliable, the simulation period should be extended so that more cells are generated and switched through the fabric.

Now, the cell loss performance under different fanout and burstiness conditions are studied. The effect of fanout on loss performance in the multicast random traffic has been studied: under the same offered load, the larger mean fanout, the better loss performance. The same argument is applied to multicast bursty traffic. However, in this case, the size of input queue to achieve the same level of loss performance increases significantly. Figure 4.22 plots the cell loss probability for an 128×128 BG switch under 90% bursty traffic with a mean fanout of 2, 4, and 8. The trend is obvious. For example, with 30-cell input queue, the multicast BG switch achieves a cell loss ratio of 6.36×10^{-7}, 4.13×10^{-7}, and 3.06×10^{-7} for the mean fanout of 2,



(a) 90% Multicast Bursty Traffic (Mean Fanout = 2, Average Burst Length = 5)



(b) 80% Multicast Bursty Traffic (Mean Fanout = 2, Average Burst Length = 5)

Figure 4.21: Loss performance vs size of the input queue for various switch sizes under 90% and 80% multicast bursty traffic

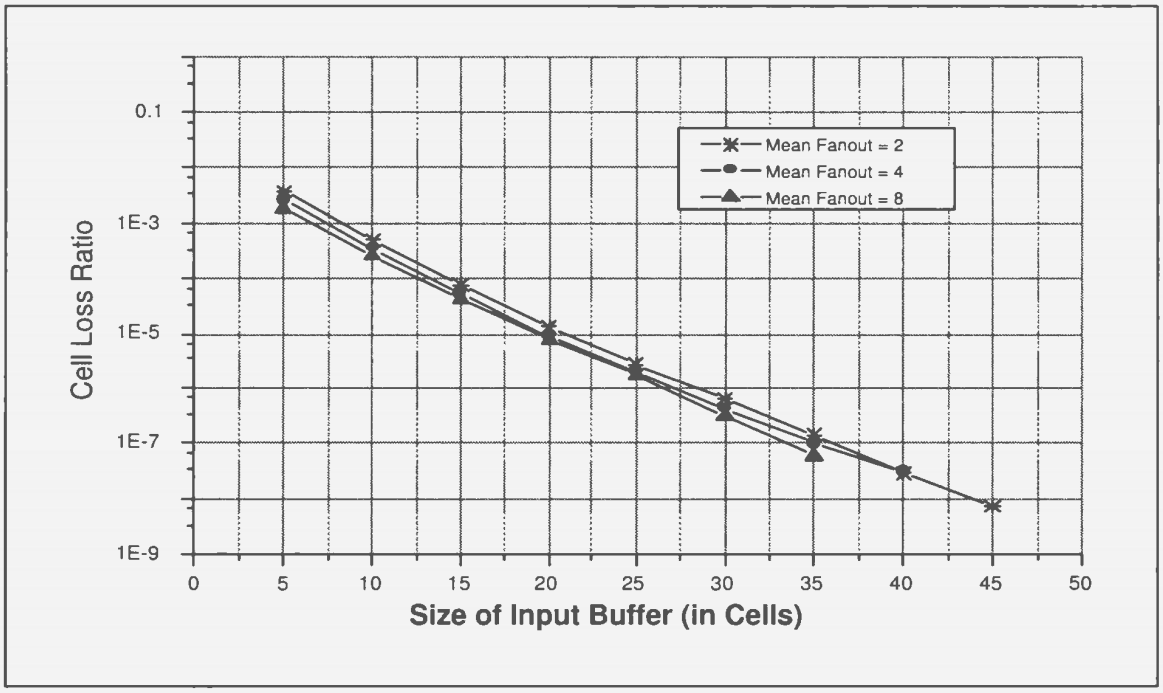


Figure 4.22: Loss performance for 128×128 BG switch under different mean fanout for 90% multicast bursty traffic

4, and 8, respectively.

The reason for traffic with larger mean fanout having a better loss performance is that traffic load is defined at the switch output and is converted to input load via the mean fanout, as shown in Equation 4.70. With the same offered load, traffic with larger mean fanout will have a lower input load. Because cell replication is performed inside the SF, the load on the interconnection links increases gradually as cells approach the output port. Before reaching the output buffer stage, the average traffic load on the link with larger fanout is always less than that from traffic with a smaller fanout. This argument is confirmed by the loss performance for different switch sizes and fanout under 90% bursty traffic, as shown in Table 4.6.

Finally, we study how loss performance is related to traffic burstiness. In Figure 4.18, the loss performance between multicast random traffic and multicast bursty

Switch Size	Size of Input Queue	Mean Fanout of Traffic		
		$F = 2$	$F = 4$	$F = 8$
32×32	5	2.2842E-03	1.4551E-03	7.4991E-04
	10	2.7354E-04	1.7094E-04	8.7222E-05
	15	3.7901E-05	2.4005E-05	1.1372E-05
	20	5.7744E-06	3.5379E-06	1.8243E-06
	25	8.4960E-07	5.8410E-07	2.5290E-07
	30	1.5840E-07	7.8300E-08	1.9800E-08
	35	2.4300E-08	1.2600E-08	5.4000E-09
64×64	5	3.0560E-03	2.0688E-03	1.2723E-03
	10	3.9234E-04	2.5935E-04	1.6865E-04
	15	5.7574E-05	3.7400E-05	2.5233E-05
	20	9.4275E-06	6.0561E-06	4.2732E-06
	25	1.8234E-06	9.6570E-07	7.0830E-07
	30	3.6810E-07	2.2050E-07	1.5840E-07
	35	4.2300E-08	2.9900E-08	1.7100E-08
128×128	5	3.7286E-03	2.6336E-03	1.7920E-03
	10	4.9397E-04	3.5038E-04	2.5865E-04
	15	7.7397E-05	5.4653E-05	4.3196E-05
	20	1.3510E-05	8.9784E-06	8.0415E-06
	25	2.6622E-06	1.9053E-06	1.7325E-06
	30	6.3630E-07	4.1310E-07	3.0600E-07
	35	1.3950E-07	9.8100E-08	5.8500E-08
256×256	5	4.3542E-03	3.1324E-03	2.2742E-03
	10	6.1061E-04	4.3498E-04	3.4542E-04
	15	1.0027E-04	7.0484E-05	5.9880E-05
	20	1.8131E-05	1.2938E-05	1.1258E-05
	25	3.6234E-06	2.6631E-06	2.4489E-06
	30	3.3309E-06	5.7780E-07	5.3460E-07
	35	8.9910E-07	1.3500E-07	1.0890E-07
512×512	5	4.9688E-03	3.6248E-03	3.3535E-03
	10	7.2809E-04	5.2532E-04	4.8828E-04
	15	1.2850E-04	8.9722E-05	7.6992E-05
	20	2.4806E-05	1.7014E-05	1.5447E-05
	25	5.1291E-06	3.5001E-06	3.3561E-06
	30	1.2366E-06	7.1550E-07	7.3620E-07
	35	1.9710E-07	2.0430E-07	1.6560E-07

Table 4.6: Loss performance for various switch sizes and different mean fanout under 90% multicast bursty traffic with average burst length of 5

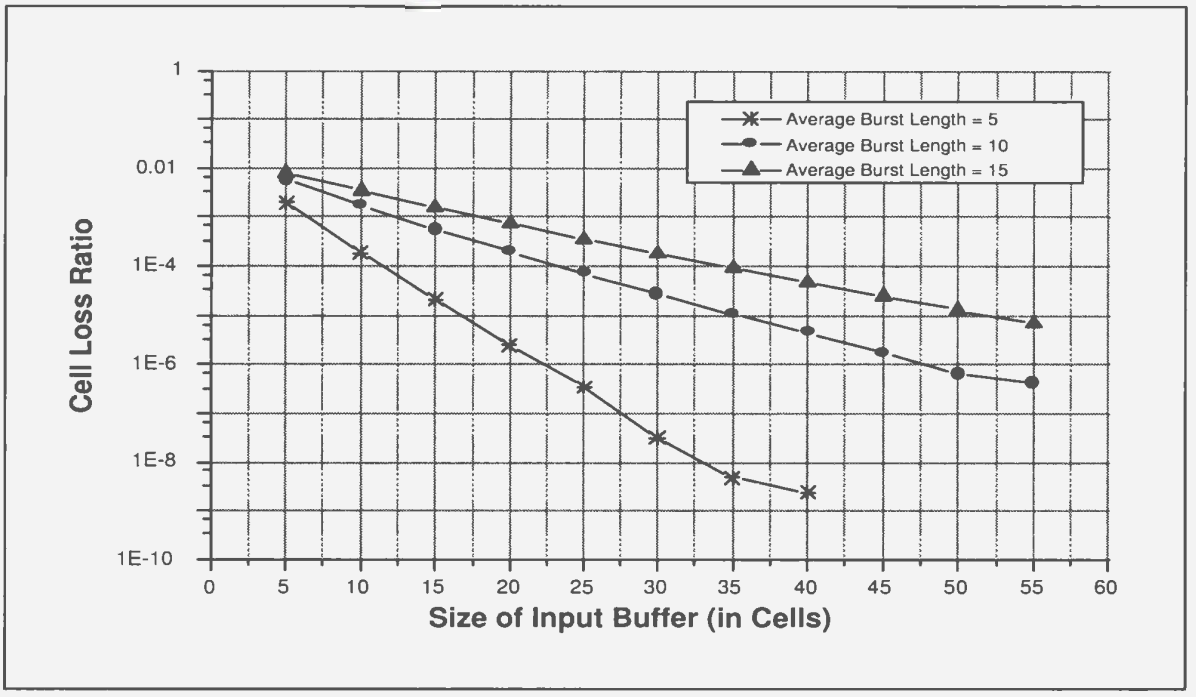
traffic with an average burst length of 5 has been compared. Random traffic can be viewed as a special case of bursty traffic in which the burst length is constant one. Table 4.7 lists the cell loss rate for the 128×128 BG switch under 90% multicast bursty traffic with the average burst length of 5, 10, and 15. A mean fanout of 2 is used. It is clear that as traffic gets burstier, the level of traffic correlation increases, and the SF internal blocking becomes larger. As a result, the demand of resource at the input queue increases in order to keep the same level of performance. From the previous example, only 6 cell spaces for each input queue are required to achieve a better than 10^{-8} loss performance in random traffic. This number increases to 45 and over 95 when the burst length becomes 5 and 10, respectively. As the burst length gets greater than 15, more than 100 cell space must be equipped in order to keep the same cell loss ratio. Similar results can be found under other load conditions, which are plotted in Figures 4.23 and 4.24. As the traffic load decreases, the required input queueing resource decreases.

4.5.2 Delay Performance and Buffer Requirement

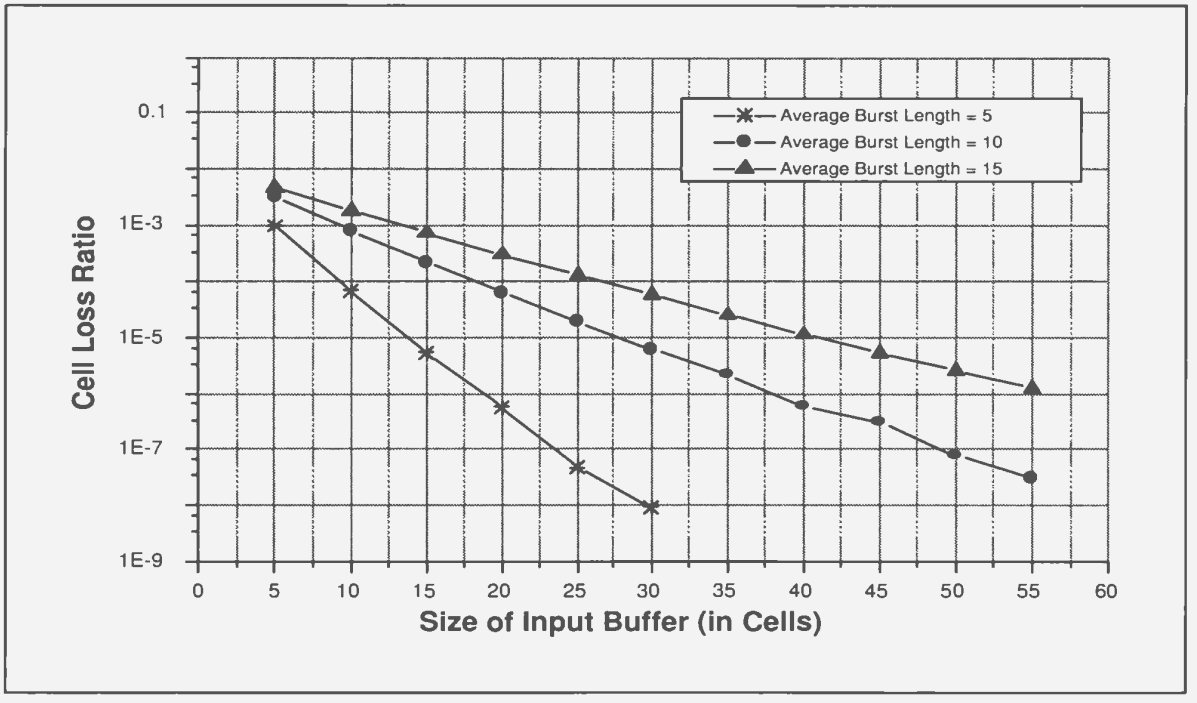
In this subsection, the delay performance of the multicast BG switch under multicast bursty traffic is studied. The delay performance breakdown for the multicast BG switch is first analyzed. Then study will focus on how the delay performance relating to various factors, including switch size, traffic load, fanout, and burstiness. Infinite input and output buffering are assumed so as to provide zero cell loss. In practice, in all simulations, a maximum value of 300 and 3000 are used for the size of input and output queue, respectively. To ensure zero cell loss, simulation results are checked afterwards to make sure that the maximum queue size is never exceeded. Throughout the study, delay performance of the multicast BG switch is compared to that of the ideal multicast switch. As stated in Section 4.4.2.1.2, the input queueing delay is

Size of Input Queue (in Cells)	Multicast Bursty Traffic ($\overline{F} = 2$)		
	Burst Length = 5	Burst Length = 10	Burst Length = 15
5	3.7286E-03	9.6149E-03	1.2852E-02
10	4.9397E-04	3.4380E-03	6.2278E-03
15	7.7397E-05	1.3317E-03	3.1927E-03
20	1.3510E-05	5.4659E-04	1.6774E-03
25	2.6622E-06	2.3537E-04	9.2090E-04
30	6.3630E-07	1.0391E-04	5.2218E-04
35	1.3950E-07	4.6551E-05	2.8922E-04
40	2.8800E-08	2.2600E-05	1.7302E-04
45	7.2000E-09	1.1093E-05	9.9399E-05
50	<1.00E-09	5.4783E-06	6.1577E-05
55	<1.00E-09	2.7990E-06	3.7040E-05
60	<1.00E-09	1.4679E-06	2.3268E-05
65	<1.00E-09	8.8110E-07	1.5152E-05
70	<1.00E-09	3.4290E-07	8.8218E-06
75	<1.00E-09	2.1510E-07	5.9238E-06
80	<1.00E-09	1.0170E-07	3.5280E-06
85	<1.00E-09	2.3400E-08	2.3949E-06
90	<1.00E-09	5.4000E-08	1.4247E-06
95	<1.00E-09	2.5200E-08	1.1475E-06

Table 4.7: Loss performance for the 128×128 BG switch under 90% multicast bursty traffic with average burst length of 5, 10, and 15

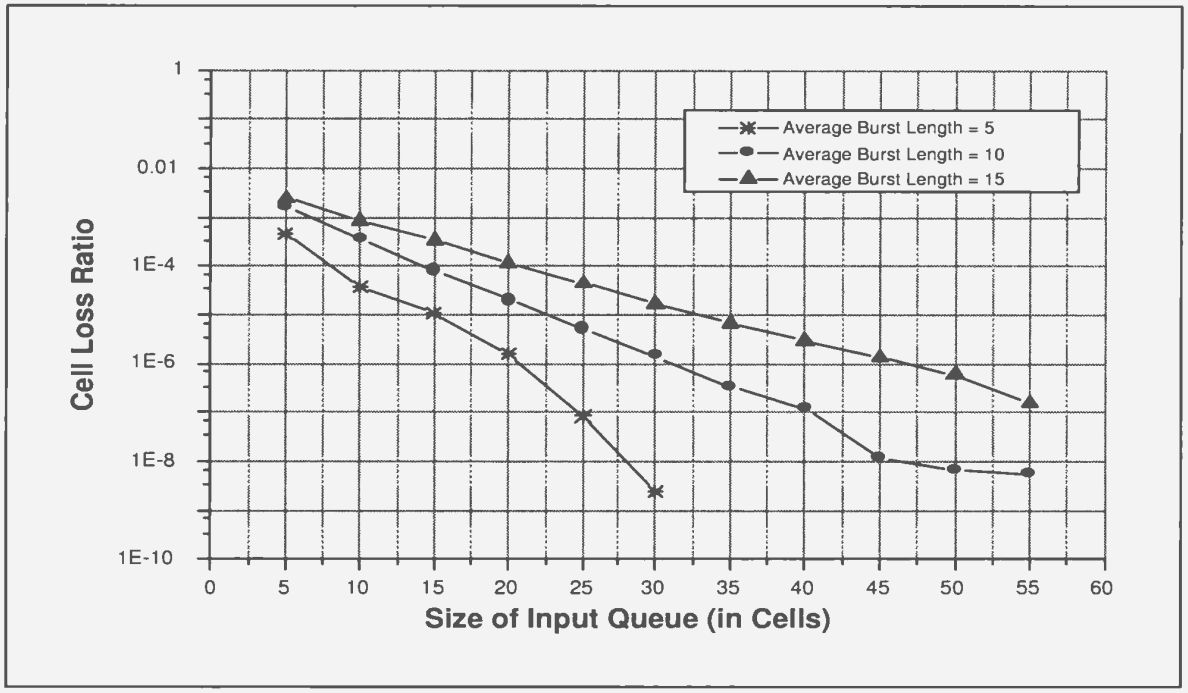


(a) 80% Multicast Bursty Traffic (Mean Fanout = 2)

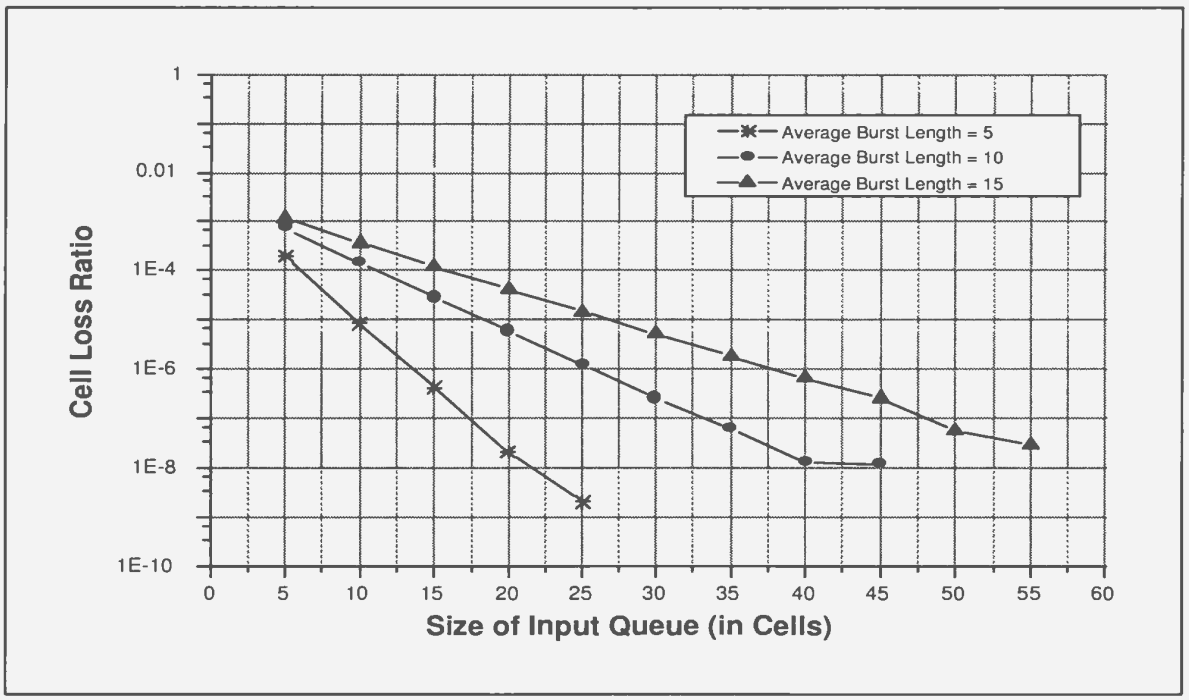


(b) 70% Multicast Bursty Traffic (Mean Fanout = 2)

Figure 4.23: Loss performance comparison for various burst lengths for 128×128 BG switch: (a) 80% multicast bursty traffic (b) 70% multicast bursty traffic



(c) 60% Multicast Bursty Traffic (Mean Fanout = 2)



(d) 50% Multicast Bursty Traffic (Mean Fanout = 2)

Figure 4.24: Loss Performance comparison for various burst lengths for 128×128 BG switch: (c) 60% multicast bursty traffic (d) 50% multicast bursty traffic

measured in terms of the master cell while the output queueing delay and total delay are calculated based on individual copies.

4.5.2.1 Delay Performance

Table 4.8 lists the delay performance breakdown for various switch sizes under multicast bursty traffic with a mean fanout of 2 and an average burst length of 5. Total delay for the ideal switch is also provided in the table for comparison. Because the ideal switch fabric can switch any incoming cell to the requested output ports in the same switching cycle, its input queueing delay is zero. Because signal propagation delay inside the SF is neglected, the output queueing delay for the ideal switch is in fact its total delay.

From the table, it is clear that the output queueing delay is the dominant part for the BG switch, and its total delay is always only slightly larger than that of the ideal switch. It is not difficult to explain this observation. The high throughput of the multicast BG SF ensures that many of the cells are transferred to the output queue within one switching cycle, thus resulting in the same delay as that in the ideal switch. Only the very few cells that are left behind due to internal blocking contribute to the input queueing delay and make the overall cell delay slightly larger than in the ideal case.

It is also observed that as the switch size grows, the total cell delay for the BG switch becomes slightly larger. At the same time, the difference between the BG switch and the ideal switch in total delay also increases. This is because the first two stages of the BG switch are strictly nonblocking, regardless of the traffic applied. Therefore, the delay performance of the 4×4 BG switch is exactly the same as that of the ideal switch. As the switch size grows, more stages are added, hence the chance of cells being blocked inside the SF increases. As a result, the total cell delay increases as

Switch Size	Traffic Load	Multicast BG Switch			Ideal Switch
		Total Delay	Input Delay	Output Delay	Total Delay
16 × 16	0.9	45.0546	0.2246	44.7828	44.7995
	0.8	20.0893	0.1483	19.9075	19.9968
	0.7	11.7416	0.0954	11.6229	11.6920
	0.6	7.5635	0.0583	7.4902	7.5381
	0.5	5.0553	0.0329	5.0135	5.0352
32 × 32	0.9	47.3710	0.3569	46.9396	47.1070
	0.8	21.1285	0.2329	20.8423	20.9897
	0.7	12.3527	0.1479	12.1688	12.2495
	0.6	7.9474	0.0900	7.8344	7.8837
	0.5	5.2947	0.0506	5.2305	5.2688
64 × 64	0.9	48.7309	0.5015	48.1258	48.4956
	0.8	21.6830	0.3225	21.2873	21.4730
	0.7	12.6915	0.2038	12.4382	12.5393
	0.6	8.1523	0.1229	7.9974	8.0624
	0.5	5.4326	0.0695	5.3443	5.3857
128 × 128	0.9	49.5026	0.6429	48.7291	48.8855
	0.8	22.0096	0.4079	21.5092	21.7178
	0.7	12.8768	0.2553	12.5586	12.6785
	0.6	8.2584	0.1529	8.0655	8.1589
	0.5	5.5095	0.0862	5.3995	5.4412
256 × 256	0.9	49.5963	0.7844	48.6546	49.0639
	0.8	22.2268	0.4927	21.6228	21.8936
	0.7	12.9635	0.3056	12.5824	12.7523
	0.6	8.3415	0.1827	8.1107	8.1962
	0.5	5.5472	0.1025	5.4164	5.4626

Table 4.8: Average delay performance breakdown for various switch sizes under 90% multicast burst traffic with mean fanout 2 and average burst length 5

the switch size grows and the difference between the BG switch and the ideal switch becomes larger.

In Table 4.9 and 4.10, the delay breakdown is presented for the 128×128 BG switch under various load, fanout and burstiness conditions. It is obvious that the change in average burst length affects the delay performance significantly while the impact of traffic mean fanout is trivial. The burst length increase means the correlation between successive cells increases because all cells belonging to the same burst go to the same destinations. Even though in the long run, destination selection is uniformly distributed, on a cycle by cycle basis, the possibility of having more cells come to the same output increases with longer bursts. Having more than two arrivals to the same output causes output queue buildup. As a result, each cell will experience longer output queueing delay. At the same time, traffic correlation will increase the chance of internal blocking because more cells are competing for the links toward the same output port, which in turn causes more blocked cells to be retained at the input queue, and thus increases the input queueing delay. Even though the internal blocking becomes worse as the traffic burstiness increases, the switching capability of the multicast BG switch ensures that most of the cells manage to reach their destinations. Therefore, although the input queueing delay increases along with the traffic burst length, it is always a small fraction of the output queueing delay. In general, average cell delay in the BG switch is very close to that of the ideal switch.

With larger mean fanout, each cell will contain more copies. It will take longer for all copies contained in the cell header to be delivered, which in turn will make cell delay in the input queue longer. But larger mean fanout also means less traffic load at the switch input which helps to reduce internal blocking, especially in the early stages. Therefore, as traffic mean fanout grows, the input queueing delay increases only slightly.

Burst Length	Mean Fanout	BG Switch			Ideal Switch
		Total Delay	Input Delay	Output Delay	Total Delay
5	2	49.5026	0.6428	48.7291	48.8855
	4	49.3910	0.7787	48.3376	48.7632
	8	49.3592	0.9459	48.0373	48.3810
10	2	94.1450	1.2324	92.5919	93.0086
	4	94.3925	1.5435	92.1378	93.9522
	8	94.6992	1.9188	91.7577	93.6238
15	2	139.1580	1.8302	136.8090	137.3470
	4	139.3400	2.3427	135.8240	136.8180
	8	138.8480	2.9285	134.1840	137.2500

Table 4.9: Average delay performance breakdown for 128×128 BG and ideal multicast switch under 90% multicast burst traffic

4.5.2.2 Input and Output Buffer Requirement

In this subsection, the buffer requirements for both BG switch and ideal switch are studied under multicast bursty traffic. As discussed earlier, the ideal multicast switch can switch any cell to its destination within the same switching cycle. The only space required for each input queue is to temporarily hold the incoming cell, which is in any case required by any switch. If this factor is excluded, the input buffer requirement for the ideal switch is always 0. Therefore, only the output buffer requirement needs to be studied for the ideal switch.

The average buffer requirement represents the mean value of cells in the buffer over long term while the maximum buffer requirement represents the largest number of cells in the queue at any given time and determines the buffer requirement in the extreme situation. Figure 4.25 plots the average and maximum input buffer requirement for BG switch, and Figure 4.26 plots the average and maximum output buffer requirement for both the BG switch and the ideal multicast switch. Multicast bursty traffic with a mean fanout of 2 and an average burst length of 5 is used. Traffic load ranges from 90% to 50%. It is noted that both measures, that is, the average and

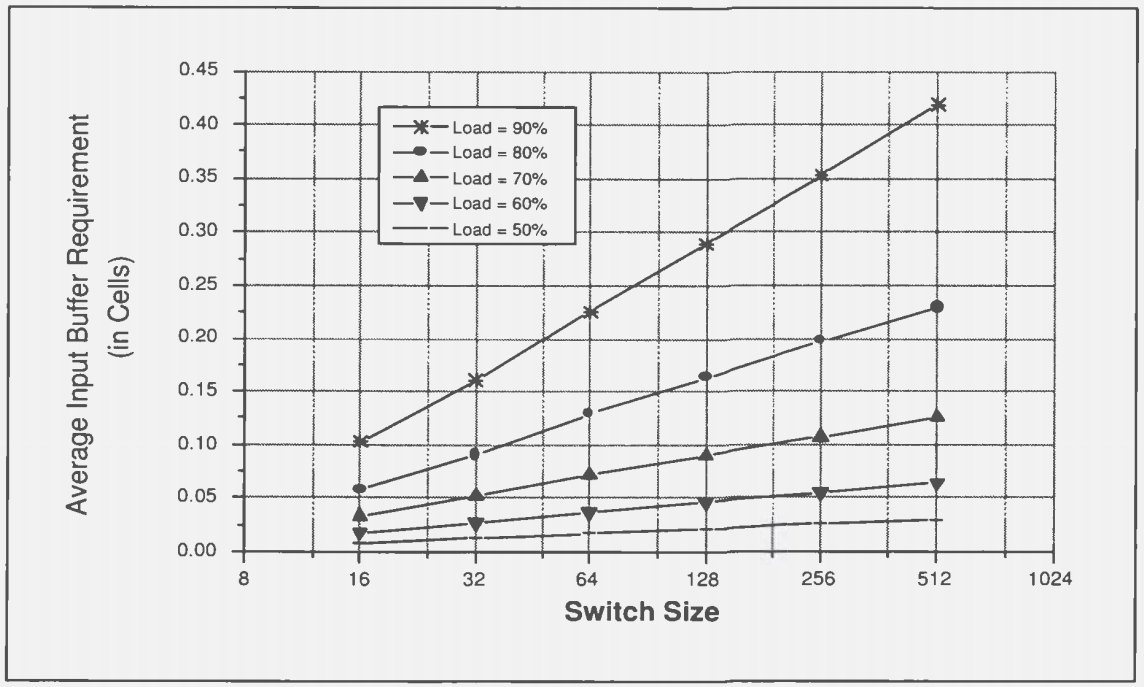
Switch Load	Burst Length	Mean Fanout	Multicast BG Switch			Ideal Switch
			Total Delay	Input Delay	Output Delay	Total Delay
80%	5	2	22.0096	0.4079	21.5092	21.7178
		4	22.1655	0.5316	21.4353	21.6526
		8	22.1372	0.6686	21.1899	21.7230
	10	2	41.9979	0.7755	40.9983	41.5858
		4	42.4465	1.0399	40.9027	41.3930
		8	42.6316	1.3319	40.5655	41.6413
	15	2	62.0604	1.1476	60.5515	61.1801
		4	62.5735	1.5590	60.1908	61.0396
		8	63.0393	2.0200	59.7879	61.1228
70%	5	2	12.8768	0.2553	12.5586	12.6785
		4	12.9386	0.3496	12.4523	12.6681
		8	12.9991	0.4556	12.3470	12.6845
	10	2	24.5935	0.4827	23.9607	24.1831
		4	24.8425	0.6771	23.8280	24.1267
		8	24.9888	0.8959	23.5829	24.2059
	15	2	36.2734	0.7118	35.3213	35.7773
		4	36.5805	1.0117	35.0170	35.8290
		8	36.9483	1.3477	34.7565	35.7772
60%	5	2	8.2583	0.1529	8.0654	8.1589
		4	8.3346	0.2190	8.0266	8.1725
		8	8.3591	0.2929	7.9364	8.1523
	10	2	15.7936	0.2895	15.4089	15.5640
		4	15.9215	0.4192	15.2860	15.5689
		8	16.0345	0.5686	15.1384	15.5048
	15	2	23.3139	0.4267	22.7350	22.9836
		4	23.5731	0.6225	22.6040	22.8763
		8	23.7710	0.8533	22.3776	22.9937
50%	5	2	5.5094	0.0862	5.3995	5.4412
		4	5.5424	0.1273	5.3622	5.4402
		8	5.5826	0.1749	5.3277	5.4457
	10	2	10.5195	0.1619	10.3015	10.3879
		4	10.6214	0.2434	10.2500	10.4054
		8	10.6943	0.3363	10.1584	10.3929
	15	2	15.5302	0.2387	15.2034	15.3056
		4	15.6948	0.3605	15.1284	15.2998
		8	15.7973	0.5013	14.9754	15.3116

Table 4.10: Average delay performance breakdown for 128 × 128 BG and ideal multicast switch under various multicast burst traffic load (80% to 50%)

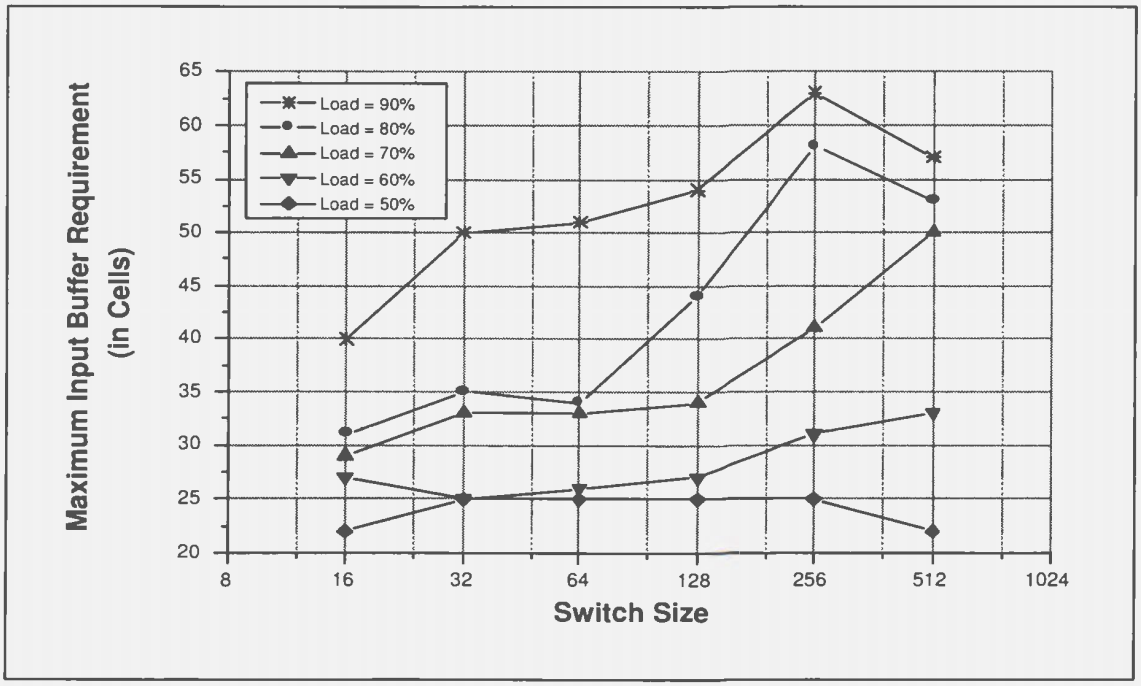
maximum buffer requirement, scale well across different switch sizes under all load conditions, especially for the output queue. In other words, the buffer requirement to achieve the same level of performance does not increase too much as the switch size grows. For the input queue, a large switch needs more stages which increases the chance of cell being blocked. However, the input buffer requirement increases only slightly. This is a desired feature of the BG switch. With this scalability feature, large switches can be easily constructed using smaller switch modules without introducing much additional hardware cost for the input and output queues.

It is also observed that in all different cases, the input buffer requirement is always a very small portion to that of the output buffer. The reason is the same as what have been concluded during the loss and delay performance analysis: the high throughput of the BG switch fabric ensures that most HOL cells are switched to their destinations within one switching cycle. Only very few cells are left behind due to the internal blocking. Hence, a small amount of buffering resource at the input port is sufficient to store the blocked cells.

Tables 4.11 and 4.12 compares the buffer requirement performance for the 128×128 BG switch and the ideal switch under various multicast bursty traffic scenarios. From the table, it is clear that the impact from traffic burstiness is apparent while that from traffic fanout is trivial. For example, under 80% traffic load with a mean fanout of 2, as the burst length increases from 5 to 10 to 15, the average input buffer requirement changes from 0.16 to 0.31 to 0.47, and the average output buffer requirement increases from 17.21 to 32.79 to 48.42. However, under the same traffic load and burst length, as the fanout changes from 2 to 4 to 8, the input buffer requirement decreases only slightly, i.e., from 0.16 to 0.11 to 0.067, while the output buffer requirement remains almost unchanged, that is, from 17.21 to 17.15 to 16.95. This is because traffic load is defined at the switch output. No matter how big the mean fanout is, as cells reach the

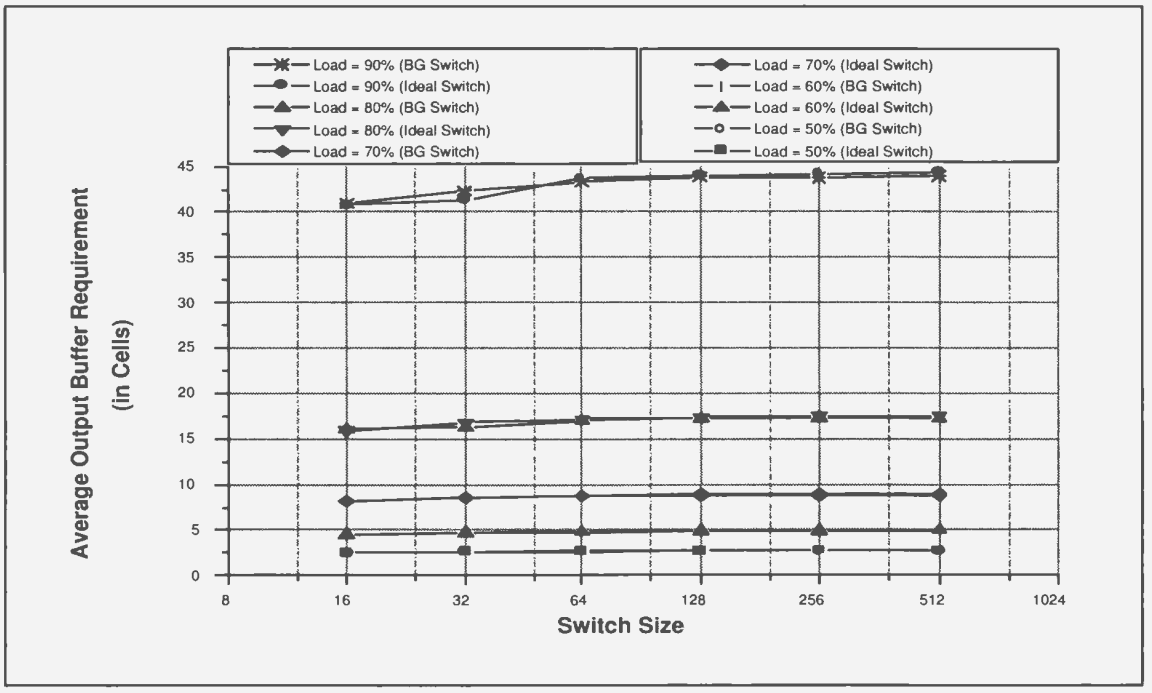


(a) Average Input Buffer Requirement under Multicast Bursty Traffic
(Mean Fanout = 2, Average Burst Length = 5)

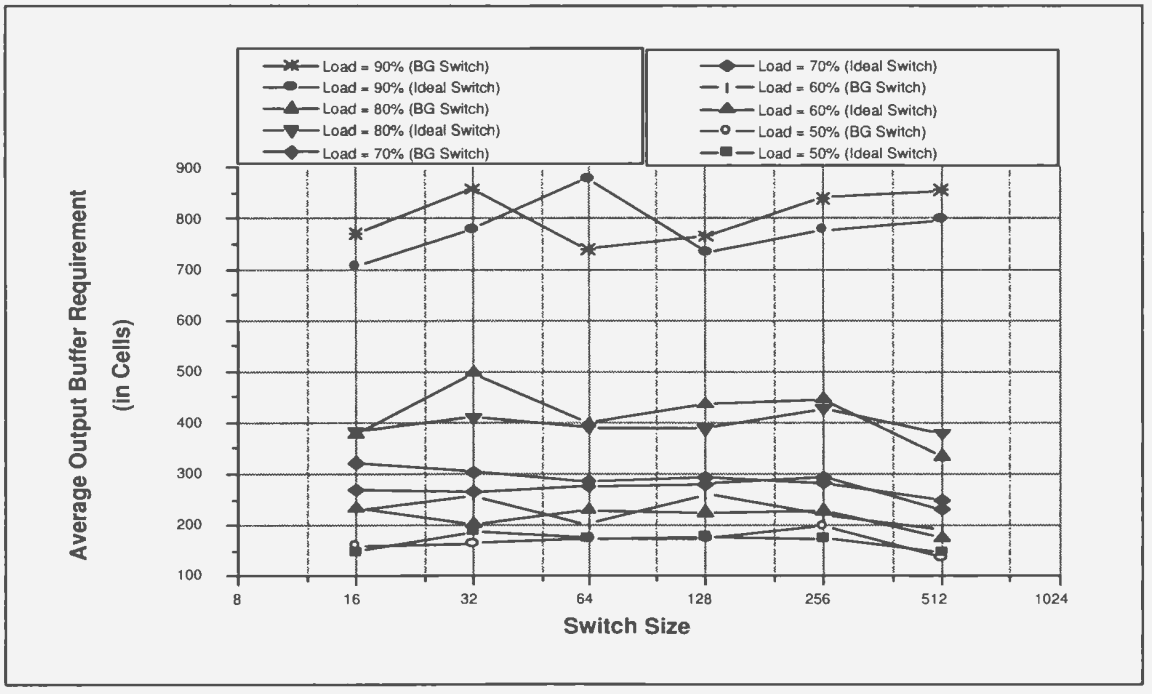


(b) Maximum Input Buffer Requirement under Multicast Bursty Traffic
(Mean Fanout = 2, Average Burst Length = 5)

Figure 4.25: Average and maximum input buffer requirement for various switch sizes and load conditions under multicast bursty traffic (mean fanout = 2, average burst length = 5)



(a) Average Output Buffer Requirement under Multicast Bursty Traffic
(Mean Fanout = 2, Average Burst Length = 5)



(b) Maximum Output Buffer Requirement under Multicast Bursty Traffic
(Mean Fanout = 2, Average Burst Length = 5)

Figure 4.26: Average and maximum output buffer requirement for various switch sizes and load conditions under multicast bursty traffic (mean fanout = 2, average burst length = 5)

Traffic Load	Mean Fanout	Average Burstiness	Multicast BG Switch				Ideal Multicast Switch	
			Input Buffer Requirement		Output Buffer Requirement		Output Buffer Requirement	
			Average	Maximum	Average	Maximum	Average	Maximum
90%	2	5	0.2893	54	43.8591	766	43.9964	733
		10	0.5546	92	83.3202	1514	83.6861	1325
		15	0.8236	143	123.0310	1871	123.5960	2009
	4	5	0.1752	48	43.5036	939	43.8797	798
		10	0.3472	93	82.8914	1352	84.6197	1636
		15	0.5271	149	122.2150	2328	123.0440	2252
	8	5	0.1064	53	43.2200	745	43.5133	777
		10	0.2159	99	82.5649	1548	84.3052	1556
		15	0.3293	144	120.4760	2070	120.7540	1902
80%	2	5	0.1632	44	17.2059	436	17.3735	389
		10	0.3101	88	32.7888	729	33.2716	705
		15	0.4589	110	48.4151	1169	48.9358	1124
	4	5	0.1063	48	17.1505	468	17.3107	446
		10	0.2080	98	32.7348	771	33.1043	771
		15	0.3119	124	48.1453	1190	48.8103	1088
	8	5	0.0668	42	16.9501	401	17.3833	411
		10	0.1332	88	32.4439	700	33.3354	892
		15	0.2020	131	47.8576	1081	48.9009	1217
70%	2	5	0.0893	34	8.7907	294	8.8727	280
		10	0.1689	71	16.7714	556	16.9217	538
		15	0.2491	97	24.7172	771	25.0473	728
	4	5	0.0612	37	8.7153	313	8.8630	317
		10	0.1186	65	16.6851	558	16.8800	576
		15	0.1771	133	24.5063	793	25.0890	747
	8	5	0.0398	42	8.6411	318	8.8790	295
		10	0.0784	73	16.5168	572	16.9502	541
		15	0.1179	102	24.3440	746	25.0463	733

Table 4.11: Average and maximum buffer requirement for the 128 × 128 BG and ideal multicast switch under various multicast bursty traffic

Traffic Load	Mean Fanout	Average Burstiness	Multicast BG Switch				Ideal Multicast Switch	
			Input Buffer Requirement		Output Buffer Requirement		Output Buffer Requirement	
			Average	Maximum	Average	Maximum	Average	Maximum
60%	2	5	0.0458	37	4.8377	260	4.8955	225
		10	0.0868	58	9.2431	372	9.3377	508
		15	0.1279	77	13.6361	608	13.7875	611
	4	5	0.0328	32	4.8154	221	4.9063	251
		10	0.0628	60	9.1686	398	9.3397	392
		15	0.0933	82	13.5590	610	13.7134	549
	8	5	0.0219	32	4.7596	219	4.8922	224
		10	0.0426	63	9.0793	398	9.2925	406
		15	0.0639	114	13.4281	650	13.7886	574
50%	2	5	0.0215	25	2.6997	173	2.7203	178
		10	0.0404	48	5.1498	324	5.1942	359
		15	0.0597	62	7.6003	551	7.6493	430
	4	5	0.0159	27	2.6805	163	2.7201	169
		10	0.0304	53	5.1282	333	5.2055	330
		15	0.0451	71	7.5651	464	7.6459	512
	8	5	0.0109	34	2.6648	179	2.7242	189
		10	0.0210	49	5.0807	318	5.1987	330
		15	0.0313	81	7.4877	453	7.6556	478

Table 4.12: Average and maximum buffer requirement for the 128 × 128 BG and ideal multicast switch under various multicast bursty traffic (continued)

output, all copies are replicated. Hence, to the output port, the loads for different fanout conditions are very close and this results in the output buffer requirements being roughly the same. For the input queue, larger fanout means more copies are contained in a single master cell and load on input links becomes less heavy. With the same switching capability from the fabric, the input buffer requirement becomes less.

It is also noticed that the average output buffer requirement for the BG switch is always slightly less than that for the ideal switch. For example, under 80% load with a mean fanout of 2, when traffic burst length is 5, 10, and 15, the average output buffer requirement is 17.21, 32.79, and 48.42, respectively, for the BG switch, and it is 17.37, 33.27, and 48.94, respectively, for the ideal switch. This implies the number of cells switched by the BG switch is only slightly less than that of the ideal switch.

From the analysis under multicast bursty traffic, it has been observed that the performance of the multicast BG switch is very close to the ideal switch. The point is, when switch size becomes large, the ideal switch is too costly to build because of the high hardware complexity and cost from the switch fabric and the high-speed memory for the output queue. While the BG switch architecture can not only achieve a performance close to optimum, but also with a reasonably low demand for hardware and memory speed.

4.6 Performance Analysis under Nonuniform Multicast Traffic

For all previous discussion, it has been assumed that all output ports are randomly selected by the incoming traffic. However, this is not always the case in the real communications networks. Very often, the destinations are selected in a nonuniform fashion. In this section, the performance of the multicast BG switch is investigated

under nonuniform multicast traffic. A modified form of the model presented in [14, 73] is used for nonuniform destination generation. The ON-OFF model and the truncated geometric distribution are used for traffic arrival process and multicast cell fanout distribution, respectively.

4.6.1 Nonuniform Traffic Model

In [73], a nonuniform traffic model is presented for the analysis of a non-blocking packet switch under unicast traffic. In this model, the output address at each input is assumed to be assigned by a binomial distribution function (except at inputs 0 and $N-1$). For input i, where $1 \leq i \leq N-2$, define

$$\begin{aligned} t_{i,j} &= \Pr[\text{Cell arriving at the } i^{\text{th}} \text{ input is destined to the } j^{\text{th}} \text{ output}] \\ &= \binom{N-1}{j} \cdot r_i^j \cdot (1-r_i)^{N-j-1}, \quad 0 \leq j \leq N-1, \end{aligned} \tag{4.71}$$

where r_i is the probability associated with input i. For a binomial distribution, the maximum probability always occurs at $j = \lfloor N \cdot r_i \rfloor$ [73]. If the maximum output is defined as the one which carries more traffic from input i than any other output, given output $N-1-i$ to be the maximum output for any input i, r_i is given by

$$r_i = \frac{N-1-i}{N-1}. \tag{4.72}$$

For input 0 and $N-1$, the probability toward output j, where $0 \leq j \leq N-1$, is given by a normalized Poisson-like distribution at rate r as below [14]:

$$t_{0,j} = \frac{r^{N-1-j}/(N-1-j)!}{\sum_{\forall j} r^{N-1-j}/(N-1-j)!}, \tag{4.73}$$

and

$$t_{N-1,j} = \frac{r^j/j!}{\sum_{\forall j} r^j/j!}. \tag{4.74}$$

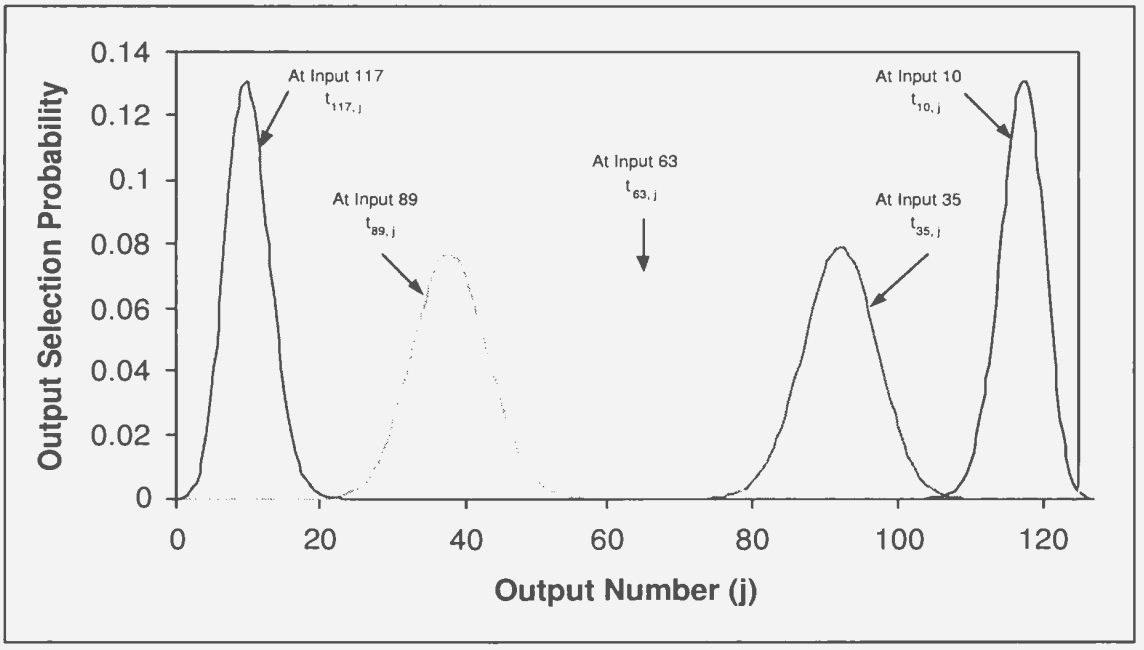


Figure 4.27: Output selection probability at inputs 10, 35, 63, 80, 117

The benefit of using this model is that, instead of providing only one or two hot destinations, it gives a substantial number of hot spots, which is more realistic when the switch size becomes very large. Figure 4.27 shows the distribution function of the output for input 10, 35, 63, 80, and 117 respectively. The traffic from each input goes mainly to 20 outputs, except for inputs 0 and 127 [73]. In [14], it has been noticed that there are a number of problems with this model: the first and last input was dealt with differently through another distribution, the value of r_i was different for different inputs which results in different distributions, and the input-output paring was restricted to a fixed pattern.

To solve the above problems, the model has been modified in [14]: using a fixed value for all r_i, the same binomial distribution can be used by all the inputs for their output selection. At the beginning of each simulation, the fixed value is determined and each input is associated with an output port by permutation. For each input, the

associated output is located at the center of the binomial distribution. These modifications help to solve the above problems and add another advantage of randomizing the output selection.

In this dissertation, the modified model is further extended to multicast environment: similar to the unicast case, at the beginning of each simulation, all the input and output ports are associated by permutation. For each input, the associated output is located at the center of the binomial distribution. For each multicast cell that appears at the input port, its fanout is calculated based on the truncated geometric distribution, as given by Equations 4.2 and 4.3. Each copy of the multicast cell selects its destination based on the binomial distribution for that input port. All destinations belonging to the same multicast cell are distinct from each other. In the case of multicast bursty traffic, cells belonging to the same burst will request the same outputs. Using this nonuniform multicast traffic model, the requested destinations from each multicast cell are more concentrated on a band of output ports instead of distributing across all output ports as in the uniform situation.

4.6.2 Loss Performance

In this subsection, the loss performance of the multicast BG switch under the above nonuniform traffic is examined. Figure 4.28 depicts the cell loss ratio of the 128 × 128 BG switch under multicast bursty nonuniform traffic with a mean fanout of 2 and an average burst length of 5. For comparison, the loss performance of the traffic's uniform counterparts, which have random and bursty arrivals, are plotted in the same figure. A very slight performance enhancement is observed when the non-uniform traffic is applied. This result is consistent with the observation made in [14], in which the unicast nonuniform traffic is studied. This is because with nonuniform traffic, the chances of having multiple input ports requesting one single

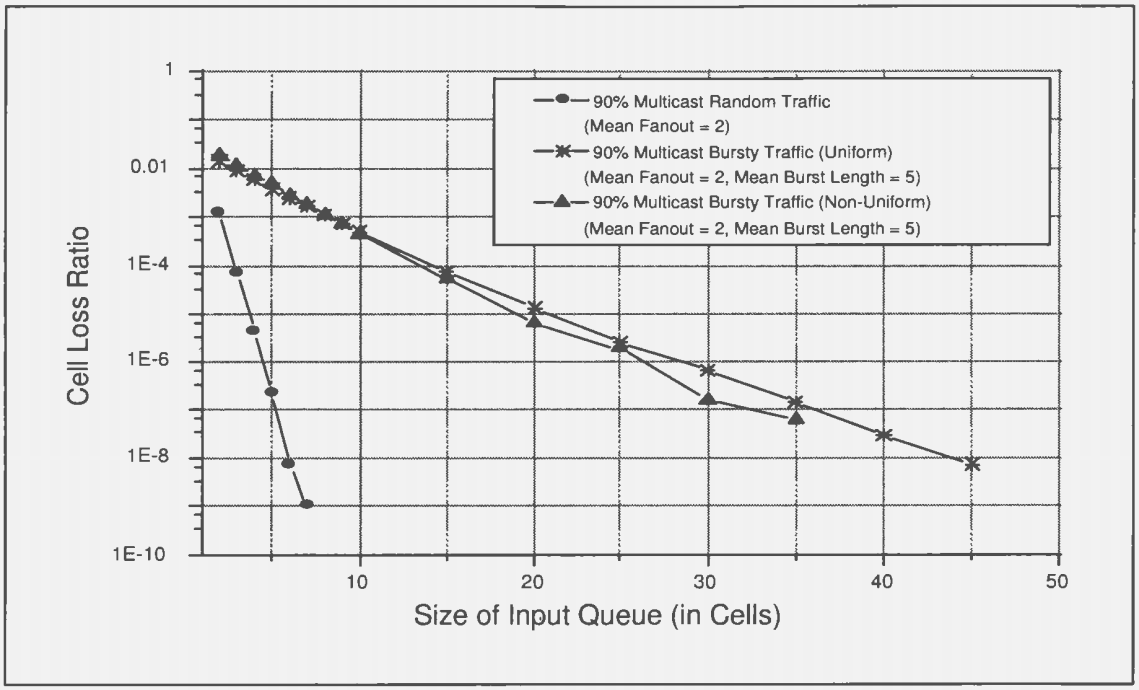


Figure 4.28: Loss performance comparison between 128×128 BG switch under 90% multicast traffic

output port are mitigated since each input port is now targeting a different band of output ports. The non-repeated destination request for all copies of a multicast cell indicates that cell replication should be performed somewhere inside the SF. Under the nonuniform traffic, incoming cells become more concentrated on a band of output ports. Therefore, cell replication only needs to be performed in the last few stages close to the output port. With fewer cells travelling across the SF, chances of internal link contention are lessened. Similar enhancement in loss performance is also observed through our simulation for non-bursty and unicast traffic conditions.

The reasons that account for the enhancement can be explained through the study of the two extreme cases of the band of output ports. When the band of the targeted output ports for each input port becomes more concentrated, incoming cells are focused on fewer output ports. In the extreme case, the targeted band narrows down

to a single output port, which is the center of the binomial distribution. Because each input port is randomly and non-repeatedly associated with an output port, in this extreme case, the traffic is moving toward permutation traffic except that the input-output match pattern is now fixed. On the other hand, when the output port band expands, the targeted output ports for each input port becomes wider. The extreme case now becomes that the targeted band from each input port expands to all output ports of the switch with equal probability to be addressed. In this case, the traffic destination request essentially becomes uniformly distributed.

Therefore, under the simple unicast and non-bursty traffic, the non-repeated input-output port mapping and the nonuniform traffic model make switch performance fit in between that of the permutation traffic and the uniform random traffic. BG and many other switches perform better under the permutation traffic than under the uniform random traffic, thus performance enhancement is observed when nonuniform traffic is applied.

The performance is also investigated under the nonuniform traffic in which each input is randomly mapped to an output at the beginning, which is the center of the output band. In this case, it is possible that multiple input ports are associated with the same output port. It has been noticed that the chances of buffer overflows significantly increases. This is because each input port is now uniformly associated with all output ports. The probability that some output ports are more heavily associated with two or more input ports increases. Therefore, those output ports are likely to receive higher traffic load which leads to higher chances of having output buffer overflow. To avoid such situation, the aggregated traffic load for those output ports should be controlled below the link consumption rate. Similarly, one may consider the two extreme cases. The worst case is when the output band narrows down to one port, in which all traffic from one input port will go to just one output

Uniform Traffic Condition				
Switch Size	Multicast BG Switch			Ideal Switch
	Input Delay	Output Delay	Total Delay	Total Delay
32 × 32	0.3541	46.9875	47.4161	47.1070
64 × 64	0.4882	48.0422	48.6329	48.4956
128 × 128	0.6161	48.4850	49.2311	48.8855
256 × 256	0.7447	48.7435	49.6442	49.0639
512 × 512	0.9306	48.8573	49.9712	49.2817
Nonuniform Traffic Condition				
Switch Size	Multicast BG Switch			Ideal Switch
	Input Delay	Output Delay	Total Delay	Total Delay
32 × 32	0.1745	42.4560	42.6672	42.4232
64 × 64	0.2588	44.3934	44.7039	44.5335
128 × 128	0.3438	45.8080	46.2171	45.9604
256 × 256	0.3996	46.8250	47.3025	46.9557
512 × 512	0.4650	47.2824	47.8384	47.7428

Table 4.13: Average delay performance breakdown and comparison between BG and ideal multicast switch under uniform and nonuniform multicast bursty traffic (mean fanout = 2, average burst length = 5)

port. In this case, the maximum possible load on the output port is equal to the load on the input link times the number of input ports that are associated with the output port. The best case is when the destination request from each input port is equally distributed to all output ports, which makes the traffic resemble uniform.

4.6.3 Delay and Buffer Requirement Performance

Table 4.13 compares the average cell delay breakdown between BG and ideal switch of various sizes under 90% uniform and nonuniform multicast bursty traffic. In Tables 4.14 and 4.15, the average and maximum buffer requirement for both switches are compared under the same traffic conditions. It is clear that there is an enhancement in both delay and buffer requirement when the nonuniform traffic is used. In both cases, input queueing delay and buffer requirement of the BG switch is always a

Uniform Traffic Condition			
Switch Size	Multicast BG Switch		Ideal Switch
	Input Buffer	Output Buffer	Output Buffer
32 × 32	0.1594	42.2806	42.3830
64 × 64	0.2197	43.2255	43.6503
128 × 128	0.2772	43.6264	43.9964
256 × 256	0.3351	43.8649	44.1490
512 × 512	0.4188	43.9661	44.3426
Nonuniform Traffic Condition			
Switch Size	Multicast BG Switch		Ideal Switch
	Input Buffer	Output Buffer	Output Buffer
32 × 32	0.0785	38.2053	38.1726
64 × 64	0.1165	39.9529	40.0753
128 × 128	0.1547	41.2254	41.3606
256 × 256	0.1798	42.1434	42.2584
512 × 512	0.2092	42.5450	42.9655

Table 4.14: Average buffer requirement comparison between BG and ideal multicast switch under uniform and nonuniform multicast bursty traffic (mean fanout = 2, average burst length = 5)

small portion of that from the output queue, which is very close to that from the ideal switch. The comparison also demonstrates the performance scalability of the multicast BG switch under nonuniform traffic, which is consistent with our previous analysis and is a very important advantage of the BG switch fabric design.

4.7 Performance Comparison with Other Switches

In Section 2.3.3, the pros and cons of the two approaches to construct a multicast switch have been discussed. In this section, the two approaches are re-visited with a focus on performance related issues. Then, the delay and loss performance of the multicast BG switch are compared with two high-performance switches published in the literature, the Abacus switch [2, 60] and the PINIUM switch [9]. Multicast cell replication is handled implicitly by both switches. As described in Sections 2.3.2.5 and

Uniform Traffic Condition			
Switch Size	Multicast BG Switch		Ideal Switch
	Input Buffer	Output Buffer	Output Buffer
32 × 32	41	855	780
64 × 64	45	902	879
128 × 128	47	822	733
256 × 256	48	877	777
512 × 512	57	856	853
Nonuniform Traffic Condition			
Switch Size	Multicast BG Switch		Ideal Switch
	Input Buffer	Output Buffer	Output Buffer
32 × 32	32	693	666
64 × 64	42	704	721
128 × 128	43	704	820
256 × 256	45	779	763
512 × 512	39	758	735

Table 4.15: Maximum buffer requirement comparison between BG and ideal multicast switch under uniform and nonuniform multicast bursty traffic (mean fanout = 2, average burst length = 5)

2.3.2.6, the Abacus switch is an input-output buffered switch, which is very similar to BG switch, while the PINIUM switch is a purely output-buffered switch.

Through the performance analysis of the BG switch, it has been noticed that with the same offered load, when pure unicast traffic is applied, all stages experience the same high load. With increasing multicast nature of the traffic, the route-and-replicate approach gives better performance. However, in the cascade approach, all stages of the routing network always face the high load. Apart from this, there is the possibility of blocking and performance degradation in the copy network. Therefore, from the switch performance point of view, multicast switches using the integrated approach are superior to those following the cascade approach. In addition to this, the copy network is normally large, expensive, and non-scalable, which make the design of a multicast switch more complicated.

4.7.1 PINIUM Switch

The basic architecture of the PINIUM switch consists of a distribution section and a concentration section, as shown in Figure 2.15. The distribution section provides the routing and multicasting functions and is made up of a stack of multicast radix-r trees. The concentration section uses the knockout principle and is made up of a row of N-to-L priority concentrating sorters.

The knockout parameter L, used in the multicast switch to describe the concentrators, decides how many cells are accepted by the output buffer in each output line in a given cycle. When $L = 4$, this situation is somewhat similar to the four-cell acceptance at each output line of the BG switch. Clearly, if a larger value was used, better performance would be obtained, but the hardware complexity of the output buffers would also increase correspondingly, and the buffer speed becomes a constraint. As the PINIUM switch does not have any input buffers, much larger values of L are required. It has been found that L should be above 8 so that cell loss rates of better than 1×10^{-6} can be obtained, and a figure of 16 is recommended. As the BG switch employs input buffers, the output buffers can be much simpler than those in the PINIUM switch.

Due to the above, as well as to other architectural differences between the PINIUM switch and the BG switch, comparing their performances is not straightforward [78]. For example, a 64×64 PINIUM switch with $L = 16$ gives a cell loss probability of better than 1×10^{-6} under 85% uniform random traffic when 37 output buffers are provided. When all other variables remain the same, providing 50 or more output buffers virtually eliminates cell loss. The 64×64 BG network under 85% uniform random traffic requires 39 output buffers as well as 5 input buffers to ensure that there is virtually no cell loss. These figures indicate that the two switches have

similar performance under random traffic with similar buffer sizes. However, the BG switch is less complex. A knockout factor of 16 in the PINIUM switch would require output buffers that operate at sixteen times the speed of the link. Given that a 64×64 PINIUM switch and the 155.52 Mbps OC-3 link are used, the required minimum memory speed is 2.48 Gbps. As switch sizes become larger and link speeds grow higher, it is not practical to build the output queue for such a switch using the current memory storage technologies.

4.7.2 Abacus Switch

The Abacus switch is modified from the purely output-buffered MOBAS switch [59]. Input buffers are equipped to temporarily store cells that have lost contention inside the multicast grouping network (MGN). A total of K feedback lines are added. Each feedback line is actually a broadcast bus, which is used to report blocking messages to all IPCs. Multicast translation tables (MTTs) are used between the MGN and small switch modules (SSMs) to generate the routing and replication tag that will be used for switching inside SSMs for cells that managed to depart from the MGN. The architecture of the Abacus switch is shown in Figure 2.14.

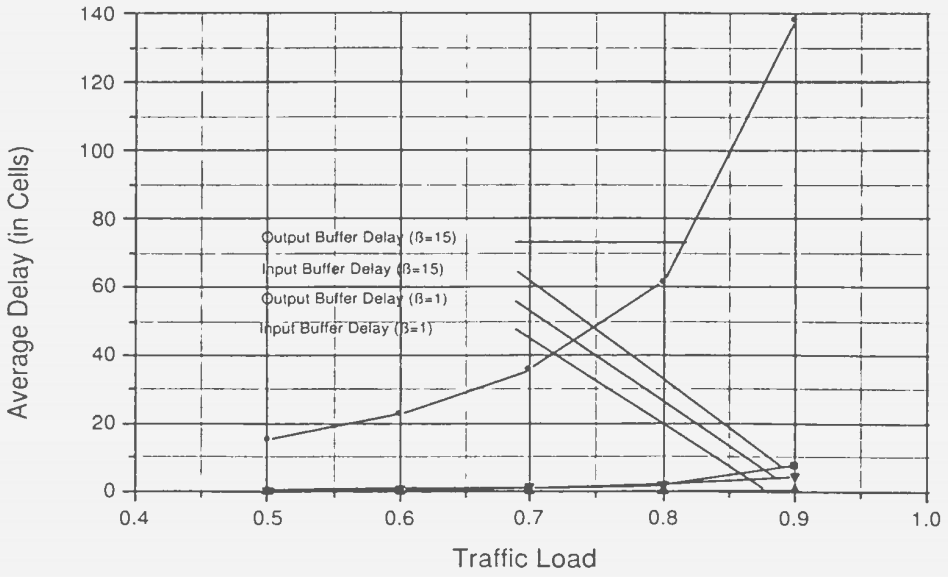
In the Abacus switch, cell replication is achieved by broadcasting incoming cells to all routing modules (RMs) of the MGN. The group expansion ratio L and the group size M can be engineered to meet the performance requirement. In Figure 4.29, the average input and output queueing delay is compared for 256×256 BG switch and Abacus switch under both unicast uniform random traffic and bursty traffic with an average burst length of 15. The results for the Abacus switch are from [2, 60]. It has been observed that the input queueing delay is almost negligible when compared to output queueing delay for both switches. Both switches perform much better under random traffic than bursty traffic. The Abacus switch performs only slightly better

than the BG switch. For example, under 70% bursty traffic, the output queueing delay for Abacus switch is around 35 switching cycles while that for the BG switch is around 36. Under 80% bursty traffic, the output queueing delays are 58 and 62 respectively for the two switches.

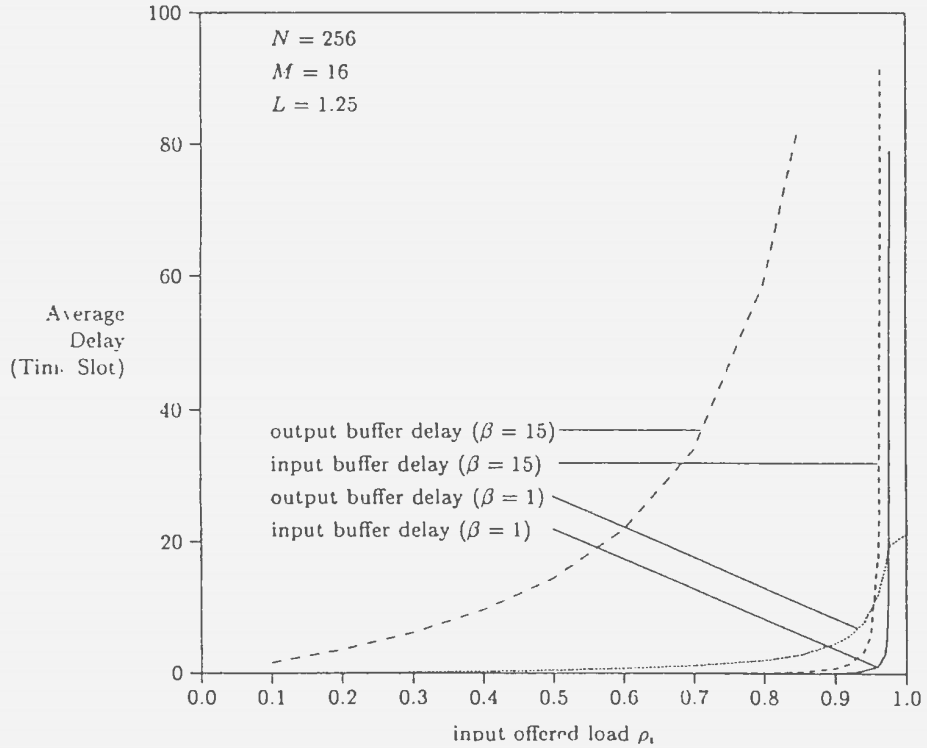
In Figure 4.30, the input queueing delay for unicast, multicast, random, and bursty traffic conditions are compared. The trends are consistent for both switches. Under the same offered load, both switches perform better under multicast traffic than unicast traffic. Again, the Abacus switch performs slightly better than the BG switch. For example, under 80% unicast and multicast bursty traffic, the average input buffer delays are around 0.6 and 0.5 switching cycles for the Abacus switch, while those numbers become 1.7 and 1.4 for the BG switch.

In Figure 4.31, input buffer overflow probability in terms of input buffer size is measured under 90% unicast bursty traffic with an average burst length of 1, 10, and 15, respectively. Although the amount of input buffering resource required by the BG switch is always only a small fraction of the output buffering, it is higher than that for Abacus switch. This implies that the internal blocking for the BG switch is slightly higher than that for the Abacus switch, especially when highly bursty traffic is used.

Through the above comparison and discussion, it seems that the BG switch is inferior to the Abacus switch in performance. However, it has been noticed that there is a big difference in hardware to construct the two switches. For the 256×256 Abacus switch, group size $M = 16$ and expansion ratio $L = 1.25$ is used. That means there are a total of 16 RMs, MTTs, and SMMs used to construct the switch. Each RM is a $256 : 20$ knockout switch and each SMM is a $20 : 16$ knockout switch. The hardware complexity is estimated in terms of the number of crosspoints for a switch. Then the complexity for the Abacus switch is $16 \times 256 \times 20 + 16 \times 20 \times 16 = 87,040$, which is

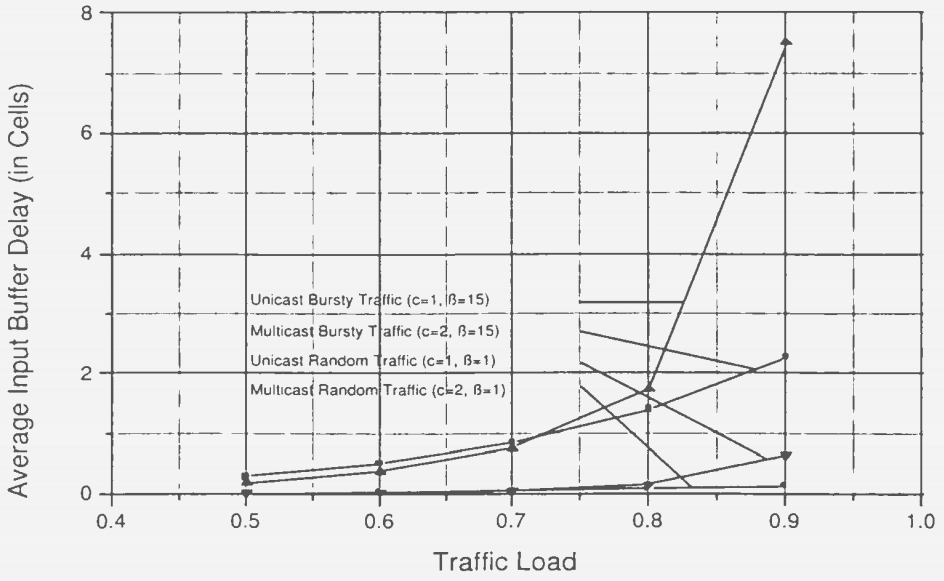


(a) Comparison of average input buffer delay and average output buffer delay of BG Switch under unicast random and bursty traffic

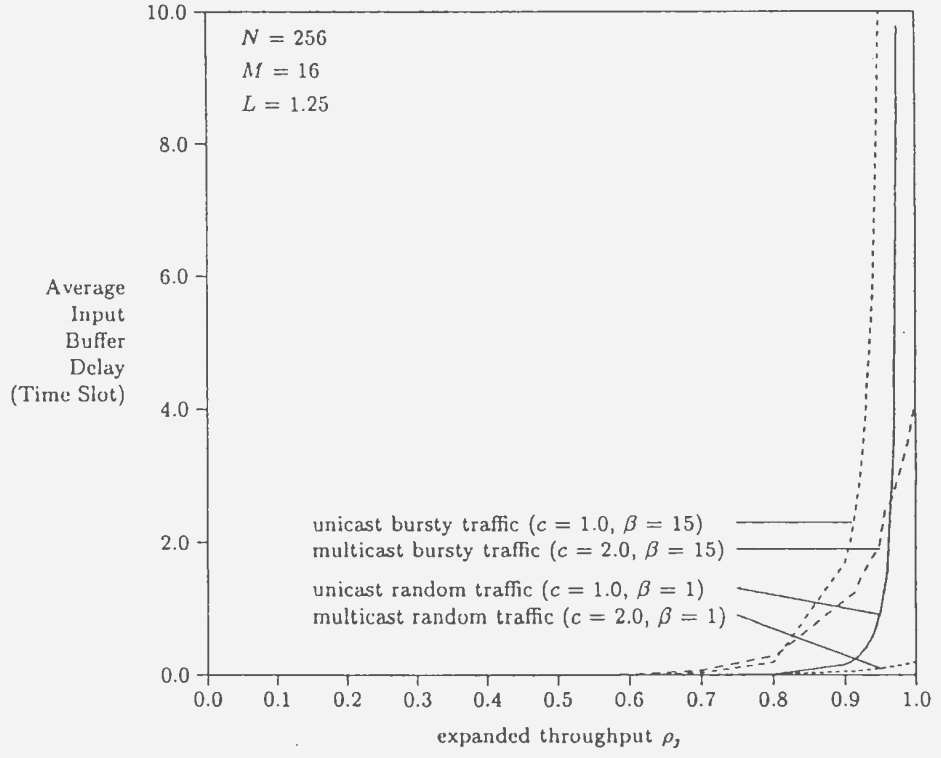


(b) Comparison of average input buffer delay and average output buffer delay of Abacus Switch under unicast random and bursty traffic (taken from [2], pp. 204)

Figure 4.29: Average input and output queueing delay comparison between multicast BG switch and Abacus switch



(a) Average Input Buffer Delay vs. Traffic Load for BG Switch



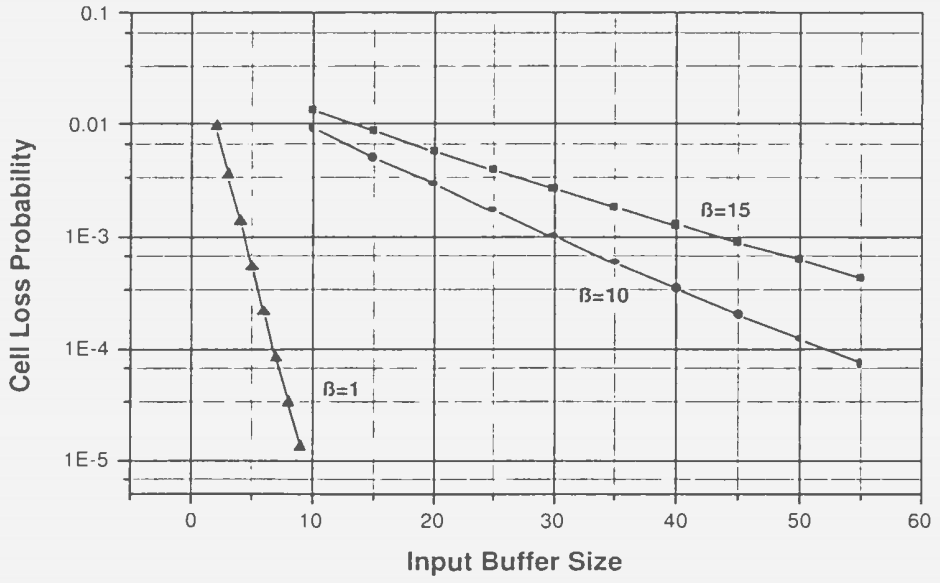
(b) Average Input Buffer Delay vs. Traffic Load for Abacus Switch (taken from [2], pp. 205)

Figure 4.30: Comparison of average input buffer delay vs. traffic load between multicast BG switch and Abacus switch

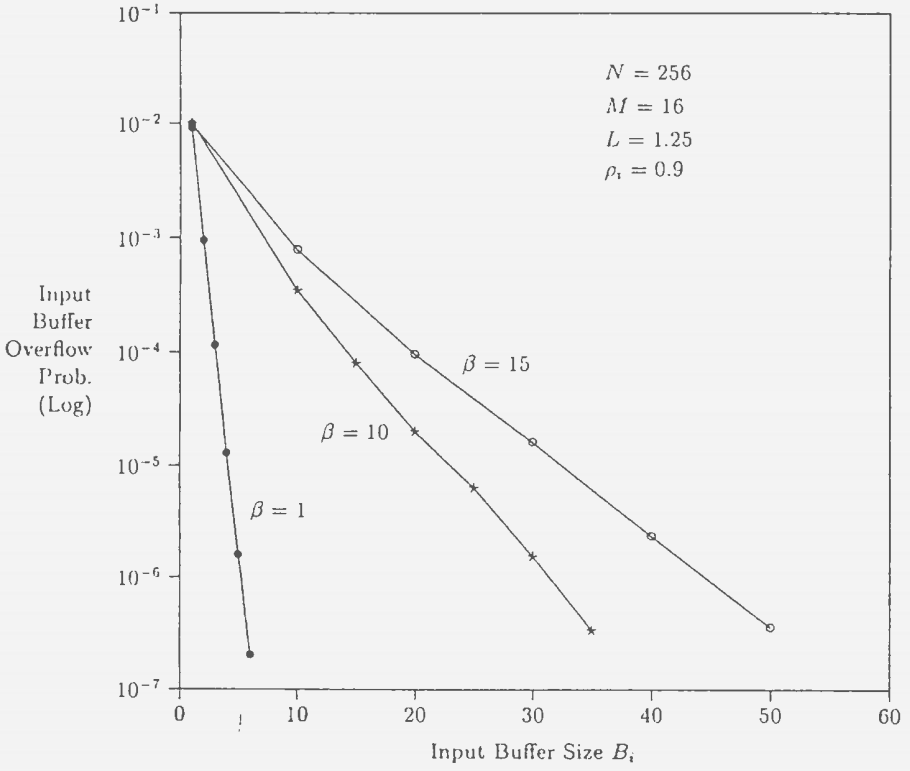
even higher than the crossbar switch ($256 \times 256 = 65,536$). The complexity for an 256×256 BG switch is given by $1 \times 256 \times 1 \times 2 + 1 \times 256 \times 2 \times 4 + 6 \times 256 \times 4 \times 4 = 27,136$. Therefore, the complexity for the BG switch is less than one third of the complexity of the Abacus switch. This number does not include the complexity associated with the translation tables and the feedback buses, which are required by the Abacus switch but not for the BG switch. As the switch size grows larger or when the Abacus switch needs to be reconfigured to achieve higher performance, the hardware complexity will become even higher.

It appears that the higher performance of the Abacus switch is actually due to its very high hardware complexity inside the fabric. In addition, high-speed memory and complicated control are also required by the Abacus switch because the output buffer is shared by all the channels within the same group. If the same 256×256 switch is used as the example, then a group size $M = 16$ means that the output buffer has to run at least 16 times the link speed. There is no doubt that the control function has to be fast enough to coordinate and achieve full sharing among the buffers. As higher link speed and larger group size are adopted, the buffer control unit will soon reach its bottleneck. Even though such sharing can improve buffer utilization, it is very costly to build for large switches. Besides this, there is no feedback mechanism to report blocking that occurs inside the SSM or due to output buffer overflow. As a result, cell loss will happen when such a blocking situation occurs. Therefore, SSM and output buffer have to be configured large enough to accommodate any kind of incoming traffic, which will result in increased cost.

Compared with the sophisticated buffer used in the Abacus switch, all building blocks used in the BG switch, including the buffer design, are simple and will not become bottlenecks for the switch. As well, it has been discovered in the study of the pipeline structure of the BG switch [14] that a single-plane BG switch is



(a) Cell loss probability vs. input buffer size of BG switch



(b) Cell loss probability vs. input buffer size of Abacus switch (taken from [2], pp.206)

Figure 4.31: Comparison of input buffer overflow probability vs. input buffer size between multicast BG switch and Abacus switch

enough to achieve a satisfactory performance in most traffic conditions. With two-plane-pipelining, the performance becomes very close to the ideal switch. Therefore, using the simple method of replicating the whole switch plane, the BG switch can be easily adjusted to much higher performance with less or comparable hardware complexity than the Abacus switch. Furthermore, such pipelining could improve the fault tolerance, robustness and reliability of the switch. It is clear that the BG switch has superior performance in comparison to the Abacus switch in relation to the level of hardware complexity.

4.8 Summary

In this chapter, the performance of the multicast BG switch is investigated in detail under various uniform and nonuniform multicast traffic conditions. A theoretical model is developed to cope with the performance analysis under multicast random traffic and is used to verify the simulation results obtained from the BG switch simulator. Performance results are compared to those of the ideal non-blocking multicast switch as well as to two other switches reported in the literature. Performance analysis confirms the architectural feature that the BG switch is a predominantly output-buffered switch. The use of very small amounts of input buffering resources not only helps to reduce the overall complexity of the switch, which makes it possible to build a practical switch based on this architecture, but also helps to achieve a close to optimum performance by reducing HOL blocking. It has also been shown that under the same offered load, with larger the mean fanout of the traffic, better loss performance will be achieved. In general, the performance of the multicast BG switch is very close to that of the ideal switch and its scalability and low complexity make it a strong candidate for use in high-speed multicast switching networks.

Chapter 5

Design and Implementation of the Multicast BG Switch

5.1 Introduction

As the size and complexity of digital systems increase, early stage paper-and-pencil design methods become no longer suitable. Instead, computer-aided design (CAD) tools, which are characterized by their sophisticated design entry, advanced verification functionality and automatic hardware generation, are introduced into the hardware design process [86]. Hardware description languages (HDLs) are used to facilitate the computer-aided system design process. Among the many HDLs, the Very High Speed Integrated Circuits (VHSIC) Hardware Description Language (VHDL) and Verilog are the two major languages which are widely used for digital hardware design.

In this chapter, the complete design and implementation process of the multicast BG switch is described with an emphasis on the front-end design issues. The general methodology of the design process is first reviewed with the focus on the digital IC design flow recommended by the Canadian Microelectronics Corporation (CMC) [11]. Then, the detailed design process is explored for the multicast BG switch using the 0.18 μm CMOS technology. The testing methodology is described with respect to the

efficient functional verification of the hardware design. Finally, the synthesis results of the design are presented with emphasis on timing and area requirements.

5.2 Digital System Design Flow

Generally speaking, an initial design idea goes through several transformations before the final hardware implementation is obtained. At each step of transformation, the designer checks the result of the last transformation, adds more information, and passes it through to the next step of the transformation [86]. When all the design and verification steps are completed, a stream file to describe the mask layer information for the circuit will be created. This is the file which is used for chip fabrication after Design Rule Checking (DRC) is completed.

Figure 5.1 shows the digital system design flow using the Deep Sub-Micron (DSM) technology recommended by CMC [11]. The design flow can be divided into two stages. The first four steps belong to the front-end design stage, in which VHDL and Synopsys tools are used. The remaining five steps comprise the back-end design stage, in which Verilog and Cadence tools are used. Simulation and synthesis are the major concerns for the front-end design. A design idea is converted to a gate-level netlist through this process. In the back-end design, the major interest is on the placement and routing of the imported gate-level netlist onto the silicon wafer. Although the main design work of the multicast BG switch described in this chapter is on the front-end design aspect, to provide a complete picture about the DSM design flow, the nine steps are briefly reviewed.

Step one is used for the Register Transfer Level (RTL) code simulation. The objective is to verify the functionality of the RTL code describing the design. The top-down design and the bottom-up implementation are the most important processes. Even

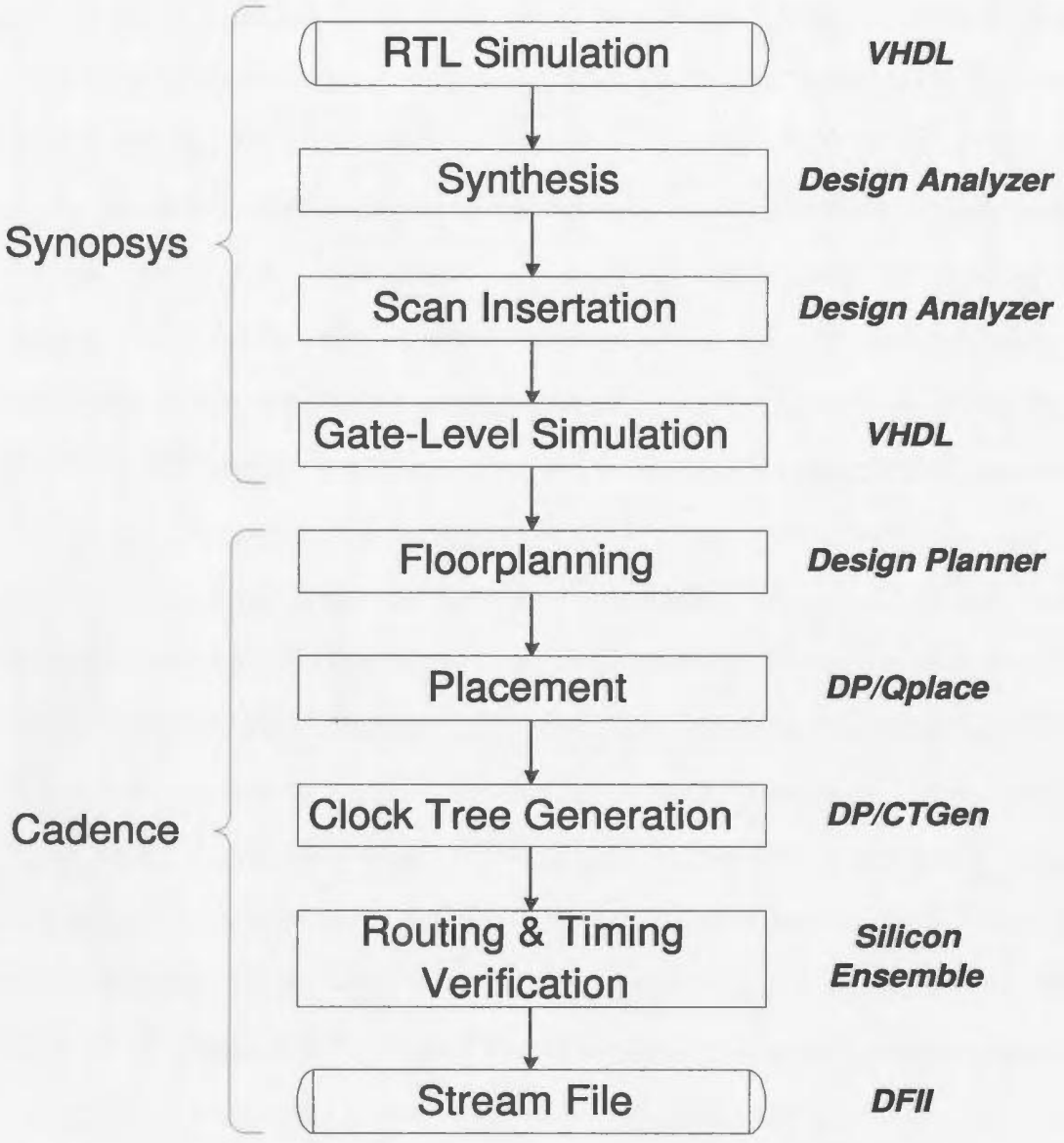


Figure 5.1: Digital IC design flow recommended by CMC [11]

though current CAD tools can support synthesis from a high-level description directly into hardware, such a synthesis process usually results in too much redundancy and therefore the results are not optimal. The situation may become even worse when synthesizing a large design, especially when the control logic is quite complicated. However, for small and simple designs, it is efficient and trustworthy for the CAD tools to achieve an excellent result. Therefore, in our design, instead of trying to implement the whole system at once, a divide-and-conquer strategy is followed in the top-down design process. The design is recursively partitioned into its components until all components become manageable parts. The design of a component is said to be manageable if the component is available as part of a library, or it can be implemented by modifying an existing part, or it can be efficient handled by a synthesizer or an automatic hardware generator [86].

When the top-down design process is completed, a partition tree and hardware implementation of the leaf components become available. Then, the bottom-up implementation process begins. During the process, hardware components corresponding to the leaves of the tree are recursively bound to make the complete system [86, 87].

The objective of the synthesis step is to import the RTL description of a design into a Synopsys database and convert it into gate-level circuits [11]. In this step, each leaf component of the partition tree is analyzed and claborated by the Synopsys Design Analyzer. Because of the hierarchical architecture, a high-level component can be imported only when all its subcomponents are available in the current database or in the active work library. After the complete design is successfully imported, it is constrained based on the designer's performance objectives. In most cases, the constraints include I/O pads specification, scan style definition, output load definition, clock definition and so on. The constrained design is then synthesized into gate-level circuits if all constraints are satisfied. Otherwise, the RTL code needs to be modified,

simulated and re-synthesized until all constraint requirements are met.

The next step is scan insertion based on the standard scan-based Design For Testability (DFT) techniques. The Synopsys Test Compiler is used. The purpose is to make the design testable. The Test Compiler first substitutes all sequential devices with scan equivalents, and connects them together to form a scan chain. Then it generates a set of test vectors which can detect "stuck at 1" (SA1) and "stuck at 0" (SA0) faults in the chip. Other tasks such as vector compaction, and fault coverage estimation will also be performed in this step [11].

In step four, gate-level simulation is performed. Recall that in step one, the RTL code was simulated to ensure the design was functionally correct. Timing was not considered because hardware timing information, which is tightly associated with the targeted technology library, is not available yet. Through synthesis, the RTL description is converted to a gate-level circuit and at this point it is possible to verify its functionality with timing information included. Successful gate-level simulation completes the front-end design process.

Step five starts the IC physical design, in which the floorplanning is performed. The objective is create a floorplan for the design including a default group of cells, I/O ring connected by abutment, and defined placement sites for all cells [11]. Cadence tool Physical Design Planner (PDP or DP) is used for physical placement of a design.

Qplace sequencer is used in step six for the timing-driven placement of standard cells. The objective is to use forward-annotated timing information from Synopsys tools to perform core cells placement.

Step seven is used for clock tree generation. The major objective is to insert clock buffer cells and nets to create a balanced clock tree. After the clock tree insertion, another gate-level netlist is generated [11].

In step eight, the new gate-level circuit after clock tree insertion is simulated again

for functionality verification. Then, the placed design, including all design libraries and constraints needed to route the design, is imported into the Silicon Ensemble environment. Using existing timing constraints, routing with antenna and crosstalk fixing is performed. The routed design is verified to meet timing goals.

In the last step, a number of tasks are performed. First, using the Diva Layout-Versus-Schematic (LVS) tool from the Cadence Design Framework *II* (DF*II*) toolbox, LVS is performed to verify the placed and routed version of the design with the gate-level netlist generated in step seven. Manual edits of the layout is carried out to fix minor DRC violations. Further DRC checking is performed to the resulting design. The feedback from the DRC checking is imported again into the DF*II* for antenna problems fixing on identified nets. Metal and poly fill are added to meet the fabrication requirements. Finally, after the LVS and DRC verification, stream output is obtained for chip fabrication, which concludes the digital system design process.

5.3 Design and Implementation of the Multicast BG Switch Fabric

In this section, the design process of the multicast BG SF is explored. Before starting, the switch architecture, which is depicted in Figure 3.1, is briefly reviewed. The basic architecture of an $N \times N$ multicast BG switch is composed of N IPCs, an $N \times N$ SF, and N OPCs. The $N \times N$ SF is composed of $n + 1$ stages, where $n = \log_2 N$. For the first n stages, 1×2 SEs are used for stage 0, 2×4 SEs are used for stage 1, and 4×4 SEs are used for all remaining stages. The last stage (stage n) is the output buffer stage, which can accept up to 4 cells per output line in one switching cycle. Each stage has N SEs numbered from 0 to $N - 1$.

As mentioned earlier, the main research interest is the SF design. The design of a fully-functioned IPC and OPC is beyond the scope of this research. However, to

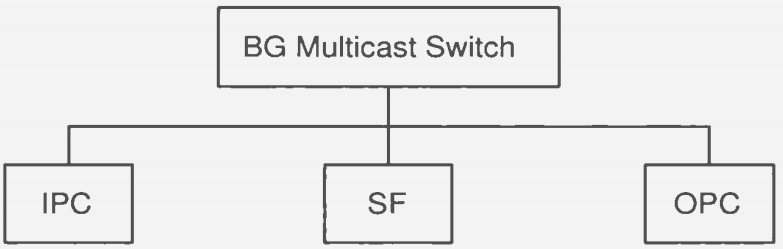


Figure 5.2: Resulting structure after first level partitioning

test the SF and to demonstrate the switching operation, simplified IPCs and OPCs, which possess basic required functions to interface with the SF, are used.

Following the divide-and-conquer strategy, the design requirements are analyzed and a suitable architecture model is developed. The switch adopts a three-phase switching method and follows a single-plane, bufferless, and non-pipelined SF architecture. The first-level partitioning is shown in Figure 5.2. Obviously, further partitioning for each leaf node is needed.

5.3.1 Input Port Controller (IPC)

The functionality of the simplified IPC are summarized as following:

1. Buffer incoming cell in the input queue if it is not full, otherwise discard.
2. Generate the internal self-routing and replication tag for HOL cell.
3. Buffer backpressure stream from the SF into the acknowledgement buffer.
4. Update HOL cell's tag information.
5. Remove HOL cell when all copies are delivered.

Function 1 requires an input queue be incorporated at each IPC. A FIFO queue running at external link speed is used. It can be implemented through shift registers or memory design. For function 2, tag generation circuitry is needed to generate the self-routing and replication tag for the HOL cell. Assuming the destination request of incoming cells has been translated to the bitmap format through table lookup

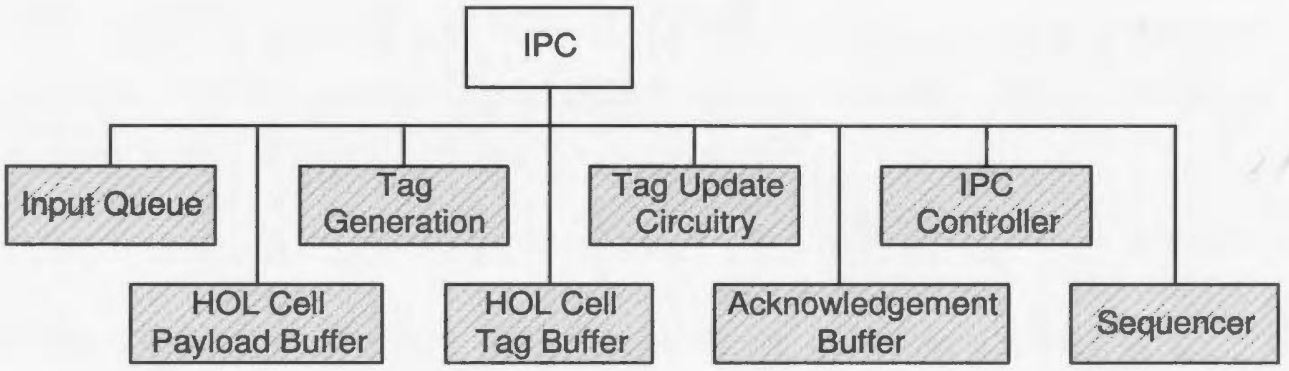


Figure 5.3: Resulting structure for IPC after second level partitioning

and using the bitmap tag encoding scheme, the destination request is directly used as the internal tag. Function 3 requires a maximum N-bit buffer space be included in each IPC to store acknowledgement information. The tag update circuit, as required in function 4, uses this information together with the HOL cell's destination request to decide whether to remove or retain the HOL cell for the next switching cycle. Finally, the IPC controller is required to generate control signals for different components, which include input queue enable, tag/payload pushout enable, acknowledgement receiving enable, and handshaking signals with the SF. The control logic can be implemented as a state machine, which utilizes a sequencer to provide timing information.

Following the above description, a second-level partitioning is performed on the IPC and shown in Figure 5.3. There is no need to continue the partitioning to the stage because modern synthesis tools, such as the Synopsys Design Analyzer, are efficient in generating a hardware circuit from a behavioral or dataflow description of the moderate-sized design. Therefore, after the second partitioning, noting that tag generation and tag update circuitry is just combinational logic, a dataflow description should suffice. The buffer design is quite standard and a behavioral description would be enough for that purpose. The IPC controller can be easily realized by using

behavioral description for a state machine. The sequencer design will be studied in the next section. Hence, all leaf nodes become manageable design modules, as represented by the shadowed boxes in the figure.

5.3.2 Switch Fabric (SF)

As described earlier, the multicast BG SF utilizes a MIN design. The routing and replication functionality is distributed into the functionality of each SE. Following the design methodology described in Section 5.2, the second and third level partitioning of the SF is carried out and presented in Figure 5.4. The SF is first partitioned into n stage components and then each stage component is partitioned into SEs, stage controller and sequencer. The sequencer provides timing information for the stage controller. The stage controller, basically a state machine, provides the control signals for all SEs within the stage. After the above two partitions, a partition tree has been obtained, which has three leaf nodes of two types: the sequencers and stage controllers are now terminal nodes, while the SEs require further partitioning.

5.3.2.1 Sequencer

The sequencer plays an important role in the operation of a single stage, as well as the whole SF. The core of the sequencer uses a counter design, which counts on the rising edge of the applied clock signal. A reset signal (Reset_In), issued by the stage controller on the falling clock edge of the previous switching cycle, initializes the sequencer to zero. At the beginning of the current switching cycle, the negative edge of the reset signal triggers the sequencer to start counting. The pattern matching circuit compares counter output with the desired state. Its output is used by the stage controller to decide the next state. The interface of the sequencer is shown in Figure 5.5. Sequencer output signals, which represent different states and are used

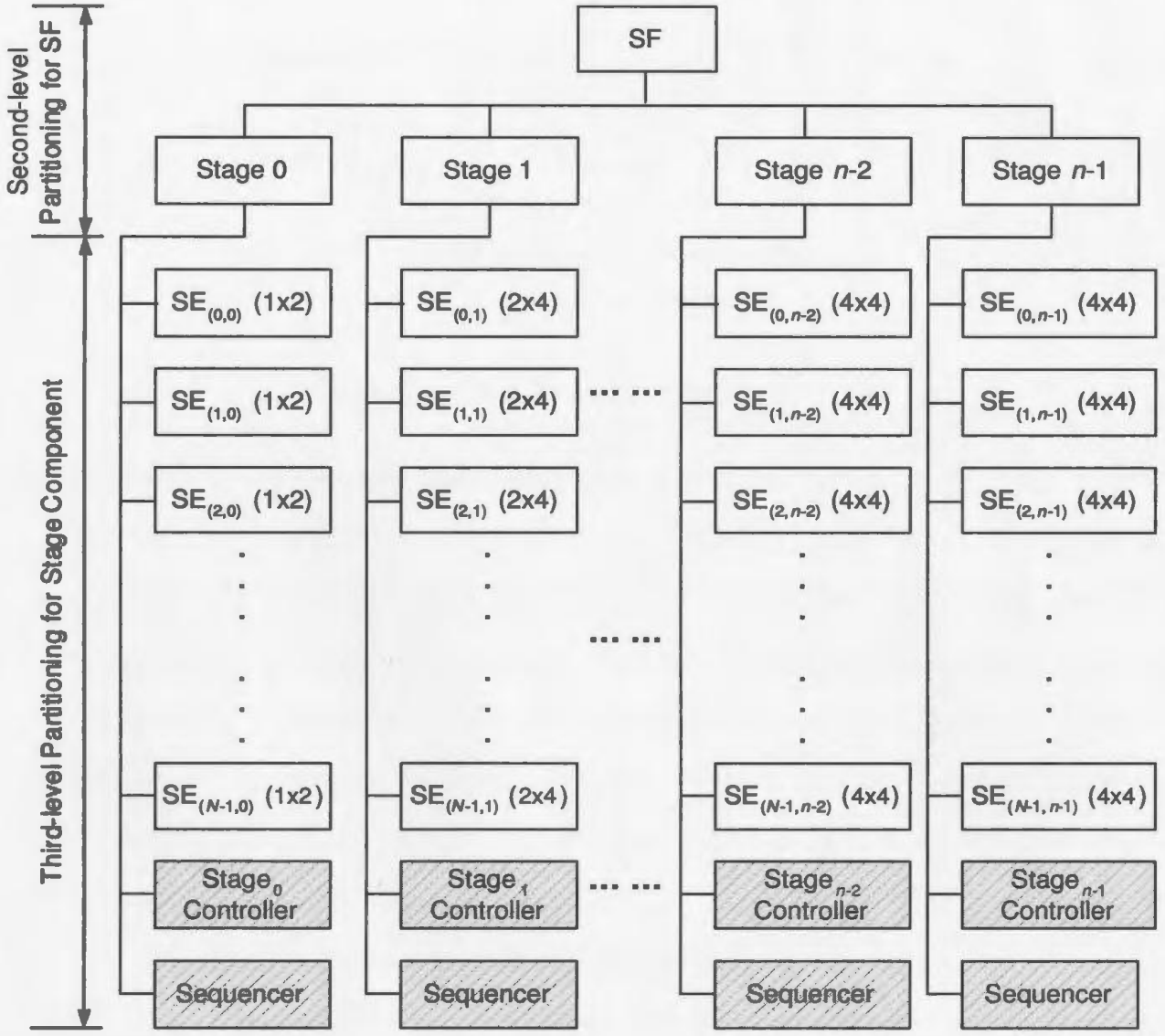


Figure 5.4: Resulting structure for SF after second and third level partitioning

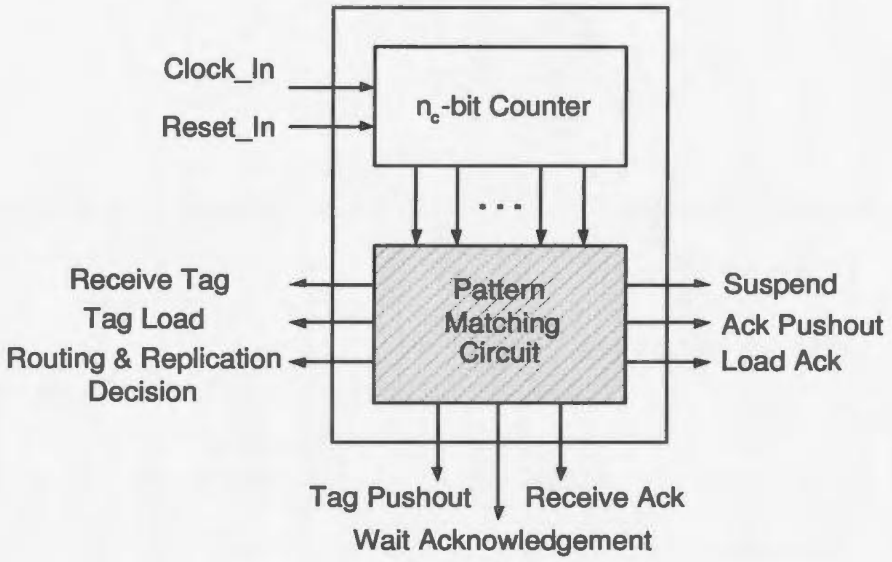


Figure 5.5: Sequencer interfacing diagram

by the stage controllers, are explained in the next subsection.

The counter size is selected based on the maximum possible range of clock cycles to be counted. It is determined by the length of the tag and acknowledgement bits exchanged between stages, the process time of the SE, and the payload. In between any two stages, including IPC and OPC, besides the tag, there are 3 fixed bits of priority to be transmitted. The process time at each SE takes 4 clock cycles to complete. To get the total number of cycles to be counted, the following notations are used:

$L_{payload}$: the length of the fixed-size payload,

L_{tag}: the accumulated tag bits exchanged in the SF,

L_{max}: the maximum possible number to be counted, in clock cycles,

and the maximum count can be calculated by

$$\begin{aligned} L_{max} &= L_{tag} + L_{payload} \\ &= [(N+3) + (\frac{N}{2}+3) + ... + (1+3)] + [N + \frac{N}{2} + ... + 1] + 4n + L_{payload} \end{aligned}$$

$$= 2(N + \frac{N}{2} + ... + 1) + 3(n+1) + 4n + L_{payload}$$
$$= L_{payload} + 2^{n+2} + 7n + 1. \quad (5.1)$$

Therefore, with some margin ζ, the counter size n_c can be selected by using

$$n_c = \lceil \log_2(L_{max} + \zeta) \rceil + 1. \quad (5.2)$$

5.3.2.2 Stage Controller

Stage controller plays the most important role in the switching operation. There are two possible ways to implement the control function: centralized and distributed.

Centralized fabric control is used in the unicast BG switch design [14]. The network main controller (NMC) is used to generate all required control signals inside the SF. The advantage is that it is easy to implement and has less hardware complexity since all stages share the same controller. However, the main disadvantage is that it does not scale well as the switch size expands. In the architectural scalability discussion in Section 3.8, it has been shown that to double switch size, we only need to duplicate the smaller switch module and add one more stage in the front. However, using centralized fabric control, the main controller has to be redesigned as the switch size grows. This would be a major obstacle for architecture expansion if the complete switch is to be implemented on a single chip.

In our design, a distributed control method is used. The SF control function is distributed into n stage controllers, which are shared by all SEs within the same stage. A stage controller is basically a state machine that changes its state based on sequencer output. Each stage controller has exactly the same number of states. The difference between them is only in the number of clock cycles associated with different states. The state that a stage controller can be in is given below:

$Wait\_Tag$: Wait state, waiting for tags to arrive;

Receive_Tag:	Active state, receive tags from adjacent upstream stage;
Tag_Load:	Active state, prepare tags for downstream stage and load into tag pushout buffer banks;
Decision:	Active state, make self-routing and replication decision based on received tags;
Tag_Pushout:	Active state, transmit tags to downstream stage;
Wait_Ack:	Wait state, waiting for acknowledgement bits from downstream stage;
Receive_Ack:	Active state, receiving acknowledgement from downstream stage;
Load_Ack:	Active state, prepare output acknowledgement for downstream stage and load into ack pushout buffer banks;
Ack_Pushout:	Active state, transmit acknowledgement to upstream stage;
Send_Payload:	Active state, payload data transmission from IPC to OPC;
Suspend:	Idle state, wait for the reset signal from the IPC to start the next switching cycle.

At the beginning of each switching cycle, an active high *Reset_In* signal from the IPC stage controller resets all fabric stage controllers to the initial *Wait_Tag* state. On the receipt of the *Reset_In* signal, each stage controller generates a *Reset_Out* signal to reset all its subordinate SEs and sequencer. The negative edge of the *Reset_In* signal triggers each stage controller to start the sequencer as well as a new round of switching operation. The ASM chart of the stage controller is shown in Figure 5.6. When switch size expands, the *Reset_In* to the smaller switch module will be delayed the number of clock cycles as required by the new added Stage 0.

Generally speaking, a stage controller closer to the IPC changes to the *Receive_Tag* state earlier, and changes to *Receive_Ack* state later. Timing information of different

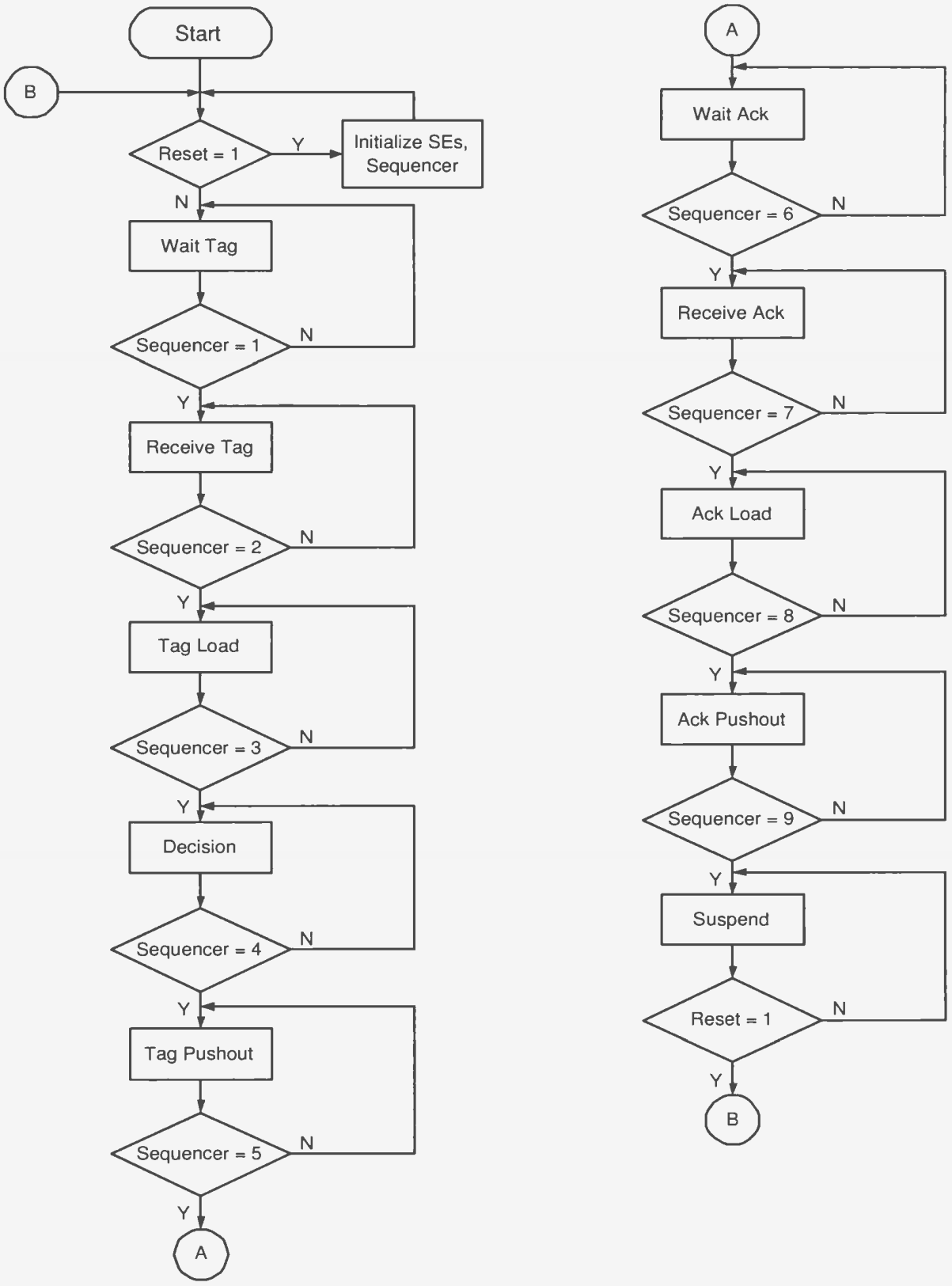


Figure 5.6: ASM chart for stage controller

states of the stage controller for an 16×16 switch is shown in Table 5.1. The number on the top of the arrow line between two blocks indicates the number of bits to be transferred between the two stages following the arrow direction. The state changes at different stages, their starting time and ending time, and the corresponding durations are also given in the table. The unit is in clock cycles.

5.3.2.3 Switch Element

As discussed earlier, the 4×4 SE is more general and will be used to describe the design process in the following discussion.

Figure 5.7 depicts the architecture of a 4×4 SE. To support the three-phase switching operation, the internal architecture of each SE is designed to provide two channels, forward and backward, and therefore contains three major functional blocks: the forward-path control unit (FCU), the admission control unit (ACU) and the backward-path control unit (BCU).

I. Forward-path Control Unit

The FCU is used for tag and payload transmission, which is comprised of the tag receiving buffer bank, tag pushout buffer bank, and source/path select multiplexer group:

1) Tag Receiving Buffer Bank

The tag receiving buffer bank is used to temporarily hold the tag bits, which are used by the ACU to make the routing and replication decision. It is a serial-in-parallel-out device and is composed of one queue in the case of 1×2 SEs, two queues in the case of 2×4 SEs, and four queues in the case of 4×4 SEs. Each queue is a shift register of $2^{n-i} + 3$ bits for each SE at the i^{th} stage, where $0 \leq i < n$. The 2^{n-i} bits are used to store the bitmap format tag while the 3 bits are used for cell priority.

Table 5.1: Timing information stage controller states in an 16 × 16 switch

Input stage	Stage 3	Stage 2	Stage 1	Stage 0	Output stage	Start	End	Duration
Tag Pushout	Receive Tag	Wait Tag	Wait Tag	Wait Tag	Wait Tag	0	19	19
Wait Ack	Tag Load	Wait Tag	Wait Tag	Wait Tag	Wait Tag	19	20	1
Wait Ack	Decision	Wait Tag	Wait Tag	Wait Tag	Wait Tag	20	22	2
Wait Ack	Tag Pushout	Receive Tag	Wait Tag	Wait Tag	Wait Tag	22	33	11
Wait Ack	Wait Ack	Tag Load	Wait Tag	Wait Tag	Wait Tag	33	34	1
Wait Ack	Wait Ack	Decision	Wait Tag	Wait Tag	Wait Tag	34	36	2
Wait Ack	Wait Ack	Tag Pushout	Receive Tag	Wait Tag	Wait Tag	36	43	7
Wait Ack	Wait Ack	Wait Ack	Tag Load	Wait Tag	Wait Tag	43	44	1
Wait Ack	Wait Ack	Wait Ack	Decision	Wait Tag	Wait Tag	44	46	2
Wait Ack	Wait Ack	Wait Ack	Tag Pushout	Receive Tag	Wait Tag	46	51	5
Wait Ack	Wait Ack	Wait Ack	Wait Ack	Tag Load	Wait Tag	51	52	1
Wait Ack	Wait Ack	Wait Ack	Wait Ack	Decision	Wait Tag	52	54	2
Wait Ack	Wait Ack	Wait Ack	Wait Ack	Tag Pushout	Receive Tag	54	*58*	4
Wait Ack	Wait Ack	Wait Ack	Wait Ack	Receive Ack	Ack Pushout	*58*	*59*	1
Wait Ack	Wait Ack	Wait Ack	Wait Ack	Load Ack	Wait Payload	*59*	*60*	1
Wait Ack	Wait Ack	Wait Ack	Receive Ack	Ack Pushout	Wait Payload	60	62	2
Wait Ack	Wait Ack	Wait Ack	Load Ack	Suspend	Wait Payload	62	63	1
Wait Ack	Wait Ack	Receive Ack	Ack Pushout	Suspend	Wait Payload	63	67	4
Wait Ack	Wait Ack	Load Ack	Suspend	Suspend	Wait Payload	67	68	1
Wait Ack	Receive Ack	Ack Pushout	Suspend	Suspend	Wait Payload	68	76	8
Wait Ack	Load Ack	Suspend	Suspend	Suspend	Wait Payload	76	77	1
Receive Ack	Ack Pushout	Suspend	Suspend	Suspend	Wait Payload	77	93	16
Send Payload	Suspend	Suspend	Suspend	Suspend	Receive Payload	93	125	32
Processing	Suspend	Suspend	Suspend	Suspend	Suspend	125	126	1

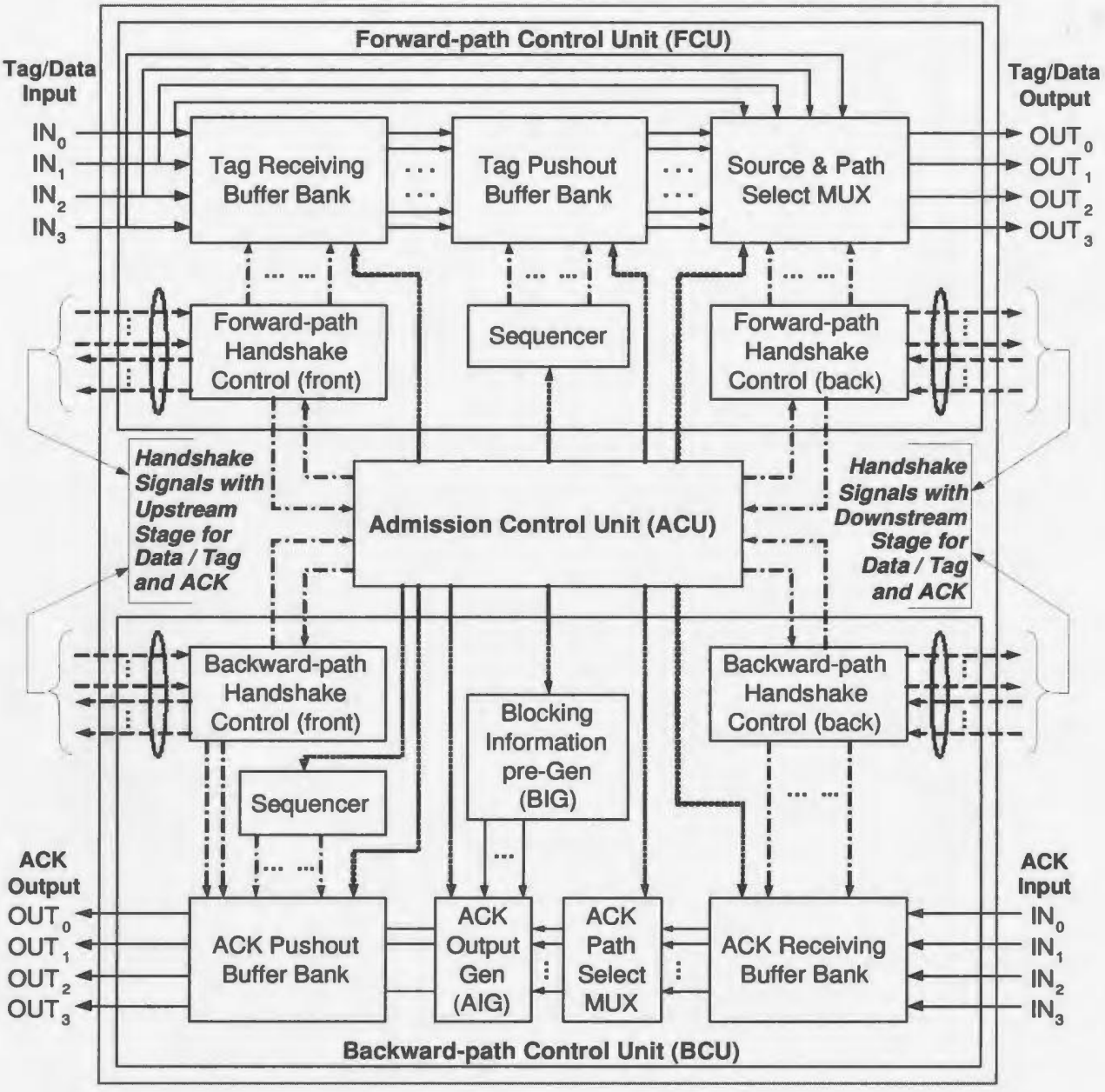


Figure 5.7: The internal architecture of an 4×4 SE

Unless otherwise stated, the length of the buffer banks used in the paper refers to the length of each single shift register. As described in the dynamic-length self-routing and replication algorithm in Section 3.4.2, to prepare tags for the next stage, current bitmap tags are halved and loaded into the pushout buffer bank together with the three priority bits, as shown in Figure 5.8 (a) for an example queue at Stage i.

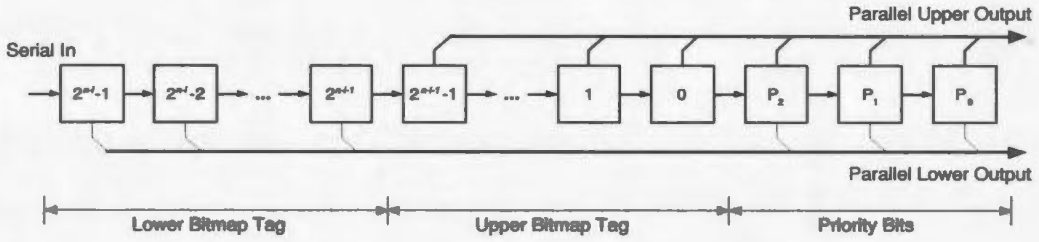
2) Tag Pushout Buffer Bank

The tag pushout buffer bank stores the tags ready to be pushed out to the next stage. Like the receiving buffer bank, it is also composed of one, two, and four queues in the case of 1×2, 2×4, and 4×4 SEs, respectively. However, it is a parallel-in-serial-out device. Each queue corresponds to a queue in the receiving buffer bank and is comprised of two shift registers, each of $2^{n-i-1} + 3$ bits. The parallel outputs of the receiving buffer bank are simultaneously loaded into the tag pushout buffer bank for transmission, as shown in Figure 5.8 (b).

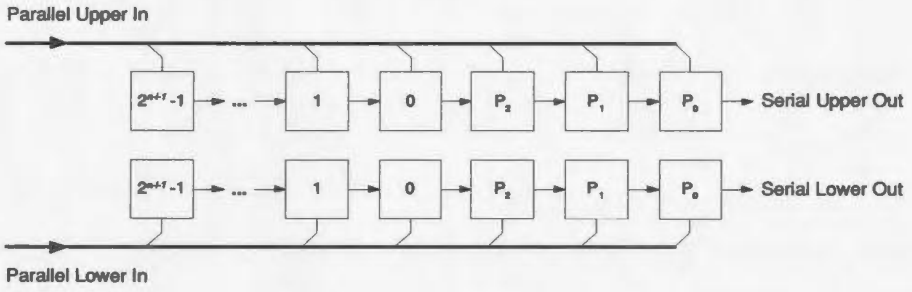
A minor difference should be mentioned for the pushout buffer bank design in the last stage (Stage $n - 1$). As the output of this stage is connected to the OPC, no more routing and replication action will be taken. Therefore, no bitmap tag is sent. However, since there are four links towards each OPC, to decide which cell will be accepted in the case of not sufficient buffering space at OPC, and to decide the order of cells that will be queued in the output queue, the 3-bit cell priority plus an activity bit are loaded into the pushout buffer bank.

3) Source/Path Select Multiplexer Group

The multiplexer group provides the path for any possible matching between the four input ports and four output ports. It also controls whether tag or payload will be transmitted to the next stage. During the reservation phase, outputs from the pushout



(a) Tag Receiving Buffer Bank Queue



(b) Tag Pushout Buffer Bank Queue

Figure 5.8: Queue example for tag receiving and pushout buffer bank in *FCU*

buffer bank are used as the inputs, while during the data transmission phase, the data input line is directly fed to the select multiplexer group and the two buffer banks are then bypassed. Multiplexer select signals are generated by the *ACU* based on the tag and priority of all active cells.

II. Backward-path Control Unit

The *BCU* is used for acknowledgement transmission. Unlike the *FCU*, as information is traversing from a downstream stage to an upstream stage, the length of acknowledgement is getting longer like a converging tree. Similar to the *FCU*, there are two types of buffer banks used in the *BCU*. For SEs at Stage i, an ACK receiving buffer bank of size 2^{n-i-1} and an ACK pushout buffer bank of size 2^{n-i} are used. Besides that, two more circuits are used in the *BCU*: the blocking information generation

circuitry (BIG) and the acknowledgement output generation circuitry (AOG).

1) Blocking Information Generation

BIG circuitry is used to generate a negative acknowledgement for a cell that loses its output contention. In this case, for a multicast cell, all switch output ports that are reachable from this SE's output link are marked as blocked. BIG results are then used by the AOG circuitry to generate the final ACK output when the acknowledgement for those successfully transferred cells comes back from downstream stages.

2) Acknowledgement Output Generation

The AOG circuitry is used to generate the final acknowledgement outputs that are passed back to the previous stage. Two scenarios should be considered: a) when a cell is successfully transferred, and b) when blocking occurs during switching. In the first case, the work for the AOG circuitry is to concatenate the received acknowledgement with 2^{n-i-1} 0s (when it is a unicast cell) or concatenate the acknowledgement received from the upper link to that from the lower link from downstream stages (when it is a multicast cell), and form its final ACK output. In the second case, the acknowledgement for the blocked part is generated by the BIG circuitry immediately while the acknowledgement for the successfully transferred part comes from downstream stages. They are concatenated by using the AOG circuitry and then loaded into the 2^{n-i} bit pushout buffer bank for transmission.

III. Admission Control Unit

The ACU is a purely combinational circuitry which forms the heart of the SE. The interface diagram is shown in Figure 5.9. The purposes of all these input/output signals are marked on the side of the signal. The most important work the ACU does is to make routing and replication decisions based on the received tag information.

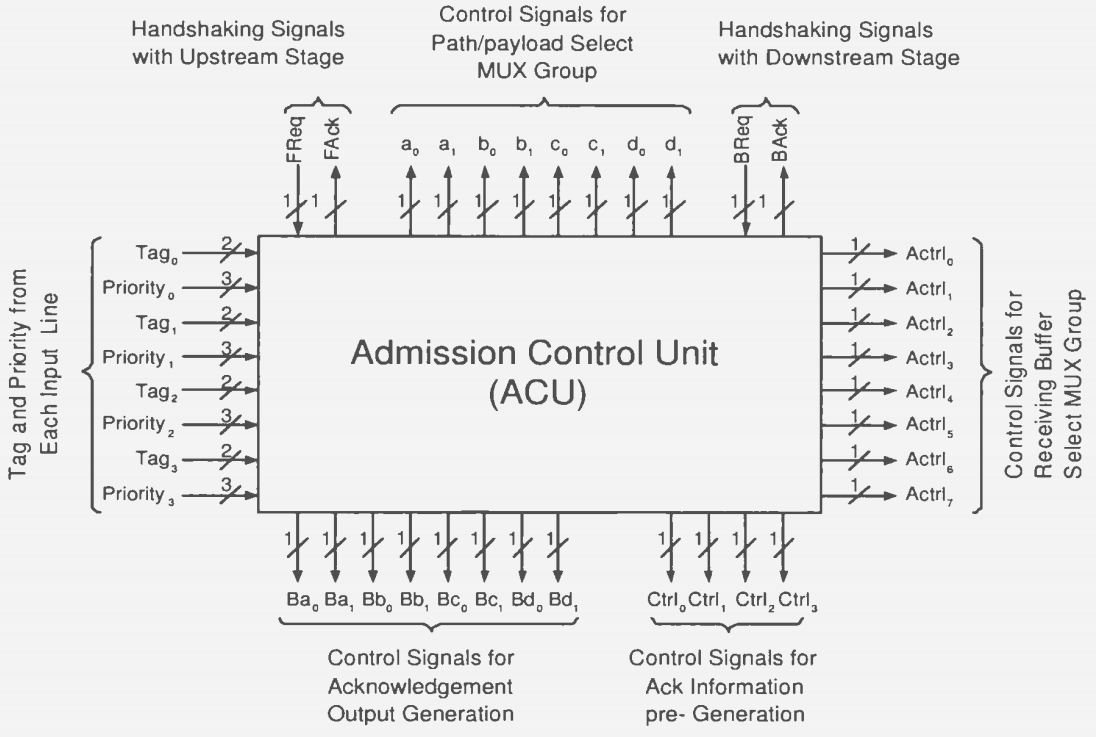


Figure 5.9: Interface diagram for admission control unit

Priority routing is a feature considered in the multicast BG switch to resolve link contentions. The cell with the highest priority is assigned to the output first and then the cell(s) of the next priority level and so on until either all the outputs are assigned or all active incoming cells are processed. When all the outputs are assigned, the remaining requests are blocked.

1) Bitonic Sorter

A bitonic sorter [38] in Figure 5.10 is used inside the *ACU* to pre-process incoming cells based on their priority levels. Let S_x, T_x and P_x, which are on the input side of the sorter, denote line number, tag and priority of the cell on input link x, where $x \in \{0, 1, 2, 3\}$. After sorting, priority bits become useless and therefore are discarded. Line number S_y and the corresponding tag T_y, where $y \in \{a, b, c, d\}$, appear on the

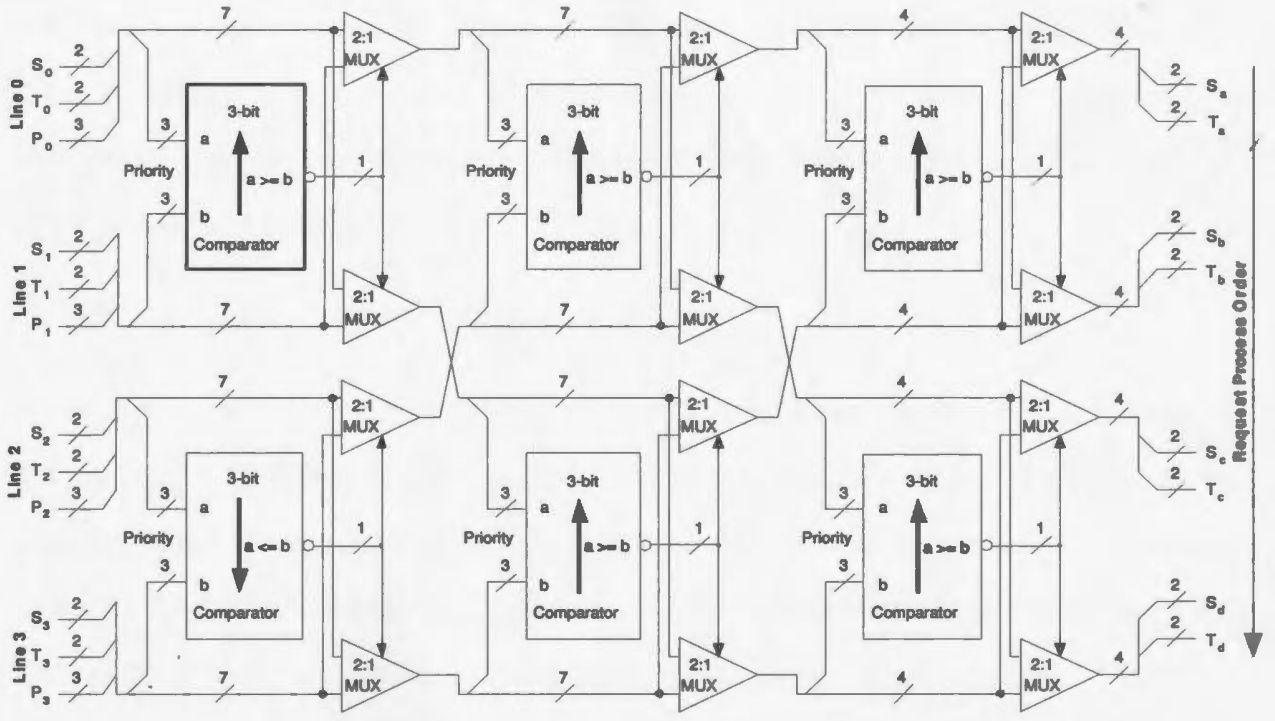


Figure 5.10: Bitonic sorter for cell priority sorting

output side of the sorter.

The *ACU* then processes the tag following the order $T_a \rightarrow T_b \rightarrow T_c \rightarrow T_d$. S_y, where $y \in \{a, b, c, d\}$, provides the input link number associated with the tag. Output resources are assigned during the process till either all incoming requests are processed or all output resources are used up.

For cells of the same priority, in the current design, the cell from smaller input line number is given a higher priority over others. Therefore, input 0 of each SE has the highest priority, input 1 has a higher priority over input 2 and 3, and so on. To achieve a better fairness, a pseudo-random number generator can be used inside each sorting element to randomly pick up one cell among the cells of the same priority.

2) Control Signals Generation

After incoming tags are sorted on their priority, the *ACU* starts the routing and

replication decision making process. It does so through the proper set/clear of various control signals. The decision is based on the request type (unicast or multicast) and the remaining available resources. The truth table for all control signals generation is provided in Appendix F.

5.3.3 Output Port Controller

Because our main research interest is on the SF design and test, the OPC functionality is realized by using the VHDL testbench and external data files. The OPC assumes that a single output queue is incorporated for each output. The service discipline is FIFO. During each switching cycle, up to four cells can be buffered at the output queue while only one cell can be dequeued to the external output link. Because the output queue accounts for the majority of buffering resources and normally requires a large amount of hardware complexity, it would be wise to separate them from the SF and put them on separate IC chips. This will also make future expansion easier when more buffering resources are required at the output queue. To facilitate such an arrangement and make the SF function verification easier, the buffer stage of the SF is included in the OPC during implementation. Therefore, each OPC can be viewed as an entity which has four input links from the SF, and one output link for cell departure to external networks. The interface diagram of the OPC is drawn in Figure 5.11. Handshaking signals are reserved control signals which will be later used for the failure detection purposes. The functionality of the OPC is summarized as follows:

1. Store incoming cell priority and activity bit to OPC tag buffer.
2. Sort active output requests based on cell priority level.
3. Decide to accept or reject cell request based on the available buffering space.
4. Generate acknowledgement for all input lines.

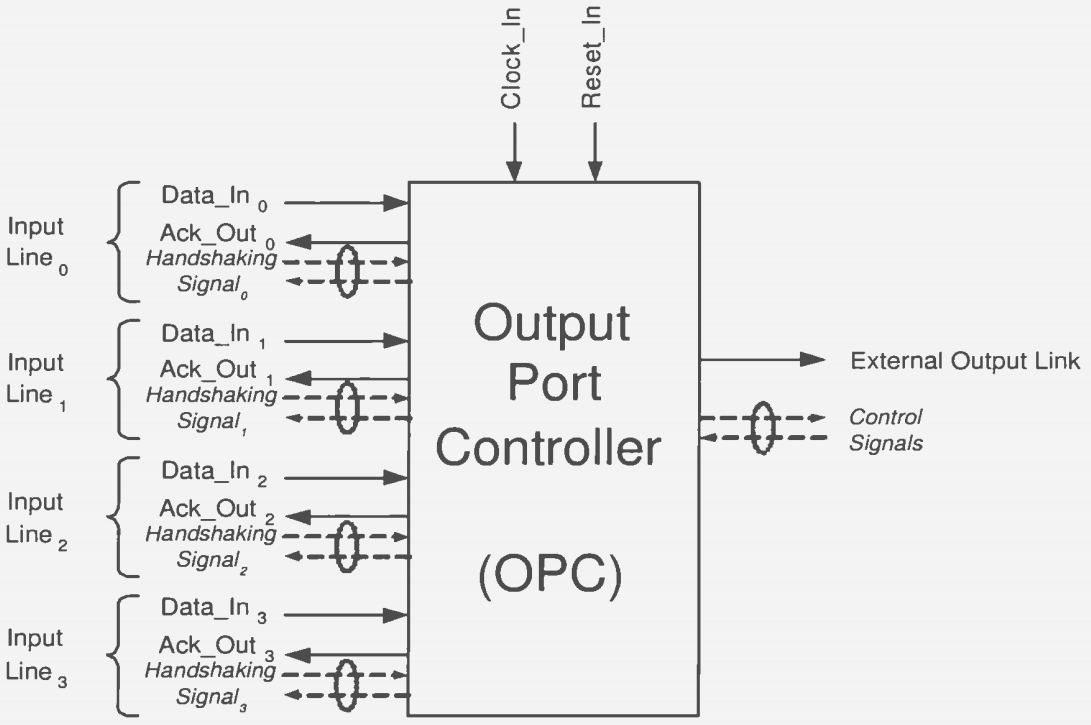


Figure 5.11: OPC Interfacing diagram

5. Store payload for successfully switched cells from IPC into its output queue.
6. Transmit HOL cell in output queue to external output link.
7. Remove the HOL cell after successful transmission.

Based on the above required functions, the second-level partitioning of the OPC design takes place and the result is shown in Figure 5.12. Each terminal block of the partitioning tree is described in the following:

Priority Tag Buffer Bank

The priority tag buffer bank, shown in Figure 5.13, performs a serial-to-parallel conversion function. It is comprised of four parallel 4-bit priority tag buffers, one for each input line. For each buffer, one bit is used to indicate the link status and the other three bits are for cell priority associated with that link. The content of the buffer bank is refreshed at the end of the reservation phase for each switching cycle.

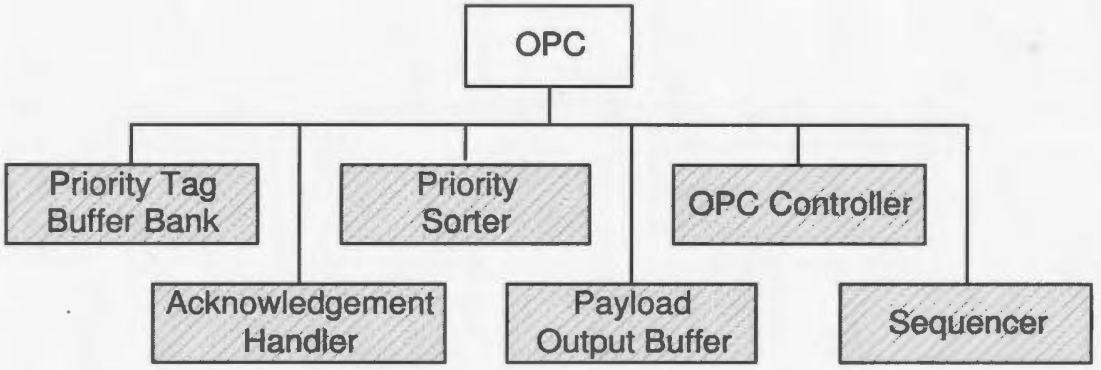


Figure 5.12: Resulting structure for OPC after second level partitioning

Unlike the buffer banks used in IPC and SEs, there is no routing and replication tag pair received, and its content need not to be further pushed out.

Priority Sorter

All incoming cells are sorted on their priority levels using a 3-bit bitonic sorter, which is exactly the same as the sorter used in the *ACU* in Figure 5.10. The sorter is used in the OPC to decide the order of cells in the output queue because there might have up to four cells coming into the OPC during the same switching cycle. The top sorter output provides the line number which has the highest cell priority while the bottom output provides the lowest. Based on the number of active cell requests and the remaining output buffer space, the OPC controller decides the cell(s) to be accepted. A blocked cell is negatively acknowledged while a succeeding cell's payload is then queued directly into the output queue during the data transmission phase.

Acknowledgement Handler

At each OPC, the only information to report is the output contention results for the four incoming links, that is, whether the cell can be accepted in the output queue or not, hence, one bit suffices. An acknowledgement is generated for each incoming link by the acknowledgement handler circuitry and is transmitted to the fabric at the beginning of the acknowledgement phase. The output of the priority sorter is

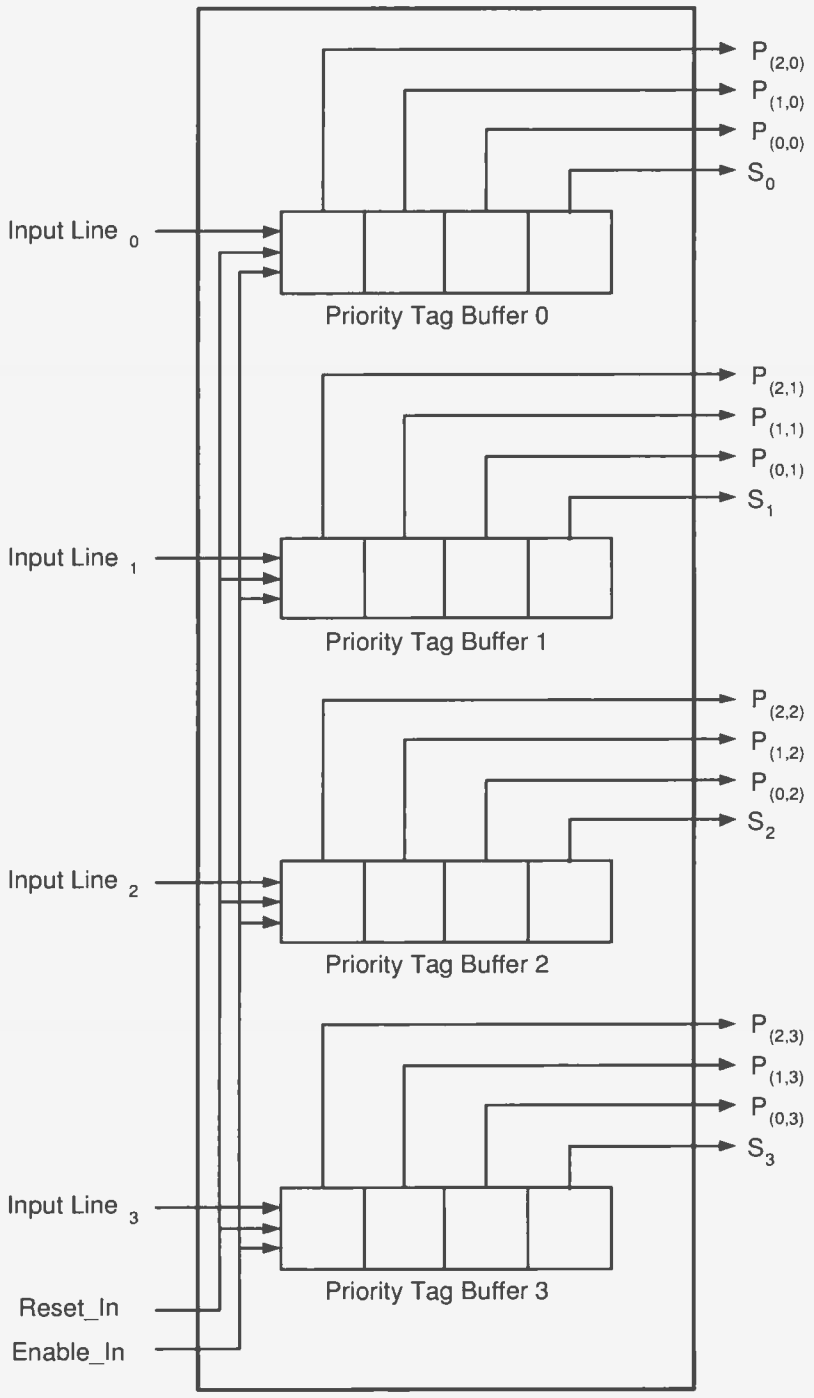


Figure 5.13: Block diagram for priority tag buffer bank

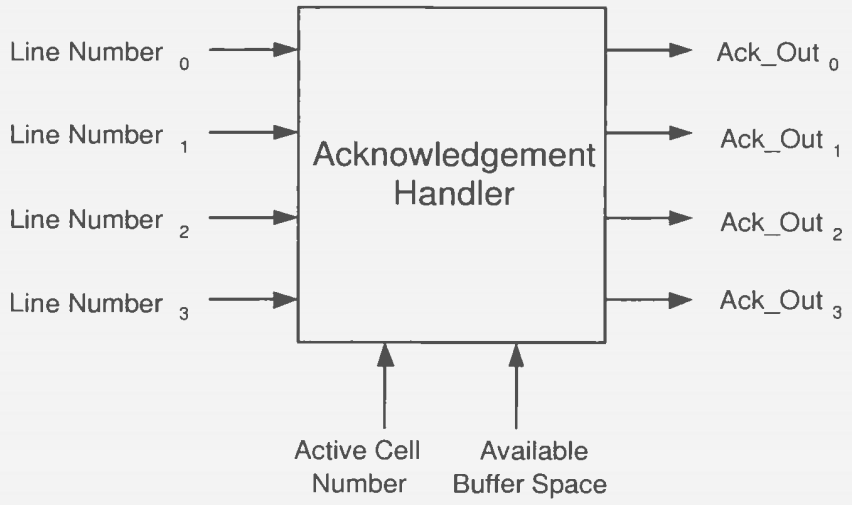


Figure 5.14: Interfacing diagram for acknowledge handler

used by the acknowledgement handler. Based on the number of active cell requests and the remaining buffering space for the OPC, the handler circuitry decides whether to positively acknowledge (ACK) or negatively acknowledge (NAK) the requests for each of the lines. An ACK will be issued until all buffering resources are used up. The acknowledgement handler interface diagram and its ASM chart is shown in Figure 5.14 and Figure 5.15, respectively.

Payload Output Buffer

The payload output buffer runs four times a single link speed on receiving, although it transmits only one cell to the output link during each switching cycle. It is comprised of a chain of cell buffers with each cell buffer constructed by a chain of shift registers. The output buffer size B_o, which is given in cells, should be carefully selected based on the performance requirement as described in Chapter 4. The OPC controller controls the operation of the output queue with the help of the output queue pointer.

OPC Controller

The OPC controller is the heart of the OPC and coordinates the work of different units. Similar to the controllers used in IPC and SF, it follows a state machine design.

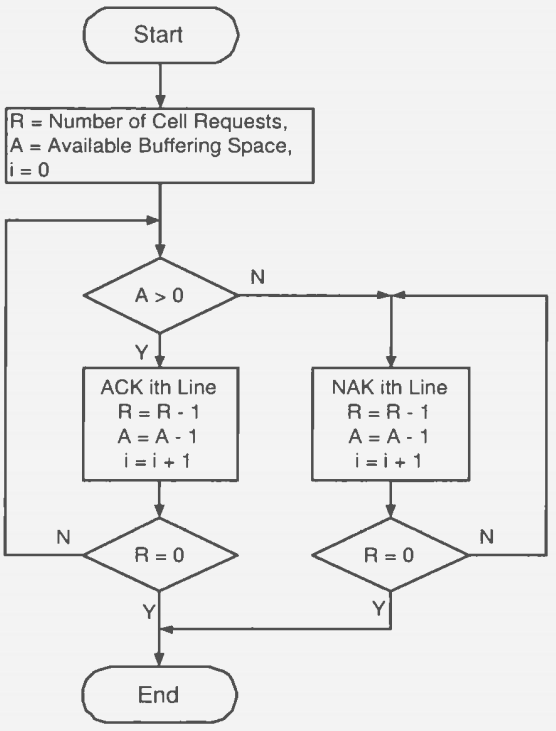


Figure 5.15: ASM chart for acknowledge handler

It uses the output of a sequencer to decide the next state. The ASM chart for the OPC controller is shown in Figure 5.16.

An important component inside the OPC controller is the output queue pointer. It points to the next empty location in the output queue which will be used by the output queue controller. To calculate the pointer for the next switching cycle, the following notations are defined:

$pointer(t)$: Pointer at the beginning of the current switching cycle,

$arrival(t)$: Number of cell arrivals during the current switching cycle,

$departure(t)$: Number of cell departure during the current switching cycle.

Then, the pointer for the next switching cycle ($pointer(t+1)$) is given by:

$$pointer(t+1) = pointer(t) + arrival(t) - departure(t), \tag{5.3}$$

where $pointer(t) \in \{0, 1, ..., B_o\}$, $arrival(t) \in \{0, 1, 2, 3, 4\}$, and $departure(t) \in$

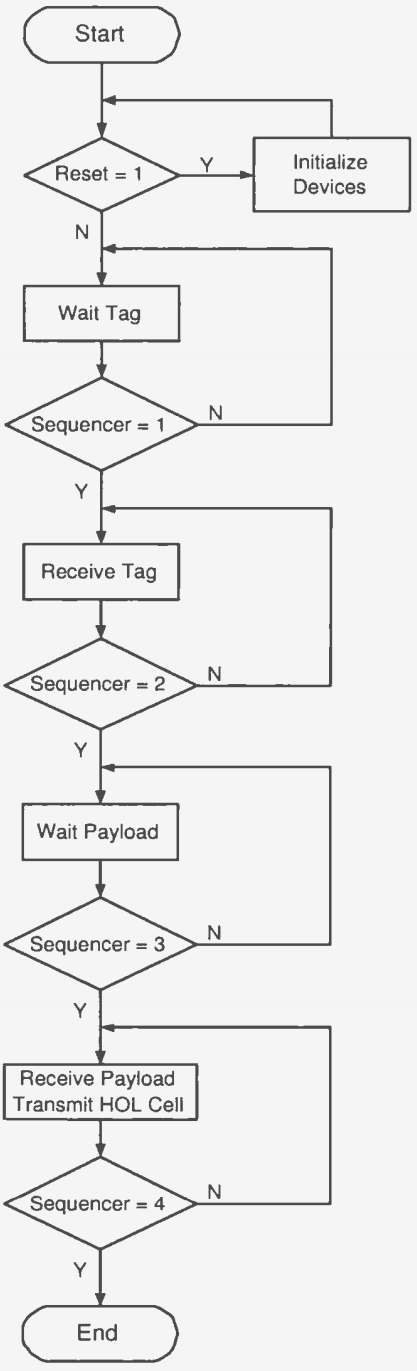


Figure 5.16: ASM chart for OPC controller

$\{0, 1\}$. B_o is the output queue size at each OPC.

The pointer circuitry also provides the information about the remaining buffering space available for use. The value is given by $B_o - pointer(t)$ and used by the acknowledgement handler to make the proper acknowledgement decision.

5.3.4 Bottom-up Implementation of the Design

At the end of the top-down design process, the whole switch design is broken down into the design of a number of smaller but manageable terminal components of the partition tree. All terminal components are first designed, simulated, and tested using VHDL. Then, following a bottom-up implementation approach, the lower level components are used to build higher level components until the top entity of the design is reached. The complete switch design is shown in Figure 5.17 for an 16×16 multicast BG switch.

5.4 Hardware Functional Simulation and Testing

During the hardware design and implementation process, simulation, testing and verification are carried out at different levels to ensure a correct result. It is important that all lower level components are thoroughly tested and proved functionally correct before moving to higher level components. For small leaf components such as flip-flops, multiplexers and shift registers, testing is straightforward. However, for large entities such as stage components or the whole SF, it is not straightforward to ensure a working function by looking at waveforms because so many input, output, and intermediate signals are involved. In this section, the VHDL simulation results are presented for SEs at different stages of the 16×16 BG SF. Because it is almost impossible to verify the correctness of switching through waveform observation, therefore, another testing methodology is adopted to test whole SF. The testing method

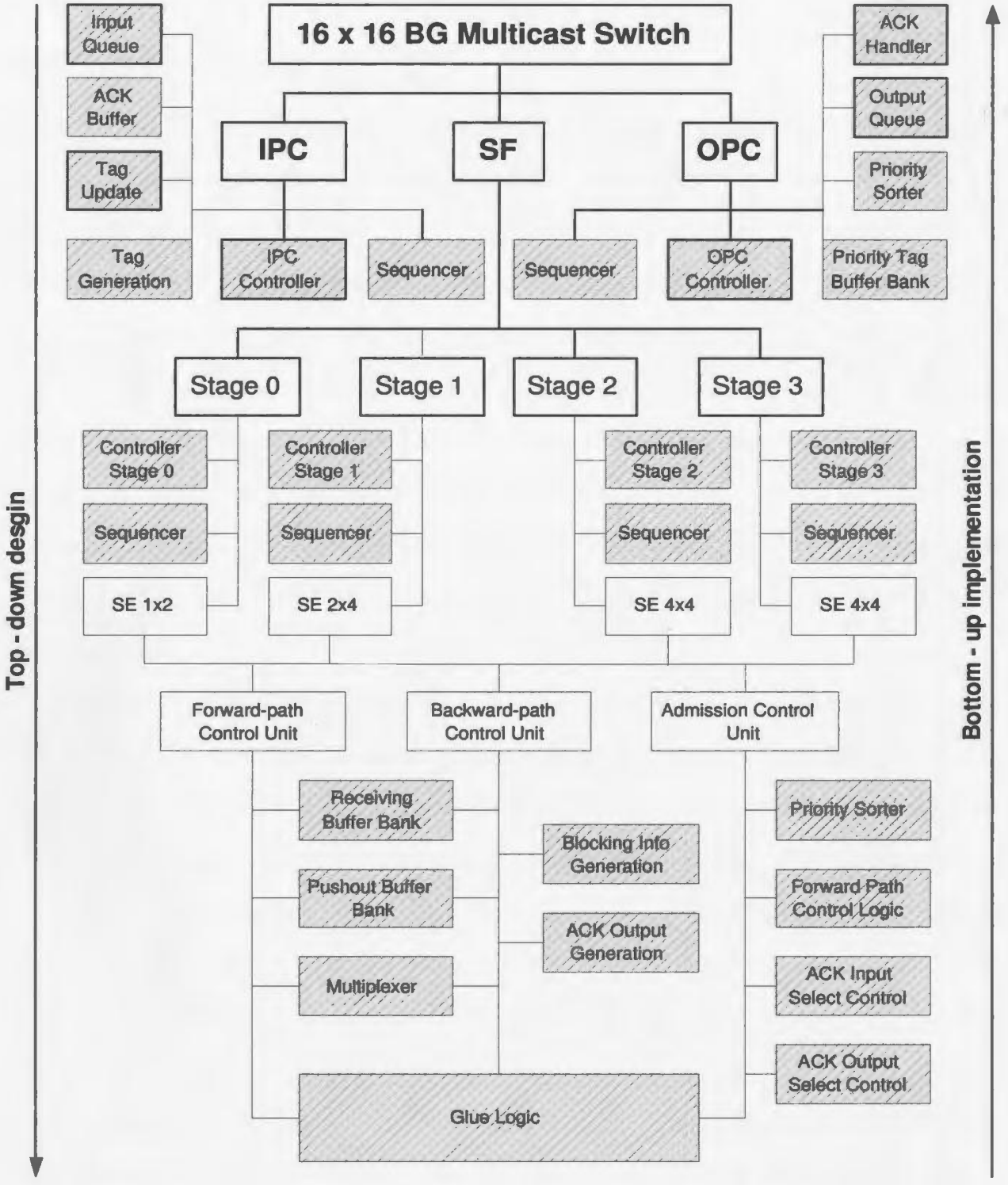


Figure 5.17: Divide-and-Conquer strategy for BG switch design and implementation

can be easily extended to other switch sizes.

5.4.1 Hardware Functional Simulation

SEs at different stages of the implemented 16×16 BG switch are simulated to ensure the routing and replication function are performed correctly. Figure 5.18 shows the simulation results for a 4×4 SE in Stage 2. A $5ns$ clock signal is applied. Signal $CURRENT$ gives the current state of the SE. It follows the timing specified in Table 5.1. In this example, there are three active cells arriving at this SE from input 0, 2 and 3. Cell tag and priority are marked on the waveform for clarity. The cell from input 2 is a multicast cell, but because it has the lowest priority, the destination request to the upper link can not be satisfied. The acknowledgement for this cell is a concatenation of the received ACK (01) for the successfully delivered copy and the blocking information from BIG circuit (11) for the blocked copy, which is 1101 to the upstream stage.

To observe the switching function performed by SEs at different stages in the SF, a testing scenario is formed to mimic the interconnection condition so that it is possible to observe how a SE performs through waveform observation. As shown in Figure 5.19, four 4×4 SEs are concatenated back-to-back. Solid lines are used to represent intermediate signals for the forward path, while dotted lines are for the backward path. Figure 5.20 describes how the tag traverses from the input to the output while Figure 5.21 provides the information for the acknowledgement. The four $CURRENT$ signals display the status of each SE when the tag and acknowledgement signals traverse back and forth and demonstrate how adjacent stages communicate. Even though this is not the true case for the real BG switch, however, it helps to demonstrate that the switching functions of SEs at different stages are correct.

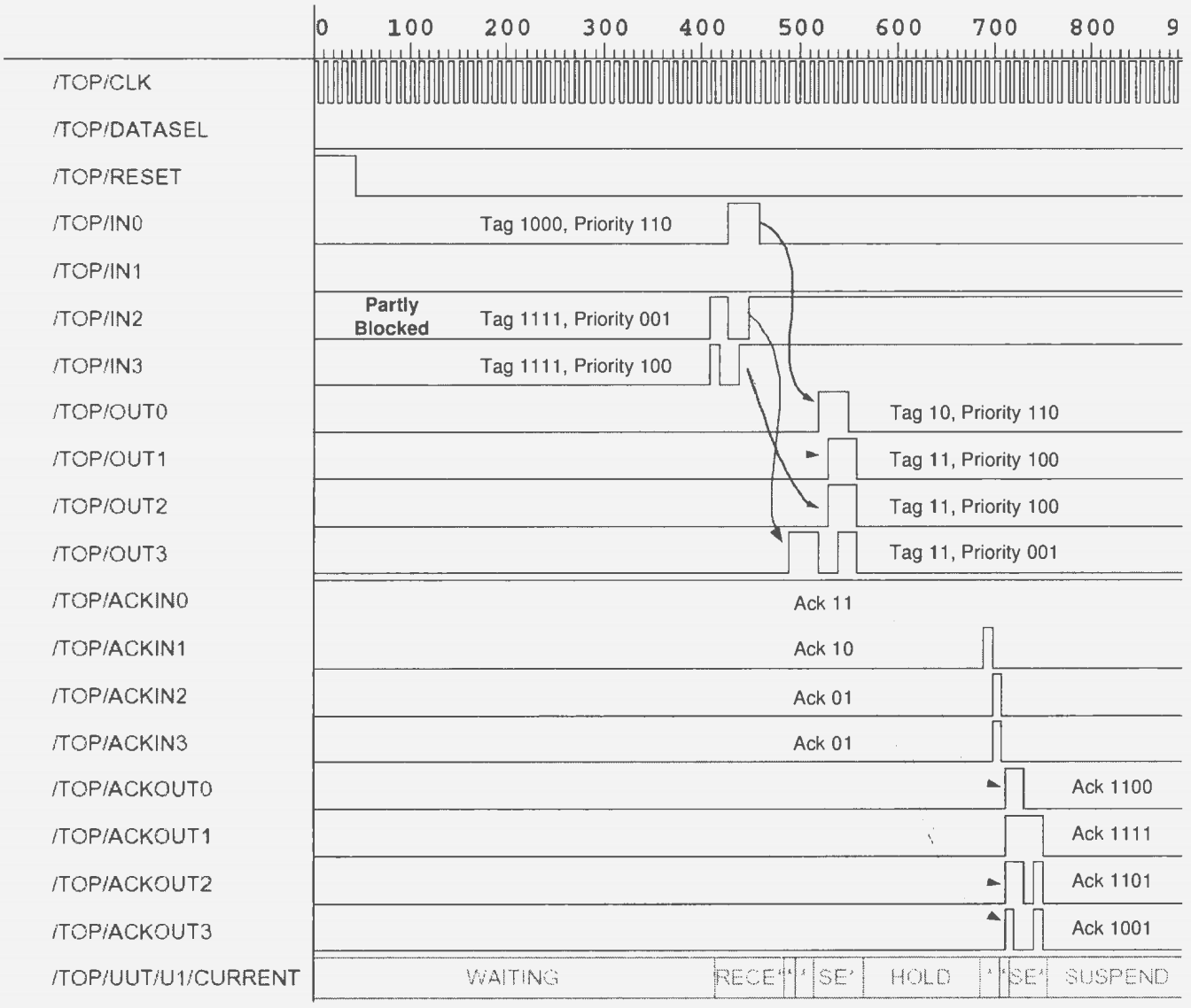


Figure 5.18: Simulation for single SE at $Stage_0$ of 16×16 BG multicast switch

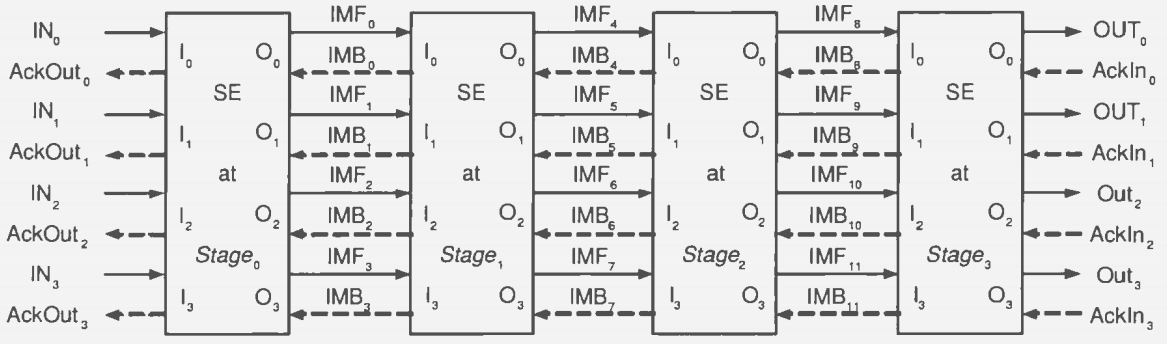


Figure 5.19: Single SE concatenate testing environment

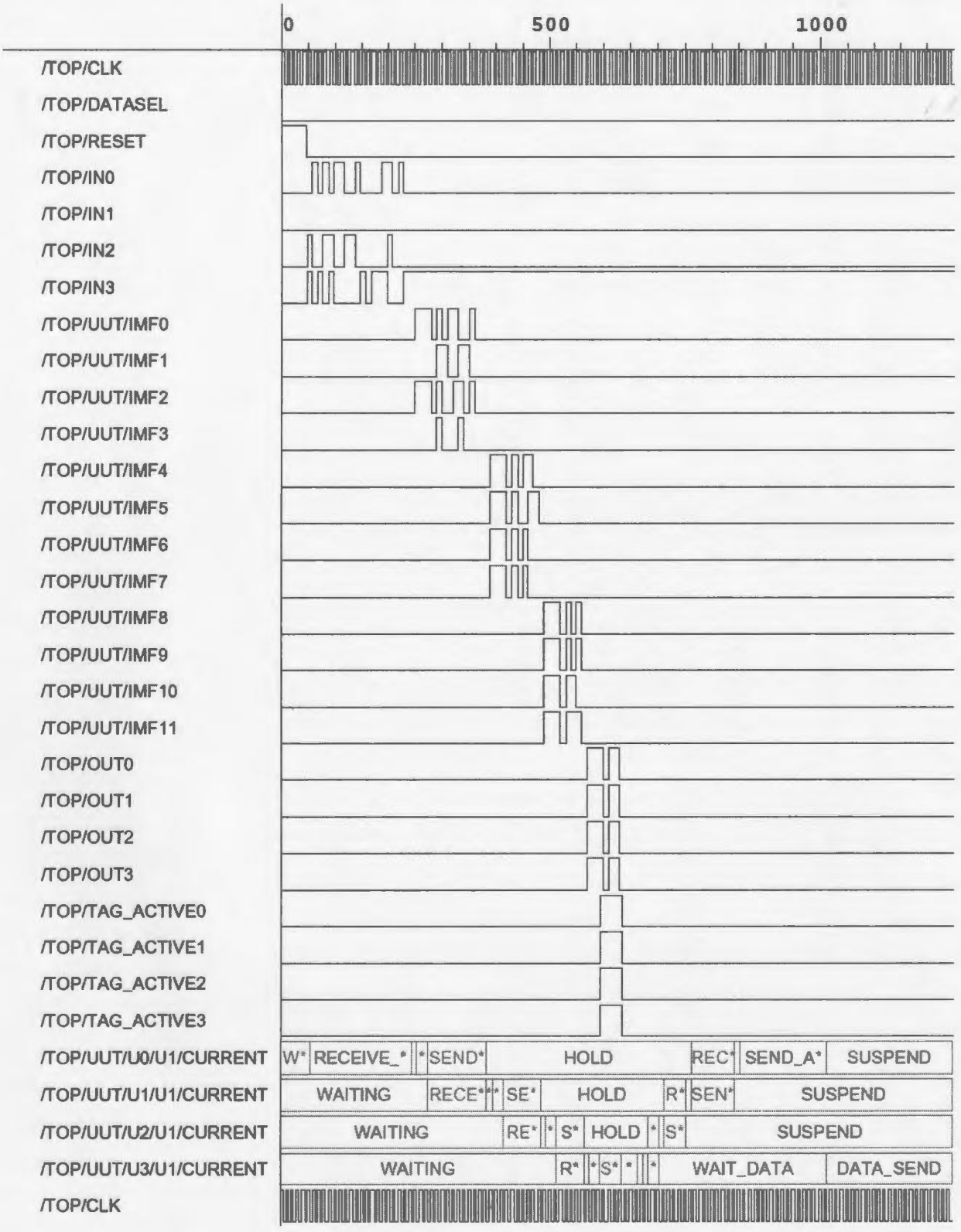


Figure 5.20: Tag transmission in single SE concatenate testing environment

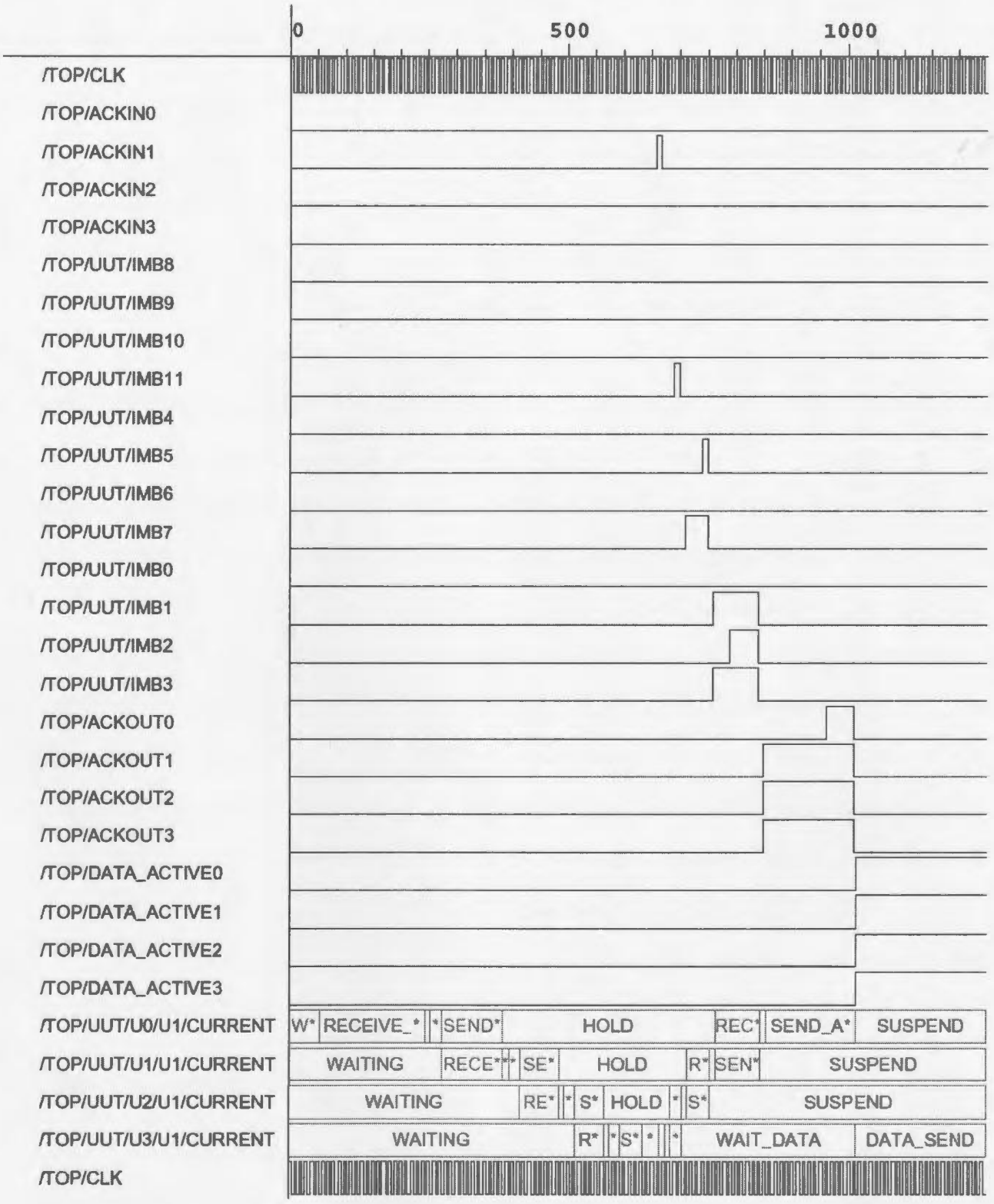


Figure 5.21: Waveform of acknowledgement signal transmission in single SE concatenate testing environment

5.4.2 Testing Environment

For the complete 16×16 multicast BG switch, it is too complex and difficult to conclude correct switching operation through waveform observation. Alternative and automatic verification method should be considered. Since the main interest is on the functional testing of the SF, a combined high-level language and hardware description language test method is used. The testing environment is configured as following:

1. Basic functions which are used to interface the SF are implemented in the simplified IPC. The IPC at each input link only provides a buffering capacity for the HOL cell. The input queue is emulated through data fetching from an external data file by the testbench program. Other functions include acknowledgement receiving and processing, and routing and replication tag loading and updating.

2. Two major functions for the simplified OPC are buffering and acknowledgement. A buffer of one cell size is used to receive the payload. When the complete cell is received, it is then stored in the output queue, which is emulated by an output data file. Note that there are four output lines connected to each OPC. Therefore, the buffer is constructed to be able to accept up to four cells simultaneously in a single switching cycle. The acknowledgement signal is issued by the testbench program at the end of reservation phase depending the remaining output queue space.

3. Testing is made on an 16×16 multicast BG switch. The bitmap tag encoding scheme is used. An ATM cell size of 384 bits is used as the payload length.

5.4.3 Testing Methodology

A C/C++ program has been developed to generate cells to be switched by the fabric. The generated data is stored in a data file which emulates the input queue, one for each input port. During each switching cycle, the testbench program checks the acknowledgement output from the IPC. If it is a NAK, that means there is a cell

retained at the HOL position which will be switched again during the next cycle. Otherwise, the testbench will read a new cell from the data file until either all data in the file is read out or the testing time is reached.

To verify the correctness of switching by the hardware, data at three different places are recorded into different output files by the VHDL testbench program. Data recording is performed on a switching cycle basis. The first file, called *payloadOutput*, stores data that appears at each output line of the SF. Note that there are four lines destined to the same OPC. Therefore, for a 16×16 switch, there are $4 \times 16 = 64$ blocks of data, each includes a 1-bit activity field, 3-bit priority field, and 384-bit payload field, to be recorded during each switching cycle. The content of the output queue is a subset of this data file. The second data file is called *payloadHOL*. It is used to record the 384-bit payload at the HOL position for every input line during each switching cycle. The third file is used a keep a record the HOL cell's destination request at the beginning of each switching cycle and is named as *tagHOL*.

When simulation is completed, the three data files are collected and sent to a C/C++ program for analysis and verification. In this program, each input link uses an array called *tagArray* to trace copy delivery status of the HOL cell, where each array element represents an output port. During each switching cycle, all *tagArrays* are initialized to zero and the delivered cell in the *payloadOutput* file is read out. If it is an active cell, then the payload is used to compare with the HOL payloads at different input lines of the same cycle, which is recorded in the *payloadHOL* file. When they are matched, the *tagArray* of that input line is updated for the delivered cell. This process continues until all the output lines are checked, in the case of the 16×16 SF, there are 64 rounds of output matching for each switching cycle.

Since the payload is randomly generated, there exists a possibility that two or more HOL cells have the same payload so that the matching results may not be

correct. However, the probability that 16 such 384-bit number are exactly the same is negligible. And for the rare case that might occur, the software will simply choose a different payload value because our purpose is to test the switching function of the fabric, the content of the payload does not help.

Once the output matching is completed, each *tagArray* is updated with the delivered cell(s) sent from its input line. For each *tagArray*, it is ANDed with the corresponding HOL cell tag recorded in the *tagHOL* file. If the resulting array elements are all zero, the next cell from the input queue file is read out and used as the HOL cell for the next switching cycle, hence the next data tag from the *tagHOL* file for this input should be the same as the tag for the next cell. Otherwise, if the resulting *tagArray* elements are not all zero, the HOL cell should be retained and switched again, therefore, the next data in the *tagHOL* file for the input should the same the resulting tag array. Whenever there is an mismatching, an error indication will be generated by the C/C++ program.

This process repeats for every switching cycle until all the data is processed. In our testing, a total of 100 cells were generated for each input queue file. So, the switch is loaded with 100% input load at least for the first 100 switching cycles. Because the fanout for each HOL cell is randomly generated, the switch is significantly overloaded. It takes much longer time than 100 switching cycles to transfer all the cells. However, this case could occur when the switch is momentarily overloaded. The experiment proves that the hardware design of the SF is correct.

5.5 Hardware Complexity and Timing

Hardware complexity and timing are the two most important measures to evaluate a hardware design. The results are collected from the Synopsys synthesis tool

DesignAnalyzer, using the library targeting at 0.18 μm CMOS technology. Table 5.2 presents the hardware complexity of the 4×4 SE at different stages as well as it subcomponents. The synthesized circuit, which is in square microns (μm^2), is converted to gate count for comparison. The two-input nand gate, which is widely used in the literature [87], is used as a reference for the conversion.

To estimate the complexity of larger switches, the three types of SEs used for different stages are first designed and synthesized. The results are provided in Table 5.3, Table 5.4 and Table 5.5 for 1×2 SE, 2×4 SE, and 4×4 SE, respectively. The same type of SE which is used for switches of different sizes will have different complexity. This is because the buffering space for tag and acknowledgement for different switch sizes are different. The hardware complexity of the stage controller is roughly the same for different stages because they have the same number of states. The synthesized result is shown in Table 5.6.

Based on the hardware complexity of SEs and stage controller at various stages, the overall complexity of the SF can be estimated. Table 5.7 and Table 5.8 summarizes the complexity for different stage components, as well as the whole SF of various switch sizes. There will be a minor difference between the estimation and real results. However, because the glue circuit, i.e., some combinational logic circuit which connects the SEs, stage controller and stage component, is quite small, the estimated result should be quite close to the actual implementation.

To demonstrate this, for the 16×16 BG switch, the whole SF is synthesized with *Design_Analyzer* using a clock of 5 ns, and compared to the above estimation. The results for the whole SF directly from the synthesis tool is 112,413 gates, while the estimation is 110,400 gates.

With the $5ns$ clock, the 16×16 SF can run at the speed of 200 Mbps. For other switch sizes, SE and stage controller at different stages are synthesized using a

Stage and SE Type	**Forward-path Control Unit**						**Backward-path Control Unit**						**Admission Control Unit**				**Overall SE Complexity**	
	Tag Receiving Buffer Bank		Tag Pushout Buffer Bank		Source/Path Select MUXs		Ack Receiving Buffer Bank		BIG Circuitry + AOG Circuitry		Ack Pushout Buffer Bank		Bitonic Sorter		Control Signals			
	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.	Comb.	Seq.
$Stage_0$ 1×2 SE	5.7	141.7	44.7	161.3	19.3	0.0	1.3	117.3	91.0	0.0	43	117.3	288.7	0.0	344.3	0.0	838.0	537.7
$Stage_1$ 2×4 SE	9.0	166.0	33.3	0.0	56.0	205.3	2.7	117.3	168.7	0.0	42.7	117.3	404.7	0.0	452.3	0.0	1169.3	606.0
$Stage_2$ 4×4 SE	17.3	206.7	74.7	0.0	80.0	293.3	2.7	58.7	104.3	0.0	44.0	117.3	434.0	0.0	530.0	0.0	1287.0	671.0
$Stage_3$ 4×4 SE	24.7	143.3	106.7	0.0	64.0	234.7	2.7	29.3	57.7	0.0	22.7	58.7	455.0	0.0	480.0	0.0	1213.3	466.0

Table 5.2: Hardware complexity of the various SE type and its subcomponents for 16×16 BG switch

Switch size	Number of stages	Combinational part	Sequential part	Total gates
4×4	2	727.0	184.3	911
8×8	3	743.3	303.0	1046
16×16	4	838.0	537.7	1376
32×32	5	1065.3	1007.0	2072
64×64	6	1524.3	1946.3	3471
128×128	7	2339.3	3818.9	6158
256×256	8	3657.0	7577.5	11235
512×512	9	6060.6	15086.7	21147
1024×1024	10	11274.9	30104.4	41379

Table 5.3: Hardware complexity of 1×2 SE of initial stage for various switch sizes

Switch size	Number of stages	Combinational part	Sequential part	Total gates
8×8	3	966.0	367.3	1333
16×16	4	1169.3	606.0	1775
32×32	5	1407.6	1074.6	2482
64×64	6	1980.3	2014.0	3994
128×128	7	3235.0	3885.9	7121
256×256	8	6295.9	7649.9	13946
512×512	9	10307.5	15155.1	25463
1024×1024	10	19700.0	30199.1	49899
2048×2048	11	17778.8	60209.5	77988

Table 5.4: Hardware complexity of 2×4 SE at second front stage for various switch sizes

Switch size	Number of stages	Combinational part	Sequential part	Total gates
8×8	3	1213.3	466.0	1679
16×16	4	1287.0	671.0	1958
32×32	5	1599.6	1090.3	2690
64×64	6	2064.3	1911.3	3976
128×128	7	2701.0	3556.6	6258
256×256	8	5611.6	6843.9	12456
512×512	9	10664.8	13428.8	24094
1024×1024	10	27367.6	27583.3	54951
2048×2048	11	43575.0	53239.7	96815
4096×4096	12	94263.6	107150.8	201415

Table 5.5: Hardware complexity of 4×4 SE at different stages for various switch sizes

Switch size	Number of stages	Combinational part	Sequential part	Total gates
2×2	1	282.0	171.3	453
4×4	2	292.0	172.7	465
8×8	3	232.7	182.0	415
16×16	4	284.7	171.3	456
32×32	5	300.3	193.0	493
64×64	6	224.0	196.0	420
128×128	7	256.3	190.3	447
256×256	8	283.3	203.3	487
512×512	9	270.7	193.0	464
1024×1024	10	214.6	188.3	503

Table 5.6: Hardware complexity of different stage controller for various switch sizes

Switch Switch Fabric Switch	Stage 9	Stage 8	Stage 7	Stage 6	Stage 5
8×8	-	-	-	-	-
16×16	-	-	-	-	-
32×32	-	-	-	-	-
64×64	-	-	-	-	222539.0
128×128	-	-	-	788699.4	911893.0
256×256	-	-	2876515.2	3570562.6	3189018.5
512×512	-	10827890.1	13037323.9	12336366.6	6377617.0
1024×1024	42372921.1	51097193.7	56270124.1	24672286.6	12754814.0
Switch Fabric Size	**Stage 4**	**Stage 3**	**Stage 2**	**Stage 1**	**Stage 0**
8×8	-	-	8785.2	11131.1	13887.8
16×16	-	22466.3	28819.5	31792.1	27322.2
32×32	66806.9	79889.3	86493.2	63119.6	54191.1
64×64	256126.3	254894.3	172571.7	125774.6	107928.8
128×128	801461.3	509332.7	344728.8	251084.4	215404.3
256×256	1602429.4	1018209.4	689042.2	501704.2	430355.3
512×512	3204365.1	2035962.8	1377671.0	1002943.8	860257.2
1024×1024	6408236.9	4071469.5	2754927.4	2005422.9	1720061.1

Table 5.7: Hardware complexity for stage components of various switch sizes

Switch Fabric Size	Estimated Hardware Complexity
8×8	33804.1
16×16	110400.1
32×32	283693.2
64×64	661169.4
128×128	1320550.2
256×256	2639311.7
512×512	5276834.8
1024×1024	10551881.0

Table 5.8: Estimated hardware complexity for switch fabric of various switch sizes

clock of 5 ns. With this clock rate, it has been observed that no positive slack has been generated from the report file. Therefore, the SF can comfortably run at 200 Mbps link speed, which is enough for the targeted OC-3 links. With more advanced technologies, such as the $0.13\mu m$ and $0.09\mu m$ CMOS technology, the speed that the design can support will be even higher.

5.6 Summary

In this chapter, the digital system design methodology and the digital IC design flow recommended by CMC are studied. The complete design process is explored for the multicast BG switch using the 0.18 μm CMOS technology. Testing methods are discussed and verification software is provided for large designs such as the complete switch fabric. Synthesis results are provided for switches of different sizes as well as for the each subcomponent inside the switch element. The implementation results indicate that the core of a 16×16 multicast BG switch fabric can be easily fabricated into a single IC chip and can comfortably run at OC-3 link speed, which yields a switching capacity of closing to 3 Gbps for the overall switch.

Chapter 6

Conclusion and Future Work

This dissertation has described the design of a new multicast switch architecture, analyzed its performance through analytical modelling and simulation, and demonstrated results from the hardware implementation of the switch fabric module.

6.1 Summary of Thesis

The main contributions of this research are summarized as follows:

- **New Multicast Switch Fabric Architecture**

  A new multicast Balanced Gamma switch architecture is proposed. The switch fabric adopts a MIN structure. Instead of using a dedicated copy network, multicast cell replication, which is the key characteristic of a multicast switch, is handled implicitly through each switch element along with the routing function. The architecture design is described in detail in Chapter 3 in which the distributed control and modular structure are highlighted. The priority switching capability provides the switch with the flexibility in handling traffic with different quality of service requirements. It is also demonstrated that the proposed switch architecture is scalable and can be made fault-tolerant when equipped with proper fault detection mechanisms.

- **Two Important Algorithms**

  The concept of self-routing is widely utilized in switches using a MIN structure, in which the control of switching operation is distributed over the switch elements and handled in a parallel fashion. To handle cell mulitcasting inside the switch fabric, dynamic-length routing and replication algorithm is developed to combine self-routing with self-replication. Using this algorithm, multicast cell replication decisions can be made on the fly based on the routing tag pair along with routing decisions. To be efficient, the tag is halved before being passed to the next stage. That is where the term "dynamic-length" tag comes from.

  Collecting the acknowledgement information in the multicast environment is challenging because cell replication can take place at any stage, and cell blocking can occur at any time as well. The acknowledgement information should reflect the delivery status of all copies of the multicast cell. To obtain such information properly and effectively, a dynamic-length backpressure algorithm is designed. The algorithm works like a converging tree in which only a simple concatenate operation is required. Therefore, the information can be processed on the fly.

- **Multicast Traffic Model**

  An important part of the performance analysis is the study and development of a multicast traffic model. The traffic model can be described using three random processes: arrival, fanout distribution and destination selection. Combination of different choices for these three processes produces different traffic types, as shown in Figure 4.1. Unicast and broadcast traffic can be properly represented using this model as special cases of multicast traffic where the average fanout is one or switch size N, respectively. The nonuniform destination selection model provides a substantial number of hot spots, which better resembles traffic in

real networks, especially when the switch size is large.

- **Analytical Modelling**

  The most significant theoretical contribution of this research is the performance analytical model of the BG switch under non-bursty multicast random traffic. The analysis follows the three-phase switching operation. The cell blocking probability at SEs of different stages is analyzed first. With this information, the cell blocking probability for the whole switch fabric and traffic arrival probability toward each output queue are obtained. In the next step, the output queue is analyzed based on the resulting traffic arrival condition using the discrete-time Markov chain. The cell blocking probability, average number of cells in the queue and average queueing delay for the output queue can be obtained. Then, the overall cell blocking probability for the combined switch fabric and output queue is calculated, which provides the probability of cells being kept in the HOL position of the input queue. Finally, the input queueing analysis is performed to get the loss performance, delay performance, and buffer requirement performance of the input queue as well as the whole switch.

- **High Performance of the Multicast BG Switch**

  A comprehensive study is conducted in Chapter 4 to evaluate the performance of the BG switch. The analysis has demonstrated that the multicast BG switch is an outstanding switch architecture under a wide range of traffic loads and various types of multicast random, bursty, and nonuniform traffic. The analytical model is used to verify simulation results under multicast random traffic. For bursty and nonuniform traffic conditions, simulation is used to obtain the performance measures. The results are compared to that of a hypothetical ideal multicast switch, which has the best possible performance, as well as two other

multicast switches. It has been observed that in all the above traffic types and conditions, the performance of the multicast BG switch is very close to that of the ideal multicast switch and its performance scales very well along with the switch size. This is important because we not only want a switch architecture that has a performance close to ideal, but also has a scalable architecture so that larger switches can be realized easily using small switches as construction modules. Modular structure and architecture scalability will become meaningless if the performance deteriorates too much as the size scales.

Throughout the analysis, the impact of various factors on switch performance are studied, including traffic load, switch size, traffic fanout, burstiness, and destination request distribution. In particular, performance analysis focuses on traffic fanout and burstiness to demonstrate how switch performance is affected in multicast environments. Average burstiness indicates the level of traffic correlation. As traffic gets more correlated, the internal blocking becomes more severe. Switch performances such as cell loss ratio, cell delay, input and output buffer requirement will be significantly affected. Larger mean fanout means that more copies will be indicated in cell headers, which implies that the traffic load at the switch input is reduced. Since replication is performed internally as cells traverse the switch fabric, less blocking will be encountered. Therefore, the output queueing delay and buffer requirement of the BG switch are very close to those of the ideal switch. At the switch input, a larger fanout means that a cell takes a longer time to be dequeued, which counteracts the effect of a lighter load, resulting in the input queue performance being little changed. As was concluded in Chapter 4, the reason for the high performance of the multicast BG switch is the design choice of accepting up to four cells in one switching

cycle at the output queue.

- **Realizable and Scalable Switch Architecture**

  Besides all the attractive features that have been concluded above, in the research, I demonstrate the feasibility of realizing the proposed switch using the existing CMOS technology. A comprehensive study of the front-end design of multicast BG switch has been performed in Chapter 5. The modular design and the implementation scalability of the BG switch has enabled us to focus on the basic building blocks, i.e., the switch elements, and then interconnect them to construct higher switch fabric module in the design hierarchy. A general testing method is proposed to facilitate automatic function verification for the complete switch fabric. The implementation results based on the $0.18\mu m$ CMOS technology indicate that the core of a 16×16 multicast BG switch fabric can be easily fabricated into a single IC chip and can comfortably run at OC-3 link speed, which yields a switching capacity of closing to 3 Gbps for the overall switch.

6.2 Future Research

Even though a comprehensive study of the multicast BG switch has been completed in this dissertation, there are still several areas to be further explored.

Firstly, in the dissertation, an analytical model has been developed to obtain the performance measures of the multicast BG switch under multicast random traffic. In addition to an effective practice in the switch architecture study, the analytical model has provided a good method to verify the simulation results and predict the switch performance. Even though the two bounds from the analytical model can provide strict sense boundaries for any traffic condition, the range is too wide and it does

not reflect the performance change for traffic fanout. It is highly desired that a more precise model to be developed so that switch performance for different random traffic conditions can be better predicted.

Secondly, as traffic in real high-speed networks behaves differently from random traffic, the limitation of the proposed analytical model is obvious. Therefore, it is worthwhile to extend the modelling work to the bursty traffic condition, especially when a nonuniform destination selection distribution is considered. Moreover, it is desired that the analytical model reflect the dynamic information exchange inside the switch fabric along with the long term effect.

Hardware implementation has become a general requirement for high-speed switches and routers. However, inflexibility is normally a problem associated with hardware design. As what has been shown from the analysis, performance of the BG switch and the demands of resources under different applications and traffic conditions vary significantly. Therefore, a high performance switch which incorporates a dynamically reconfigurable architecture will possess a noticeable advantage as the network traffic and applications continuously drive hardware performance requirements to higher levels. The reconfigurable architecture can also be used together with the scheduling mechanism in the port controllers to provide a dynamic buffering resource allocation.

Priority routing is an important feature for the multicast BG switch. By associating different priority levels with different traffic sources or network applications, different QoS requirements can be fulfilled. Priority routing can be considered as a type of scheduling mechanism used inside the SF. In the performance analysis, cell priority is not considered. But up to 8 priority levels have been designed to be supported in the switch hardware and this number can be easily expanded with slight modifications in the SE. It would be interesting to investigate how switch performance will be affected when cell priority is considered. It would be another interesting topic

to study how priority routing can cope with the scheduling mechanism located at the port controllers to improve the scheduling capability of the switch.

Buffer sharing is a well-known approach to reduce the buffering requirement and improve switch performance. However, it is at the cost of incorporating a complex control mechanism and higher buffer speed requirement as the level of sharing increases. From performance comparison with the Abacus switch in Chapter 4, it has been observed that a main reason for its high performance is that the output buffering resources are fully shared by all M output ports belonging to the same group; the value of M used in the example is 16. Since the multicast BG switch is mainly an output-buffered switch, it is worthwhile to investigate how its performance and per-output-link buffer requirement will be affected as the level of output buffer sharing increases, and identify the most cost-effective approach when both performance and implementation issues need to be addressed.

Finally, since the multicast BG switch is a strong candidate for use in the future high-speed networks, it is worthwhile to prototype the multicast BG switch and test its performance by using the traffic specified in the switch fabric benchmark framework. It would be a significant step toward in the research to integrate a working switch fabric with two fully-functioning port controllers, and thus amplify the present research effort by a study of a working switching system.

References

[1] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling Internet Routers Using Optics," tech. rep., Stanford HPNG Technical Report TR03-HPNG-080101, May 2003.

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

[3] R. Y. Awdeh and H. Mouftah, "Survey of ATM Switch Architectures," *Computer Networks and ISDN Systems*, vol. 27, pp. 1567–1613, November 1995.

[4] A. Huang and S. Knauer, "STARLITE: a Wideband Digital Switch," in *Proceedings of Global Telecommunications Conference (GLOBECOM'84)*, pp. 121–125, December 1984.

[5] K. Y. Eng, M. G. Hluchyj, and Y. S. Yeh, "Multicast and Broadcast Services in a Knockout Packet Switch," in *Proceedings of IEEE INFOCOM'88*, pp. 29–34, 1988.

[6] M. H. Guo and R. S. Chang, "Mulitcast ATM Switches: Survey and Performance Evaluation," *ACM SIGCOMM Computer Communication Review*, vol. 28, pp. 98–131, April 1998.

[7] T. T. Lee, "Nonblocking Copy Networks for Multicast Packet Switching," *IEEE J. Select. Areas Commun*, vol. 6, pp. 1445 -1467, December 1988.

[8] D. S. Kim and D. Z. Du, "Performance of Split Routing Algorithm for Three-Stage Multicast Networks," *IEEE/ACM Trans. Networking*, vol. 8, no. 4, pp. 526–534, 2000.

[9] K. L. E. Law and A. Leon-Garcia, "A Large Scalable ATM Multicast Switch," *IEEE Journal on Selected Area in Communications*, vol. 15, pp. 844–854, July 1997.

[10] I. Elhanany and V. Tabatabaee, "Benchmarking Next-Generation Switch Fabrics," *IEEE Computer Magazine*, pp. 109–110, October 2003.

[11] Canadian Microelectronics Corporation, *Tutorial on CMC's Digital IC Design Flow*, October 2002. Document ICI-096.

[12] C. Metz, *IP Switching: Protocols and Architectures.* New York: McGraw Hill, 1998.

[13] M. Guizani and A. Rayes, *Designing ATM Switching Networks.* New York: McGraw Hill, 1999.

[14] Yaser El-Sayed, *Performance Analysis, Design and Reliability of the Balanced Gamma Network.* PhD thesis, Memorial University of Newfoundland, 2000.

[15] M. Schwartz, *Broadband Integrated Networks.* New Jersey: Prentice Hall, 1998.

[16] T. Harrington, "Network News: Journey's End for ATM?," *http://www.vnunet.com/Features/1101912*, May 2000.

[17] K. Ahmad, *Sourcebook of ATM and IP Internetworking.* New Jersey: IEEE Press, 2002.

[18] A. Pattavina, *Switching Theory: Architecture and Performance in Broadband ATM Networks.* New York: Wiley, 1998.

[19] R. Wittmann and M. Zitterbart, *Multicast Communication Protocols and Applications.* San Francisco: Morgan Kaufmann Publishers, 2000.

[20] M. Banikazemi and R. Jain, "IP Multicasting: Concepts, Algorithms, and Protocols; IGMP, PRM, CBT, DVMRP, MOSPF, PIM, MBONE," tech. rep., Computer and Information Science, Ohio State University, Columbus, Ohio, U.S.A., 1997.

[21] J. Turner, "Design of a Broadcast Packet Switching Network," *IEEE Transactions on Communications*, vol. 36, pp. 734–743, June 1988.

[22] W. T. Chen and T. W. Deng, "PPCN: A High-Performance Copy Network for Large Scale ATM Switching Systems," *IEICE Transactions on Communications*, vol. E82-B, January 1999.

[23] J. N. Giacopelli, J. J. Hickey, W. S. Marcus, W. D. Sincoskie, and M. Littlewood, "Sunshine: a High-Performance Self-Routing Broadband Packet Switch Architecture," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1289–1298, October 1991.

[24] R. Venkatesan and H. Mouftah, "Balanced Gamma Network - A New Candidate for Broadband Packet Switching Architectures," in *Proceedings of the IEEE INFOCOM'92*, vol. 3, pp. 2482–2488, 1992.

[25] Harinath Sivakumar, "Performance, Fault Tolerance and Reliability of Multistage Interconnection Networks for Broadband Packet Switch Architectures," Master's thesis, Memorial University of Newfoundland, 1995.

[26] R. Venkatesan, Y. El-Sayed, R. Thuppal, and H. Sivakumar, "Performance Analysis of Pipelined Multistage Interconnection Networks," *Informatica – International Journal of Computing and Informatics*, vol. 23, Spetember 1999.

[27] Y. El-Sayed, R. Venkatesan, and H. Sivakumar, "Fault Tolerance and Reliability Analysis of the Balanced Gamma Network," *International J. of Parallel and Distributed Systems and Networks*, vol. 2, no. 4, pp. 244–254, 1999.

[28] J. Turner and N. Yamanaka, "Architectural Choices in Large Scale ATM Swiches," *IEICE Transactions on Communications*, vol. E81-B, pp. 120–137, February 1998.

[29] H. Ahmadi and W. E. Denzel, "A Survey of Modern High-Performance Switching Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 1091–1103, September 1989.

[30] F. A. Tabagi, "Fast Packet Switch Architectures for Broadband Integrated Services Digital Networks," *Proceedings of the IEEE*, vol. 78, pp. 133–167, January 1990.

[31] H. Suzuki, H. Nagano, T. Suzuki, T. Takeuchi, and S. Iwasaki, "Output-Buffer Switch Architecture for Asyncronous Transfer Mode," in *Proceedings of IEEE ICC'89*, pp. 99–103, June, 1989.

[32] FORE Systems, Inc., "White paper: ATM Switching Architecture," November 1993.

[33] Y. Shobatake, M. Mptpyama, E. Shobatake, T. Kamitake, S. Shimizu, M. Noda, and K. Sakaue, "A One-Chip Scalable 8 x 8 ATM Switch LSI Employing Shared Buffer Architecture," *IEEE Journal of Selected Areas in Communications (JSAC)*, vol. 9, pp. 1248–1254, October 1991.

[34] T. Kozaki, N. Endo, Y. Sakurai, O. Matubara, M. Mizukami, and K. Asano, "32 x 32 Shared Buffer Type ATM Switch VLSI's For B-ISDNs," *IEEE Journal of Selected Areas in Communications (JSAC)*, vol. 9, pp. 1239–1247, October 1991.

[35] Y. S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The Knockout Switch: A Simple, Modular Architecture for High-Performance Switching," *IEEE J. Select. Areas Commun*, vol. 9, pp. 1173–1193, October 1987.

[36] C. Clos, "A Study of Nonblocking Switching Networks," *Bell System Technical Journal*, vol. 32, pp. 406–424, March 1953.

[37] J. Hui, *Switching and Traffic Theory for Integrated Broadband Networks*. Boston: Kluwer Academic Publishers, 1990.

[38] K. Batcher, "Sorting Networks and Their Applications," in *Proceedings of AFIPS Spring Joint Computer Conference*, pp. 307–314, 1968.

[39] V. E. Benes, "Optimal Rearrangeable Multistage Connecting Networks," *Bell Systems Technical Journal*, vol. 36, pp. 1641–1656, June 1964.

[40] F. A. Tobagi, T. Kwok, and F. M. Chiussi, "Architecture, Performance, and Implementation of the Tandem Banyan Fast Packet Switch," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1173–1193, October 1991.

[41] S. C. Liew and T. T. Lee, "N log N Dual Shuffle-Exchange Network with Error-Correcting Routing," in *Proceedings of IEEE ICC'92*, vol. 1, pp. 1173–1193, June, 1992.

[42] S. Nojima, E. Tsutsui, H. Fukuda, and M. Hashimoto, "Integrated Services Packet Network Using Bus-Matrix Switch," *IEEE Journal on Selected Areas in Communications*, vol. 5, pp. 1284–1292, October 1987.

[43] M. A. Marsan, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, "On the Throughput of Input-Queued Cell-Based Switches with Multicast Traffic ," in *Proceedings of IEEE INFOCOM2001*, April 22-26, 2001.

[44] Nicholas W. McKeown, *Scheduling Algorithms for Input-Queued Cell Switches*. PhD thesis, University of California at Berkeley, 1995.

[45] N. F. Mir, "A Survey of Data Multicast Techniques, Architectures, and Algorithms," *IEEE Communications Magazine*, September 2001.

[46] J. Turner, "A Practical Version of Lee's Multicast Architecture," *IEEE Transaction on Communications*, vol. 41, pp. 1166–1169, August 1993.

[47] J. W. Byun and T. T. Lee, "The Design and Analysis of an ATM Multicast Switch with Adaptive Traffic Controller," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 288–298, June 1994.

[48] D. X. Chen and J. W. Mark, "Multicast in the SCOQ Switch," in *Proceedings of IEEE INFOCOM'94*, pp. 406–424, 1994.

[49] W. D. Zhong and K. Yukimatsu, "Design Requirements and Architectures for Multicast ATM Switching," *IEICE Transactions on Communications*, vol. E77-B, November 1994.

[50] H. S. Kim, "Design and Performance of Multinet Switch: A Multistage ATM Switch Architecture with Partially Shared Buffers," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 571–580, December 1994.

[51] W. D. Zhong and Y. Onozato, "A Copy Network With Shared Buffers For Large Scale Multicast ATM Switching," *IEEE Transactions on Networking*, vol. 1, pp. 157–165, April 1993.

[52] M. Hashemi and A. Leon-Garcia, "A Multicast Single-Queue Switch with a Novel Copy Mechanism," in *Proceedings of IEEE INFOCOM'98*, March, 1998.

[53] S. Urushidani, S. Hino, Y. Ohtomo, and S. Yasuda, "A High-Performance Multicast Switch and Its Feasibility Study," *IEICE Transactions on Communications*, vol. E81-B, February 1998.

[54] T. T. Lee, R. Boorstyn, and E. Arthurs, "The Architecture of a Multicast Broadband Packet Switch," in *Proceedings of IEEE INFOCOM'88*, pp. 1–8, 1988.

[55] T. H. Lee and S. J. Liu, "A Fair High-Speed Copy Network for Multicast Packet Switch," in *Proceedings of IEEE INFOCOM'92*, pp. 886–894, 1992.

[56] P. U. Tagle and N. K. Sharma, "A Multicast Switching Network for B-ISDN," in *Proceedings of IEEE ICAAAPP'92*, pp. 916–919, 1995.

[57] P. U. Tagle and N. K. Sharma, "Multicast Packet Switch Based on Dilated Network," *IEICE Transactions on Communications*, vol. E81-B, pp. 258–265, February 1998.

[58] R. Cusani and F. Sestini, "A Recursive Multistage Structure for Multicast ATM Switching," in *Proceedings of IEEE INFOCOM'91*, pp. 1289–1295, 1991.

[59] H. J. Chao and B. S. Choe, "Design and Analysis of a Large Scale Multicast Output Buffered ATM Switch," *IEEE/ACM Trans. Networking*, vol. 3, pp. 126–138, April 1995.

[60] H. J. Chao, B.-S. Choe, J.-S. Park, and N. Uzun, "Design and Implementation of Abacus Switch: A Scalable Multicast ATM Switch," *IEEE Journal on Selected Area in Communications*, vol. 15, pp. 830–843, June 1997.

[61] R. J. F. Vries, "ATM Multicast Connections Using the GAUSS Switch," in *Proceedings of IEEE GLOBECOM'90*, pp. 211–217, 1990.

[62] D. J. Marchok, C. E. Rohrs, and R. M. Schafer, "Multicasting in a Growable Packet (ATM) Switch," in *Proceedings of IEEE INFOCOM'91*, pp. 850–858, 1991.

[63] W.-T. Chen, C.-F. Huang, Y.-L. Chang, and W.-Y. Hwang, "An Efficient Cell-Scheduling Algorithm for Multicast ATM Switching Systems," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 517–525, August 2000.

[64] K.-S. Chan, S. Chan, K. L. Yeung, K.-T. Ko, and E. W. M. Wong, "Clos-Knockout: A Large Scle Modular Multicast ATM Switch," *IEICE Transactions on Communications*, vol. E81-B, pp. 266–275, February 1998.

[65] X. Chen and V. Kurner, "Multicast Routing in Self-Routing Multistage Networks," in *Proceedings of IEEE INFOCOM'94*, pp. 306–314, 1994.

[66] S. Kothari, G. Prabhu, and R. Roberts, "The Kappa Network with Fault-Tolerant Destination Tag Algorithm," *IEEE Transactions on Computers*, vol. 37, pp. 612–617, May 1988.

[67] Y. El-Sayed and R. Venkatesan, "Modeling and Simulation of the Pipelined Balanced Gamma Network," in *Proceeding of the Fourth IEEE International Conference on Electronics, Circuits, & Systems (ICECS'97)*, vol. 1, pp. 97–101, 1997.

[68] H. Sivakumar and R. Venkatesan, "Blocking in Multistage Interconnection Networks for Broadband Packet Switch Architectures," in *Proceedings of the Sixth Annual Newfoundland Electrical and Computer Engineering Conference*, 1995.

[69] N. P. Forum, "Switch fabric benchmarking implementation agreement," *http://www.npforum.org/techinfo/approved.shtml*, July 2003.

[70] H. Hlavacs, G. Kotsis, and C. Steinkellner, "Traffic Source Modeling," tech. rep., Technical Report No. TR-99101, Inst. Of Applied Computer Science and Information Systems, Univ. of Vienna, Austria, 1999.

[71] A. Rueda and W. Kinsner, "A Survey of Traffic Characterization Techniques in Telecommunication Networks," in *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 830–833, May 1996.

[72] Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "On the Implications of Zipf's Law for Web Caching," in *Proceedings of IEEE INFOCOM '99*, March 1999.

[73] S. Q. Li, "Nonuniform Traffic Analysis on a Nonblocking Space-Division Packet Switch," *IEEE Transactions on Communication*, vol. 38, pp. 1085–1096, July 1990.

[74] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast Scheduling for Input-Queued Switches," *IEEE Journal on Selected Area in Communications*, vol. 15, pp. 855–866, June 1997.

[75] A.Pattavina, "Design and Performance Evaluation of a Packet Switch for Broadband Central Offices," in *Proceedings of IEEE INFOCOM'90*, pp. 1252-1259, May, 1990.

[76] M. Hluchyj and M. Karol, "Queuing in High-performance Packet Switching," *IEEE Journal on Selected Area in Communications*, vol. 6, pp. 1587–1597, December 1988.

[77] C. Li, "Modeling, Analysis and Design of the Balanced Gamma Multicast Switch," in *Ph.D. thesis proposal*, Memorial University of Newfoundland, February, 2001.

[78] C. Li, R. Venkatesan, and H. M. Heys, "Performance Evaluation of the Multicast Balanced Gamma (BG) Switch," in *the 2002 International Symposium on Performance Evaluation of Computer and Telecommunication System (SPECTS 2002)*, pp. 118–125, July 2002.

[79] C.-M. Chiang and L. M. Ni, "Multi-Address Encoding for Multicast," *http://citeseer.nj.nec.com/5854.htm*, 1994.

[80] S. H. Dyun and D. K. Sung, "A General Expansion Architecture for Large-Scale Multicast ATM Switches," *IEICE Transactions on Communications*, vol. E80-B, pp. 1671–1679, November 1997.

[81] G. D. Stamoulis, M. Anagnostou, and A. Georgantas, "Traffic Source Models for ATM Networks: A Survey," *Computer Communications*, vol. 17, no. 6, pp. 428–438, 1994.

[82] M. Saleh and M. Atiquzzaman, "Accurate Modeling of the Queueing Behavior of Shared Buffer ATM Switches," *International Journal of Communication Systems*, vol. 12, pp. 297–308, July-August 1999.

[83] L. Kleinrock, *Queueing System.* New York: John Wiley & Sons, 1975.

[84] Waterloo Maple web site: http://www.maplesoft.com/main.shtml.

[85] H. F. Badran and H. T. Mouftah, "Performance of Broadband Integrated Switch Architectures with Input-Output-Buffering under Backpressure Mechanisms," Tech. Rep. 90-7, Queen's University, Kingston, Ontario, Canada, 1990.

[86] Z. Navabi, *VHDL: Analysis and Modeling of Digital Systems.* New York: McGraw Hill, 1998.

[87] J. R. Armstrong and F. G. Gray, *VHDL Design: Representation and Synthesis.* New Jersey: Prentice Hall, 2000.

Appendix A

Balanced Gamma Network Topology (Data Path)

The notation used in the algorithm:

$SE_{i,j}$: Switch Element at the i^{th}position in the j^{th}stage;

$Queue_i$: Queue at the $i^(th)$position in the last stage;

n: Total number of stages of the switch fabric;

N: The size of the switch fabric;

%: Modular operation;

$\lfloor\rfloor$: Floor operation.

IL_i: input link i.

OL_i: output link i.

Algorithm pseudo code:

For all middle stages:

for ($j = 0$; $j < n - 1$; j++) // j is the stage index

for ($i = 0$; $i < N$; i++) // i is the row index

$k = i$ % 4, $l = \lfloor \frac{i}{2^{n-1}} \rfloor$, $\alpha = \lfloor \frac{l}{4} \rfloor + l \times \frac{N}{2^i}$, $\beta = \frac{N}{2^{i+2}}$

if ($2 > k \geq 0$)

connect OL_0 of $SE_{i,j}$ to IL_k of $SE_{\alpha,j+1}$.

connect OL_1 of $SE_{i,j}$ to IL_k of $SE_{\alpha+\beta,j+1}$.

connect OL_2 of $SE_{i,j}$ to IL_k of $SE_{\alpha+2\beta,j+1}$.

connect OL_3 of $SE_{i,j}$ to IL_k of $SE_{\alpha+3\beta,j+1}$.

else if $(4 > k \geq 3)$

connect OL_1 of $SE_{i,j}$ to IL_k of $SE_{\alpha,j+1}$.

connect OL_0 of $SE_{i,j}$ to IL_k of $SE_{\alpha+\beta,j+1}$.

connect OL_3 of $SE_{i,j}$ to IL_k of $SE_{\alpha+2\beta,j+1}$.

connect OL_2 of $SE_{i,j}$ to IL_k of $SE_{\alpha+3\beta,j+1}$.

For last stage $j = n - 1$:

for $(i = 0; i < N ; i{+}{+})$ // i is the stage index

if $(i \ \% \ 2 = 0)$

connect OL_0 of $SE_{i,j}$ to IL_0 of $Queue_i$.

connect OL_1 of $SE_{i,j}$ to IL_1 of $Queue_i$.

connect OL_2 of $SE_{i,j}$ to IL_0 of $Queue_{i+1}$.

connect OL_3 of $SE_{i,j}$ to IL_1 of $Queue_{i+1}$.

if $(i \ \% \ 2 = 1)$

connect OL_0 of $SE_{i,j}$ to IL_2 of $Queue_{i-1}$.

connect OL_1 of $SE_{i,j}$ to IL_3 of $Queue_{i-1}$.

connect OL_2 of $SE_{i,j}$ to IL_2 of $Queue_i$.

connect OL_3 of $SE_{i,j}$ to IL_3 of $Queue_i$.

Appendix B

Balanced Gamma Network Topology (Acknowledgement Path)

The notation used in the algorithm:

$SE_{i,j}$: Switch Element at the i^{th}position in the j^{th}stage;

$Queue_i$: Queue at the $i^(th)$position in the last stage;

n: Total number of stages of the switch fabric;

N: The size of the switch fabric;

%: Modular operation;

$\lfloor\rfloor$: Floor operation.

IL_i: input link i.

OL_i: output link i.

Algorithm pseudo code:

For all middle stages:

for ($j = n - 2$; $j \geq 0$; j-) // j is the stage index

    for ($i = 0$; $i < N$; i++) // i is the row index

        if ($\beta = 0$)

connect acknowledgement OL_0 of $SE_{i,j+1}$ to IL_k of $SE_{\alpha+\beta,j+1}$.

connect acknowledgement OL_1 of $SE_{i,j+1}$ to IL_k of $SE_{\alpha+\beta+1,j+1}$.

connect acknowledgement OL_2 of $SE_{i,j+1}$ to IL_{k+1} of $SE_{\alpha+\beta+2,j+1}$.

connect acknowledgement OL_3 of $SE_{i,j+1}$ to IL_{k+1} of $SE_{\alpha+\beta+3,j+1}$.

else if $(\beta = 1)$

connect acknowledgement OL_0 of $SE_{i,j+1}$ to IL_k of $SE_{\alpha+\beta,j+1}$.

connect acknowledgement OL_1 of $SE_{i,j+1}$ to IL_k of $SE_{\alpha+\beta+1,j+1}$.

connect acknowledgement OL_2 of $SE_{i,j+1}$ to IL_{k-1} of $SE_{\alpha+\beta+2,j+1}$.

connect acknowledgement OL_3 of $SE_{i,j+1}$ to IL_{k-1} of $SE_{\alpha+\beta+3,j+1}$.

For last stage $j = n - 1$:

for $(i = 0; i < N\ ; i{+}{+})$ // i is the stage index

if $(i\ \%\ 2 = 0)$

connect acknowledgement OL_0 of $Queue_i$ to IL_0 of $SE_{i,j}$.

connect acknowledgement OL_1 of $Queue_i$ to IL_1 of $SE_{i,j}$.

connect acknowledgement OL_2 of $Queue_i$ to IL_0 of $SE_{i+1,j}$.

connect acknowledgement OL_3 of $Queue_i$ to IL_1 of $SE_{i+1,j}$.

else if $(i\ \%\ 2 = 1)$

connect acknowledgement OL_0 of $Queue_i$ to IL_2 of $SE_{i-1,j}$.

connect acknowledgement OL_1 of $Queue_i$ to IL_3 of $SE_{i-1,j}$.

connect acknowledgement OL_2 of $Queue_i$ to IL_2 of $SE_{i,j}$.

connect acknowledgement OL_3 of $Queue_i$ to IL_3 of $SE_{i,j}$.

Appendix C

Self Routing and Replication Algorithm

The notation used in the algorithm:

priority[*i*]: An array of size 4, used to hold the input port number according to their priority level. The input associated with the highest cell priority is arranged to the first element. The lowest priority input will be arranged to the last element.

input[*i*]: An array of size 4, used to hold the packet from the input port.

Algorithm pseudo code:

Initialize ports and parameters;

Sort inputs based on the priority;

for ($i = 0$; $i \leq 4$; i++)

    if (*input*[*priority*[*i*]] = multicast cell)

        if (there is an available upper output link)

            Switch the cell to the upper output link;

            Modify tag to the next stage;

            Decrease the number of available upper output link by one.

Else

Generate upper blocking information for the cell;

if (there is an available lower output link)

Switch the cell to the lower output link;

Modify tag to the next stage;

Decrease the number of available lower output link by one.

Else

Generate lower blocking information for the cell;

if ($input[priority[i]]$ = unicast cell to the upper output link)

if (there is an available upper output link)

Switch the cell to the upper output link;

Modify tag to the next stage;

Decrease the number of available upper output link by one.

Else

Generate upper blocking information for the cell;

if ($input[priority[i]]$ = unicast cell to the lower output link)

if (there is an available lower output link)

Switch the cell to the lower output link;

Modify tag to the next stage;

Decrease the number of available lower output link by one.

Else

Generate lower blocking information for the cell;

Appendix D

Dynamic Length Backpressure Algorithm

The notation used in the algorithm:

$input[i]$: An array of size 4, used to hold the packet from the input port.

Algorithm pseudo code:

Initialize ports and parameters;

for ($i = 0$; $i \leq 4$; i++)

    if ($input[i]$ has an active cell)

        if ($input[i]$ is a unicast cell)

            Check blocking information and fanout correctness

            if (this unicast cell is blocked)

                Use blocking information generated at this switch element

            else

                Form blocking information from downstream stages

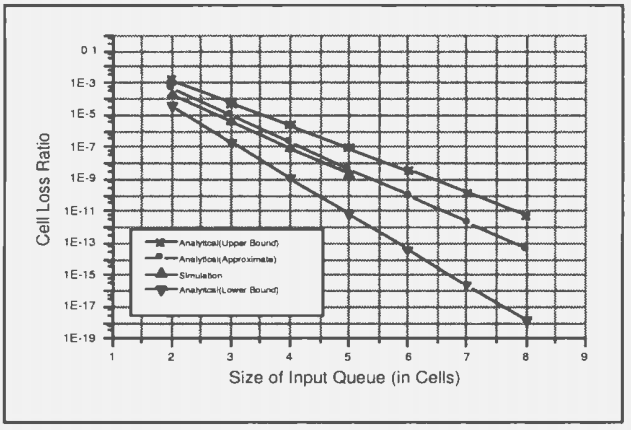
        else if ($input[i]$ is a multicast cell)

            Check blocking information and fanout correctness

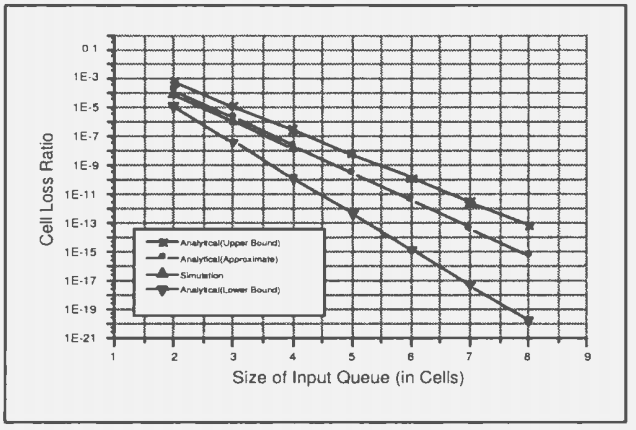
```
if (both links are blocked)
    Use blocking information generated at this switch element
else if (upper link is blocked while lower link gets through)
    concatenate the generated upper link blocking information with the
    generated lower link blocking information from downstream stages
else if (upper link gets through while lower link is blocked)
    concatenate the received upper link blocking information from
    downstream stages with the generated lower link blocking information
else if (both links get through)
    concatenate the received blocking information from both following
    downstream stages
```

Appendix E

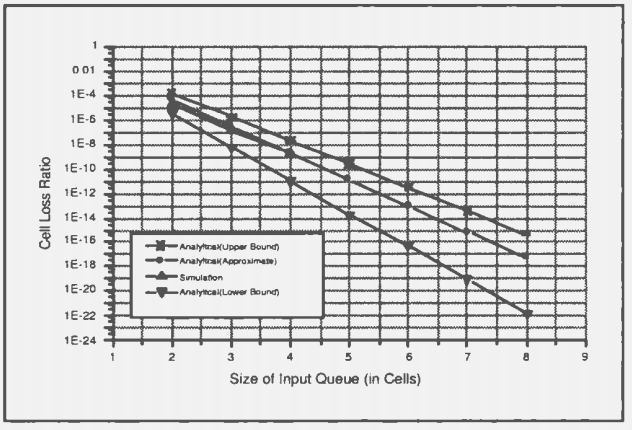
Loss Performance Comparison Between the Analytical Modelling and Simulation Results Under Multicast Random Traffic



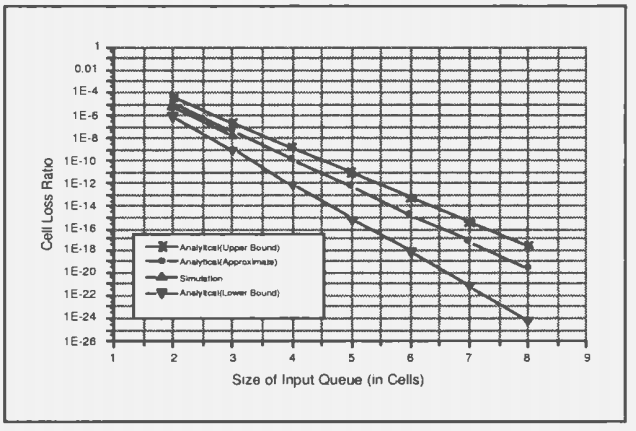
(a)



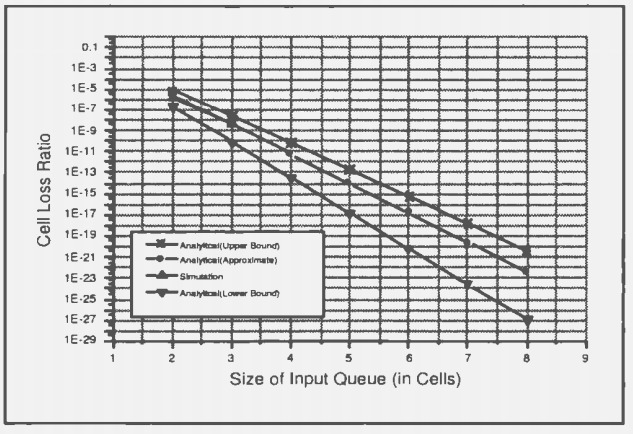
(b)



(c)



(d)



(e)

Traffic Type: Multicast Random Traffic, Mean Fanout = 2

Simulation Time: One Billion Cells

(a) Load = 90%

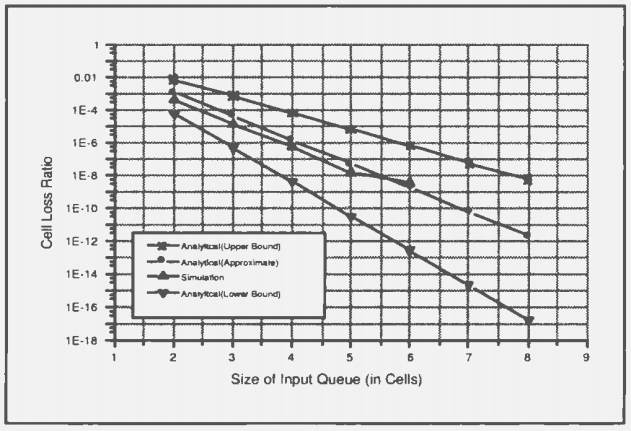
(b) Load = 80%

(c) Load = 70%

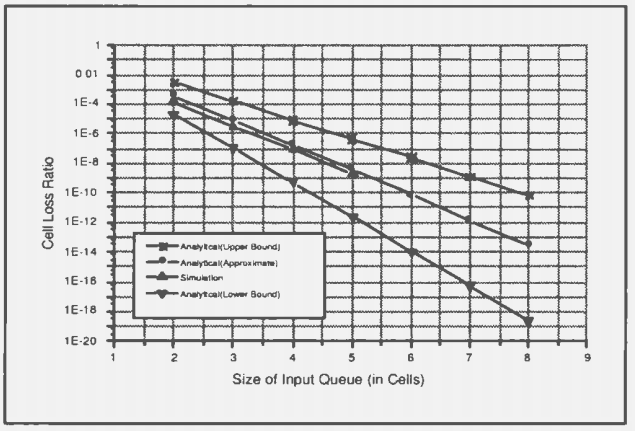
(d) Load = 60%

(e) Load = 50%

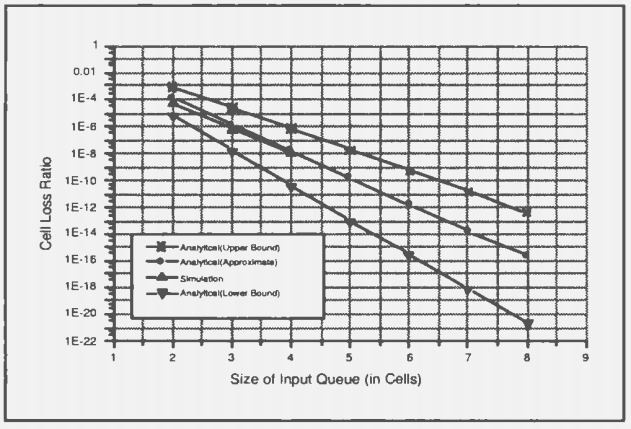
Figure E.1: Loss performance comparison between analytical modelling and simulation results under various loads for 16×16 BG switch



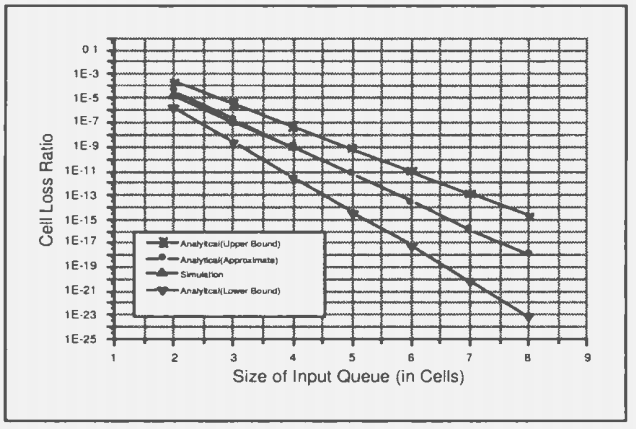
(a)



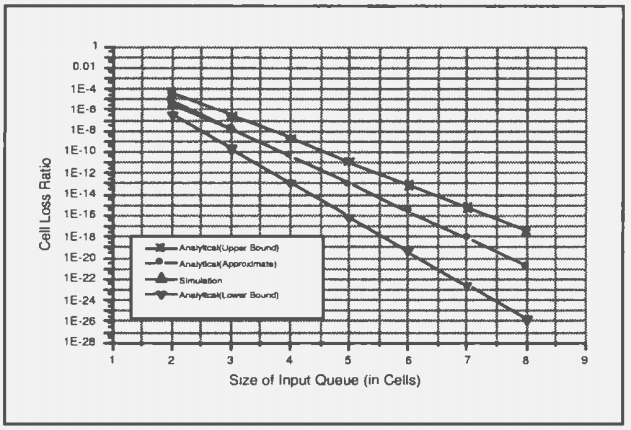
(b)



(c)



(d)



(e)

Traffic Type: Multicast Random Traffic, Mean Fanout = 2

Simulation Time: One Billion Cells

(a) Load = 90%

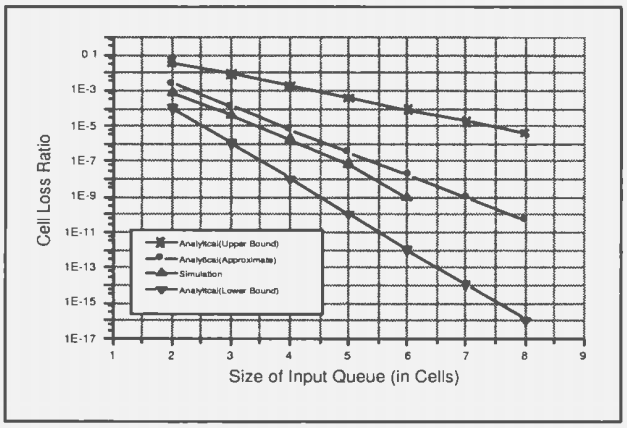
(b) Load = 80%

(c) Load = 70%

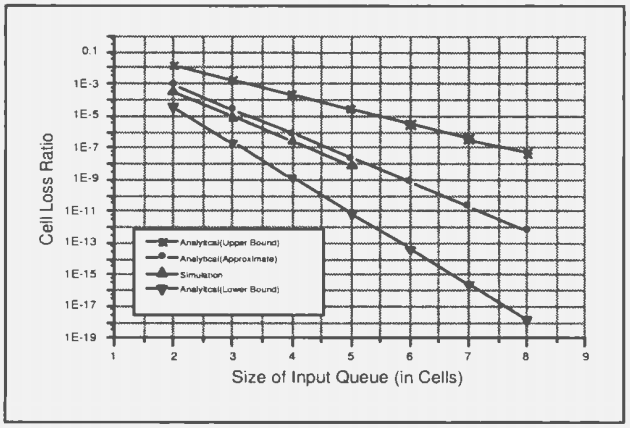
(d) Load = 60%

(e) Load = 50%

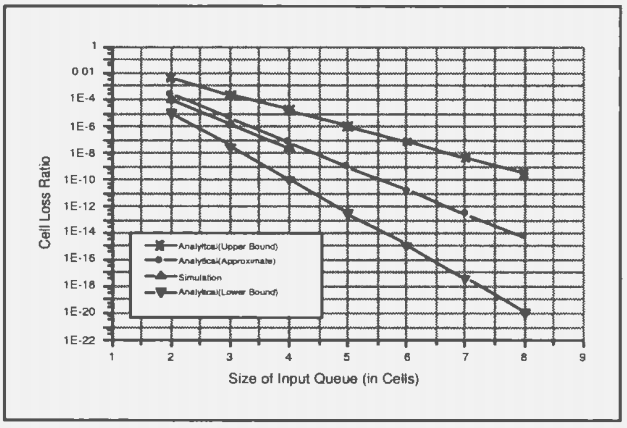
Figure E.2: Loss performance comparison between analytical modelling and simulation results under various loads for 32×32 BG switch



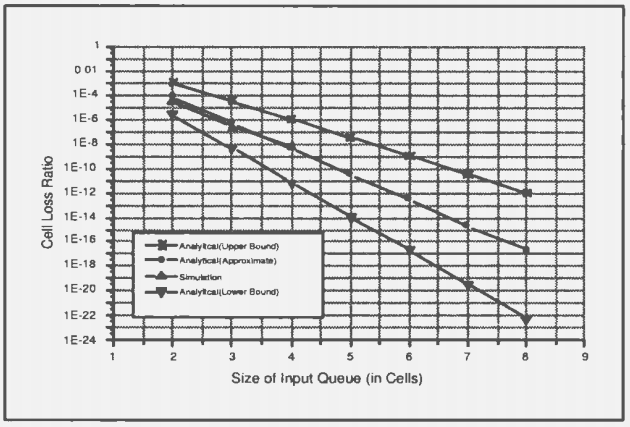
(a)



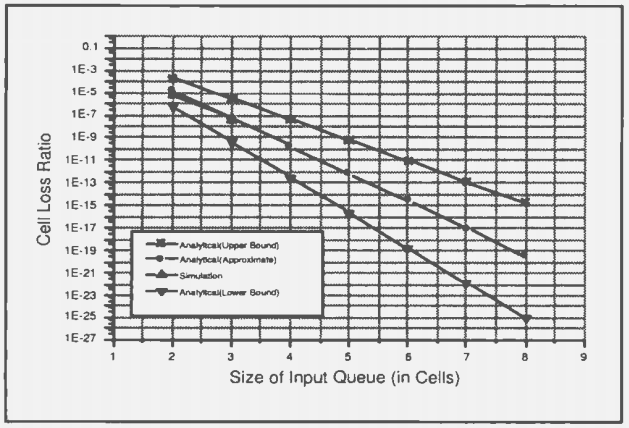
(b)



(c)



(d)



(e)

Traffic Type: Multicast Random Traffic, Mean Fanout = 2

Simulation Time: One Billion Cells

(a) Load = 90%

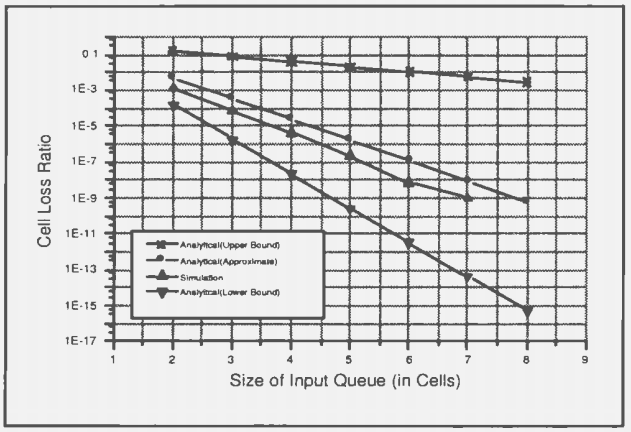
(b) Load = 80%

(c) Load = 70%

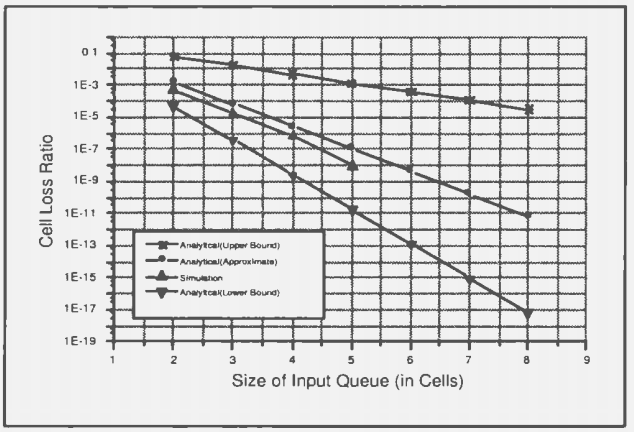
(d) Load = 60%

(e) Load = 50%

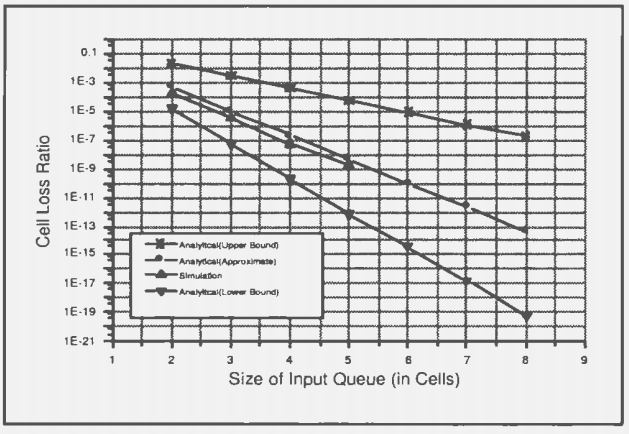
Figure E.3: Loss performance comparison between analytical modelling and simulation results under various loads for 64×64 BG switch



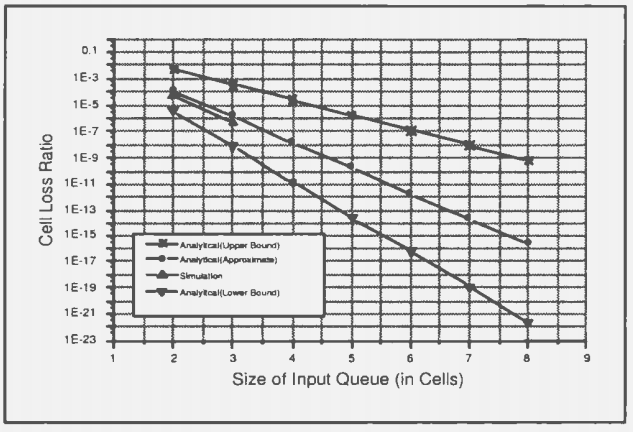
(a)



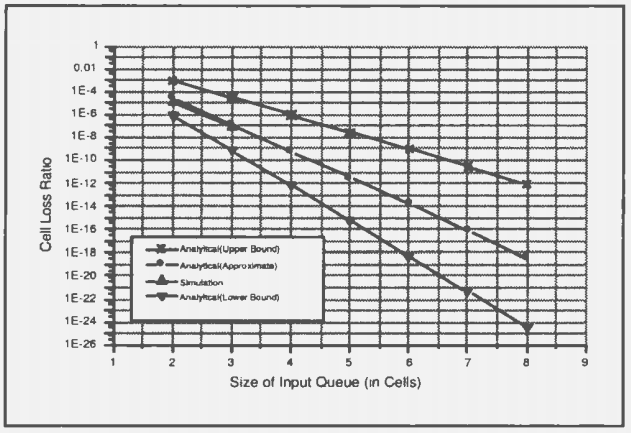
(b)



(c)



(d)



(e)

Traffic Type: Multicast Random Traffic, Mean Fanout = 2

Simulation Time: One Billion Cells

(a) Load = 90%

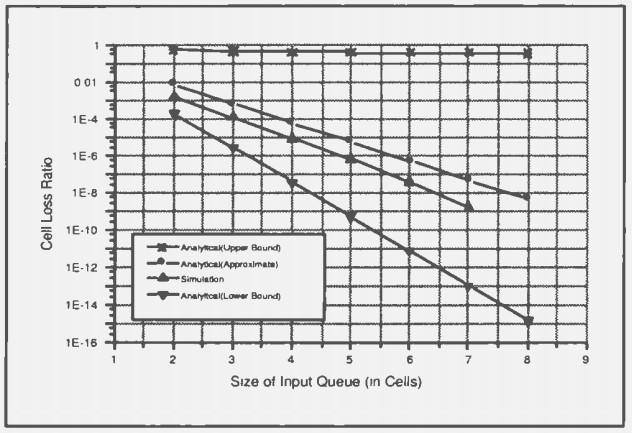
(b) Load = 80%

(c) Load = 70%

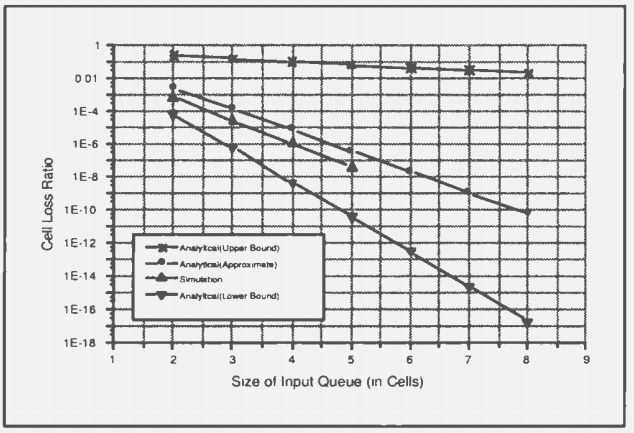
(d) Load = 60%

(e) Load = 50%

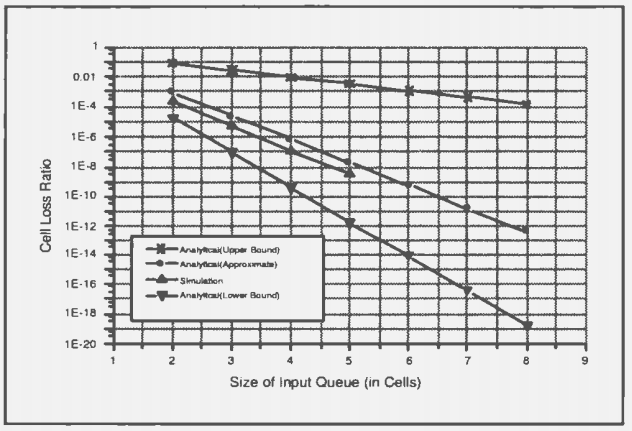
Figure E.4: Loss performance comparison between analytical modelling and simulation results under various loads for 128×128 BG switch



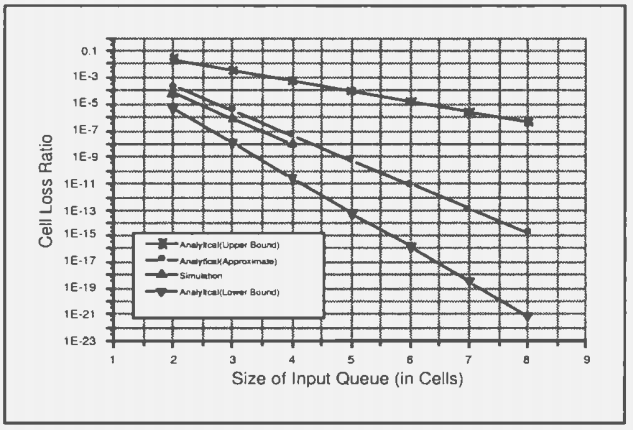
(a)



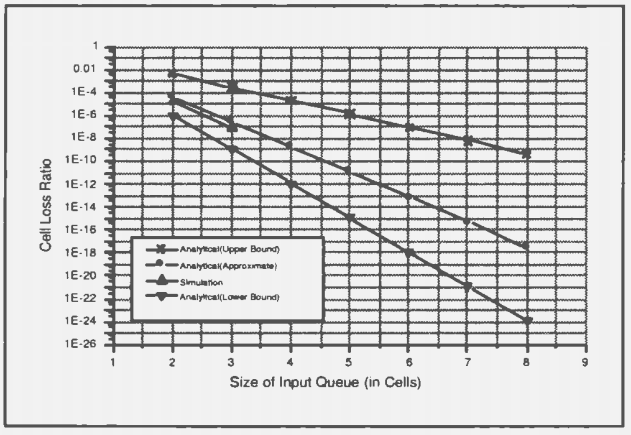
(b)



(c)



(d)



(e)

Traffic Type: Multicast Random Traffic, Mean Fanout = 2

Simulation Time: One Billion Cells

(a) Load = 90%

(b) Load = 80%

(c) Load = 70%

(d) Load = 60%

(e) Load = 50%

Figure E.5: Loss performance comparison between analytical modelling and simulation results under various loads for 256×256 BG switch

Appendix F

Control Signals Generation for Admission Control Unit (ACU) in SEs

Tag Bits		Line		Output Line Status				Forward Path								Backward Path															
Tag_1	Tag_0	S_1	S_0	ST_0	ST_1	ST_2	ST_3	a_0	a_1	b_0	b_1	c_0	c_1	d_0	d_1	$Ctrl_0$	$Ctrl_1$	$Ctrl_2$	$Ctrl_3$	$Actrl_0$	$Actrl_1$	$Actrl_2$	$Actrl_3$	$Actrl_4$	$Actrl_5$	$Actrl_6$	$Actrl_7$	Ba_0	Bb_0	Ba_1	Bb_1	Ba_2	Bb_2	Ba_3	Bb_3
0	0	×	×	×	×	×	×	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	0	0	×	×	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
		0	0	1	0	×	×	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
		0	0	1	1	×	×	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	×	×	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
		0	1	1	0	×	×	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
		0	1	1	1	×	×	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Upper		1	0	0	0	×	×	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Link		1	0	1	0	×	×	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
Request		1	0	1	1	×	×	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Only		1	1	0	0	×	×	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		1	1	1	0	×	×	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
		1	1	1	1	×	×	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		0	0	×	×	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
		0	0	×	×	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0
		0	0	×	×	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	×	×	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
		0	1	×	×	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
		0	1	×	×	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lower		1	0	×	×	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Link		1	0	×	×	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
Request		1	0	×	×	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Only		1	1	×	×	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
		1	1	×	×	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
		1	1	×	×	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table F.1: Truth Table for Control Signals Generation in SE *ACU*, Part I

Tag Bits	Line	Output Line Status	Forward Path	Backward Path
Tag_1 Tag_0	S_1 S_0	ST_0 ST_1 ST_2 ST_3	a_0 a_1 b_0 b_1 c_0 c_1 d_0 d_1	$Ctrl_0$ $Ctrl_1$ $Ctrl_2$ $Ctrl_3$ $Actrl_0$ $Actrl_1$ $Actrl_2$ $Actrl_3$ $Actrl_4$ $Actrl_5$ $Actrl_6$ $Actrl_7$ Ba_0 Bb_0 Ba_1 Bb_1 Ba_2 Bb_2 Ba_3 Bb_3
	0 0	0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
	0 0	1 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0
	0 0	1 1 0 0	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
	0 0	0 0 1 0	0 0 0 0 0 0 0 0	0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0
	0 0	1 0 1 0	0 0 0 0 0 0 0 0	0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0
	0 0	1 1 1 0	0 0 0 0 0 0 0 0	1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0
	0 0	0 0 1 1	0 0 0 0 0 0 0 0	1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
	0 0	1 0 1 1	0 0 0 0 0 0 0 0	1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
	0 0	1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	0 1	0 0 0 0	0 1 0 0 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
	0 1	1 0 0 0	0 0 0 1 0 1 0 0	0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0
	0 1	1 1 0 0	0 0 0 0 0 1 0 0	0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
	0 1	0 0 1 0	0 1 0 0 0 0 0 1	0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0
	0 1	1 0 1 0	0 0 0 1 0 0 0 1	0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0
	0 1	1 1 1 0	0 0 0 0 0 0 0 1	0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0
	0 1	0 0 1 1	0 1 0 0 0 0 0 0	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
1 1	0 1	1 0 1 1	0 0 0 1 0 0 0 0	0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0
	0 1	1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Request	1 0	0 0 0 0	1 0 0 0 1 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0
Both	1 0	1 0 0 0	0 0 1 0 1 0 0 0	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0
Links	1 0	1 1 0 0	0 0 0 0 1 0 0 0	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
	1 0	0 0 1 0	1 0 0 0 0 0 1 0	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1
	1 0	1 0 1 0	0 0 1 0 0 0 1 0	0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0
	1 0	1 1 1 0	0 0 0 0 0 0 1 0	0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0
	1 0	0 0 1 1	1 0 0 0 0 0 0 0	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
	1 0	1 0 1 1	0 0 1 0 0 0 0 0	0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0
	1 0	1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
	1 1	0 0 0 0	1 1 0 0 1 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
	1 1	1 0 0 0	0 0 1 1 1 1 0 0	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1
	1 1	1 1 0 0	0 0 0 0 1 1 0 0	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
	1 1	0 0 1 0	1 1 0 0 0 0 1 1	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1
	1 1	1 0 1 0	0 0 1 1 0 0 1 1	0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1
	1 1	1 1 1 0	0 0 0 0 0 0 1 1	0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0
	1 1	0 0 1 1	1 1 0 0 0 0 0 0	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
	1 1	1 0 1 1	0 0 1 1 0 0 0 0	0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1
	1 1	1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Table F.2: Truth Table for Control Signals Generation in SE *ACU*, Part II