Novel Techniques in Large Scaleable ATM Switches

Prepared by: M. A. Lawrence, Masters Student in the Department of Electrical Engineering.

Prepared for: The Department of Electrical Engineering at the University of Cape Town.

This thesis is submitted to the University of Cape Town in fulfillment of the requirements for the degree of Masters of Science in Electrical Engineering.

Acknowledgements

This work was performed for the Telkom Broadband and ATM Center of Excellence at the Department of Electrical Engineering, University of Cape Town. The Center of Excellence receives sponsorship from Telkom, Siemens and THRIP. I wish to thank them for their support.

Special thanks to Neco Ventura for his help, support and guidance as a supervisor.

Thanks go to my colleagues at the center for their incredible patience while I discussed my ideas, and the other students at UCT for the same reason.

And most importantly to my parents for their continual support and friendship.

Synopsis

This dissertation explores the research area of large scale ATM switches. The requirements for an ATM switch are determined by overviewing the ATM network architecture. These requirements lead to the discussion of an abstract ATM switch which illustrates the components of an ATM switch that automatically scale with increasing switch size (the Input Modules and Output Modules) and those that do not (the Connection Admission Control and Switch Management systems as well as the Cell Switch Fabric). An architecture is suggested which may result in a scalable Switch Management and Connection Admission Control function. However, the main thrust of the dissertation is confined to the cell switch fabric.

The fundamental mathematical limits of ATM switches and buffer placement is presented next emphasising the desirability of output buffering. This is followed by an overview of the possible routing strategies in a multistage interconnection network. A variety of space division switches are then considered which leads to a discussion of the hypercube fabric, (a novel switching technique). The hypercube fabric achieves good performance with an $O\left(N.(\log_2 N)^2\right)$ scaling. The output module, resequencing, cell scheduling and output buffering technique is presented leading to a complete description of the proposed ATM switch.

Various traffic models are used to quantify the switch's performance. These include a simple exponential inter-arrival time model, a locality of reference model and a self-similar, bursty, multiplexed Variable Bit Rate (VBR) model.

FIFO queueing is simple to implement in an ATM switch, however, more responsive queueing strategies can result in an improved performance. An associative memory is presented which allows the separate queues in the ATM switch to be effectively logically combined into a single FIFO queue. The associative memory is described in detail and its feasibility is shown by laying out the Integrated Circuit masks and performing an analogue simulation of the IC's performance is SPICE3. Although optimisations were required to the original design, the feasibility of the approach is shown with a $15\eta s$ write time and a $160\eta s$ read time for a 32 row, 8 priority bit, 10 routing bit version of the memory. This is achieved with $2\mu m$ technology, more advanced technologies may result in even better performance.

The various traffic models and switch models are simulated in a number of runs. This shows the performance of the hypercube which outperforms a Clos network of equivalent technology and approaches the performance of an ideal reference fabric. The associative memory leverages a significant performance advantage in the hypercube network and a modest advantage in the Clos network. The performance of the switches is shown to degrade with increasing traffic density, increasing locality of reference, increasing variance in the cell rate and increasing burst length. Interestingly, the fabrics show no real degradation in response to increasing self similarity in the fabric.

Lastly, the appendices present suggestions on how redundancy, reliability and multicasting can be achieved in the hypercube fabric. An overview of integrated circuits is provided. A brief description of commercial ATM switching products is given. Lastly, a road map to the simulation code is provided in the form of descriptions of the functionality found in all of the files within the source tree. This is intended to provide the starting ground for anyone wishing to modify or extend the simulation system developed for this thesis.

Glossary

AAL	ATM Adaptation Layer
AAL1	The constant bit rate, synchronous ATM Adaptation Layer.
AAL2	The variable bit rate, ATM Adaption Layer - To be used for compressed voice and video, not yet completely specified.
AAL3/4	A connectionless data oriented AAL, unfortunately overspecified and inefficient. (See SEAL).
AAL5	The most commonly used connectionless data oriented adaption layer.
ADM	Add Drop Multiplexer - A physical layer synchronous TDM switch used in the Synchronous Digital Hierarchy (SDH).
ADSL	Asymmetrical Digital Subscriber Line - A modem technology capable of achieving from 1.5 Mbps to 7 Mbps over the copper local loop.
AIS	Alarm Indication Signal - This informs a downstream ATM host of failure in the ATM network.
ARQ	Automatic Repeat Request - a system in which if data is not received or received in error, the downstream end asks the upstream end to resend.
ASN.1	Abstract Syntax Notation 1 - This is used to define MIBs in a SNMP based network management system.
ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network.
bit	A **bi**nary dig**it** - it holds either a zero or a one.
bus	A collection of wires and signals that allow processors and peripherals to communicate.
byte	An set of eight bits, can store 256 different states.
CAC	Connection Admission Control - This is used to determine whether a SVC should be accepted by the ATM switch.
CBR	Constant Bit Rate
CLP	Cell Loss Priority - a one bit field in the ATM Cell Header.
CLP	Cell Loss Probability

CLIP	Classical IP over ATM - this is a traditional method of obtaining IP transport over ATM.
CMIP	Common Management Information Protocol
CMOS	Complementary Metal Oxide Semiconductor - A digital logic family characterized by the use of both n-channel and p-channel MOSFET transistors in opposition.
CO	Central Office - The central switching and management office of a large carrier.
CPCS	Common Part Convergence Sublayer - The lower layer of the Convergence Sublayer, is specialised by the SSCS
CRC	Cyclic Redundancy Check - a statistically correct form of checksum.
CS	Convergence Sublayer - The layer above the SAR sublayer in an AAL, provides frame-wide data.
CSMA/CD	Carrier Sense Multiple Access / Collision Detect - The mechanism used in an Ethernet network to prevent contention in the physical medium.
DMAC	Direct Memory Access Controller - This is a digital system that allows hardware to write directly to memory which is shared with a microprocessor.
EFCI	Explicit Forward Congestion Indication - This bit is used to signal to a downstream node that a switch is experiencing congestion.
EPD	Early Packet Discard
fBm	Fractional Brownian Motion
fBn	Fractional Brownian Noise
FIFO	First In - First Out
FR	Frame Relay
FTTC	Fiber to the Curb
FTTH	Fiber to the Home
FUNI	Frame User Network Interface - A frame based access to an ATM network.
GFC	Generic Flow Control - a four bit field in the UNI ATM Cell Header.
Gigabit	One billion bits. Normally taken to mean Gigabits per second. (Gbps)
HEC	Header Error Control - An 8 bit CRC in the ATM Cell Header
HOL	Head of Line - A description of blocking in an ATM switch caused by FIFO queues at the switch's inputs.
IC	Integrated Circuit
JPEG	Joint Photographic Experts Group - An image encoding format.
ILMI	Integrated (or Interim) Local Management Interface - This was initially intended to be a stop gap ATM management interface but has been extended. Hence its original name of Interim LMI and its newer name of Integrated LMI.

IM	Input Module - The physical medium dependant interface of an ATM switch to the transmission system
IPv4	Internet Protocol version 4. - The current internet protocol. It is expected that this will be supplemented by or possibly overtaken by IPv6 (IPng)
IPX/SPX	Internetwork Packet Exchange / Sequenced Packet Exchange. The networking protocols originally designed by Novell Networks. A LAN protocol suite.
ISDN	Integrated Services Digital Network - This is generally understood to mean N-ISDN where N stands for narrowband.
ITU-T	International Telecommunication Union - Telecommunications
Kbps	Kilobits per second - A thousand bits per second
KB	Kilobyte - 2^{10} or 1024 bytes.
LAN	Local Area Network
LANE	LAN Emulation
local loop	The final section of twisted pair copper wire to the telephone in a residence
Mbps	Mega bits per second - A million bits per second.
MIB	Management Information Base - A set of objects defined for network management on a network node.
MIN	Multistage Interconnection Network - A network which switches ATM cells through multiple stages of normally identical elements.
MOSFET	Metal Oxide Field Effect Transistor - The type of transistor that is used in CMOS digital logic (as well as other devices).
MPEG	Motion picture experts group - This is a compressed motion picture file format.
MPOA	Multi-protocol over ATM - A standard that allows Layer 3 protocols to be switched at the ATM layer once a flow has been identified.
MPSR	Multi-path Self Routing
NIC	Network Interface Card
NNI	Network to Network interface
OAM	Operations and Maintenance
octet	Eight bits, a standard byte.
OC3	A physical layer specification for 155 Mbps transmission over fiber optics.
OSI	Open Systems Interconnect - A commonly used network reference model.
OSPF	Open Shortest Path First - A routing algorithm used to determine the optimal route through a network.

PDH	Plesiochronous Digital Hierarchy - A physical layer standard which uses a semi-synchronised clock and stuff bits in order to maintain synchronisation.
PNNI	Private Network to Network interface.
PMD	Physical Medium Dependant Sublayer - The transmission layer of the ATM model.
PPD	Partial Packet Discard
PSTN	Public Switched Telephone Network
PTI	Payload Type Identifier - a 3 bit field in the ATM Cell Header
Q.2931	The UNI signalling protocol for B-ISDN, based on the N-ISDN protocol Q.931.
QoS	Quality of Service
REs	Routing Elements - An element used in the hypercube switching fabric, it is more complex than the SEs which interconnect them.
RDI	Remote Defect Indication - This cell is returned by a downstream host to the upstream host to indicate that it has received an AIS signal.
RM	Resource Management - A type of management ATM cell used in ABR flow control.
RMD	Random Midpoint Displacement - An algorithm for producing self-similar traffic (fractional brownian noise)
S-AAL	The signalling ATM Adaption Layer - The adaptation layer used to support signalling services, essential AAL5.
SAR	Segmentation and Reassembly - A sublayer used for chopping frames into cells, found in the AAL.
SDH	Synchronous Digital Hierarchy - This is a physical layer standard generally used for the transmission of data over optical fiber links.
SDU	Service Data Unit
SE	Switching Element
SEAL	Simple and Efficient Adaptation Layer - a nickname for AAL5 and a critical evaluation of the other adaptation layers.
SM	Switch Management
SMDS	Switched Multimegabit Data Service - A reasonably high speed connectionless data service.
SNMP	Simple Network Management Protocol
SONET	The American equivalent of SDH, they are compatible where the transmission speeds match.
SRAM	Static Random Access Memory - This is a type of memory built up of flip flops rather than capacitors. It is faster than the latter (DRAM).

SSCOP	Service Specific Connection Oriented Protocol - A protocol used for signalling in ATM, it is the SSCS above AAL5.
SSCS	Service Specific Convergence Sublayer - This layer above the CPCS, specialises an AAL to services not explicitely handled by the CPCS and SAR.
STM	The electrical equivalent of an OC speed.
TC	Transmission Convergence Sublayer - The layer above the Physical Medium in the ATM reference model
TCP	Transport Control Protocol - This layer 4 protocol of the IP protocol suite provides an error free streamed data service between two IP hosts.
TCP/IP	Transport Control Protocol / Internet Protocol - **The** Internet Protocol
TDM	Time Division Multiplexing
UBR	Unspecified Bit Rate
UDP	User Datagram Protocol - This is a peer protocol layer to the TCP layer in the internet protocol suite. In provides for a best effort, message based connection service.
UNI	User to Network Interface
VBR	Variable Bit Rate - In turn divided into RT (Real Time) and nRT (non Real Time) services.
VC	Virtual Circuit
VCI	Virtual Channel Identifier - a sixteen bit field in the ATM Cell Header
VoD	Video on Demand
VP	Virtual Path
VPI	Virtual Path Identifier - a 8 bit field in the UNI ATM Cell Header, a 12 bit field in the NNI ATM Cell Header.
WDM	Wavelength Division Multiplexing - A very high speed optical switching technique where data streams are identified by the wavelength of the light that carries them.
X25	A connection oriented data protocol developed in the 70s.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This dissertation is to explore the conceptual space presented by "Large Scale Switching Systems" and presents two Novel techniques (the hypercube fabric and the associative memory) discovered as a result of this research. Their performance is evaluated by means of simulation; these simulations include more realistic models than those typically used for analytical purposes. It thus attempts to interweave a discussion of the demands and constraints placed on an ATM system with an overview of the large body of research already performed on the topic of ATM switching systems, and in the process to lay the bedrock for a discussion of the extensions to this research presented in this dissertation.

ATM is an architecture in which small, fixed length packets, called cells, are switched through a network along virtual circuits (VCs). Unlike most datagram networks, the route through the network is not determined for every cell that enters an ATM switch, but rather a virtual circuit (called virtual since the channel is not implicitely associated with a time slot) is established for the duration of the connection. All ATM cells travel down the same links and arrive in the correct order. The use of a virtual circuit enables a Quality of Service (QoS) to be negotiated at the call set up phase. For this purpose, ATM traffic is divided into four different classes, Constant Bit Rate (CBR), Variable Bit Rate (VBR) in which the traffic has a mean cell rate the may be exceeded for short bursts, Available Bit Rate (ABR) in which a flow control mechanism ensures fair use of the remaining resources and Unspecified Bit Rate (UBR) in which no guarantees whatsoever are presented. Each of these traffic classes in turn has a number of parameters that can be used to specify the quality presented to a class. These parameters include the Peak Cell Rate, the Sustainable Cell Rate, the Cell Delay and the Cell Delay Variation. ATM can assure QoS since every cell in a virtual circuit travels the same path; for this reason sufficient resources can be reserved for a VC at call setup at the ATM switch. (This would be difficult to ensure in a datagram network).

An ATM network consists of a number of ATM switches connected to ATM hosts and interconnected through some physical layer interface (typically optical fiber is used) and ATM

was designed for the high data rates and low bit error ratios of an optical fiber environment. At the ATM switch, ATM cells arrive on input ports, are switched through some sort of switching fabric (for small switches a fast bus or a shared memory system is used) and are buffered when contention occurs. (Since in ATM a cell is not confined to a particular slot as in a synchronous network, two ATM cells might wish to leave the switch through the same output port in the same time slot). This buffering might occur at the input or preferably the output of the ATM switch.

There are two factors that present an impediment to scaling ATM switches to very large sizes. The one is the Call Setup and Switch Management overhead of handling a large number of ports, this is not the primary focus of this dissertation. However an architecture that could potentially alleviate some of this overhead is presented in chapter 3. The other is the cell switch fabric, that part of the switch that is responsible for moving the ATM cells from the input to the output of the switch. (This generally does not perform a buffering function which is generally reasonably independent of the size of the switch). Due to the high speeds typical of ATM (155Mbps to 2.4Gbps), a limit is reached on a technique which relies only on switching in time (such as a shared memory or shared medium switch). Thus, it becomes necessary to switch in space. This is normally done by interconnecting a number of Switching Elements (SEs) in some sort of network [4]. Obviously, the cost of the ATM switch then is determined more or less by the number of SEs times the complexity of each SE (possibly with another overhead represented by the interconnection complexity of the elements). The cost of a switching fabric is in practice not simple to determine and may be compounded when considering the performance trade-offs that each architecture may present. In this thesis, the order of scaling for a large switch is the primary metric used and occasionally the complexity of each SE is also determined.

It is not especially obvious how switching in space can be performed at reasonable cost in an ATM switch, especially if only output buffering is considered, since the cells arrive asynchronously. Thus, even a crosspoint network [82] which has $O(N^2)$ scaling cannot perform the task of ATM switching [28]. If one allows a certain number of ATM cells to be routed to one output simultaneous (the rest are dropped), then the knockout principle can be utilised (this principle is that it become extremely unlikely for a large number of ATM cells to be destined to the same output if their destinations are uniformly distributed) to allow an $O(N^2)$ network, the knockout concentrator. This achieves an output buffering switch while retaining $O(N^2)$ scaling in exchange for an extremely low cell loss probability.

Obviously, an $O(N^2)$ scaling is unacceptable for a large number of ports. (A 1000 port switch would require roughly 1000000 elements), thus alternative means of switching in space have been sought. Derived from the classic synchronous switches, Clos networks [18] have been used [23], [48], [67]. These normally consist of an interconnection of shared memory SEs in a Clos network. Buffering is thus performed internally in the network. In order to allow

a reasonable throughput to be sustained, the Clos network must be sped up in relation to the link speed (often, an internal speed twice that of the link rate is used for convenience in synchronisation). A Clos network has an $O\left(N^{\frac{3}{2}}\right)$ scaling.

Further improvements can be made, the Banyan class of networks switch on a single bit and require only $O(N \log_2 N)$ SEs, [76], [38], [39], [45], [81], [44], [62]. A Banyan network, suffers from blocking, however, and unacceptable throughput degradation with increasing size if internal buffers are used. Many solutions have been presented for this. Banyan networks are connected in series to form the tandem Banyan network, those cells unable to be correctly routed in the first Banyan network vie for attention in the next. This architecture presents delay problems however since cells arriving out of succeeding Banyan networks must be resynchronised when buffered. Banyan networks may be dilated and demultiplexing and concentration may be used to distribute cells to parallel Banyan networks. If cells are presented sorted and without gaps to a Banyan network (and none are destined to the same port) then the Banyan network is capable of routing all of the cells to their correct destinations. Thus if a sorting network (a Batcher network [6]) is placed before the Banyan network, the network is capable of routing the cells to the output. The resulting class of networks (the Batcher-Banyan) networks [59], [32], [31], [29] are well studied within academic circles and representatives include the Moonshine, Starlite and Sunshine switches (it has $O\left(N (\log_2 N)^2\right)$ SEs).

The novel technique proposed in the Thesis combines time domain and space domain switching. The speedup required in time is limited to $\log_2 N + 1$. This results in the design being practical for a very large switching size. Unlike most other space division fabrics, the hypercube fabric consists of two different types of elements, buffered routing elements (REs) and extremely simple switching elements (SEs). The REs perform the routing function of the cells while the SEs simply interconnect them in a hypercube fabric. The number of REs is N , however, the speedup required effectively results in a complexity of $O(N \log_2 N)$ the number of SEs required is $\frac{1}{2} N \log_2 N$, however, the speedup effectively results in a complexity of $O\left(N (\log_2 N)^2\right)$. This is of the same complexity as that presented by the Batcher-Banyan network, however, the SEs are extremely simple (realisable with a couple of CMOS gates). This switch thus represents a viable method of building very large scale ATM switches. Its performance is compared against a Clos network (of similar technology) and against an idealised perfect switching fabric. The variation of the network used does not guarantee cell order (like the Roxanne network and the Clos network to which it is compared), thus a cell resequencer is also presented. The result used is a hybrid technique between a time based and sequence count based resequencing method.

Within a buffered Multistage Interconnection Network (MIN) logical (or physical) FIFO buffers are often used to buffer cells destined for a particular output port. This can result in a cell being delayed at a particular logical queue. However, no compensation is made for

this delay and the cell may arrive at another long queue and suffer a further delay. The end result of this is that the Cell Delay Variation (CDV) or jitter may be exacerbated in this type of network. Another complication arises in network in which the cell may have more that one valid output port to which it may be destined. In this case, the queue on which the cell will be placed must be determined when the cell arrives, rather than when it is dequeued. Unfortunately, this may result in the routing of the cell being based on inaccurate information. This is especially prevalent where back-pressure signals are used in the network to indicate congestion at later stages in the network.

A novel technique is proposed which solves or alleviates both of these problems. This involves the use of an associative memory to queue the cells. When a cell is dequeued, a cell is selected through logic within the associative memory which effectively allows the highest priority cell to be selected at every cell time. This priority can be used in conjunction with a global clock to ensure that older cells are selected in preference to newer cells, this allows the combined buffer space of the switch to be used as a single effective queue since if a cell is delayed at an earlier queue it automatically receives higher priority at a later one. The feasibility of this associative memory is demonstrated by laying out the IC masks in an IC layout tool and simulating their performance in SPICE3. For $2\mu m$ technology, an 8 bit priority field and a buffer capacity of 32 cells, a read time of $160\eta s$ is demonstrated. The performance of this queueing scheme against the FIFO scheme is compared in both the Hypercube and the Clos networks and it is shown to result in either smaller Mean Cell Delay or Standard Deviation of Cell Delay, or both in these networks.

The last contribution made in this thesis is a presentation of a parameter driven, complex VBR model. This model is capable of simulating both uniform and bursty traffic with both a mean and variance of cell rate. The burstiness of the traffic can be characterised both by a time driven and a data driven model and can be varied. The model can also produce self similar traffic. The amount of self-similarity is driven by the use of a Hurst parameter [22], [33] the changing value of which alters the self-similarity of the traffic. The VBR model is first used to obtain plots of multiplexing gain against various parameters and is also used to evaluate the performance of the switching techniques presented in this thesis under these more interesting (and realistic) conditions.

The objective of the thesis was to explore the area of large scale switching in ATM. This was refined over time to an exploration of a novel technique of switching which resulted in the development of the hypercube fabric. Thus, the primary objective of the thesis was fulfilled in the development of this fabric. The performance of this fabric needed to be evaluated and this resulted in the development of more realistic traffic models in order to compare the switches performance under more realistic load conditions. Another natural development of this primary aim was an investigation of the queueing methods used in the ATM switches, this led to the development of the associative memory concept. This technique was found

to lead to improved performance both in the hypercube fabric and in a Clos network.

1.1 Roadmap of the Thesis

In chapter 2, the technology of ATM is introduced in a broad overview. The intention of this chapter is to allow the reader to become familiar with the architecture and evolution of an ATM network and thus to determine the requirements that are placed on an ATM switch, whether large or small. From this basic description of the ATM network, the theme of the various chapters will be developed. Since the topic is very large it also allows an isolation of what will be dealt with in this dissertation and what falls out of its scope.

Chapter 3 provides an abstract overview of an ATM switch. The intention of this chapter is to view the process of ATM switching at its most abstract and thus to determine the possible constraints of Large Scale ATM switches in particular. It provides some suggestions for the possible implementation of the control logic and data paths of a Large Scale ATM Switch and presents an architecture that allows the control of the ATM switch to be distributed over a Large Number of processors.

The intent of chapter 4 is to provide the reader with the standard analyses which expose some fundamental considerations and constraints in the design of ATM switches. In particular it exposes the desirability of output buffering (while also suggesting methods whereby Input buffering can achieve reasonable performance). The limitations imposed on the scalability of shared medium and shared memory architectures are then discussed. This leads to the necessity of space division switching in Large Scale ATM switches. The possible routing schemes through an abstract MIN are then discussed along with the advantages and disadvantages of this scheme. One of the schemes is selected for exploration in this dissertation due to the many advantages it posseses over the other schemes.

Chapter 5 presents the topic of space division fabrics. In the first part of the chapter the knockout principle is shown, leading into a discussion of the knockout fabric. The poor scaling of this fabric is shown. (Despite its use of the knockout principle) This leads to speculation that a better order of scaling could be obtained using another architecture. The next section presents an overview of the possible techniques that can be used if contention for the outputs of a Switching Element occurs in a space division fabric. The various possibilities are discussed and references to switching fabrics which utilise all of the possibilities are given. The intent of this section is both to discuss the possibilities in the abstract and to allow the reader to research the fabrics which utilise the various possibilities. This is followed by a brief overview of the Benes network and the minimum number of SE's determined by Benes. This allows the lower bound on a switching network to be determined and leads to the discussion of the Banyan network followed by the non-blocking properties exploited by the

Batcher-Banyan network variants. Next discussed is the Clos network. This is done since this network is used as a comparison to the Hypercube fabric since it has attained the largest practical implementation in Large Scale Switches of all of the space division fabrics. The next section introduces the novel Hypercube switching fabric and presents a discussion of its structure and evolution. It also presents a simple FIFO queueing and routing algorithm that can be used in the Hypercube fabric.

Chapter 6 presents a discussion of the Output Module of an ATM switch. Since the practical performance of an ATM switch is determined by this element it presents an important component. An abstract overview of buffer management strategies (and that used in the simulation) is given as well as the cell output scheduling technique used in the simulations. The cell resequencer is presented next and the chapter rounds off by joining the pieces together in a discussion of the input and output modules used in the simulations.

Chapter 7 presents the traffic models used in the simulation. This includes the standard exponential interarrival time model as well as locality of reference models. The phenomenon of self-similarity and the use of the RMD algorithm in order to generate self-similar time series is discussed. The Variable Bit Rate model is then presented which utilises both self-similarity and two different bursty models in order to generate traffic. A composite model using both CBR and VBR sources is then presented.

A novel queuing approach is discussed in Chapter 8. This utilises an associative memory to more optimally select cells for routing through an ATM network than a simple FIFO scheme. The idea behind the scheme is presented as well as a discussion of its various elements. Next the optimisations necessary to allow the memory to perform in an adequate time are presented followed by simulation results that show the feasibility of the memory for ATM switching.

The methodology used in simulating the various components of the ATM system is discussed in chapter 9. This includes such decisions as the choice of language and the structure of the source tree and the most important class hierarchies as well as the scheduling and control techniques used within the simulation itself. In addition, some of the abstractions used in modeling the switch are discussed. Some of the structures that are used solely for the purpose of obtaining simulation results are discussed.

The simulation results for the various traffic models, switching fabrics, the resequencer and other is presented in the second to last chapter, 10. The significance of the results is then discussed.

The final chapter, chapter 11 concludes the thesis with an overview of the results and their implications. It then presents further work that can be performed on the ideas presented in this dissertation.

The appendices are intended to provide information that falls outside of the main thrust of

the thesis. Appendix A presents suggestions for how fault tolerance and reliability can be attained in the hypercube fabric. Appendix B presents a very brief overview of a possible implementation of multicast in the hypercube fabric. Appendix C is intended to provide those who are interested in exploring the Associative memory farther a very basic introduction to the topic of integrated circuits. Appendix D presents a brief summary of the types of Commercial ATM switches and the players in the ATM market. It also presents a "real world" view of where a Large Scale ATM switch might find use. Finally, Appendix E contains a description of the class hierarchies created and used as a result of these simulations. This is intended for those who intend to extend this work by using the libraries and techniques developed.

Chapter 2

Introduction to ATM

This chapter is intended to provide a basic primer for the remainder of the thesis. It is not an exhaustive introduction to all of the aspects of ATM switches, but provides a basic introduction to all of the aspects of ATM switching that will follow. Its purpose is to provide a sufficient overview of the ATM network architecture that the requirements imposed on ATM switches can be identified and their impact on the architecture examined. It is for this reason that the author decided to place this material as a chapter rather than an appendix. Since this chapter presents introductory material pooled from many different sources and represents information that is generally common knowledge within the ATM and networking worlds, explicit references to every concept will not be given, however for good introductory material on networking see [74]. For information on ATM networks see [66], [13], [68].

The Asynchronous Transfer Mode had been selected for the switching layer of the B-ISDN (Broadband Integrated Services Digital Network), normally specified to run over SDH/SONET. From the private network perspective it has been adopted as a switching technology for the Local Area Network in the form of LANE (LAN Emulation) Switches. ATM promises to be a unifying technology. It is the only technology that scales from the Local Area to the Enterprise Backbone and Carrier Backbone environments. In addition, unlike other contenders to the title of unifying technology (such as the Internet Protocol TCP/IP) ATM provides Quality of Service (QoS) capabilities, thus it is capable of transporting different types of traffic such as Video, Voice and Data over the same transmission structure.

ATM has characteristics of both circuit switched and packet switched networks. ATM is an asynchronous switching architecture, thus unlike a normal PSTN (Public Switched Telephone Network), traffic is not switched in a fixed time slot across the network. It is also not switched in integral multiples of a base bandwidth (such as the 64Kbps used in synchronous transmission network). ATM is designed to run in a high bandwidth, low bit error environment, i.e optical fiber. In fact it is specified that the bit error rate for the transmission

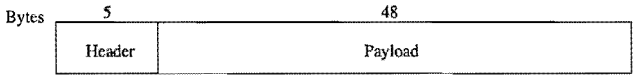
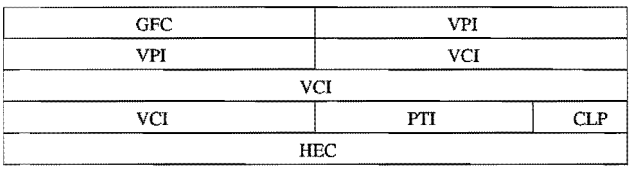
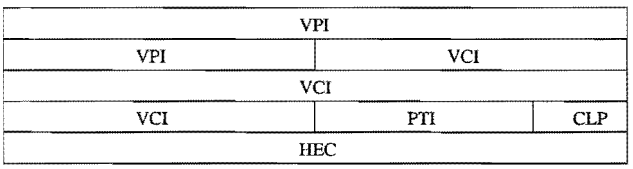


Figure 2.1: An ATM Cell



UNI ATM Cell Format



NNI ATM Cell Format

Figure 2.2: The ATM Cell Header

system in ATM be below 10^{-12}. In addition, ATM imposes very low cell loss probabilities on the switching network, some classes of service are specified at a cell lost probability of 10^{-8} down to 10^{-10}.

2.1 The ATM Cell

ATM is carried in a fixed size packet of 53 octets called a cell. A cell consists of a five octet header and a forty eight byte payload. An ATM cell is shown in figure 2.1.

The payload in an ATM cell is not checked for errors. Instead, end to end transport protocols perform the error correction. In addition, unlike X25 the ARQ function is performed end to end across the network (for certain services) and not between two nodes in a network. The reason for this is that the low bit error rate in an ATM network makes the overhead of performing ARQ between nodes in a network prohibitively expensive. Also due to the high speeds and proportionately high latencies of an ATM network, the buffering requirements for this type of service would be prohibitively large. The 5 byte ATM header has the formats shown in figure 2.2.

Unlike the payload of an ATM cell an 8 bit Cyclic Redundancy Check (CRC) is inserted into the cell header. The header provides the switching information (amongst other data) to the ATM switch, thus if any data were to be corrupted in the cell header, the cell might be switched incorrectly and arrive at the incorrect destination. This will have the effect of corrupting the data between two users of the network instead of just one. Thus in order the prevent this occurrence, the HEC is inserted. It is capable of correcting one bit error in the header and detecting multiple bit errors. The size and meaning of these fields is shown in

Cell Field	Meaning
GFC	Generic Flow Control (4 bits)
VPI	Virtual Path Identifier (8 bits at UNI, 12 bits at NNI)
VCI	Virtual Channel Identifier (16 bits)
PTI	Payload Type Identifier (3 bits)
CLP	Cell Loss Priority (1 bit)
HEC	Header Error Control (8 bits)

Table 2.1: The ATM Cell Header Fields

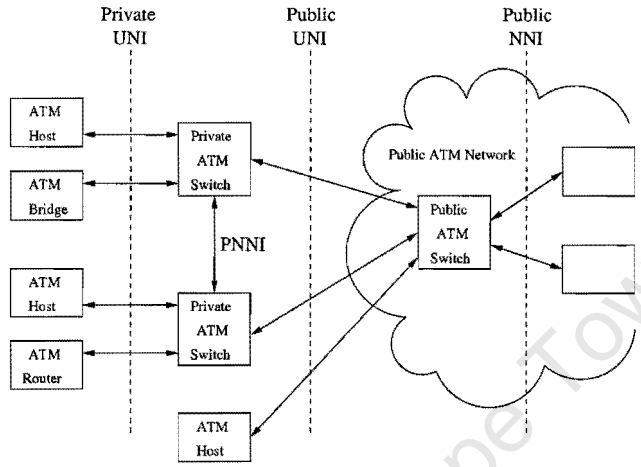


Figure 2.3: The ATM network interfaces

2.1.

The ATM Cell header has two different formats, one at the UNI (User Network Interface) and the other at the NNI (Network to Network Interface). The ATM network model consists of a number of different interface, each with its associated signalling protocols and the two different ATM cell types. The reason for this division is that there are different needs amongst the various users of the ATM network. A corporate user of a private network is not particularly concerned with billing amongst his various users for example, while to a telecommunications service provider, this is essential. In addition, the formation of closed user groups will not be as important within the private network as it will between the various branches of a corporate customer using a network providers ATM network. The various interfaces are shown in figure 2.3.

The network interfaces names are in fact almost completely self explanatory. The Private UNI (Private User Network Interface) forms the interface between ATM devices (such as bridges, routers and hosts) and an ATM switch owned by a private user (for example a bank or a university campus), the Public UNI forms the interface between user switches and devices and the public ATM cloud. The Public NNI forms the interface between the switches within a public carriers backbone. Worthy of mention here is the PNNI which is always understood to mean Private Network to Network interface. This is the interface between the switches within an organisation's network. Note that PNNI could also mean

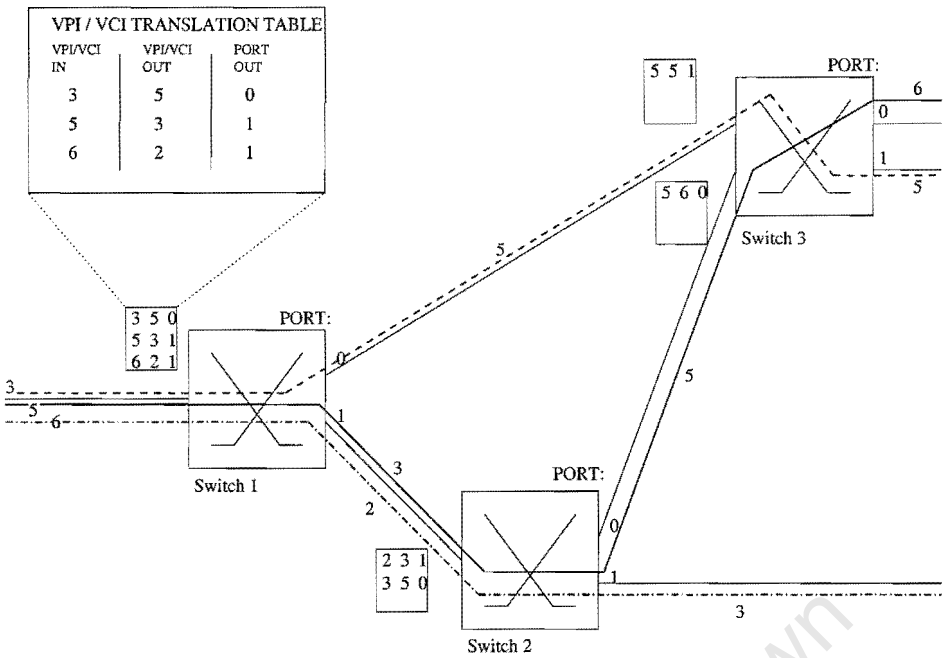


Figure 2.4: ATM Virtual Circuits

Public NNI but is never used in this way.

Returning back to figure 2.1 one can thus see that at the level of the cell header, the distinction between an ATM cell at the UNI and at the NNI is the existence of a GFC field at the UNI, these four bits become used for extra switching information in the form of four extra VPI bits at the NNI. The majority of the space in the ATM cell header is devoted to switching information, this is contained in the VPI/VCI fields. ATM, unlike many other packet based protocols (such as IP and IPX), is connection oriented. Thus, the VPI/VCI value indicates a virtual circuit number. In order to understand how this value is used for switching, it is sometimes convenient to think of the two values as a single value (at least at first).

A small network with three switches and three virtual circuits running through it is shown in figure 2.4. Here three virtual circuits - 3, 5 and 6 enter the leftmost ATM switch. At the entrance of every ATM switch there is a VPI / VCI translation table. The VPI/VCI value of every cell is looked up and the corresponding output port for that cell is found. The cell is sent to the output port through the switching fabric of the ATM switch and its VPI/VCI header value is changed to the value indicated in the table. Thus if we trace the route followed by VPI/VCI 5 we see that it it translated to output port 1, VPI/VCI 3 at Switch 1. At Switch 2, virtual circuit (VC) 3 is translated to port 0 and VC 5. Then at Switch 3 it is translated to port 0 and VC 6.

The important point to notice is that the VPI/VCI value has only a local significance, its significance does not even extend to the entire switch and is associated with an input port. Thus, switch 3 has two incoming VC 5's, one per port, and in fact the outgoing VC's at

Switch 3 could have both been on VC 5 since they were directed to different output ports. To summarise, the majority of the cell header space is devoted to switching information fields, the VPI/VCI value. At the UNI there are 8 bits (256 values) devoted to the virtual paths. At the NNI there are 12 bits (4096 values) devoted to virtual paths. There are 65536 (16 bits) possible Virtual Channels per Virtual Path.

2.1.1 The GFC field

The Generic Flow Control Field is only used at the UNI, it is intended for flow control purposes between a ATM host and the ATM network. It is only relevant over a single link. Currently it is unused and is set to 0 (uncontrolled access).

2.1.2 Virtual Paths and Virtual Channels

The Virtual Circuit consists of two values, the Virtual Path Identifier and the Virtual Channel Identifier, and there are in fact conceptually two different type of ATM switches - Virtual Path Switches and Virtual Channel switches. (In practice ATM switches will normally perform both functions and the same switch on the same port might perform virtual path switching or virtual channel switching depending on its configuration). A virtual path cross connect switches based only on the VPI value. The VCI values are unaltered. Thus, a virtual path identifies a group of virtual circuits. This is useful to reduce the overhead of setting up a virtual circuit across the entire network for every single connection request. For example, a VP could be set up from Cape town to Johannesburg across the backbone network. Then, when a user connected to the ATM switch at the Cape Town end signalled that she wanted to connect to another Host in Johannesburg, a virtual circuit could be allocated within the virtual path that travelled to Johannesburg, saving the signalling on intervening switches. The other reason for virtual path and virtual channel switching rose from the need to conserve the VPI/VCI space. Although at the NNI there are 28 bits = 2^{28}= 268435456 virtual circuits, in practice ATM switches cannot support this entire space. For example, a FORE ASX 200 ATM switch is restricted to 256 virtual paths and 256 virtual channels. Thus, commonly travelled segments of the network can be consolidated into virtual paths. In fact, it is likely that in the future, the greatest restriction on ATM switches will be the virtual circuit space.

2.1.3 PTI Field

The Payload Type Identifier Field provides information on the type of ATM cell that is being transported and whether the ATM switch through which it is being switched is experiencing

PTI Coding	Interpretation
000	User Data Cell, congestion not experienced, SDU-type = 0
001	User Data Cell, congestion not experienced, SDU-type = 1
010	User Data Cell, congestion experienced, SDU-type = 0
011	User Data Cell, congestion experienced, SDU-type = 1
100	Segment OAM F5 flow related cell
101	End-to-end OAM F5 flow related cell
110	Reserved for future traffic and resource control
111	Reserved for future functions.

Table 2.2: Payload Type Identifier Field Values

congestion or not. A Full discussion of the use the traffic management bits will have to be delayed until a fuller understanding of ATM switch management and Operations and Maintenance (OAM) is discussed. A summary of all of the PTI bit values is presented in table 2.2.

The OAM cells will not be discussed here. Note that all user data cells begin with a zero bit. Then two other bit values can be set, the second bit indicates whether congestion was (1) or was not (0) experienced. This bit is also known as the EFCI (Explicit Forward Congestion Indication). It is set by an ATM switch when congestion is being experienced in order to alert the receiving node to this congestion. (The downstream node would then ask the sending node to reduce its data rate). In the original ATM flow control model this was the only form of flow control, however, the extremely high bandwidth/latency product of ATM meant that this form of flow control was essentially inadequate. More intricate and powerful methods have been devised as a result. Also of interest is the last bit of the User Data Cell PTI field, here indicated as SDU (Service Data Unit) type. This bit is also informally known as the "AAL 5" bit because of its importance in this Layer of the ATM architecture. In order for a frame to be transmitted over the network, it has to first be segmented into cells. The last cell of the frame will have this bit set to a one. This plays an important role in three important features of ATM switches to be explored later. These are in the SEAL (Simple and Efficient Adaptation Layer - AAL5), Partial Packet Discard (PPD) and Early Packet Discard (EPD).

2.1.4 CLP Bit

The CLP bit is used to establish priority amongst ATM cells in the network, normal ATM cells have a '0' in this field position while lower priority traffic will have a '1' in this position. It is used to mark nonconforming ATM cells and UBR cells. The true priority of an ATM cell is really given by the virtual circuit to which it belongs and the ATM switch will use the Connection Setup information to prioritise and schedule cells internally. Contrast this with the very crude mechanism of setting the CLP bit.

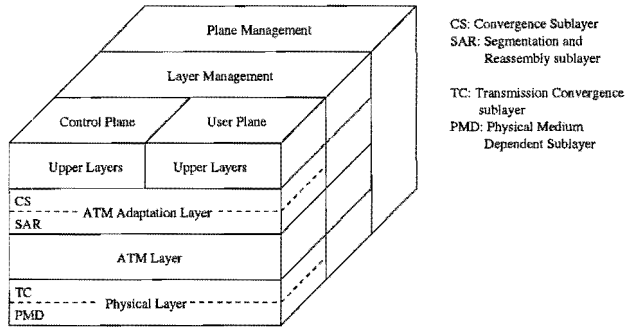


Figure 2.5: The B-ISDN reference model

2.2 The ATM Reference Model

Like any other network architecture, ATM has a reference model. It differs substantially from both the TCP/IP and OSI models. There are a number of reasons for this. One of them is that ATM can be viewed as essentially a Layer 2 network and as a technology to provide integration between Layer 3 and Layer 2 in the network. Thus, ATM is by definition not concerned with the more abstract upper layers (such as presentation and session layers) and it does not in fact specify a transport layer protocol (in practice TCP is often used). The reason that ATM can be seen as a layer 2-3 integrator can be seen in the specification of such standards as Classical IP over ATM (CLIP) and Multi-protocol over ATM (MPOA), here the intention is to remove the overhead associated with the use of routers in the network and replace certain of the routing functions with a switched virtual circuit.

The ATM model is also unlike traditional network protocol stacks in that it is portraid as a cube rather than a simple layered protocol. The standard B-ISDN ATM Reference model is shown in figure 2.5.

The Scope of this thesis will largely be confined to the Physical Layer and most especially the ATM layer. The ATM Adaptation Layers will be considered mainly in their impact on the signalling protocols and what this requires from an ATM switch that wishes to support ATM signalling and routing protocols. The higher user layers (for example the transport layer e.g. TCP) will not be considered except where flow control in ATM switches is discussed.

2.2.1 Physical Layer

The ATM Reference model defines two sublayers in the Physical Layer, the Physical Medium Dependant Sublayer (PMD) and the Transmission Convergence Sublayer (TC). The PMD function is to provide access to the physical interface and would include such considerations as bit timing (i.e. synchronous in an SDH network, plesiochronous in a PDH network or even possibly CSMA/CD in a "cells in frames" interface) and the actual transceivers, physical dimensions, framing information that is defined in a transmission layer specification. This

layer does not provide much interest in the actual ATM switch design since the sublayer above it (the Transmission Convergence Sublayer) is intended to isolate these considerations from the ATM layer. However, it does rear its head on occasion particularly in AAL1 where different algorithms are used for bit-level timing depending on whether the cells are transported over a synchronous or less synchronous physical layer.

The Transmission Convergence Sublayer is used to isolate the ATM layer (i.e. the actual ATM switch) from the lowest layer in the network, just as the higher ATM Adaption Layer does. Its function is rather like the interface layer in the TCP/IP reference model. It's functions include:

- Cell Rate Decoupling - This includes identifying cell boundaries and extracting cells from the physical network
- Header Checksum generation and verification
- Cell generation - It must be able to pack cells into whatever physical medium dependent form is appropriate.
- Frame generation

In practice, in order to attain the flexibility over the physical medium layer that this implies, an ATM switching fabric will generally have its own physical interface to interface cards that provide the TC and PMD functions for the ATM switch (e.g. as a standard edge connected card). However, there is fundamentally no difference in the function of an Input Module (IM) or an Output Module (OM) in a large or a small scale switch except where issues such as redundancy and reliability are considered.

2.2.2 The ATM Layer

Since the majority of this dissertation will focus on this layer, this section will be kept brief. The ATM layer provides the functions necessary to switch and flow control the ATM cells. This list includes the various types of ATM flow control (there are conflicting requirements between maintaining "dumb" ATM switches with flow control at the edge of the network as per the original ATM specification and the practical difficulties involving flow control in the high bandwidth / high latency environment of ATM). The ATM layer must also be able to process the ATM cell headers and use the information provided at connection setup to manage the virtual paths and virtual circuits in the ATM network. This includes such functions as traffic policing, buffer management and queuing disciplines required to provide qualities of service to the various virtual paths and virtual circuits. It must also handle Payload Type Identification by the correct use and interpretation of the PTI field values In

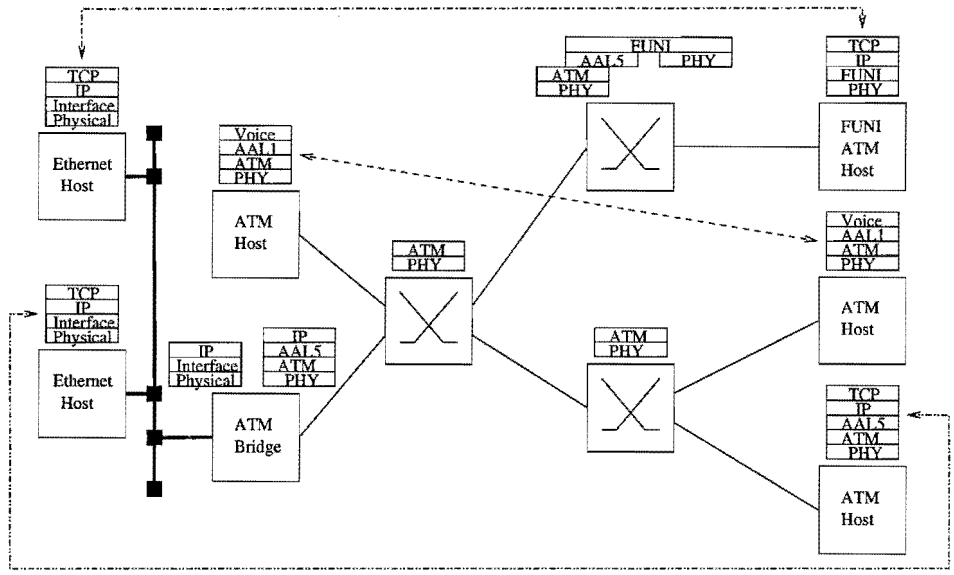


Figure 2.6: The AAL in the ATM network

general, due to the high speeds of ATM and the small cell size (resulting in a small time to process an ATM cell, 2.7μs in the case of an OC3c 155Mbps link) the functions in this layer will have to be implemented in hardware. However, the simple ATM cell header and the use of virtual circuits to effectively store QoS information (rather than in a special options field of a datagram header such as is used in IPv4 for example) mean that at least for small switches, this is not too difficult.

2.2.3 The ATM Adaptation Layer

The ATM Adaption Layer (AAL) is intended to provide service specific interfaces to higher protocol layers or applications such as voice over ATM. The intentions in establishing this adaption layer were to provide a mechanism to allow ATM networks to evolve to new applications without requiring any change to the ATM network itself. In addition, almost all services supported by ATM cannot be explicitely mapped into cells. For example, a voice circuit is a streaming service involving the synchronous transmission of bits while a UDP (User Datagram Protocol) datagram being transmitted over the network consists of a large (up to 64KB) best-effort frame of data transmitted over the network. The ATM adaption layer does not appear in a network except when used for signalling (in which case it is known as the Signalling AAL, S-AAL). The normal position of the AAL in the ATM network is shown in figure 2.6.

This shows a relatively complete view of the operation of the AALs in an ATM network, it also demonstrates how ATM can combine and integrate different services (in this case voice and IP datagram traffic). The figure illustrates two TCP connections between four hosts, two ethernet, one pure ATM host and one Frame based ATM host. The topmost hosts

	Class A (Voice)	Class B (Video)	Class C (X25)	Class D (SMDS)
Timing beween source and destination	Required		Not Required	
Bit Rate	Constant	Variable		
Connection mode	Connection oriented			Connectionless

Table 2.3: ITU-T Service Classes

communicate over a TCP connection through an ATM Bridge. This ATM bridge is capable of understanding the IP datagram protocol over an Ethernet link on the one interface and on the other it is capable of using AAL5 to transmit the datagram across the ATM network as cells. The FUNI ATM host is capable of ATM signalling, but due to efficiency reasons does not transmit its payload across its access link in cells, but rather in Frames. This requires the Access Switch connected to it to implement AAL5 in order to translate the frames to and from ATM cells. The bottommost pair of hosts are also using TCP/IP but here the ATM based host performs the AAL5 function and passes the resulting frame up to the IP network. Notice that all of these hosts are shown as using AAL5, this is due to the fact that it is the preferred datagram AAL. Although AAL3/4 performs a similar function, it is far less efficient.

The final communicating pair shown is that of the voice over ATM pair. These use AAL1 for communication as this is specified for Constant Bit Rate (CBR) services.[1]

The ITU-T originally defined 4 types service classes which where differentiated on whether timing was required, whether the bit rate was constant or variable and whether the service was connectionless or connection oriented, these are shown in table 2.3.

Although details of the various services will not be given here, the various AAL layers, the classes of service they support and the bit rates that they are used for are detailed in table 2.4. The AAL type that has been selected for Signalling in ATM is AAL5 (with a signalling specific service specific convergence sublayer). Since all of the other AAL types will generally be found on an ATM host this is the only one that will be discussed in any detail in this introduction.

[1]It is interesting to note the similarity in function between the TCP layer and the AAL layer which are both found only at the endpoints of the network and provide for multiplexing of data, error detection and recovery etc. The same applies to the IP and ATM layers, both of which are found on routers and switches respectively, but are only required in the center of the network. The only essential differences are that the TCP layer has to cater for misordered datagrams which the AAL layer needn't since ATM cells always follow the same path, and that the AAL layer must segment its data in cells for transmission.

AAL Type	Class of Service	Typical Bit Rate Type	Typical Service
1	A	CBR	Circuit Emulation
2	B	VBR	Video on Demand (Obsoleted)
3/4	C/D	ABR	Data Transport (SMDS)
5	C/D	ABR / VBR	Data / Video Transport

Table 2.4: Summary of AAL Types

The Segmentation and Reassembly Sublayer and Convergence Sublayer

Referring back to figure 2.5, we see that the AAL Layer is divided into a Convergence Sublayer (CS) and a Segmentation and Reassembly sublayer (SAR). In a nutshell, the SAR layer for a particular AAL type divides the frame presented to in by the CS, adds a (hopefully small) overhead per cell (for example a sequence count). The ATM cells are then sent over the ATM network and the peer SAR layer reassembles the cells and passes them up to the CS layer on the other side. The CS is responsible for providing information that is required across the entire AAL frame. For example it provides a multiplexing identifier for the AAL 3/4 CS as well as a CRC and other data. Not shown in figure 2.5 is the fact that the CS is in fact broken up into two parts, the Common Part Convergence Sublayer (CPCS) and the Service Specific Convergence Sublayer (SSCS). The use of these two layers is in fact reasonably obvious, the CPCS provides generic AAL processing, while any overhead needed to handle a particular service is placed in the SSCS, an example of this is the Signalling SSCS which rests on top of the combined CPCS and SAR function of AAL5.

2.2.4 AAL5

AAL5 is covered in more detail here since it is the specified AAL for ATM signalling and thus has to be implemented as an interface to the Control Processor of an ATM switch. It also provides a good indication of the use of the final bit of the PTI field in the User ATM Cell (see table 2.2) - this bit is critical to the implementation of congestion control features such as EPD and PPD. It is also a particularly efficient layer for the transport of data and sports zero overhead on all but the final ATM cell of a data frame. The basic process used by AAL5 is shown in figure 2.7. The user payload is simply segmented into cells, the final cell of the frame is differentiated by having its final PTI bit set to '1'. The last 64 bits of this ATM cell then provide the framing information for the cell, including a 32 bit CRC, a 16 bit Length Indication field (giving a maximum user frame size of 65536 bytes) and a control field - currently unused. If the user frame does not completely fill the remaining 40 bytes, then empty padding is inserted.

This AAL is very efficient and is also reasonably simple to implement, it has thus earned the nickname SEAL (Simple and Efficient Adaptation Layer), which is also a derogatory reference

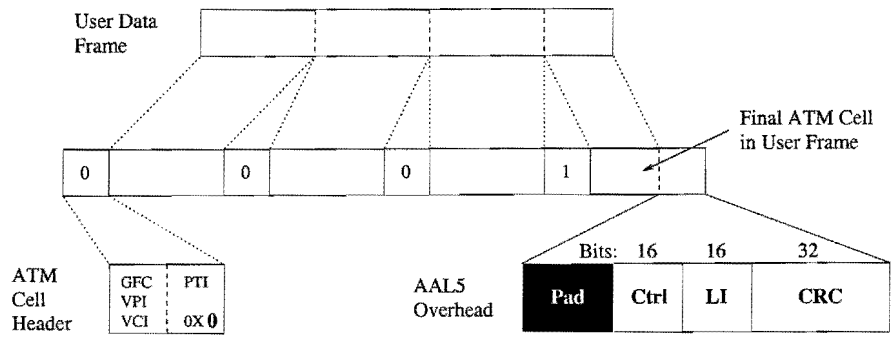


Figure 2.7: AAL5 - CS and SAR Operation

to the massive overhead imposed by AAL 3/4. The simplicity of processing makes it ideal to place in hardware and thus unburden the Control Processor from having to perform this function, this is especially important considering the complexity of the ATM Call Admission Control (CAC), signalling (Q.2931 for example) and routing (PNNI for example).

2.3 Traffic Control in ATM

ATM unlike synchronous switched architectures allows the user to insert a cell into the network on an asynchronous basis. What, then prevents all of the users from simply opening their floodgates and inundating the ATM network with traffic? The three answers to this are the functions of Connection Admission Control, Traffic Policing and Flow Control.

Connection Admission Control is the process of determining whether an ATM switch has sufficient resources in terms of buffer capacity, latency and bandwidth both internally and on its links to allow a virtual circuit with a set of traffic parameters (e.g. required cell delay, cell rate, cell delay variation) to be set up. The ATM switch must ensure firstly that the added load of maintaining the extra virtual circuit does not unacceptably impact its existing virtual circuits, it must then ensure that it has sufficient resources to buffer and switch the new connection. The process of Connection Admission Control can be performed in an ATM network where it cannot be performed in a non-connection oriented datagram network such as IP since each connection can have the connections required for it reserved in the network. This will involve the evaluation of a complex, switch specific algorithm to determine if the virtual circuit can in fact be supported by the ATM switch. It will thus almost definitely be required to be supported in software.

Traffic policing is the process of ensuring that once a contract has been established for a Virtual Circuit the sender abides by the parameters that have been agreed upon. The specified algorithm for traffic policing is a rate based method known as the Dual Leaky Bucket algorithm. If a source is found to exceed its contract the the ATM switch has two possible alternatives, it can discard the cell, or it can mark it as a lower priority cell for

the rest of the network by setting its CLP bit to '1'. In practice, slightly non-conforming cells will be marked as lower priority and if a virtual circuit becomes highly nonconforming then the cell will be dropped. Due to the high speed of cell arrival, the Dual Leaky Bucket algorithm will in practice almost certainly need to be implemented in hardware.

The above two mechanisms, Connection Admission Control and Traffic Policing are sufficient for the two classes of traffic known as Constant Bit Rate (CBR) and Variable Bit Rate (VBR) traffic types since both of these traffic types can specify parameters that characterise their behaviour. For example their Sustainable Bit Rate, Peak Cell Rate, Cell Delay, Cell Delay Variation (jitter) amongst others. However due to the difference between the Burst Cell Rate and the Sustainable Cell Rate in VBR traffic, another layer of traffic, a best effort service, can be delivered with the remaining bandwidth. Thus, the two remaining types of ATM traffic are Unspecified Bit Rate and Available Bit Rate (UBR and ABR). Unspecified bit rate is singularly uninteresting, there is absolutely no traffic contract with the user (explicit or implicit) and all other traffic takes immediate precedence over it (In fact at the ingress to the ATM network, the first ATM switch is required to set the CLP bit immediately to '1' - Low priority). It should only used when overlaid by a robust transport protocol such as TCP which will automatically adjust to the changing network conditions without explicit scheduling. ABR traffic specifies only a minimum cell rate when it connects to the ATM switch. However flow control mechanisms are used on the virtual circuit to dynamically adjust the cell rate to whatever both the network (and the receiver if that is the bottleneck) can support. The original ATM model placed flow control at the boundary of the network with the principal of moving whatever possible complex processing was possible to the network periphery. All that the ATM switch had to do was to set the EFCI bit in the PTI field if it was in a congested state. The receiver would then send a Resource Management (RM) cell back to the sender requesting that it slow its rate of cell transmission. However, due to the high bandwidth / high latency environment in which ATM is positioned, highly efficient flow control techniques are necessary in order to effectively provide flow control in ATM. Thus, the basic mechanism has been considerably altered and its complexity has increased. However, due to high speed at which ATM cells arrive, this function is best performed in hardware. Thus, backward compatibility has been maintained amongst the various flow control mechanisms.

2.4 Signalling and Routing in ATM

The precise nature of the signalling and routing protocols used in ATM are immaterial. The only requirement is an appreciation of the complexity of signalling in general and ATM signalling in particular and the difficulties involved in routing in a QoS based network. Signalling protocol are complex and in ATM, many different signalling protocols might have

to be supported since different protocol are specified on the various network interfaces. (See figure 2.3). Thus there is B-ICI (Broadband Inter-Carrier Interchange) in the public NNI between different carriers and B-ISUP (derived from ISUP) on the same interface. There is ISI between the switches within a carriers network. There is private UNI between an ATM host and a privately owned ATM switch and the public NNI between an ATM host or ATM switch and the public ATM network and there is the PNNI between private ATM switches, and that is just the signalling protocols in ATM.

Routing is less developed, there is the PNNI routing specification for private networks, however this is a very complex protocol indeed involving QoS and network abstraction and uses the OSPF (Open Shortest Path First) routing algorithm, a Link State based routing algorithm.

In addition, the legacy support of ATM essentially encompasses additional signalling complexity. For example, an IP switch uses a virtual IP router and complex software for determining when a set of packets in IP network becomes a flow. When this state is detected, a virtual circuit is created and the remaining packets in the flow are switched rather than routed. Additional protocols such an LANE, MPOA and CLIP provide additional complexity. Essentially the complexity and rapid change of the ATM protocols require their implementation in software on a dedicated fast microprocessor on the ATM switch.

Another consideration is how signalling is differentiated from other ATM traffic. The simple technique used is to reserve a particular Virtual Path and Virtual Circuit for signalling. The selected values are VPI = 0 and VCI = 5 for signalling with a local exchange, for another VPI value, the signalling will be connected to whatever switch the network user wished to signal to. The AAL used for signalling is known as the S-AAL and is AAL5 enhanced with a SSCS called the SSCOP (Service Specific Connection Oriented Protocol) which performs classical OSI Data Link Layer functions to ensure reliable transport of the ATM cells. Various VPI, VCI and PTI values are used to differentiate between the ATM cells, these are summarised in table 2.5.

2.5 Operations and Maintenance in ATM

OAM in ATM is performed by sending various cells through the ATM network that signal an ATM switch to perform some sort of function. Since the ATM header is available to the ATM layer it is obvious that these cells will have to be differentiated from the other ATM cells by some data in their cell headers. The three fields used are VPI/VCI/PTI. The various reserved values in ATM are shown simplified and summarised in table 2.5. From this it should be obvious that for example the Segment OAM Flow F4 cells must deal with Virtual Paths (since the VCI value is fixed and thus cannot indicate the VCI) and it should

Cell Type	VPI	VCI	PTI	
Unassigned Cells	00000000	00000000 00000000	-	
Meta-signalling cells	xxxxxxxx	00000000 00000001	0A0	
General broadcast cells	xxxxxxxx	00000000 00000010	0AA	
Point to point signalling cells	xxxxxxxx	00000000 00000101	0AA	
Segment OAM Flow F4 cells	xxxxxxxx	00000000 00000011	0A0	VCI 0-15 reserved by ITU-T
End-to-end OAM Flow F4 cells	xxxxxxxx	00000000 00000100	0A0	
Segment OAM Flow F5 cells	xxxxxxxx	yyyyyyyy yyyyyyyy	100	
End - to end OAM Flow F5 cells	xxxxxxxx	yyyyyyyy yyyyyyyy	101	
Resource Management cells	xxxxxxxx	yyyyyyyy yyyyyyyy	110	
ILMI cells	00000000	00000000 00010000	0AA	VCI 16-31 reserved by ATM Forum
User Cells	xxxxxxxx	yyyyyyyy yyyyyyyy	0AA	Remaining VCI can be used by user

Table 2.5: Reserved VCI/VPI and PTI values in ATM

also be obvious that Segment OAM Flow F5 cells must deal with Virtual Channels since the VPI and VCI values are both reserved. Note that for signalling cells, in general, if the VPI is zero, the signalling is destined for the local switch, if it is another value then the signalling will be used at whichever switch the VP terminates.

The OAM cells in the table are referred to as OAM F4 and F5 flows. The ITU-T terminology included five flows numbered from F1 to F5 where F4 and F5 are ATM layer OAM Cells. For completeness the flows are:

- F1 - Regenerator Section - this is the smallest possible flow path. It represents a channel (optical fiber generally) between two signal regenerators.
- F2 - Digital Section - this is section between two digital (rather than simple amplifier) transceivers, for example between two SDH Add Drop Multiplexers (ADMs).
- F3 - Transmission path - This represents a path between two ATM switches. For example, a SDH ring might have multiple ADMs that define a transmission path between two ATM switches.
- F4 - Virtual Path - this is the first of the ATM layer flows. It extends from one Virtual Channel switch to another through one or more Virtual Path switches.
- F5 - Virtual Channel - this defines an ATM flow from one end to the other through the ATM cloud.

Two other terms used for ATM flows include:

- Segment - A segment is defined between two ATM switches. It might encompass other transmission equipment however.
- End-to-end - An end to end flow extends either down an entire Virtual Path through a possibly fairly large number of ATM switches, or from host to host across an entire Virtual Channel.

Now that this has been dealt with, what functionality can be accomplished through the use of the OAM cells? All OAM cells have a particular format which begins with a four bit type field. This gives sixteen possible OAM cell types.

Two illustrative OAM cells are the Alarm Surveillance Cells and the Loopback cells.

The Alarm surveillance cells work as follows: when a link goes down, AIS (Alarm Indication Signal) cells are sent by the ATM switch to the downstream receiving node. This node notes the failure and sends a RDI (Remote Defect Indication) cell back its peer ATM hosts, thus both hosts are notified of the alarm condition. They might decide to try to re-establish the connection to avoid the point of failure.

Loopback is performed by special ATM cells that indicates to a switch that the loopback cell must be returned to the sender, this allows the management system to verify that the connection has been set up.

Note that both of these cell operations could potentially involve a large number of cells (particularly AIS since on a link failure **all** VCs and VPs will need to be notified of the failure) and represent a mission critical functional part of the ATM switch. Due to the relative simplicity of the cell formats and the possibly quick interarrival time that the cells may represent, this functionality is recommended to be implemented in hardware. However a dedicated fast microprocessor could possibly perform this function and has the advantage of being more easily upgradable in the event of new OAM cells being defined by the ITU-T.

2.6 Network Management in ATM

All of the above functionality, switching, signalling, routing and OAM is useless to a service provider and almost all private users if it is not complemented by a network management platform. Network management is a complex function and explicitely includes human beings in the management loop, since the function of any network is ultimately determined by a human component, otherwise its function can be logically argued to be meaningless. However, in this section will deal with the automatable components of the network management system.

The main network management specification for ATM is defined in the ILMI or Integrated Local Management Interface. This interface was originally the Interim Local Management Interface, but in the face of its obvious persistence and increasing functionality it was renamed. The ILMI uses both SNMP (Simple Network Management Protocol) which is commonly used in the IP world and the OSI CMIP Common Management Information Protocol for network management. The reasons for the duplication of this functionality are many. The CMIP protocol is more powerful, much more complex and more reliable and is thus generally used by the large backbone carriers. SNMP, however is well know and used throughout the IP

world. It is the simpler protocol requiring considerably less effort to implement and it plugs in automatically to many popular network management platforms such as Hewlett-Packards HP-OpenView that are available on reasonably priced hardware platforms. Thus, it is the obvious candidate for private, IP biased, ATM networks.

To illustrate a typical difference between CMIP and SNMP, CMIP supports **inheritance** amongst the objects that it defines. Thus for example, one can define that if link fails, then all virtual circuits on the link will fail. This can dramatically reduce network traffic and provides for filtering of trivial information (that a VC is disconnected) from the more important (that the link is down). SNMP does not support inheritance, thus if a link goes down, the network management workstation will also be informed unnecessarily that all the virtual circuits are disconnected.

The other primary difference is that CMIP is connection oriented while SNMP is based on the best effort UDP protocol. Thus, there is no guarantee in SNMP that alarm messages (known as traps) will in fact reach the network management station. A proposed solution to this deficiency of SNMP is to define it over TCP in addition to UDP. This would increase the software complexity on the network node where it the complexity of implementing TCP is avoided with relief, but it would result in SNMP achieving the same reliability of CMIP.

The complexity and rapid change and development of network management evidently dictates that this functionality be implemented in software on an ATM switch.

2.6.1 Management Information Bases

This discussion follows the SNMP protocol in its particulars, but many of the concepts are transferable to CMIP. A MIB consists of a hierarchically grouped set of objects which are identified by a globally unique number. This uniqueness is guaranteed by assigning various prefixes to international standards bodies, companies and reserved experimental spaces under which these institutions may define any unique hierarchy that they wish. The object number may include fields known as tables in which multiple objects may be identified by their indices. These objects are defined in a computer readable format known as SMI which is a subset of ASN.1 (Abstract Syntax Notation 1). The network management station can read this SMI file and learn the contents of a MIB located on a network node.

Various operations are defined on the objects defined in this hierarchy of which the most important are Get, Set and Trap. The get operation gets the current value from an object on the host, it might also cause an operation to be performed on the network node. The set option sets a value on the network node. Thus, for example, if a loopback was to be tested on an ATM switch, its SNMP server could be instructed to start sending the loopback cells. The last operation is an event trap. This defines a way for a network node that some event

has occurred without the management station querying the node. For example, a link failure will generally be indicated with a trap. The trap will in turn be indicated in the SMI code.

2.7 Multicast in ATM

ATM supports a point to multipoint multicast architecture. The Call Setup signalling is modified to enable a process known as leaf join in which once a point to multipoint call has been set up ATM nodes may join and leave the multicast tree. The ATM network itself performs the necessary cell duplication and will attempt to do so in such a way as to minimise the overall network traffic. It thus creates a multicast tree throughout the network. The ATM switch itself is responsible for the cell duplication and special modifications to certain ATM switch architectures will have to be performed in order to allow this functionality to be supported. Note that multicast in ATM is not bi-directional. If an ATM host receives a multicast Virtual Channel and wishes to communicate with the sender, a separate point to point VC will have to be established.

In addition to multicast, a switch broadcast function is also supported through a reserved VPI/VCI value of (x/2). (See figure 2.5).

Chapter 3

An Abstract view of an ATM Switch

An ATM Switch has to perform all of the functions given in Chapter 2. Some of the functions (e.g. switching, traffic policing, flow control) have to be performed in hardware due to the speed with which they have to be performed . Others must be performed in software due to their complexity or the rapidly changing nature of the standards (e.g. signalling, routing and connection admission control). Still others might be performed in either hardware or software, for example the handling of OAM cells.

The essential question is how the functions indicated in figure 2.5 (The B-ISDN reference model) can be performed in an ATM switching architecture. One of the first hurdles to cross is the separation of the Transmission Layer from the ATM layer. This can be accomplished as shown in figure 3.1. Here the isolation of the ATM functions is achieved by dividing them into four basic components, Input Modules (IMs), Output Modules (OMs), the Cell Switch Fabric, a Connection Admission Control Processor and a Switch Management Processor. This division is discussed in [13].

Each of these modules have a clearly defined set of functions:

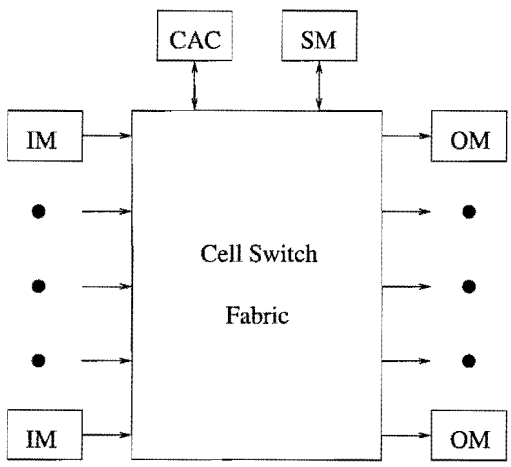


Figure 3.1: Abstract view of an ATM switch

Conceptually the Simplest is the Cell Switch Fabric, this is responsible for transporting cells from the input modules to the output modules. The Cell Switch Fabric is normally synchronous, i.e. all the ATM cells to be transported must be presented at its input at the same time, this dramatically simplifies the design of the Cell Switch Fabric. It is mainly practical because of the small fixed size of the ATM cell. Thus, although there need be no timing relationship between the ATM cells on the various Input and Output Modules and their links, the Cell Switch Fabric is normally synchronous. There are many possible ways for the Switch Fabric to switch an ATM cell. The most common is that a routing tag specifying an output port is appended to the ATM cell before it is submitted.

The Input Modules are responsible for both the TC and PMD functions of the ATM switch. They will also perform the VPI/VCI translation for the incoming ATM cells and append a routing tag to the cell that the Cell Switch Fabric uses to switch the cell to an Output Module. In addition they will normally perform the traffic policing function of the ATM layer since this may prevent the Cell Switch Fabric and the Output Modules from being presented with a load that they cannot switch or buffer. Note that in order to present the cells synchronously at the Cell Switch Fabric the input modules will have to implement double buffering in order to allow a cell to be buffered while another is being received before submitting it to the Cell Switch Fabric. The use of changeable Input Modules results in ATM achieving independence from the transmission system.[1]

The Output Modules are responsible for scheduling the cells they receive from the cell switch fabric and buffering them before submitting them to the output link. In addition, in the case of multicast ATM cells, the output modules will need to be responsible for the VCI/VPI translation of the cells since the Cell Switch Fabric will have duplicated them. Some of the functionality of the output modules may be shared with the Cell Switch Fabric especially in the case of a shared memory switching fabric. This is done to reduce the overall memory usage of the ATM switch. Note that although the Input and Output Modules are shown here as separate entities, they will in practice be implemented in general as one plug in card (an interface card) since it is very unlikely that communication will only be received from or sent to a host (although some channels exhibit Asymmetrical bandwidths, the primary example being ADSL on the local loop).

The Connection Admission Control Module is responsible for handling the Connection Admission Control Algorithms, the Connection Setup and Connection Teardown functions. It

[1] A consideration here is that the VPI/VCI translation tables will be likely to be stored on chip within the Input module, the reason for this is the fast access that is required to it and also to avoid the extra cost in terms of pin count that would be required to connect to an external memory for this function. The end result of this is that although the VPI/VCI space in the ATM cell header is large, it is likely that only a fraction of this will be utilisable. This has important implications for the design of such protocols as MPOA and LANE and also will in the authors opinion, require the eventual setting up of smart Virtual Paths that can be automattically set up by the routing protocol when a sufficient number of virtual circuits follow the same path.

is shown as a separate logical component to the SM module although both functions could be implemented on the same fast microprocessor (this also eliminates the communication overhead between the two modules), or on two microprocessors using a shared memory system. The CAC module will normally have a hardware AAL5 processor since this offloads this tedious and intensive function from the processor. As stated previously, the complexity of signalling and routing in ATM requires this function to be implemented in software on a fast microprocessor.

The Switch Management Function controls the various elements of the ATM switch, it sets up the VCI/VPI translation tables in the Input Modules, collects usage statistics from the Output Modules and obtains information about the transmission system from the PMD subsystems of both the IMs and the OMs. It will handle the alarm notifications and loopback cells. Some of this functionality might be implemented in software and some in hardware or a combination of the two. For a Carrier Access Switch it might also accumulate billing information. As mentioned previously, some of the functionality might be shared with the CAC processor, this is especially the case since the CAC will require information from the SM in order to determine whether to accept an ATM virtual circuit or not.

The most important considerations after this abstract overview of an ATM switch, especially in the context of large scale ATM switches are the factors that limit the scaling of this ATM architecture and how the communication is accomplished between the various modules of the switch in order to allow the switch to function (this can also naturally limit scalability). The most obvious limitation to the scalability is the cell switch fabric itself. It may be implemented as either a time or a space switch. In the case of a time switch, the scalability is limited by the speeds that might be required to perform the switching function as the number of input and output ports increases. In the case of a space switch the limitation is likely to be the escalating costs of providing more ports since the number of Switching Elements (SEs) may grow in a polynomial fashion. The complexity of an ATM space division switch is exacerbated by the fact the cells can arrive asynchronously at the ATM switch.

It is obvious that the input and output modules present a linear (or better if economies of scale can be utilised) increase in cost with increasing ATM switch size. This is useful since many of the more complex hardware aspects of ATM switching are performed in these modules (such as cell scheduling, output buffer management etc). Thus these do not respresent a limitation in terms of scaling an ATM switch to large sizes.

The next candidates that can present a scaling bottleneck are the CAC and SM functions. For a sufficiently large ATM switch the amount of signalling overhead might become too large for a single processor and this burden might need to be shared. Additionally the number of OAM cells and the amount of hardware that the SM module has to monitor and control could become excessive leading this component to be a bottleneck. The obvious solution to this is to parallelise these functions. This concept is illustrated in figure 3.2. Here the

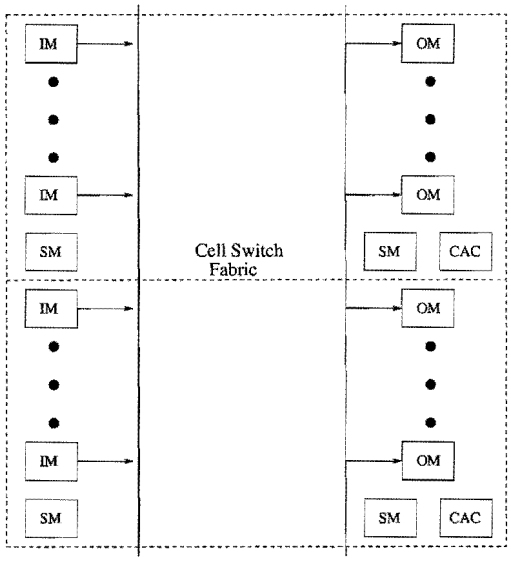


Figure 3.2: Parallelisation of CAC and SM functions

dashed boundary illustrates the control boundary of the CAC and SM modules. Note that a separate SM module is shown conceptually grouped with the input modules. Although it is unlikely to be physically duplicated in this way this is to conceptually show that the function applied to the switch inputs as well.

Another reason why the SM function might be parallelised is in order to place the SM processor in closer proximity to the device it is controlling. For example, if the cell switch fabric itself needed occasional independent configuration a SM processor could be dedicated to this task and could communicate via the cell switch fabric itself or another communication medium to another SM processor and thus indirectly influence the CAC function. (the assumption is that the SNMP or CMIP agent will be run on the CAC processor and not on the SM processor which will be dedicated to hardware control and configuration).

The difficulty of attaining this parallelisation is that the communications costs must be contained. This can in turn impose further constraints on the design of other components in the ATM switch. For example if the Cell Switch Fabric can fail to deliver cells to a particular input/output port combination due to existing traffic conditions and this spans different CAC processor boundaries, there will be a polynomial complexity in the communication between the various CAC processors in order to determine if a connection should be allowed. Thus, it becomes necessary to design the switch fabric in such a way that the CAC processing boundaries do not interfere with each other. In this case the connection can be processed without excessive communication with other CAC and SM processors. This could be a fertile area of research if the demand for large scale ATM switches develops and only some rudimentary suggestions will be presented here. For the Q.2931 signalling, a CAC processor might be required to become a router which passively forwards a packet to the correct CAC processor that does in fact have sufficient information to decide on whether a call should be

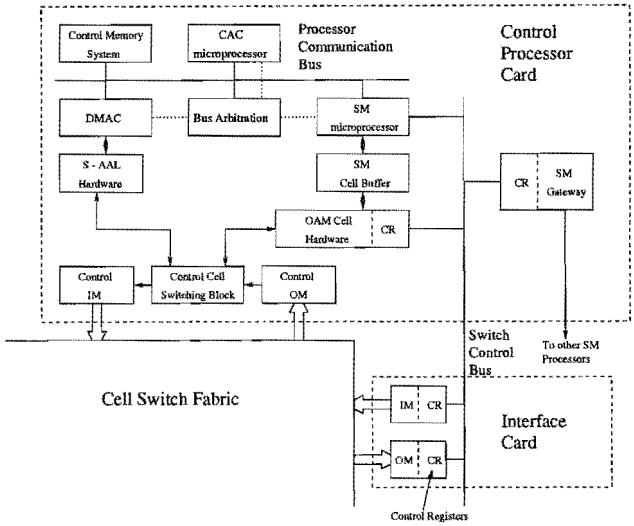


Figure 3.3: Inter-element communications in an ATM switch.

accepted or not.

3.1 Control in the ATM Switch

Since much of the ATM switch must be implemented in hardware and controlled with software, there must be a control path between the Control Processor on the ATM switch and the various hardware elements which perform the various ATM functions. These elements could conceivably communicate with each other through the cell switch fabric itself, however, this would greatly increase the complexity of the individual hardware elements. We will assume that the communication between the various elements will take place as shown in figure 3.3. Note that this design is by no means the only possible one, however it has the advantage of being scalable to a large switch size and using the cell switch fabric for most of its communication, which results in both simplicity and allows the number of Control Processors to scale to whatever size the Cell Switch Fabric can scale to (assuming that distributed control algorithms can be developed simply). Also the design aims to offload whatever processing burden is possible from the microprocessors and thus allow one control processor card to control the largest possible number of ports.

This system illustrates one control processor card, an interface card and the cell switch fabric. The intention of the design, however is to allow multiple control processor cards, each of which controls a group of interface cards to collectively control an ATM switch. The Input and Output modules are grouped logically together rather than on opposite sides of the Cell Switch Fabric as shown in figure 3.1. The reason for this is that this is the logical packaging of hardware in an ATM switch if one considers the unlikelyhood of a one-way communications interface. In order to aid the discussion, a data flow view of the above block diagram is presented in figure 3.4. Here some of the filtering components of the

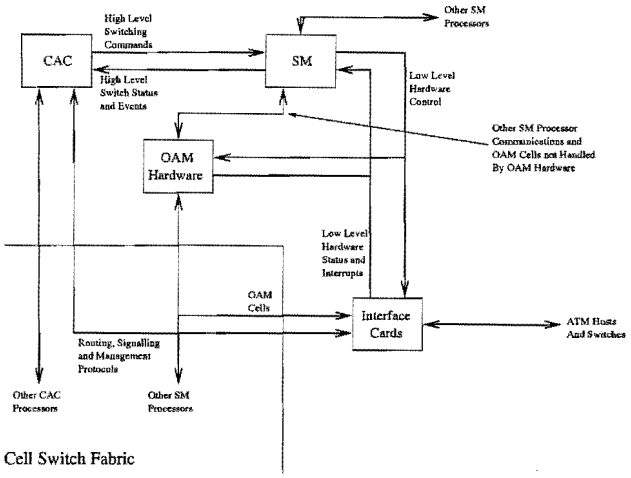


Figure 3.4: Inter-element Data Flow in an ATM Switch

control processor card of figure 3.3 have been removed and some technicalities such as the communication through a shared memory system of the SM, S-AAL and CAC processors (requiring the use of a Bus Arbitration mechanism) and the existence of the S-AAL and SM Cell Buffer hardware has been ignored since it does not add to the data path communication.

The functionality of the various elements in figure 3.3 are itemised below, refer to figure 3.4 if the discussion becomes unclear.

- Control IM and OM - The control processor is a port of the cell switch fabric like any other interface to the ATM switch. This is in fact a logical way to manage the control port since signalling, routing and OAM cells are all differentiated by either their VPI/VCI values or by their PTI field. Thus the input modules can simply have their VPI/VCI translation table destine these cells to the port on which the Control Processor is placed and another memory location which indicates on which port to send OAM cells and RM cells. Note the origination port number will have to be appended to the cell before it is transmitted through the switch in order for this scheme to work since a reference to a VC is port dependent.

- Control Cell Switching Block - Some cells should be destined for the SM processor and some to the CAC processor, this block uses the PTI and VCI/VPI field information to decide which of the two subsystems to forward the cell to. Note that this functionality could also be attained by simply using two ports of the cell switch fabric. However, this would probably be excessive and depending on the design of the cell switch fabric could necessitate the use of a larger form factor for the control processor card.

- OAM cell hardware, this is used to allow loopback cells and other simple OAM and flow control cells to be handled or generated in hardware at less cost to the SM processor. Note that in the case of an AIS signal, this might require the SM processor to first

be notified of the event by an interface card and then to instruct the OAM hardware to start generating the required cells on the required virtual circuits. If the OAM cell hardware cannot handle the cell or has not had its tables correctly updated to handle the cell yet, then it passes the cell on to the SM processor.

- SM Cell Buffer - This is a buffer memory that is used to buffer data between the SM processor and the OAM hardware subsystem. It is bi-directional in that the SM Processor can both place cells into the buffer for the OAM cell hardware to return to the Cell Switch fabric and can read any buffered cells from the buffer.

- S-AAL Hardware - Cells that are destined for the CAC microprocessor will almost invariably need to be reconstructed as frames before being processed by the CAC microprocessor. In addition, frames presented by the CAC microprocessor to be sent to other CAC microprocessors, ATM hosts and ATM switches will be processed by this hardware subsystem. Since AAL5 frame processing is fairly simple but intensive it is logical to place this functionality in dedicated hardware to remove the burden from the CAC microprocessor.

- DMAC - The Direct Memory Access Controller would probably be integrated with the S-AAL hardware but is shown here as a separate functional block. This interacts with the Bus Arbitration Controller to gain access to the system bus and either read frames required for conversion into ATM cells for the S-AAL hardware or write reassembled frames into the control memory.

- Control Memory System - This is merely a shared memory that the S-AAL, the CAC and the SM processor use to communicate with each other.

- Processor Communication Bus - This is the bus which the S-AAL, the CAC and the SM processor use to gain write and read access to the shared memory system.

- CAC system - This processes all of the signalling, routing and network management protocols. It receives an abstracted image of the ATM switch through the data presented to it in the Control Memory System by the SM system. It ultimately controls the switch but is protected from the complexity of the switch hardware by the SM system which achieves control of the switch either through direct control of the hardware through the Switch Control Bus or, if the required hardware is not under its control by communicating with the appropriate SM microprocessor in order to achieve this control. Since all communications and actions of the SM system are low level, except where high level information is exchanged with the CAC microprocessor, it is assumed that the communication between SM processors can be encapsulated within a single cell. Although not shown here, the CAC system could have its own unshared memory

independent from the Control Memory System in which data that need not be shared between the subsystems can be stored.

- SM microprocessor - This performs low level control of the ATM switch by directly communicating with hardware over the Switch Control Bus and by communicating with other SM microprocessors either through the cell switch fabric or through a SM Gateway.

- Bus Arbitration - Since three components, the CAC microprocessor, the S-AAL and the SM microprocessor all share the processor communication bus, some mechanism will be needed to allocate bus cycles to these units.

- Switch Control Bus - This bus will extend to all the interface cards and other hardware that a particular control processor card is responsible for. The SM microprocessor will control the various subsystems through the use of Control and Status hardware registers that interface to their respective controlling state machines. Exception conditions or other asynchronous control will be enabled by allowing the subsystems to interrupt the SM microprocessor through interrupt request lines. Note that the control on this bus will be low level in order to simplify the design of the other subsystems. In order to explain the various levels of abstraction maintained in the switch, consider how usage accounting could be implemented:

    - a simple counter of the number of cells switched per VC could be incremented by the interface card hardware whenever a cell is received or sent by the IM and OM.
    - At regular intervals (once every few milliseconds for traffic management purposes, once a minute for accounting purposes), the counters are read and reset by the SM microprocessor. The SM microprocessor then abstracts this information and stores it in a database in the Control Memory System.
    - This database can then be queried through the use of SNMP or CMIP by a network management system in order to process billing information, this function would be performed by the CAC microprocessor.
    - If a CAC microprocessor receives an SNMP or CMIP request for information about a VC that is not in a hardware group that it controls from a host that is connected to an interface in its control domain, it would reroute the request frame through the cell switch fabric to the correct CAC microprocessor that could then reply to the request. This technique imposes a constant cost in extra overhead and could scale to an arbitrary large switch for this example (Note that the CAC and SM microprocessors do not necessarily need to set up a VC in order to communicate with each other since it will be assumed that a routing tag is

appended to each cell that is routed through the cell switch fabric, thus if the port on which the destination CAC processor is positioned is known, the cells can simply be sent with this port number appended as the routing tag. This prevents a polynomial increase in the number of interconnections between microprocessors.)

- SM Gateway - Due to hardware considerations it is possible that the Control Processor might need to control or configure hardware that cannot be connected either to the Switch Control Bus or to the Cell Switch Fabric. An example of this might be a very fast cell switch fabric core that only needs to be configured when it is first connected and which does not support ports internally. Another example might be an ADSL modem bank which presents a switch interface to each of the connected ADSL hosts and which then multiplexes its traffic to be switched to higher capacity links through the cell switch fabric. An interface added to the Switch Control Bus would allow the SM microprocessor (under the control of its CAC microprocessor) to control and gather information from such devices without necessitating the complexity of providing them with their own independent SNMP or CMIP hosts).

Although this particular implementation of the control pathways within the ATM switch is certainly not the only possible one it is presented here as an architecture that could scale to enable the control and management of a very large scale ATM switch. Through the remainder of this dissertation it will be assumed that this control system is in place even when not specifically mentioned and it is assumed that the higher layer signaling, routing and management protocols can be successfully scaled to meet large switch sizes. There is a large body of research that could be performed on the scalability of these protocols in a large scale ATM switch, however the remainder of this dissertation will mainly be concerned with another bottleneck to the scalability of ATM switches, the Cell Switch Fabric.

Chapter 4

Fundamental Considerations in ATM Switches

Before the different methods of implementing the cell switch fabrics are discussed, some basic limiting factors on the design of ATM switching fabrics will be presented here. These mathematical analyses are interesting and important since they provide a conceptual framework of the limits of the various possible switching fabrics. However, in order to allow the analysis to be tractable, the assumptions about traffic patterns often have to be dramatically simplified and thus can give overly optimistic results for certain traffic patterns. A large portion of this discussion is taken from [68], however this has been extended with the authors own comments and analyses. We will assume that an ATM switch is constructed as indicated in figure 3.1. Here an Input Module performs the VPI/VCI address translation and submits a cell to the switching fabric. The cell is then switched to the correct output port in the cell switch fabric. In ATM, cell arrivals are asynchronous, thus it is possible that another ATM cell might be destined for the same output at the same time. This situation is shown in figure 4.1. Thus, fundamental to ATM is the need to buffer in an ATM switch in order to handle these eventualities.

The buffering can be performed at two possible positions in the ATM switch, i.e. at the

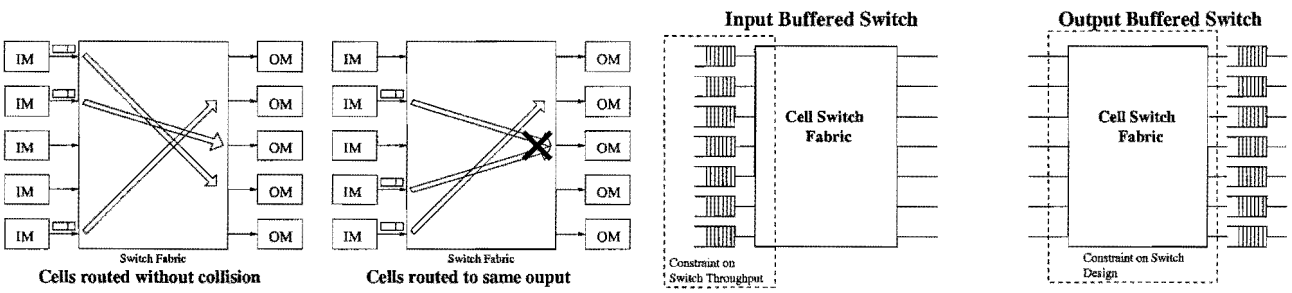


Figure 4.1: Collision at output of ATM switch

Figure 4.2: Input buffering and Output buffering in an ATM switch

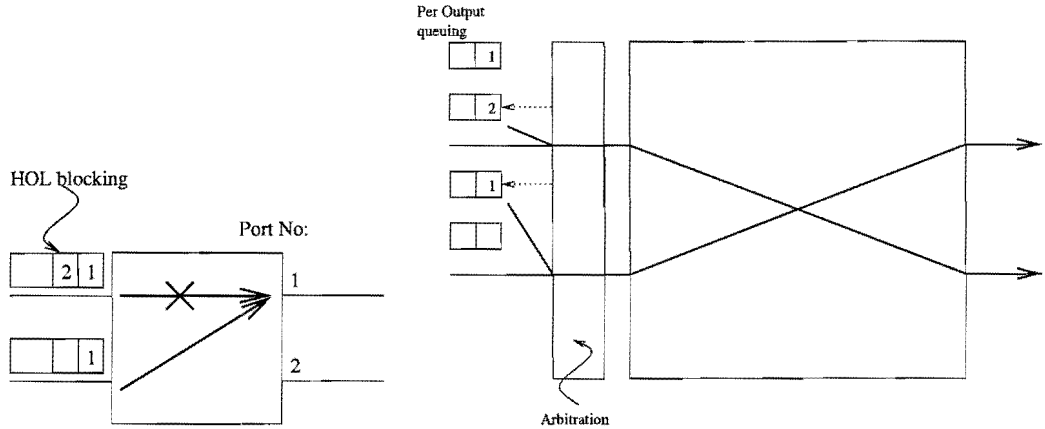


Figure 4.3: HOL Blocking in a two port switch

Figure 4.4: HOL Blocking solved with arbitrator and multiple FIFO queues

inputs or at the outputs of the switch. This is shown in figure 4.2. Note that the position of the buffers affects the ATM switch in different ways. If the buffers are placed at the input to the switch (and the buffers are simple FIFO buffers), then Head of Line (HOL) blocking occurs, however, in this case the fabric is not required to transport more than one cell to an output in any cell time slot. If the buffers are placed at the output of the switch, then, as shall be demonstrated, the behaviour of the switch is very close to that of a simple single server FIFO queue. Note, however that in this case, the cell switch fabric must be capable of delivering more than one cell to the output of the ATM switch in one cell time slot. This must be accomplished in one of two ways, by a speedup in the ATM switch fabric itself, or by a dilation in the number of outputs from the cell switch fabric in space.

4.1 Buffer placement

Why does HOL blocking occur? Figure 4.3 illustrates, here both input ports have a cell destined for port 1, however, the lower one is selected to reach the output. This means that the second cell on input port 1, which could profitably be switched down to port 2, cannot in fact be switched. Notice that HOL blocking can be avoided if multiple input FIFO queues (one per output port) are present at the input port and a central arbitrator determines which input port can service an output at any one time, as shown in figure 4.4. This presents two difficulties, the arbitration unit might not scale well, and it is a single point of failure. A proposed input arbitration scheme includes a ring reservation scheme [9], and neural networks (neural networks can be shown to scale to an arbitrary number of inputs, see [26] for a neural network arbitration scheme that approaches the performance of output buffering, earlier work was done by [56] and [14] using different neural network schemes).

Note that the commonly used mathematical symbols in ATM switches are presented in

Symbol	Common Usage
p	The probability a cell is present in a time slot
T	The length of a cell time slot
N	The number of inputs and output
γ	Throughput
μ	The service rate (or link speed)
ρ	Link utilisation (similar to and often equal to p)
$E(x)$	The expected value of x
$G(z)$	The moment generating function
$P(x)$	The probability of x
Q	Queue Length
n	The number of customers (equivalent to Q)
i	Input (used as a subscript)
o	Output (used as a subscript)
s	Switch (used as a subscript)

Table 4.1: Commonly used mathematical symbols in ATM

table 4.1 for assistance in the analysis that follows (taken from [68], page 189), this analysis assumes that an ATM cell that is blocked in the switch fabric is blocked.

Input buffering analysis with cell discard:

Assume that the traffic at each input link is Bernoulli, independent and homogeneous from link to link. There is thus the same probability p across the links that a cell is present in any time slot. The probability that an output will receive a cell in a time slot is then:

$$1 - \left(1 - \frac{p}{N}\right)^N$$

The throughput at any output point is thus:

$$\gamma_o = \left[1 - \left(1 - \frac{p}{N}\right)^N\right] \frac{1}{T}$$

For $N \gg 1$ this becomes:

$$\gamma_o = \left(1 - e^{-p}\right) \frac{1}{T}$$

If we let the probability of a cell arriving in a particular time slot increase to maximum capacity ($p = 1$), then we obtain:

$$\gamma_o \mid_{max} = \left(1 - e^{-1}\right) \frac{1}{T} = 0.632 \frac{1}{T}$$

Thus for a large number of ports and large traffic volumes, the best that an input buffered switch can do is a throughput of 63%. One is tempted to believe that preventing a cell being lost by blocking it at the input will improve this situation. (It would certainly improve the very dismal cell loss probability). However, the analysis on page 39, gives even less scope for optimism. It is convenient to consider output queueing before continuing with input buffering with HOL blocking.

Output buffering analysis

Of fundamental importance to ATM switching is the performance of an output queued ATM switch. This case is in fact optimal, and thus bounds on the behaviour of an output queued switch are extremely useful. This analysis is greatly reduced from [68] pages 190-200.

The traffic model of the preceding section is used. The number of cells A transfered to a queue j must obey a binomial distribution:

$$P(A=j) = \binom{N}{j} \left(\frac{p}{N}\right)^j \left(1-\frac{p}{N}\right)^{N-j} \qquad j = 0, 1, ..., N$$

note that as $N \to \infty$, $P(A=j)$ approaches a Poisson distribution with parameter p. If Q_m is the number of cells in a given output queue at the end of time slot m, and we use A_m to represent the number of arrivals at this output port in time slot m; the statistics of A_m are then given by the above equation. Then, Q_m must have the following form:

$$Q_m = \max\left(0, Q_{m-1} + A_m - 1\right) \tag{4.1}$$

This can be shown to lead to the following moment generating function, when we let $m \to \infty$,

$$G_Q(z) = (1-\rho)(z-1)\left[z - G_A(z)\right]$$

It can also then be shown that since A has a binomial distribution, its moment generation function must be:

$$G_A(z) = \left(1 - \frac{p}{N} + \frac{zp}{N}\right)^N$$

in addition, the utilisation ρ can be shown to be

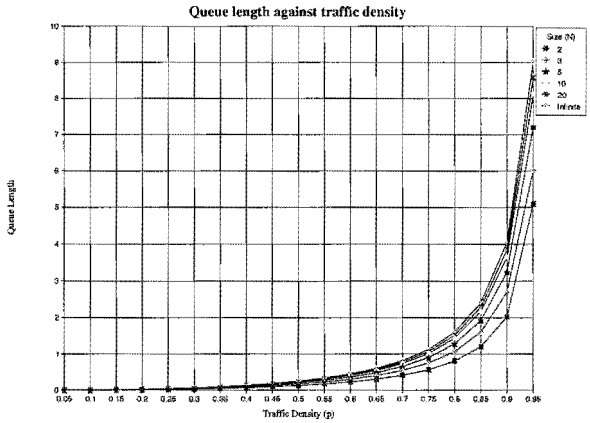


Figure 4.5: Mean Queue length for Output queued switch

$$\rho = E(A) = N\left(\frac{p}{N}\right) = p$$

Using the moment generating functions allows any of the queue statistics to be calculated. In particular the expected queue length can be shown to be:

$$E(Q) = \frac{N-1}{N}\frac{p^2}{2(1-p)}$$

Notice that for $N \gg 1$this approaches the queue length for a normal M/D/1 queue.

This result shows that an output queueing switch of any reasonable size has the same characteristics per output port of a single server deterministic FIFO queueing system. This result, coupled with the severe limitations imposed on a switches performance by input queueing illustrates that in general, output queueing is the desired solution in an ATM switch. The mean queue lengths (and hence cell time slot delays) for various sizes of output queued at switches are shown in figure 4.5.

The case where cells are not discarded, but blocked at the inputs in an input buffered switch are illustrated next.

Input buffering analysis, with input blocking

This analysis is taken from [68], pages 202-204. To simplify the analysis, let us assume that a unit time interval is equal to T to prevent a redundant $\frac{1}{T}$ appearing throughout the analysis.

Let F_{m-1} be the total number of cells transmitted through the switch over all output in time slot $m-1$. Clearly $F_{m-1} \leq N$. F_{m-1}input queues must have transmitted a cell to the outputs, freeing up F_{m-1} input spaces at the head of the input queues. Since we assume $p = 1$ each free head of a queue must be filled by a cell in the next interval. If we let A_m^i be the the total number of the cells moving to the head of their queues in slot m that are destined for output slot i we must have:

$$F_{m-1=} \sum_{i=1}^{N} A_m^i$$

Since cells are equally likely to be destined to any output, the probability that k of the cells moving to the head of the queue, is given by the binomial probability:

$$P(A_m^i = k) = \begin{pmatrix} F_{m-1} \\ k \end{pmatrix} \left(\frac{1}{N}\right)^k \left(1 - \frac{1}{N}\right)^{F_{m-1}-k}$$

Since a cell cannot be sent to the same output port in the same slot, some cells must be held at the head of their queues. If we let B_m^i be the number of cells destined for output i in timeslot m that are blocked at the head of their respective queue, then B_m^i must have the following characteristics over time:

$$B_m^i = \max[0, B_{m-1}^i + A_m^i - 1]$$

But this equation is precisely of the form of (4.1). If we let N become large and we let $m \to \infty$ (stationary conditions), then the statistics of A_m^i become Poisson. Then we must have $B_m^i \to B^i$ independent of m. Then:

$$E(B^i) = \frac{\rho_0^2}{2(1-\rho_0)} \tag{4.2}$$

where the utilisation ρ_0 is the desired utilisation. In order to determine ρ_0 we need another form for $E(B^i)$. We do this by determining by continuity of throughput that:

$$F_{m-1} = N - \sum_{i=1}^{N} B_{m-1}^i \tag{4.3}$$

In the stationary case, $F_{m-1} \to F$, thus $E(F)$ is the mean number of cells moving through the switch and $\frac{E(F)}{N}$ must be the utilisation per output port. But, by the same argument $E(A^i)$ is also the utilisation of an output line. Thus:

$$E\left(A^i\right) = \frac{E\left(F\right)}{N} = \rho_0$$

Then, using equation (4.3), and letting $m \to \infty$ so that stationarity applies and expectations can be taken:

$$E(F) = N\rho_0 = N\left[1 - E\left(B^i\right)\right]$$

thus,

$$E\left(B^i\right) = 1 - \rho_0$$

substituting this back into (4.2) we then obtain

$$(1 - \rho_0) = \frac{\rho_0^2}{2\left(1 - \rho_0\right)}$$

solving for ρ_0 then results in

$$\rho_0 = 2 - \sqrt{2} = 0.586$$

Thus, in the case where cells are not discarded in an input buffered switch, the best throughput that can be attained at full traffic density is 59%.

The above clearly illustrates the advantages of output queued switches over input queued switches. If an input queued switch is to be used, then the maximum utilisation that can be attained is 59%. Thus, in order to use an input queued switch, the switching fabric must be sped up by a factor of two. In an output queued switch, this utilisation is limited by the behaviour of any deterministic queued system. As illustrated in figure 4.5, the queue lengths grow without bound as the traffic density approaches 1 and the system is in fact unstable at full traffic density.

4.2 Time Division Switching

4.2.1 TDM Bus architecture

One of the most logical ways to create an ATM switch is to use a bus based architecture, increase the number of parallel lines on the bus (or decrease the bus cycle length) until it

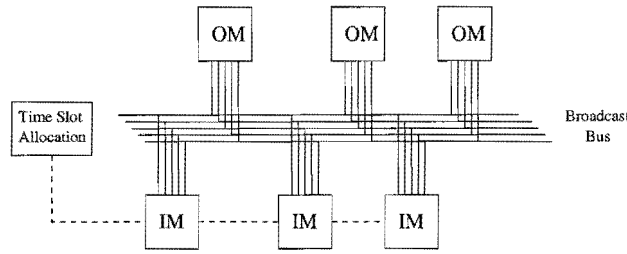


Figure 4.6: Bus based switching fabric

has sufficient speed to handle all of the ATM traffic and then simply allocate a time slot in which every input module places its data on the bus. Any output module could then read it off the bus and if necessary buffer it before sending the cell off on the transmission system. Thus, in this case, the "cell switch fabric" simply consists of a bus and a time slot allocator as shown in figure 4.6. Let us assume that the bus width is W and that the cell time slot per IM is T ($2.7\mu s$ assuming a 155Mbps OC3 transmission links). Since there are $53 \times 8 = 424$ bits, if there are N input and output modules, then it is self evident that the bus cycle length is constrained to be: $\frac{T.W}{424.N}$. This means that for a 32 bit bus, and 16 inputs the bus cycle cannot exceed $13\eta s$ or 79MHz. This goal is by no means unreachable, but it becomes apparent that there is a natural absolute upper bound on the size of the switch that can be accommodated by this fabric. The OM in a broadcast bus architecture must be capable of writing N cells to its memory in one complete cycle and reading one out. Thus, the minimum memory access time becomes: $\frac{T.W}{424.(N+1)}$. Thus for this example , a $11\eta s$ memory access time would be required. This could be accomplished with off the shelf commercial SRAMs.

There is in addition, in ATM, a finite limit on the amount of simple bitwise parallelism that can be exploited, since an ATM cell is no longer than 424 bits. For an indication of the upper bound on the bandwidth of a bus, let us assume that the width of the bus is one ATM cell and that the maximum reachable bus speed is $100MHz$, the largest number of OC3 ports that could then be supported is 270. However, it must be remembered that busses exhibit reflection due to imperfect termination and impedance mismatch with their connectors. In addition they are limited by the number of inputs that a module would be required to drive (fan out). Thus a broadcast bus cannot in practice be scaled to even these sizes. This switching architecture therefore cannot be used for large scale switches, despite its simplicity and ease of implementation. A great advantage of this architecture is its ability to automatically support multicast, since any group of OMs can read the cell information off the bus in the same time slot.

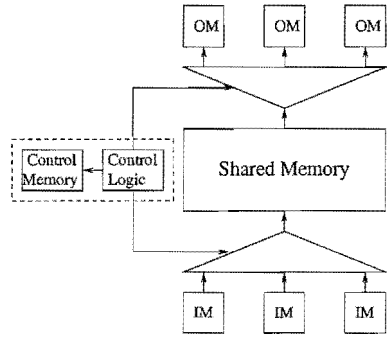


Figure 4.7: Shared Memory cell switch fabric

4.2.2 Shared Memory Architecture

Another possible architecture for an ATM switch is to pool all of the output buffers together into a single common shared memory pool. This is shown diagrammatically in figure 4.7. This architecture obviously has the same switching characteristics as an output buffered architecture. However, it also has the advantage that due to the statistical multiplexing of the traffic that it buffers, it can achieve the same performance as an output buffered switch with smaller buffer sizes [66]. It has much the same scaling limitations of a bus architecture switch, except that it does not suffer as much from fan out problems and does not suffer from termination and connection reflections. It is also more suited to be placed on an integrated circuit.

It should be obvious that in every time slot, N writes and N reads must be performed on the memory, one for every IM and one for every OM. Thus, the minimum access time for this type of architecture is $\frac{T.W}{424.2.N}$ if a single ported memory is to be used. (I.e. one which only allows one reading and writing device per access cycle). If a dual ported memory is used, then obviously this minimum access time is $\frac{T.W}{424.N}$.

Shared memory architectures self evidently do not scale to large switch sizes. However, the reduction in buffer capacity is a very useful characteristic of this type of switch (at the expense of more complex buffer management), and thus often the building blocks of larger scale switches will in fact be implemented as small shared memory switching elements integrated onto a single IC. This approach will be demonstrated in the two switching fabrics that will be simulated as part of this thesis, the Clos switching architecture (section 5.4) and the Hypercube switching architecture (section 5.5).

4.3 Classes of ATM MIN switch

It becomes obvious after considering the limits imposed on ATM switching by a bus or shared memory architecture in sections 4.2.1 and 4.2.2 that time division switching cannot scale to

Routing information place / Routing information time	Cell based (Routing tag)	Network based (Routing table)
Connection based	I	III
Cell based	II	IV

Table 4.2: Classification of ATM Switching Fabric routing

the speeds required by large scale ATM switches. Thus, switching in space is required. In this technique switching elements are connected together in some network topology in order to route the cells from one input to the other. This architecture is known as a Multistage Interconnection Network (or MIN).

[66] identifies four possible routing strategies in order to route the cell from the input to the output. These are shown in table 4.2. This classification makes use of two variables, the time when the routing decisions are made and where the routing information is stored. (Here routing means the route that a particular virtual circuit or cell will take through the switching fabric). There are two times when the routing decision can take place; it can take place when the virtual circuit is first established or it can take place whenever a cell arrives. There are also two places where the routing information can be stored; it can be stored in tables in the switch fabric itself (i.e. at every SE of the switching element), or it can be appended to the cell as a routing tag before the cell is introduced into the switching fabric.

First the implications of when the routing decision is made will be discussed.

- **Connection based routing** - The primary advantage of this approach is that if the path through the switch taken by all the cells belonging to a particular VC is determined at connection time, then, so long as a FIFO discipline is maintained within all of the buffers inside the switch, the cells are guaranteed to arrive in order at the output of the switch. There are a number of disadvantages to this approach however. The advantages of statistically multiplexing cells within the switching fabric will be lost, (in other words, a larger amount of switch hardware will be required to support a given per port throughput). The number of internal nodes and links of a switch will far exceed the number of ports, thus, the routing information required to represent a path through a switch is far larger than that required to represent an output port. The final disadvantage is that connection decisions cannot be made without considering all of the other connections in the ATM switch. In particular, the decision is hard to parallelise. Since a major bottleneck in ATM switches is likely to be the number of VC's to be set up and torn down in a unit of time, this is a particular disadvantage.

- **Cell based routing** - In this technique, the route through the switch for every cell is determined every time a cell appears at the switch. The primary advantage of this

approach is that a statistical multiplexing gain can be realised over connection based routing. However, if internal buffers are used, this approach can result in cells losing their cell order in transit through the switch. This can require resequencing hardware at the outputs of the switch in order to allow the cells to be correctly reordered, (for a self-routing switch that preserves cell order at the outputs, although it is internally buffered, see [35]). Severe implementation difficulties present themselves when a Self-Routing switching fabric is not used since central control logic might be required to create the path through the switching fabric for every cell that arrives on every input, this presents a natural bottleneck to the scale of the switch.

The routing information can be stored either in the cell, or in the cell switch fabric itself. The advantages and disadvantages of both of these approaches are discussed next.

- **Routing information stored in switch -** In this approach, the route that a cell must take is stored in the switch itself. This could be done by having a central control module create a path through the switch, or by storing routing tables at every point in the switch and in effect creating a virtual circuit through the switch. This approach has the advantage that less overhead is needed for every cell since the routing tag either becomes unnecessary, or can be reduced in size (this approach almost invariably becomes necessary for multicast in the switch).

- **Routing information stored in the cell** - In this approach, a routing tag is appended to the cell before it is submitted to the ATM switching fabric. This will normally consist simply of the port number to which the cell is destined. The disadvantage of this approach can arise if the routing tag becomes very large, in this case it might be better to combine a reduced routing tag with information stored in the switch.

To complement these descriptions the characteristics of each class of switch will be discussed:

- **Class I** - In this class the route through the switching fabric is determined at connection set up time and the route is appended to every cell before it is placed in the network. This is not generally recommended since the number of possible routes through a switching fabric is normally considerably larger that the number of ports in the fabric, thus the routing tag is likely to become very large. This approach also fails to take advantage of any statistical multiplexing within the fabric itself since all the cells are obliged to follow the same route. This technique is guaranteed to preserve cell order.

- **Class II** - In this class, the route through the switching fabric is determined for every cell. The routing tag is appended to the cell itself. It normally consists of an output

port and some control information. The advantage of this switch is that the routing overhead can be small since the cell does not need to specify its entire route through the switch, in addition, full advantage of the statistical multiplexing of the switch can be attained. The primary disadvantage is that it is difficult to guarantee the correct in-order deliver of cells if there are buffers in the switching fabric itself. Thus, an output resequencer might be required.

- **Class III** - In this approach, the route through the switching fabric is determined when the connection is established, and the routing information is stored within the switching fabric itself. In essence, the ATM switch is divided up into many sub-ATM switches that then switch the ATM cells through the switching fabric. The prime disadvantage of this scheme is the lack of statistical multiplexing. However, cells are guaranteed to be correctly sequenced.

- **Class IV** - In this approach, the route through the ATM switch is determined every time a cell arrives, however, the routing information is stored in the network. This would normally be the result of having a central controller determining the route through the network (as in a Benes network [7]) or having the input port determine a route through the switching fabric. In general this type of fabric is not appropriate to large scale switches since the overhead required to set the route of each cell through the network is too large to perform for every cell.

From the above description of the classes it becomes obvious that class I and class IV switches are not appropriate for large scale ATM switches and that class II and class III can be used. In fact, it is often better to use a combined strategy where class II routing is used for point to point traffic and class III is used for multi-point traffic to reduce the size of the routing tag. For this reason the sole focus in this dissertation will be on class II switches. A hybrid class II / III scheme will be discussed for multicasting in the hypercube fabric.

4.4 Self Routing Switches

In general switching fabrics can be classified as non-self routing or self routing switching fabrics. A non-self routing switch would include a class I or III switch, or, a switch in which the route through the fabric is determined by a central controller (possibly through a Benes network which approaches the minimum complexity for a space switch but the complexity of routing requires an external controller). In a self routing switch, the routing is performed independently by the switching elements themselves. This obviously requires greater complexity on the part of the switching elements themselves but removes the need for an external controller. It is obvious that the presence of a central controller or route chooser

will not scale to determining the route at either the cell level or the connection level for a sufficiently large switch. Thus, non-selfrouting fabrics are not considered in this dissertation.

A farther classification of switches can be divided into multipath switches and non-multipath switches. A multipath switch is one in which there are multiple routes through the network. A counter example would be a crossbar switch in which only one possible route exists between an input and an output. Section 4.1 illustrates the throughput constraints imposed by input buffering, which is required in a non-multipath switch. In order to supply the output buffers in an ATM switch, either the switch must be dilated in time (by a speedup over the output rate), or in space (by providing multiple links to an output port). Thus, this dissertation will focus on multi-path self routing (MPSR) switches.

If internal buffering is used in an MPSR switch, it is possible that the cells may become misordered before reaching their output. In the case of the proposed switching fabric presented in this dissertation (the hypercube switching fabric), this is in fact the case. Thus, output resequencing is required.

Chapter 5

Space Division Switch Fabrics

It was shown in chapter 4 that switching in the time domain cannot scale beyond a certain internal bandwidth. It is thus necessary when considering large scale switches to restrict oneself to space division switching fabrics. Using the abstract switching fabric of figure 3.1 we will assume that a type II (see table 4.2) fabric is being used. I.e. a routing tag specifying the destination output port is prepended to the cell. Section 4.1 illustrates the disadvantages of input buffers in an ATM switch, however in practice often a compromise must be made between these two extremes. There are many different space division fabrics and an overview of all of them cannot reasonably be presented here (see [4] for a thorough survey and analysis of a large number of switching fabrics).

However, a number of principles of switching and the trade-offs to be made in the various switching architectures can be discussed by considering several different space division switches. The ultimate goal is of course to deliver an acceptable degree of performance with the minimum cost. The difficulty is presented in precisely how the cost of a cell switch fabric is determined. A traditional measure is the scaling of the switch with the number of SEs. For example, a crossbar switch evidently has a complexity of $O(N^2)$ while a Batcher-Banyan network [59] has a complexity of $O(N(\log_2 N)^2)$. Thus it becomes obvious that a Batcher Banyan will be more cost effective for a sufficiently large number of ports, regardless of the relative complexity of the Switching Elements themselves. However, many other metrics are possible especially since the switching fabrics themselves must be placed on an integrated circuit. Thus, for example [82] presents an architecture (the multi-channel deflection crossbar) that is specifically optimised for VLSI placement and compared with the batcher banyan and is shown to result in a larger number of input and output ports for a given die size and a lower clocking rate. Here, despite the indications of complexity theory, the crossbar network costs less than the Batcher Banyan network! Other sets of metrics exist for architectures that require more than one IC to be implemented. A cost comparison of a number of networks in terms of the number of pin limited chips required to implement a particular switching architecture for a defined level of performance is presented in [84].

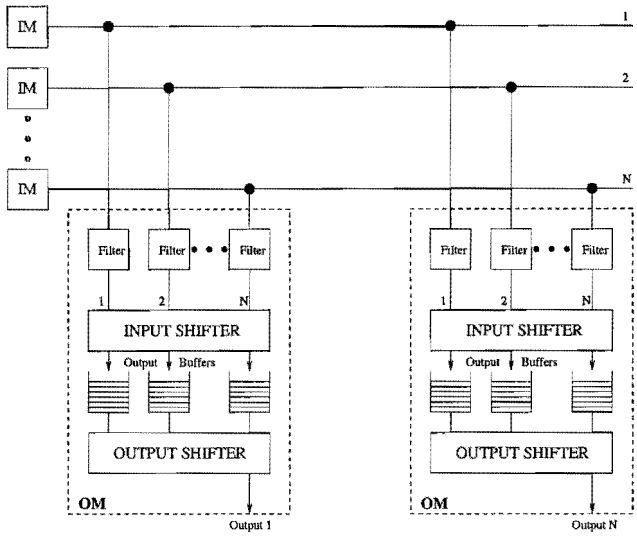


Figure 5.1: A broadcast bus architecture

The approach taken here will generally be to consider the order of complexity of a switching fabric, but also the complexity of the individual SEs and the ease of implementation of a particular architecture.

5.1 The Knockout Switch

At first glance it appears that the goal of pure output buffering will present a prohibitive expense in terms of the number of switching elements. If we have N inputs then, potentially, at any interval all N inputs could receive a cell that is destined for the same output. Thus, either every single output must be capable of delivering N cells in one cell time (which gives the same scaling problems as a shared medium architecture), or there have to be N outputs that put the cells into some sort of shared queue. (A shared N input, 1 output queue that does not require a speedup is used in the knockout concentrator which was described in [83]. This maintains cell order by shifting incoming cells in a circular fashion into a queue and removing the cells one at a time from the output in the same order). This would require there to be N^2 outputs from the cell switch fabric. In addition, the question arises as to how these N^2 outputs are to be supplied by the cell switch fabric itself. Let us consider the following architecture (based on the Knockout Concentrator [37]) shown in figure 5.1. In this architecture, every IM appends a destination tag to every cell. These cells are then broadcast onto a bus. Every OM has N filters that listen for the correct output port number and send matching cells through a shifter to an output buffer (the input and output shifters are synchronised so as to evenly fill buffers and to preserve cell order, creating a virtual shared buffer). The output shifter then reads out the cells in the correct order. This architecture is evidently completely output buffered and does not require any speedup in order to fulfill its objective. What is its complexity? It is evident that there are simply N input and

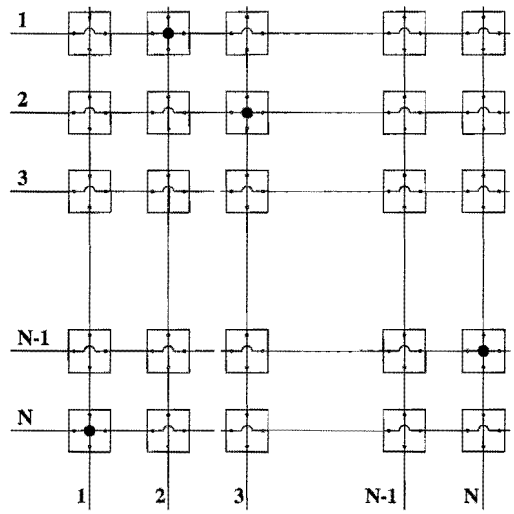


Figure 5.2: A crosspoint network (shift by 1)

output modules. It is equally obvious that there must be N^2 filters and buffers in every output module. Thus the minimum complexity of this architecture must be $O(N^2)$, (thus there will be one million filters and buffers for a thousand port switch). This architecture achieves the same complexity as a crosspoint network (although both filters and buffers are far more complex than a crosspoint). However, this still requires a million filters and buffers for a thousand port switch. Another consideration is the complexity of the input shifter and output shifter, the output shifter will evidently only require N tri-state drivers onto the output and control logic. The function presented by the input shifter is potentially far more complex since it must move up to N cells to arbitrary outputs. We will assume that its function is implemented by a crosspoint network. The crosspoint network requires N^2 crosspoints in order to switch N inputs to N outputs (see figure 5.2). (This assumption of using a crosspoint network is not arbitrary, but also takes into account the fact that these networks are logical candidates for placing on an IC's surface).

Taking into account the input shifters of the elements, this switch requires N^3 crosspoints. Although these are very simple elements (a couple of logic gates), this represents a generally unacceptable number of crosspoints since a 1000 port switch would require 10^9 crosspoints across the network.

Our initial conjecture that pure output buffering could become impractical for a large number of inputs and outputs appears confirmed. However, it is worth considering that zero cell loss is not required in an ATM switch, merely a very low cell loss probability (10^{-8} to 10^{-10} for some services). It is therefore worth considering the probability that a certain number of ATM cells will be destined for the same output in the same cell time slot. The following analysis is take from [68], pp. 207-208.

> We assume that each of the N inputs has a Bernoulli probability p that a cell appears in any one cell time slot, independent from slot to slot. If A represents

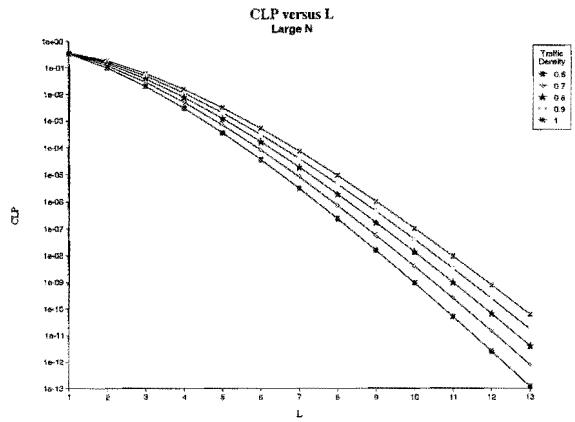


Figure 5.3: CLP against L and p

the number of cells to be transfered in any one time slot then the probability that $A = j$ cells will be transfered to any one output is binomially distributed:

$$P(A = j) = \binom{N}{j} \left(\frac{p}{N}\right)^j \left(1 - \frac{p}{N}\right)^{N-j} \qquad j = 0, 1, ..., N$$

From this it can be shown that if only L cells can be accepted at any one output (and for $N \to \infty$) then the resulting cell loss probability must be:

$$P_L = \frac{p^L e^{-p}}{L!} + \left(1 - \frac{L}{p}\right)\left[1 - \sum_{k=0}^{L} \frac{p^k e^{-p}}{k!}\right]$$

For full link utilisation, this becomes:

$$P_L = \frac{e^{-1}}{L!} - (L-1)\left[1 - \sum_{k=0}^{L} \frac{e^{-1}}{k!}\right]$$

This becomes very low for a surprisingly low L (the number of outputs that may receive a cell in a given time slot). The CLP is shown against L and p in figure 5.3. This indicates that for surprisingly low values of L very low CLPs may be attained. In fact for a CLP below 10^{-10} every output port need only accept 11 cells in every cell time slot.

The knockout concentrator makes use of this phenomena by only allowing L of the N cells to be buffered at any output. The switch accepts a small cell loss probability in order to reduce the complexity of the switch elements. The mechanism to select which of the N inputs to accept and place in the L buffers is to hold a tournament (very much like a tennis

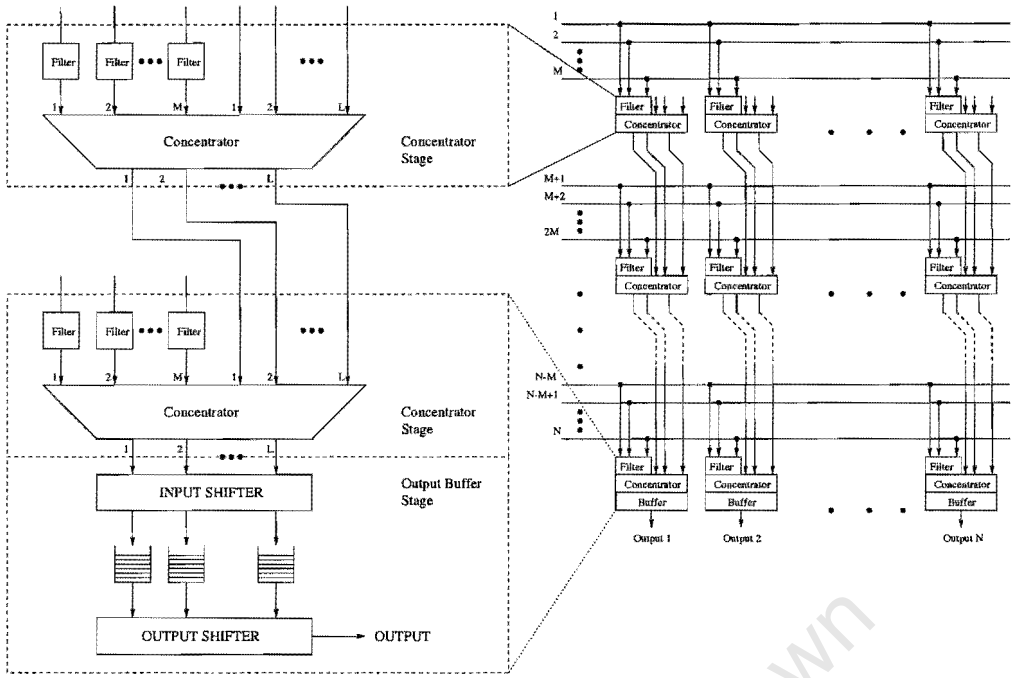


Figure 5.4: Knockout Concentrator Network

tournament) in which losing and winning cells are selected. The details of this stage will not be presented here, see [37] and [59], see also [66] for a very good analysis and description of this switching fabric. The most important features of the fabric are shown in figure 5.4.

The resulting complexity of an ATM switch based on this knockout principle is much improved. It is evident that the number of crosspoint needed to implement the input shifter is now L^2 where L is the knockout constant. Since this is constant for any size of switch, it can effectively be ignored for the purposes of the complexity of the fabric. This corresponds to the great advantage that the knockout concentrator (figure 5.4) has against a simple broadcast bus architecture (figure 5.1) namely that regardless of the number of elements in the network, the structure of the SEs is the same. It is obvious that the number of Filters in the network is N^2. Less obvious is that the number of SEs in the network (the Filter and Concentrator units) is also $O(N^2)$ (despite some claims to the contrary, i.e. [68], p. 206).

In order to illustrate this consider the total number of SEs requires to implement the switch where $N > M$ and $\frac{N}{M}$ is an integer. (This isn't necessary but helps visualisation). Then the number of SEs required per output port is obviously $\frac{N}{M}$. There are N of these outputs and thus the total number of SEs must be $\frac{N^2}{M}$which is $O(N^2)$. Here M is a constant of scaling which represents a technology constraint since in this case it is obviously desirable to package every SE as an IC. To reduce the total number of ICs (often the primary determinant of cost) we would desire M to be large.

It is apparent that using the knockout principal, a complexity of $O(N^2)$ can be attained for the purely output queuing case. This complexity is still prohibitive for the constructor of a very large switch. Other Switching Fabrics will thus be discussed in order to pave the way

for the discussion of the Hypercube switching fabric, a novel ATM switching architecture.

5.2 Contention in a Space Division Switch

A space division switch consists of a number of switching elements (SEs) interconnected in some kind of topology. In general, an ATM cell will be received by the switching element and then routed out of one of the (generally two) outputs of the switching element. The question arises as to what should be done within the switching element when two ATM cells are destined for the same output in the same time slot. There are number of possible solutions to this problem and they will be discussed here. All of the various solutions are shown diagrammatically in figure 5.5.

- **Internal Buffering** - in this solution, one of the cells is sent on the correct output of the switching element, while the other is simply placed in a buffer. If cells are already buffered, then they are obviously selected first. If the buffer overflows, then one of the remaining solutions will have to be adopted. This solution also naturally limits switch throughput. But, this solution has been used in banyan, crosspoint and Clos networks. Indeed, many practical large scale ATM switches employ this solution. One of the primary difficulties is that in MPSR switching fabrics, this technique cannot in general guarantee cell order. This technique has been used in crosspoint switches with the bus matrix switch [58], combined with input buffers in the Limited Intermediate Buffer (LIB) switch [25] and has many variations within Banyan networks (the first being [76], see also [77]).

- **Dilation** - In this technique, the number of internal links in the ATM switch are increased between elements. Thus, instead of one link interconnecting the elements of a banyan network, 2 are used. Then, in this case, if contention results in the ATM switch, the ATM cells can simply be sent to the same output. There are a large number of versions of this ATM switching architecture mainly found in banyan networks since in this architecture the blocking within the network is very acute and cannot be solved with internal buffering. [38], [39], [45], [2]. For an overview of these scheme see [4] pp. 1592-1593. More complex variants of the dilation scheme have been proposed including versions which are fault tolerant (for example see [85]) by using the extra internal links to provide redundancy.

- **Speed up** - This technique is often used with internal buffering in a switch where the buffering imposes a throughput constraint on the switching fabric. It can also be used for fabrics with throughput constraints by reducing the amount of traffic that each switching fabric receives and having many switching fabrics in parallel (as used with

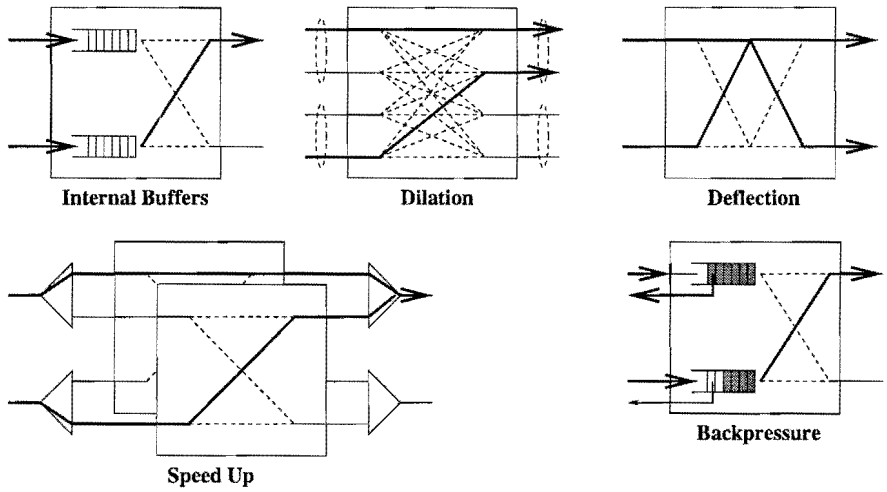


Figure 5.5: Possible responses to output contention in an ATM SE

Banyan networks in the Replicated Banyan network ([38], [39]) . This technique must be used with Clos networks [18] in order to allow them to reach reasonable throughput performances. (See [23]). This technique was proposed in [34] for a $N \log_2 N$ element switch (unfortunately it required a speed up of N). The hypercube switch proposed in this thesis also requires a speedup in order to be implemented, but restricts this to $\log_2 N + 1$. Speedup has also been proposed to improve the throughput of buffered banyan networks (for example [10]).

- **Backpressure (Blocking)** - In this technique, when the buffers of an ATM switch become full, a signal is propagated towards the input SEs indicating that it should not receive an ATM cell until buffer space is available. This essentially allows the buffering to be distributed over a number of the SEs resulting in a smaller required buffer size per SE. However, if the backpressure signal causes a tree saturation effect [65]), it can result in traffic hotspots under non-uniform traffic conditions since nodes towards the input are intermediate nodes for other output elements. This signal can either have local significance or global significance. This technique is obviously most useful for internally buffered Switching Elements. At the ultimate extreme, backpressure can be used with no buffers in the network in order to select cells from a queue in a purely input buffered switch, as used in [3]) in the form of the two phase algorithm.

- **Deflection** - In this case, when two ATM cells need to be routed to the same output of the SE, one of the cells is routed to the correct output, and the other is simply routed to the incorrect one (with the intention that at a later point in time or space it will be correctly routed). This approach is used in the tandem banyan network [75], the sunshine switch [29] and the shuffleout switch [5], [20]. The hypercube switch implements a combination of deflection and dilation in which a number of possible outputs from a particular SE can be valid destinations.

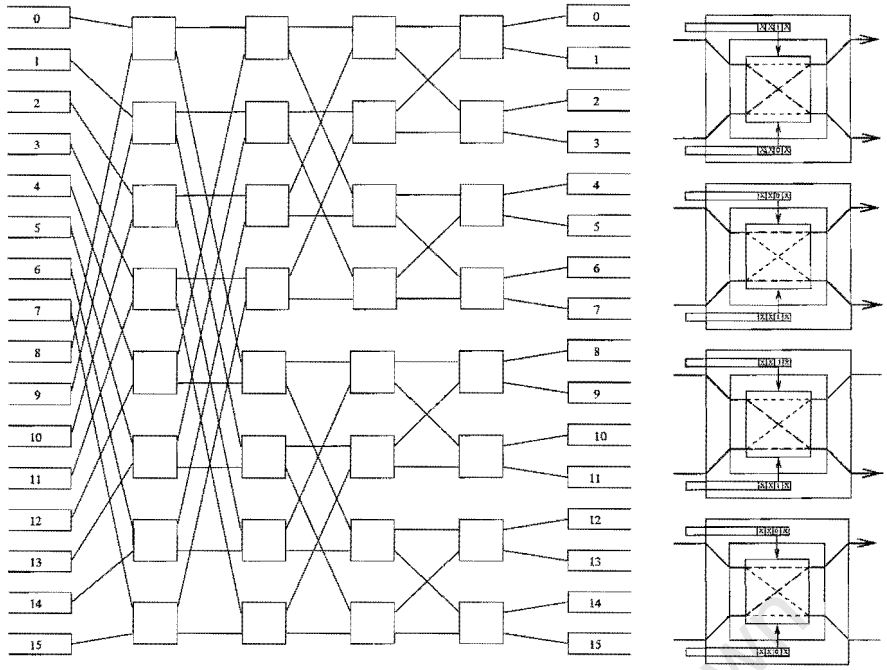


Figure 5.6: Banyan Network

- **Discard** - Obviously, when buffer space has been exhausted, or there are are no alternative links for the cell in the form of dilation or deflection the cell must be discarded. In any practical ATM switch there will be a finite but small probability of discarding an ATM cell, this is acceptable as long as the probability is sufficiently small. Note, however, that the cell switch fabric of an ATM switch should have a far smaller cell loss probability than the switch with output buffering in the OM. The reason for this is that the sophistication of buffer management and cell scheduling obviously cannot be reasonably performed in every SE since these expensive but vital functions must be confined to elements that scale linearly with the size of the switch. For a discussion of the functions that must be performed in the OM see chapter 6. Techniques might be used that ensure that cells with a high CLP bit are preferentially discarded and buffer management techniques such as pushout [17] and queue length limits [80] can ensure that cells from badly behaved traffic sources are discarded preferentially to well behaved sources.

5.3 The Banyan Network

It was shown by Benes [8]), that the minimum possible complexity required to implement a switching network with 2×2 crosspoint SEs is $N \log_2 N - 1.44N + \frac{1}{2} \log_2 N$ or $O(N \log_2 N)$. He then developed the Benes network, which has a complexity of $N \log_2 N - \frac{N}{2}$. The difficulty of this network topology is the complexity required to route elements through it which necessitates a central routing controller. Although this is feasible in a synchronous network,

the scaling difficulty presented is too large in an ATM switch. However, it is interesting to observe that this lower bound is much lower than the $O(N^2)$ attained with the knockout concentrator, thus it is tempting to believe that large scale switches can be constructed with a much lower complexity (at the expense, possibly, of additional delay). The difference in scaling is illustrated in figure 5.20. This illustrates the difference in growth of a $O(N^2)$ network (crosspoint or knockout concentrator), an $O(N\sqrt{N})$ network (Clos network), an $O\left(N\left(\log_2 N\right)^2\right)$ network (Batcher-Banyan) and $O\left(N\log_2 N\right)$ network (the Banyan network and some of its relatives). This graph is not intended to be to scale, but it is intended to provide an impression of the various scaling properties of the networks.

A 16×16 Banyan network is shown in figure 5.6. (There are also other equivalent topologies that have the same properties, these belong to the "delta" class of networks [63], this includes the Baseline [81], the Reverse Baseline [81], Omega [44] and others, these networks are generally used interchangeably since they have been shown to be topologically equivalent [81], [70] and to have the same performance under uniform traffic [21]). A Banyan SE in all of its four possible states is illustrated to the right of the figure. As this shows, the SE switches based on the value of only one bit in the routing header appended to the ATM cell. If the bit is '1' then it is routed towards the lower output, if it is '0' then is is routed towards the upper output. Every layer of the banyan network switches on a different bit. This can be implemented in two ways, either the SE simply refers to a different bit, or, the header may be rotated at each SE position. The latter solution is the more complex but has the advantage that all SEs are then identical throughout the switching fabric. As indicated it is possible for an ATM cell to be blocked at the crosspoint. It is self evident that the number of layers in the switching element must be $\log_2 N$. Each layer of the network contains $\frac{N}{2}$ SEs, thus the total complexity is $\frac{1}{2}N\log_2 N$. This is below the minimum number of SEs required to route any input to any output under all permutations, thus certain permutations of inputs will cause the switch to block. If this occurs then the cell must be buffered, deflected, blocked or lost. All of these approaches to handling contention in an ATM switch are discussed in section 5.2. Turner proposed the first broadband packet switch as a buffered banyan switch [77]. However, the buffered solution presents a throughput degradation of $\sqrt{N}$ and thus requires a speed up [10].

The primary advantages of a Banyan network is the low complexity of the switching fabric and the simplicity of each of the SEs (each SE is required only to inspect a single binary digit). The primary disadvantages are the degradation in throughput with size and a tendency to develop hot spots in non-uniform traffic. For another use of a Banyan network for ATM switching see [62].

A very useful property of the Banyan network is exploited in the Batcher-Banyan type of network in order to provide the same interconnection capabilities as a crosspoint network but with a far smaller number of switching elements. The property that is exploited is that

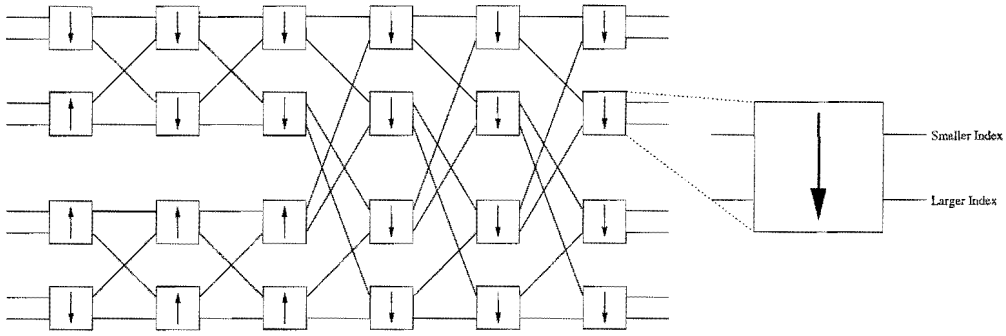


Figure 5.7: Batcher Sorting Network

when ATM cells are presented to the inputs of a banyan network in a sorted order with no gaps in the cells the banyan network can route all of the remaining cells to their respective outputs without a network contention. Thus, if the banyan network is preceded by a sorting network, every input can output a cell in every time slot. The sorting network normally used is the Batcher network [6] illustrated in figure 5.7. The resulting network is then known as a Batcher-Banyan network. This network topology does allow any input to communicate with any output and has fewer than N^2 elements (it in fact has $O(N(\log_2 N)^2)$ SEs) (However the SEs are reasonably complex, some of them having to implement a comparator) and it still does not solve the problem that the network is incapable of supplying an output in more than one time slot with an ATM cell. This problem can be solved in a number of ways, input buffering can be used (as in the Moonshine (or three phase) Switch [32]) resulting in HOL blocking, or in the most famous example, a recirculating buffer may be used (as in the Starlite Switch [31] and the Sunshine Switch [29]). The sunshine switch in particular achieves the throughput of output buffering by the use of recirculating buffers (however this results in added complexity in the switching elements). Although Batcher-Banyan networks have resulted in great interest from the academic community, little commercial interest has followed and all commercial large scale switches have used the Clos network architecture described in the next section.

5.4 Clos Networks

The Clos network was first proposed by Clos [18] as a method that could be used to dramatically reduce the number of crosspoints in a circuit switched network. The network has also found use in many ATM switches, both in commercial use [24] and as a research switching fabrics (for example, the Roxanne Switch). See also [23], [49], [67] for applications of this switching fabric to ATM switching, A Clos network take the general form shown in figure 5.8.

As shown in the diagram, a Clos network has three layers. A first layer consisting of K SEs each of which is capable of switching n inputs to m outputs (an $n \times m$ crosspoint is often

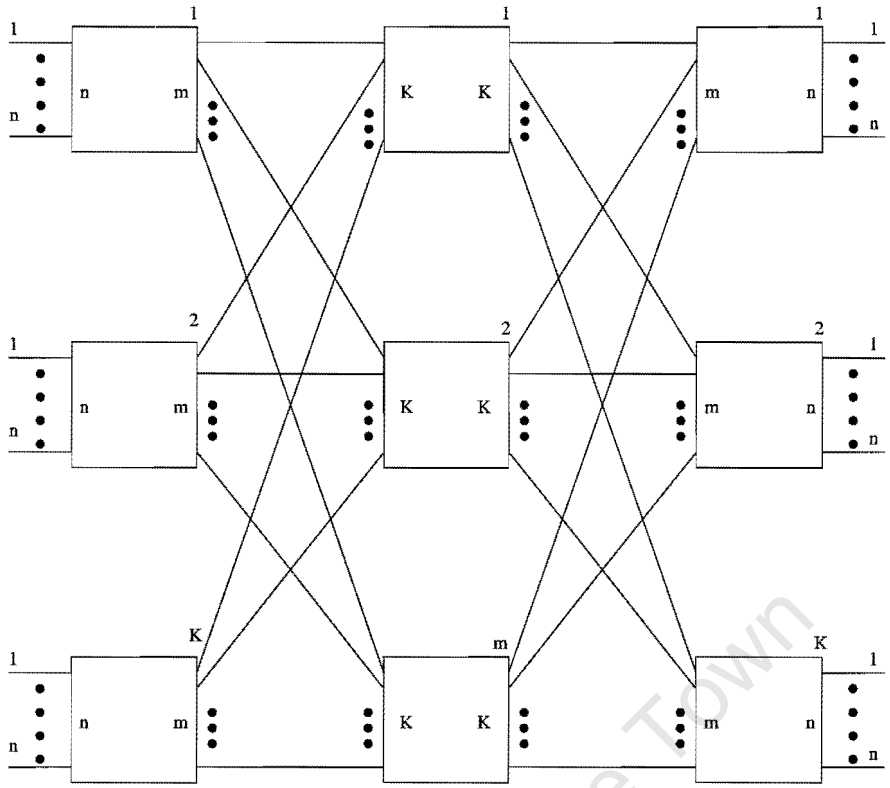


Figure 5.8: The Clos network

assumed to be used) a central layer of m, $K \times K$ SEs and an identical final layer. It is obvious that the number of inputs and outputs in this network is $Kn = N$. The complexity of the Clos network can be calculated more simply if we assume $K = n = m$ (which will also be used in the simulations). Then the size of the network is simply $N = n^2$. This means that the number of crosspoints in each switching block is $\sqrt{N}\sqrt{N} = N$. The total number of switching blocks in the network will be $3n$ or $3\sqrt{N}$. It then becomes obvious that the equivalent number of crosspoints in the network is $O\left(N\sqrt{N}\right)$.

In practice, the Clos network is normally internally buffered if it is to be used in an ATM switching network. Thus, rather than use a crosspoint based switching fabric, the smaller switching blocks of figure 5.8 are used in what amounts to a smaller shared memory switch is used. Here the memory is shared in order to reduce the overall memory usage of the buffers. This can be quite feasibly accomplished since the SE will typically be packaged as single IC.

A possible implementation for the IC is shown in figure 5.9. (The function of the various elements is described in table 5.1.) Here, the incoming cells are first buffered before being multiplexed onto a TDM bus. A control unit then determines where in the shared memory buffer the cells should be stored and stores information in the control memory to determine which logical output queue the cell should be placed on. Cells within the memory buffer (which could in this case be profitably duel ported) are then writing out on the TDM bus to the output buffers which then send the cells towards the next switching element. The size of the output buffers need not be an entire ATM cell since they can be supplied at

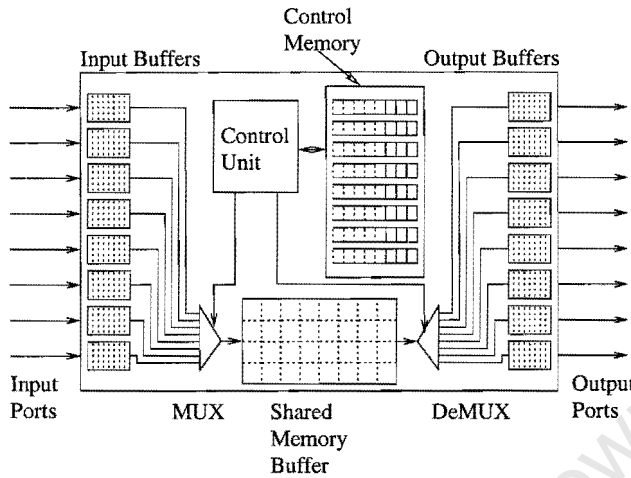


Figure 5.9: Large Buffered Switching Element

Sub-Element	Function
Input Ports	Receive ATM cell - electrical conversion
Input Buffers	Allow synchronisation of Cells at MUX
MUX	Select Cells from Inputs and write to Buffer
Shared Memory Buffer	Buffer Cells in Random Access Memory
DeMUX	Select Cell from Buffer to write to Output Buffers
Output Buffers	Handle speed difference between SE and Output Ports
Output Ports	Transmit ATM cell - electrical conversion
Control Unit	Provide control signals to various sub-elements
Control Memory	Store Queueing information for various output ports

Table 5.1: Function of Large Buffered SE sub-elements

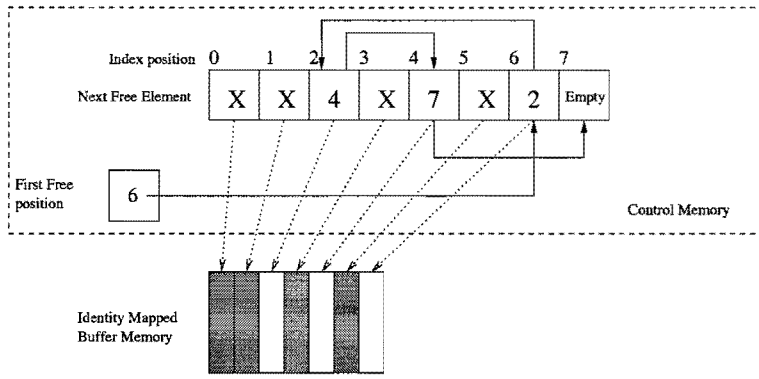


Figure 5.10: Free Buffer Management in Control Unit

any rate by the shared memory buffer and control unit. The allocation of buffer memory can be performed very simply by maintaining a singly linked list of indexes to free memory locations (one of the main advantages of ATM switching is the relative simplicity of buffer management engendered by the use of a fixed size ATM cell). This allocation scheme is assumed to be used in all shared memory buffered SEs in both the simulation and in the rest of this dissertation. (In some of the simulation a simpler analogue to this can be used in which a counter is simply used to indicate the number of elements in a buffer, since, for performance reasons, pointers to cells are manipulated in the simulation rather than the cells themselves). The data structure is shown in figure 5.10. (Note that in this case the highest representable number in the number of bits required to represent the shared buffer space is assumed to be a NULL pointer). The principle advantages of this approach to buffer management is that the cost in order to both find a free buffer position and to reclaim it is $O(1)$. Note that in some fabrics [30], a slot based approach to memory management may be used rather than allocating memory on a cell by cell basis. In this approach the memory is allocated in slots and the ATM cell occupies the slots rather than an entire cell sized buffer. The main advantage of this approach is that is further reduces the amount of memory required since an ATM cell is read from multiple slots, as the slots are evacuated, a new cell can be written into them. In the case of cell based buffering this cannot easily occur. The drawback of this technique is obviously the added complexity of the Control Logic and the increased number of slots to be managed. Also, the slot loss probability must be lower than the cell loss probability. Thus, this extention to buffer management is not considered in the simulations of the switching fabrics in this thesis.

The Clos network can be used typically as either a class I or a class II network. In other words, the routes through the network can either be determined at connection setup time or, the routes can be determined per cell. The latter approach will be taken in simulating this fabric since it most closely approximates the routing scheme in the hypercube fabric and thus provides a more relevant model for comparison.

In order to consider its use as a class II network it is necessary to realise that the first layer

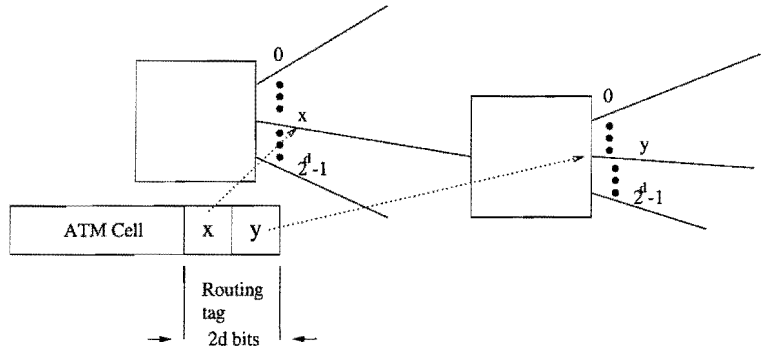


Figure 5.11: Routing in a Class II Clos fabric

of the Clos network can perform a distribution function since no matter what output the cell leaves on, the cell is capable of reaching its destination by being routed through the later two layers. The next two layers do not have to same latitude in routing the ATM cells. The routing is simplified if the number of inputs and outputs in all of the Clos elements are an exact power of two. This is illustrated in figure 5.11, here $n = k = m = 2^d$, if a routing tag with the output is appended to the cell and the cell simply sent to the output indicated in each of the fields, it will reach its correct destination. This property makes routing the cells relatively easy within the network. Since the Clos network is internally buffered, this will result in cells being delivered out of order to the output and will result in a resequencing overhead. This also illustrates how the network can be used as a class I network. If another field is simply added to the routing tag for the first layer of the network (rather than having the elements randomly routed), then all cells belong to a particular VC can be routed along the same path, this will result in the cells arriving in the correct sequence at the output but will have the disadvantage of not taking full advantage of the multiplexing gain and traffic smoothing achieved by randomly distributing the cells in the first layer.

The performance of a configuration of the Clos network with both a FIFO queueing scheme and an alternative non-FIFO queueing scheme is compared in chapter 10.

5.5 The Hypercube Fabric

The hypercube fabric is a novel switching fabric, presented for the first time in [42] and supplementally in [43]. It takes a different approach to switching than most pure space division switches and implements both time and space division switching. The departure the hypercube fabric takes from most space division fabrics is not to consider the switching process as taking place through a number of layers of a network, instead it takes the form shown in figure 5.12.

The fabric consists of two different types of elements, N routing elements (REs) and and large number of much simpler switching elements. The REs determine the route that an

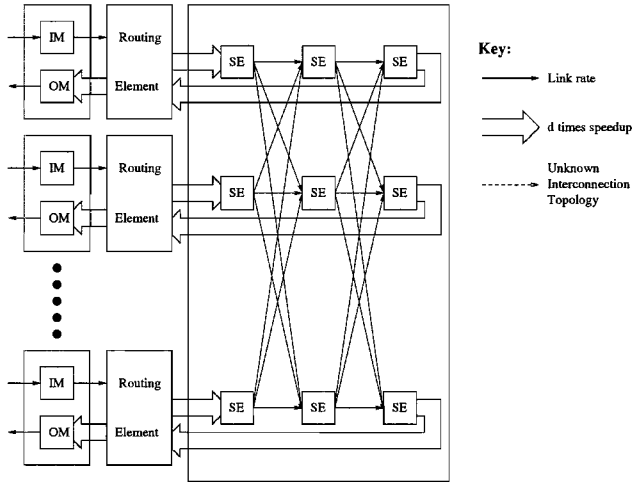


Figure 5.12: Concept of hypercube switching fabric

ATM cell should follow and the switching elements interconnect the routing elements in some manner. It is not required that the SEs be capable of connecting every RE with every other RE, but merely that a subset of the REs can communicate with each other (hereafter known as the neighbouring nodes or neighbouring REs). Obviously if the REs cannot communicate directly with every other RE, an ATM cell might have to traverse more the one RE in order to get to its destination. Let the number of hops that the cell has to traverse be called d.

If a cell must traverse d hops in order to reach the output node it is obvious that in order to maintain the link throughput rate the REs must be sped up by a factor d. In fact in any one cell arrival time, the RE must be capable of receiving one cell and either routing out, or passsing to the OM d cells.

The question then is can we find an interconnection topology between the REs that allows d to remain small and also allows the number of SEs to scale reasonably with the size of the network (also, the interconnection should allow the SEs to remain as simple as possible). Obviously we require d to remain small in order to restrict the speedup needed in the REs otherwise the switching fabric may become technologically unattainable.

A useful interconnection between the SEs may be realised through the use of a hypercube switching fabric. A four dimensional hypercube is shown in figure 5.13. A number of properties of hypercubes can be observed from this figure.

Every hypercube has:

- A dimensionality d.
- A number of interconnected nodes $N = 2^d$, each connected to d neighbours.
- A numbering sequence such that every node's number in base 2 differs from its neighbours by precisely one binary digit (they are Grey Code sequenced).

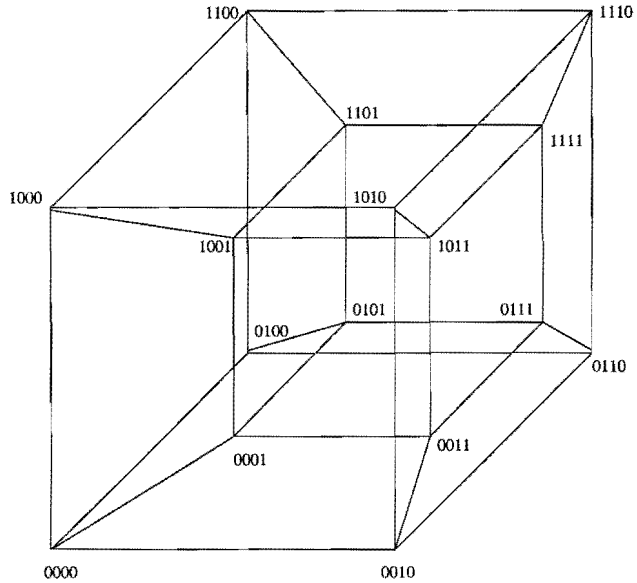


Figure 5.13: A four dimensional hypercube

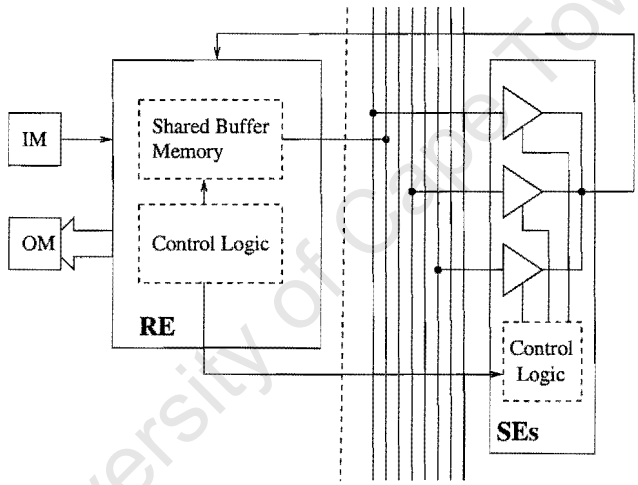


Figure 5.14: RE and possible SE of the hypercube switch

- Each link between nodes represents one dimension of the hypercube and one index position in a base 2 number system.
- If you are at a particular node X and need to read node Y, then all the positive binary digits in X XOR Y represent a valid dimension along which you may travel, and following this route, there are at most d hops.

A hypercube interconnection thus appears to fulfill the requirements needed for this alternative approach to ATM switching since the number of interconnections between each RE is limited to d and the number of hops required to move from any RE to any RE is also $d = \log_2 N$. This value grows very slowly with the size of the switching fabric and thus this speedup can be maintained up to a very large scale ATM switch. A possible interconnection and switching architecture is shown in figure 5.14.

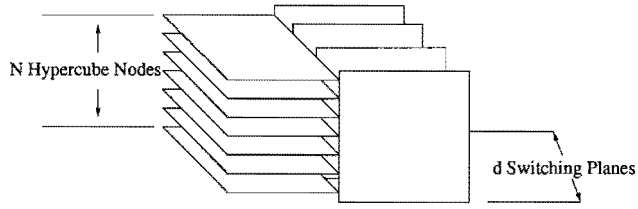


Figure 5.15: Hypercube Speedup

The d speedup of the hypercube switch can be attained by simple bitwise data parallelism. This involves duplicating the number of switching planes in the network by d and submitting all data needed in parallel to these planes. This is illustrated in figure 5.15. A speedup of d is the minimum required if a dual ported memory is used in the RE. The reason for this is that although the RE must be sped up by d in order to achieve the routing, in every cell time slot it must also receive a cell from the input. This means that if non dual ported memory is used, the hypercube node must process $d+1$ cells in every cell arrival time. Since dual ported memory can use twice the chip real estate of non-dual ported memory, it is suggested that the REs simply be sped up. However, synchronisation would be difficult to maintain between the REs and the SEs if one were operating on a cycle of d cells and the other were operating on a cycle of $d+1$ cells. It would be simpler to speed up the operation the switching fabric to $d+1$ cells per cell arrival time as well. The extra cycle time could be used to exchange reachability information among cells in order to detect failed nodes to increase the reliability of the switching fabric. (See appendix A for some suggestions on how fault tolerance can be achieved in the hypercube fabric).

For thoroughness a completely interconnected three dimensional hypercube using the SEs of figure 5.14 is shown in figure 5.16.

The complexity of the hypercube fabric can be determined as follows:

The number of REs is self evidently N, however if the speedup of d is implemented by bit wise parallelism, then the complexity of every RE is self-evidently d. The complexity of the complete set of REs is $Nd = N \log_2 N$. This is of the same order as that required for a minimum interconnection network as shown by Benes (see section 5.3 on 55). There are $d+1$ switching planes of SEs. As illustrated in figure 5.14 there are N SEs each of which contains d tri-state buffers. The total number of tri-state buffers required in the hypercube switch is $Nd(d+1)$ or $O\left(N.(\log_2 N)^2\right)$. Thus, the complexity of the overall switching fabric is the fastest growing of these terms, $O\left(N.(\log_2 N)^2\right)$.

Note that in figure 5.14, the control logic of the SE and RE are not specified. The reason for this is that there are a number of possible routing schemes and interconnection schemes to allow this routing to occur. Two schemes for the hypercube will be dealt with in this dissertation, first a simple FIFO queueing routing scheme. In chapter 8 on page 90 an alternative queueing and routing scheme is presented which allows non-FIFO queuing, a

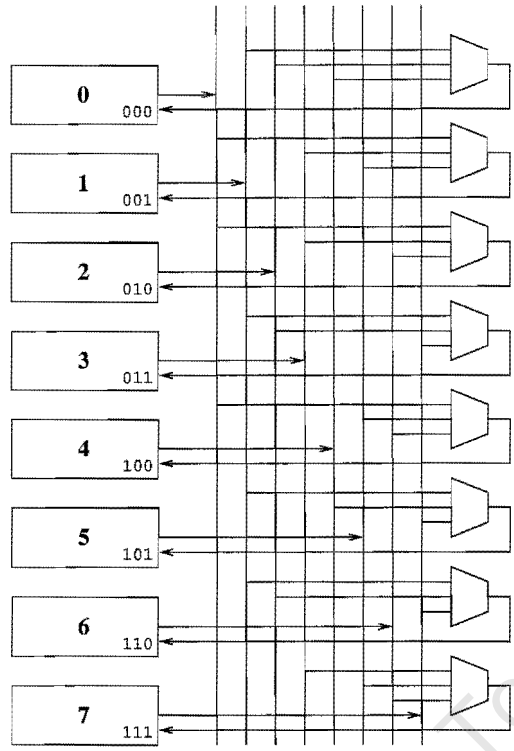


Figure 5.16: Fully interconnected three dimensional hypercube

more flexible approach to selecting the next hop in the network and improved delay and loss performance to the FIFO queued scheme.

It is obviously desirable to ensure that the complexity of the most numerous element of the switch is the simplest. For this reason, tri-state buffers were selected in figure 5.14 for the most numerous of the switching elements. However, as shown in both figure 5.14 and in figure 5.16 this requires every RE to place its data onto a bus for collection by the SEs. However, buses present difficulties in terms of the fan out from the REs and also from reflection from the ends of the bus due to imperfect termination as well as reflection from the connectors to the bus due to imperfect impedance matching. It would be desirable to find an interconnection network in which every RE communicated with only a single SE. This can be done if the interconnection sequence between the REs is made strictly sequential, i.e. every RE is connected to its neighbour in order. This has a number of advantages, the Control Logic for the tri-state buffers can be centralised in the switch (further reducing the complexity) and, a form of the banyan network can be used for interconnecting the switching elements. This interconnection scheme is illustrated in figure 5.17.

This is referred to as a crosspoint banyan since the elements within the network perform no processing of the ATM cell themselves. Instead they are either simply crossed (they switch the incoming cell over) or open in which case the ATM cells do not change direction. Since every column in a Banyan network is capable of routing an ATM cell according to one bit of a routing mask. This means that by crossing all the ATM cells in one column only, the ATM

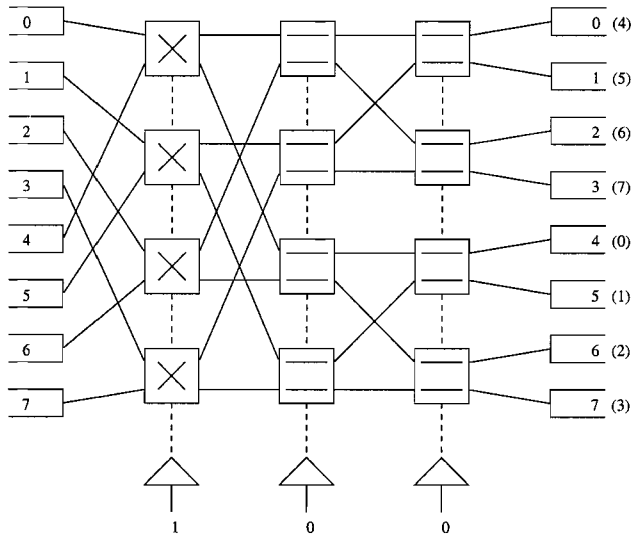


Figure 5.17: Crosspoint Banyan interconnection network

cells will be routed to the neighbouring RE of each RE in the dimension that is crossed. This is illustrated in figure 5.17 where the highest order column is crossed. The figures on the right hand side indicate the output port number and the corresponding input port it is connected to in this configuration. In this case, as expected, they all differ in their highest order bit.[1]

This network cannot result in a cell loss since each crosspoint is either crossed or open. It also removes the problems of echo and timing presented by the bus solution. In addition, it results in the centralisation of the control logic of the SE planes, further reducing their complexity. The crosspoint elements in the Banyan network are in addition extremely simple (a couple of CMOS gates), further reducing the complexity of the switching fabric.

Now that it has been determined to simply sequentially switch the SEs and thus sequentially connect every RE to its neighbour, a routing algorithm for the ATM cells in the REs must be determined. The RE used in the Hypercube routing scheme has the form shown in figure 5.18. Unlike the generic SE used in the Clos networks (figure 5.9) it has only two inputs and outputs and does not require output or input buffers, resulting in a much simpler SE than that attainable using a Clos network. This is because it is synchronised both with the IMs and OMs and with the hypercube fabric itself (for a sufficiently large hypercube, this synchronisation could become problematic and might need to be relaxed, reintroducing input buffering in the SE, this complication will not be considered here however).

A routing tag is added as overhead to the ATM cell. This is shown in figure 5.19. The various fields are:

[1] An interesting observation is that all hypercube nodes can be connected to each other simultaneously through this network as long as every node is connected to a node whose node number differs to its by the same number of bits. A possible use of this could be found in the input cycle where second neighbours could communicate with each other in a $d^2 - 1$ count cycle in order to trade information about their common neighbours (for example to collaboratively determine a node's failure) (see appendix A).

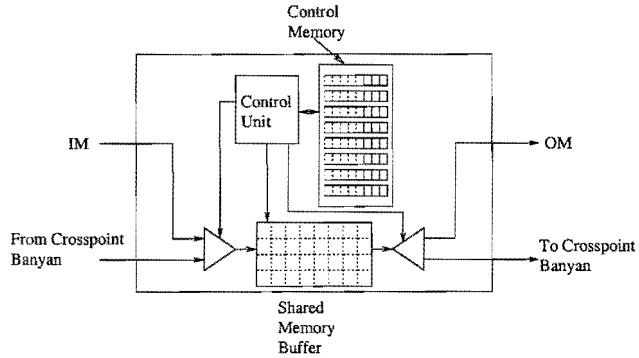


Figure 5.18: The Hypercube Switching Element

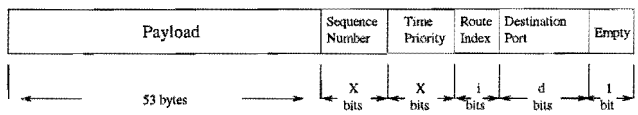


Figure 5.19: FIFO Routing Algorithm cell overhead

- A Sequence number of X bits: This is used for resequencing at the output. See chapter 8 on page 90 for a description of the use of this field and a discussion of an appropriate size for it.
- A time priority field of X bits: This is discussed in the same chapter, it is used both within a variant of the hypercube switching fabric and for the output resequencing
- A destination port of d bits: Here $d = \log_2 N$. This simply indicates the port on which the ATM cell will leave.
- A route index of i bits: Here $i = \lceil \log_2 d \rceil$. This is the route that the ATM cell will follow through the hypercube network.

When a cell is received at a RE, the following routing algorithm is performed in order to determine which logical FIFO queue to place the cell onto:

- The destination port address is exclusive-ored with the node's address
- If the result is zero, the correct port has been reached and the cell is sent directly on to the OM
- Counting forward from the route index amongst the exclusive-ored bits, the first bit which is '1' is found (the count is recirculated to the beginning if no bit is found)
    - The cell is placed in the corresponding logical queue.

This algorithm can in fact be used for both Class II and Class I routing. In the case of Class I routing, a route index is determined for every VC that is set up (the commitment of all

of the RE nodes must be determined to see if the route can be supported). In this case the cells will arrive in sequence, but the advantages of statistical multiplexing will not be gained in the switch.

In Class II routing a route index counter is kept at every RE, for every cell received the index counter in incremented. This technique has the advantage that the cells are evenly distributed throughout the switch fabric.

This routing algorithm attains the following objectives:

- The traffic is evenly distributed throughout the switch
- Every node has d parallel routes to every other node (i.e. routes that share no internal nodes)
- The longest route is $d+1$ hops while the average route has $\frac{d}{2}$ hops.
- Every input/output port pair distributes their buffering requirements through on average d^2 nodes.
- It is simple enough to implement in hardware
- It can be modified to route around malfunctioning nodes (see appendix A on page 142 for a description of this process)

The performance of this algorithm and the hypercube fabric is compared with other fabrics and techniques in chapter 10.

5.6 Order of Switching Fabrics

In order to illustrate the effect of the order of different network topologies, figure 5.20 has been devised. This illustrates the growth of various network topologies of $O(N \log_2 N)$, $O(N(\log_2 N)^2)$, $O(N^{\frac{3}{2}})$and $O(N^2)$. (The order of the majority of ATM switching fabrics). Order represents the single fastest growing term of the formula describing the number of elements. In addition, any scaling factors in that fastest growing term are ignored. As illustrated in figure 5.20, although a switch fabric might have a higher order of complexity than another, over a particular portion of its growth it might have a lower cost than another, lower order of complexity fabric. This might be due to the complexity of the SEs themselves (leading to a much smaller scaling factor in one of the terms), or it might be inherent in another aspect of the switching architecture (for example the SEs in a crosspoint network are much simpler than in a Banyan network, although it requires a much greater number of SEs). What order notation does indicate accurately is the relative cost of the network as N

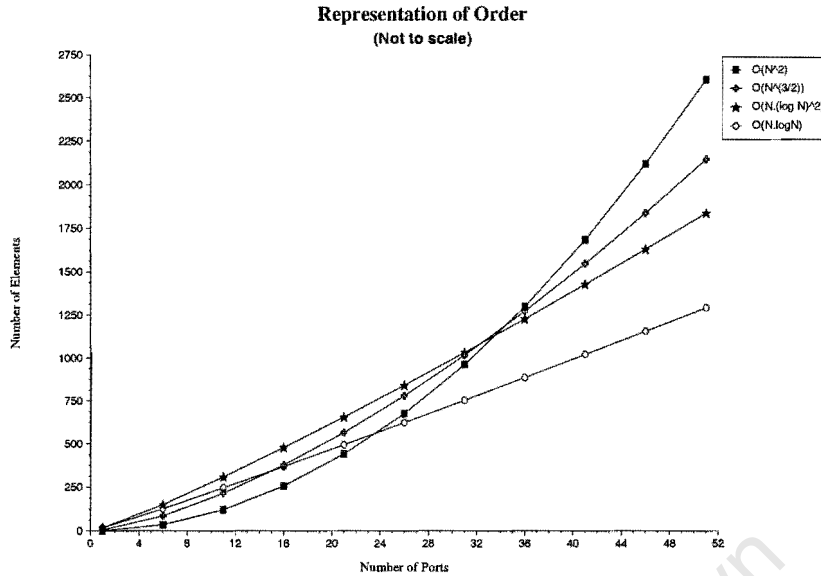


Figure 5.20: Representation of Order in a network

becomes very large (which justifies its use in this thesis), however a higher order switching fabric might be more cost effective if the size of the switch is small.

Chapter 6

The Output Module

The output module is responsible for queueing the cells before submitting them into the network. This module is not influenced particularly by the size of the ATM switch and generally falls out of the scope of this thesis. However, the studied switch fabrics do not guarantee cell order (unlike in most smaller switching fabrics) and thus require a cell resequencer which will be described in this section. In addition, many of the greatest impacts on the performance of an ATM switch are in fact to be found in the component of the switch since it is responsible for buffer management and output cell queueing (also typically it is responsible for such features as EPD and PPD). These topics will be discussed in the abstract and references to further reading given, then the approach taken for simulation in this thesis will be discussed.

6.1 Buffer Management

Due to the asynchronous nature of ATM, cells must fundamentally be buffered at the output (or, less preferentially, at the input) of the the ATM switch. We will not focus on input buffering in this chapter due to the reasons given in chapter 4. However input buffering and input/output buffer management techniques have been proposed [69].

The original concept for ATM was that QoS restriction would be enforced on a per VC basis and that Usage Parameter Control would ensure that traffic would not be submitted to the network that would overwhelm the ATM switches in its path. However, in practice, ATM switches carry a large proportion of ATM traffic which is characterised by a large amount of burstyness and lack of any QoS of service constraints or guarantees. It is in this light that buffer management has become very important in modern ATM switches. There are two essential tasks that buffer management must perform. The first is to ensure that there is sufficient buffer capacity to guarantee the QoS parameters that are agreed upon when the Virtual Circuit is set up. The second is to ensure that badly behaving virtual circuits do not jeopardise the buffer requirements of well behaved circuit. There are two essential techniques

that are used to accomplish this, queue length limits [16], [49] and hot spot pushout [17], [80].

The simplest technique is queue length limits. In this technique, every VC has a reserved queue size that is determined upon connection set up. When a cell is received, the current queue length is first inspected. If there is sufficient reserve capacity for the cell, the cell is queued, otherwise it is discarded. This technique has the advantage of being comparatively very simple to implement. However, it can be overly conservative and waste buffer capacity if there are VCs that are consistently under-utilising their buffer capacity. Thus, the modification of dynamic queue length thresholds has been proposed [16]. In this technique the queue lengths are increased dynamically depending on the amount of unused capacity and allowed to drain away naturally to smaller limits if other VCs become more active. This approach retains most of the simplicity of static queue length thresholds, while conferring many of the advantages of hot spot pushout (discussed next) without the costs.

Hot spot pushout takes a more direct approach to punish misbehaving virtual circuits. In this technique, all cells are buffered on being received at the output so long as there is sufficient buffering capacity. If the buffer capacity is exhausted, then a cell is robbed from the most badly behaved VC (i.e. the one with the longest buffer length) and discarded. This technique is self evidently more efficient than that of queue length limits since the buffer capacity is always used to capacity if necessary. However, it is considerably more complex to implement.

The approach used in the thesis to simulate buffer management is very simple. VCs are not simulated (in order to reduce complexity and to allow the steady state of the switch to be more easily studied). The approach taken is to reserve a fixed amount of buffer space for every traffic class in the final output buffer stage. When a cell is received at the output, the space left in the buffer for that class of service is checked, if there is sufficient buffer space, the cell is allowed through, if there is not, the cell is discarded. This model is extremely simplistic, however, the main focus of this thesis is on the scalability of the switch fabric itself, and as noted, the buffer management is essentially independent of the switch size if output buffering is used.

6.2 Cell Output Scheduling

The next function that must be performed is the scheduling of the ATM cells before being submitted to the output. This is essentially where the rubber of the QoS negotiations meets the road of the transmission system. There are many possible cell output scheduling algorithms, examples include Leap Forward Virtual Clock, [72] and QLWFQ, [61]. The objective of the Cell Output scheduler is conceptually simple; ensure that the cells are

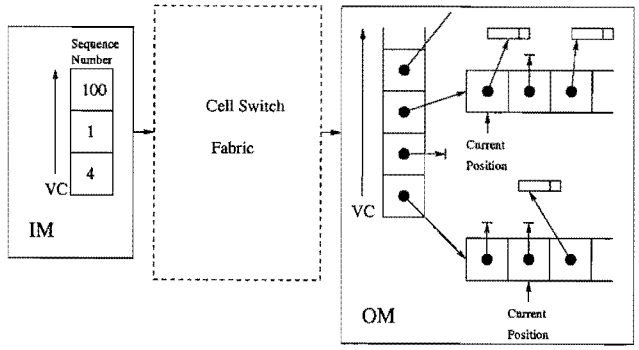


Figure 6.1: Sequence count based resequencing

scheduled at the output in such a way that they meet the QoS agreements set up by the VC. There are, however, many different techniques used to attain this goal and it presents the scope of a full scale thesis in itself. The reader is thus encouraged to consult the references for further information.

The technique used for cell output scheduling in the ATM switch is in fact extremely simple. It consists of a scheduler which will always schedule cells of a higher priority traffic class if there are any available in the queue. A cell belonging to a traffic class is simply inserted into the appropriate FIFO queue when it is received.

6.3 Cell Resequencing

Due to the fact that the routing networks considered in this thesis are all Class II and contain internal buffers, cell may arrive out of sequence at the outputs. Since ATM requires the in order delivery of cells, the cells must be resequenced at the outputs. Two different techniques will be discussed, sequence count reordering and delay reordering.

Sequence count reordering is straight forward. In this technique, every virtual circuit maintains a sequence count number at every input to the switch. When a cell is received at the input, this sequence count number is inserted as an overhead into the ATM cell and the cell is submitted into the cell switch fabric. When the cell arrives on the output it is inserted onto a per VC sequence count "time line" in the position indicated by the sequence count number. Every time line has a current sequence count pointer which will only remove the ATM cell from the queue when one in the correct sequence is found. When this occurs it will advance the pointer to the next slot (which could immediately cause the next ATM cell to be output if it were found in the correct order). This is illustrated in figure 6.1.

It is obvious that this approach will result in the ATM cells being resequenced at the output. However, the cost of this resequencing algorithm is high. Every virtual circuit must have its own resequencing line and in addition, keeping track of all of the sequence pointers is difficult (and requires a speed up to remain stable at full load). The other disadvantage is

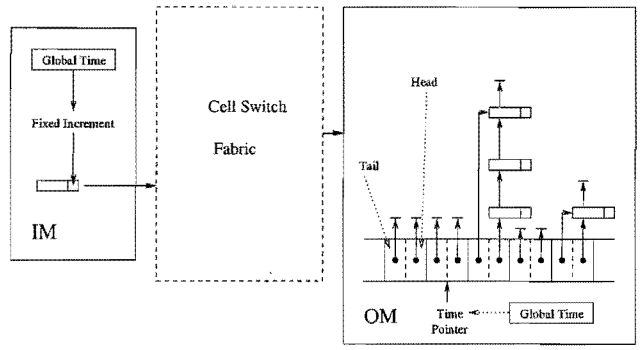


Figure 6.2: Time Output Resequencer

that if a cell is lost in the cell switch fabric (although the probability should be negligible), the virtual circuit will not be able to dequeue the next cell. This is especially problematic since the output resequencer has no way of determining if a cell has been lost.

As a result of these disadvantages, another resequencing technique must be considered, a time based resequencing technique. In this technique, a globally synchronised clock is kept throughout the switch. When an ATM cell is received, a fixed increment is added to the global time and inserted as a sequence count to the ATM cell. This increment is larger than the largest delay through the switching fabric. At the output of the ATM switch, a time line is kept, each of the potential time slots is represented in the time line and consists of a linked list of cells that become eligible in that time slot. A time pointer in the output module is also tied to the global time. It may extract ATM cells from the timeline but it may never exceed the global time. Cells are added to the tail of the output resequencer list and removed from the head. The time pointer advances at a faster rate than the link rate in order to preserve stability in the output resequencer, a value of twice the link rate is assumed for the simulations. The cell switch fabric ensures that any cell that is delayed past its maximum delay is automatically dropped.

This technique requires only a single time line in order to provide resequencing and also results in a very small jitter at the output. However, the disadvantage is the delay that must be imposed on every cell. This can unnecessarily delay cells at the outputs and results in a larger buffer capacity being required. Two techniques are proposed in order to prevent this. One is to treat the time that the cell must be resequenced by as a priority within the cell switch fabric (this technique is pursued in chapter 8), this ensures that cells that have a very low delay will receive a higher priority for exiting the switch than other cells. This will improve the latency of certain classes of cell.

The other technique involves a composite of the above technique. If a large number of virtual circuits coexist on a switch, it is likely that the cells from each virtual circuit will be received in order regardless since the maximum delay through the switch fabric will be smaller than the delay between cells from the virtual circuit. In addition, even when presented with a

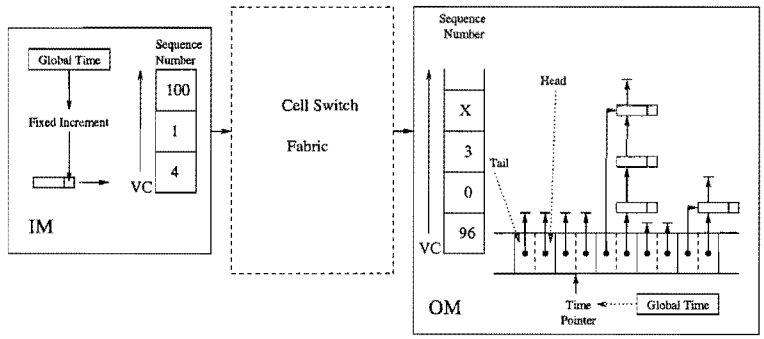


Figure 6.3: Composite Resequencing

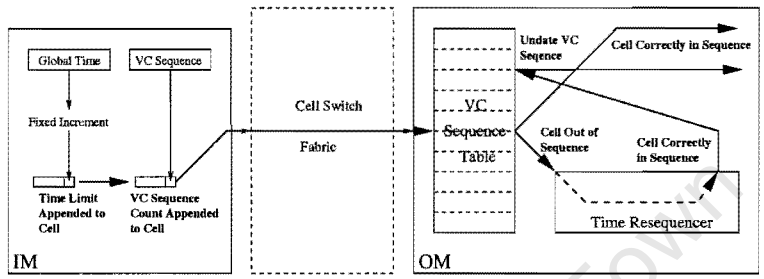


Figure 6.4: Cell Path through composite resequencer

variable delay at the Cell Switch Fabric, it is likely that the actual degree of mis-sequencing per VC will be very low. The composite technique thus uses a sequence count for every VC, if a cell is received in the correct sequence at the output, it is automatically passed on and the sequence number is incremented. If it is out of sequence it is passed to the time line resequencer. When it is removed from the time line resequencer, a sequence count one larger than that of the cell is stored in the output sequence count number. This technique is illustrated in figure 6.3.

This retains many of the advantages of Sequence count resequencing if the VCs do not consume a large proportion of the bandwidth and if the cell switch fabric does not interrupt the sequence of the cells too severely. It will thus largely release the cells as soon as the sequence is attained, resulting in a reduced buffer size at the output. However, if a cell is lost, the time line resequencing will correctly reorder it and once it is removed, correctly set the sequence at the output. Another possibility is that cells that have low delay but reasonably lax jitter requirements can use the composite resequencing methods, while those with very low jitter tolerances may receive only time line resequencing. The impacts of this are not considered in the simulations, however. The two possible paths of an ATM cell through the resequencer are shown for added clarity in figure 6.4.

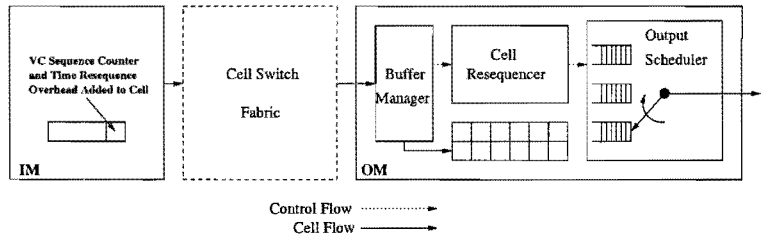


Figure 6.5: Simulation Output Module

6.4 Simulation Output Module

The simulation input and output model are shown in figure 6.5. It basically consists of the previously described modules being concatenated together to form the cell path over the entire switch. When a cell is received at the input module, it has the output time inserted into it as well as a per VC sequence number. When it is received at the output, the buffer manager determines whether there is sufficient buffer space for the cell when it arrives at the output, the cell is then submitted to the resequencer. Finally it is passed to the output scheduler.

An important idea is that although the figures of section 6.3 show the cells moving through the resequencer, in fact once the cell has been buffered, the control information for the cell is manipulated by the resequencer and the output scheduler. The cell itself simply remains buffered in the output buffer. This reduces the amount of buffer space required from that required if the cell were to be moved from one module to the other. In addition, it is in fact simpler to merely pass the control information from one component to the other.

Chapter 7

Traffic Models

Of critical importance to the characterisation of ATM switches is the traffic that the switch is likely to encounter. In particular many of the assumptions made for voice traffic networks by telecommunications service providers are no longer valid. Some of these assumptions include:

- Exponential Call Holding Times
- Exponentially distributed Interarrival Times between calls (Poisson distribution)
- Statistically independent traffic sources
- Exponentially distributed packet length
- Lack of Self Similarity in the traffic
- Even distribution of calls to switch outputs

However, the characterisation and design of modern networks and switches (particular a multi-service network such as ATM) is dependent on the correct characterisation of traffic. Thus, more and more correct analytical techniques have been devised in order to allow network traffic to be more correctly understood and predicted. These improvements include the substitution of on / off bursty traffic sources for the modeling of data traffic (using a two state Markov chain) rather than the normal Poisson process and the use of log-normal call holding times for traffic modeling. However, these more complex models naturally result in far greater complexity in the analytical result, (which was why the simpler models were used in the first place).

The approach used in this thesis is to base the exploratory stage of the work on the fundamental mathematical limits of ATM switching. (See chapter 4) (i.e. favoring output buffering solutions and understanding the delay characteristics of M/D/1 queues). However,

simulation rather than analysis has been used in order to evaluate a switches performance. The reasons for this lie both in the fact that the author is far more proficient in simulation techniques than mathematical analysis and the fact that much more complex traffic models can be developed and tested via simulation than is feasible with analysis. The primary disadvantage is that ATM specifies very small Cell Loss Probabilities (10^{-8} to 10^{-10}) for certain traffic classes, thus simulation times can become very lengthy in order to confirm that these requirements are in fact met.

The three different types of traffic model that were used were a simplistic model whose primary envisaged purpose is correlation with analytical models for other ATM switches published in the literature and for the first characterisation of a switching architecture. The next traffic model extends this simplistic model to consider traffic where there is a locality of reference between input and output ports (although the traffic distribution remains the same). The last model is a multiplexed VBR and CBR model which allows a large number of traffic types to be considered in the ATM switch. ABR traffic is not considered in these simulations, but the interaction of flow control methods with VBR and CBR models would make an interesting study.

7.1 Poisson Traffic Model

This model followed the normal analytical model for an ATM switch. Calls to and from the ATM switch are not considered. The interarrival time between cells is exponentially distributed for a mean inter-arrival time normalised to one. The pdf follows a distribution of $\alpha e^{-\alpha t}$ for a mean interarrival time in cell times of $\frac{1}{\alpha}$or a traffic density (or link utilisation) of $\rho = \alpha$. The times given from this interarrival time are not integral while the valid cell arrival times are, thus these continuous times must be truncated to integer cell inter-arrival times. This is done by the following algorithm:

1. cell_time := exponential_distribution(alpha)
2. current_time := 0
3. Repeat
   (a) if (current_time >= cell_time) then
      i. begin
      ii. SEND OFF CELL;
      iii. cell_time := cell_time + exponential_distribution(alpha);
      iv. end

(b) current_time := current_time + 1;

4. Until SIMULATION COMPLETE

When a cell is generated it is equally likely to be destined for any output port of the switch.

7.2 Locality of Reference Poisson Model

In this model, the traffic is generated as in the normal Poisson model. However, rather that evenly distributing every cell to any output port every input port is assumed to have a tendency to communicate with a set of output ports. There is no port with which it will not communicate, however, the vast majority of its traffic will be distributed to a finite set of other ports. Another possible misconception of this model is that it represents a switch in which one port is overloaded. This is not the case since it is assumed that if overloading occurs in the network on a particular link it can be aleviated by adding infrastructure (or avoided by careful planning).

The locality of reference model presented is rather based on the human tendency to communicate more frequently with one set of recipients than with another (a recipient here can also refer to a computer, such as a web site). It is assumed that the communication frequency over a number of ports can be modeled as a normal distribution. The locality of reference is then given by the value of the variance chosen for the normal curve. An illustration of the technique used to generate locality of reference patterns is shown in figure 7.1.

First a set of Normal curves are used to represent the locality of reference from every input, and a certain amount of the traffic is reserved to be evenly distributed from every port. The advantage of this approach is that it can be guaranteed that the total traffic contribution from every port sums to unity. However, a locality of reference pattern is unlikely to also represent a locality of reference in the physical distribution of the ports, therefore the port values are shuffled. This results in a pattern of traffic which is normally distributed in the sense that it can be characterised by a variance (and the proportion of the traffic that is evenly distributed) but where there is no necessary correspondence with this locality of reference in space on the output ports. It also retains the useful property that no output port will be overloaded from the combined traffic of all of its input ports and in addition, the total memory usage for the entire simulation is $O(N)$. The reference pattern of a single port is shown in the final stage of the figure.

One area in which this model is deficient is that there is a correlation between the traffic reference patterns at the input ports (specifically, if a port references an output port heavily, then a neighbouring port will also). This is an inevitable byproduct of using only one shuffling

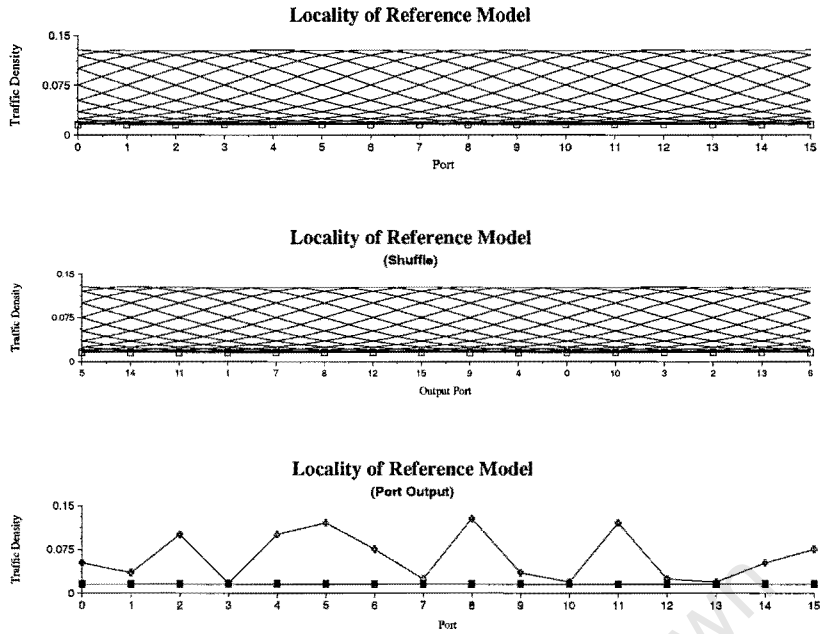


Figure 7.1: Locality of reference stages (16 port switch)

for the output port reference. However the advantages of the model probably outweigh this one deficiency.

7.3 Multiplexed Traffic Model

The multiplexed traffic model replaces the Poisson process of the models in sections 7.1 and 7.2 with a multiplexed CBR and VBR traffic source. Since a large number of CBR traffic sources multiplexed into a queue with a constant service rate will produce an exponentially distributed interarrival time, the Poisson model is still used to represent the multiplexed CBR sources (this saves computation).

However, the multiplexing of multiple VBR sources is considerably less well understood. Therefore the VBR portion of the traffic is represented by multiple VBR sources that are multiplexed together. This multiplexing could occur at an upstream Access Switch or it could also occur within the operating system of a complex traffic source (for example a video on demand server). The cell loss that can occur within this upstream multiplexer is also considered as part of the simulation.

This traffic model boasts the following characteristics:

- The CBR traffic model is characterised by:
  - Mean cell rate of the sources

- The VBR traffic model is characterised by:
  - Mean cell rate (Sustainable cell rate)
  - Variance of mean cell rate
  - Peak cell rate (this limits the rate that cells from the source can be admitted to a network).
  - Variable Self Similarity Parameter (See section 7.3.1) - the Hurst parameter
  - The rate of variation of the data source is a parameter.
  - The VBR source can be:
    - Continuous cell release - this models an ATM cell source
    - Data driven frame source - this models an ATM source which is driven by filling a higher level protocol frame whose length and variance are parameters of the model.
    - Time driven frame source - this models an ATM source which is driven by the periodic release of a variable amount of data into the network, the time interval and variance are parameters of the model.
- The multiplexer is characterised by
  - The number of queues in the multiplexer
  - A finite length to each multiplexing queue
  - A prioritised queue scheduler

It will be noted that this traffic model is characterised by a very large number of parameters. In general, there are two ways in which traffic models can be generated for ATM, the specific and the generic traffic model.

In a specific model, a particular traffic source (such as the MPEG encoding of a particular segment of film) is analysed and a traffic model is found which characterises the traffic in some way (for example the placement of I, P and B frames and their relative sizes in an MPEG stream). This type of model can accurately generate the traffic peculiar to one particular type of traffic source, however, it provides no information about any other traffic source. In addition, the accuracy of these models can be misleading, it can be insufficient to analyse the statistical properties of a traffic type if its self similarity is ignored since this can dramatically influence the success of multiplexing the traffic sources together.

The generic model is considered by the author to be the more powerful. In this a model with a number of parameters is produced and by the manipulation of these parameters approximations to a large number of other traffic sources can be derived. Thus, the behaviour

of the switch under a very much larger number of operating regimes can be studied. In addition, this approach allows one to draw conclusions about the behaviour of the system in response to traffic sources that do not yet exist and are under consideration. Thus the generic models provide more useful information for analysing the appropriateness of a particular scheme for a larger number of possible network conditions.

This traffic model does not round off its characterisation with a consideration of ABR traffic since this requires analysing the complexities of flow control mechanisms in ATM (there are at least three variants of this mechanism). However, it would be a very interesting field for future study.

The discussion of this multiplexed model will begin with the an overview of self similarity (section 7.3.1) and fractional Brownian motion (fBm) and the RMD algorithm will be presented (section 7.3.2). The two data source models for frame based traffic sources, time driven (section 7.3.3) and data driven (section 7.3.4) will then presented. Sections 7.3.1 to 7.3.4 will then be drawn together to present the VBR model in section 7.3.5. Lastly the fully multiplexed model will be presented in section 7.3.6.

7.3.1 Self Similarity

Many natural and human phenomena have patterns that can be described as self-similar or "fractal". These include the stock market [64], long term climactic patterns [33], geographical features, plant growth patterns and network traffic [53], [47], [1], [40]. A self-similar traffic pattern has the two following characteristics, an increase in a parameter is more likely to be followed by another increase (and likewise with a decrease) and more vividly, the scaling law. In time series this means that if the sampling frequency on the time axis is varied (zoomed in or out) the dependent values can be scaled so that they look the same as another sampling rate.

As a concrete example of a self similar pattern, one can use the famous example of a coast line. If one views the coast line of a continent, a country, a peninsula, a bay and a rocky outcrop, they are all similar in appearance even though they occupy vastly different areas.

The importance of self similarity lies in the fact that self-similar traffic sources can break some of the assumptions usually made when considering multiplexed traffic streams (such as are normally switched by an ATM switch). The primary assumption is that as traffic streams are multiplexed together, their burstyness decreases. The self-similarity of a traffic source can be quantified through the use of the Hurst parameter (H). (See [51] for the mathematical definition of Self Similarity) For values of ($0 < H < \frac{1}{2}$) the traffic demonstrates negative correlation, for values of ($\frac{1}{2} < H < 1$) the traffic demonstrates positive correlation and for the special case of ($H = \frac{1}{2}$) the traffic shows no correlation. Thus if the traffic model supports

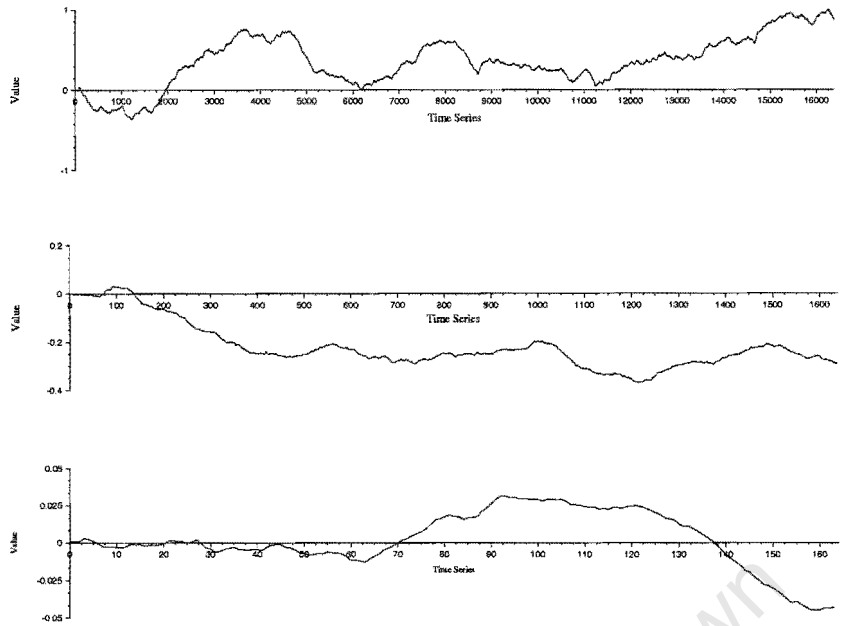


Figure 7.2: Fraction Brownian Motion time series

this parameter, the degree of self similarity can be varied. The Hurst parameters can be estimated when a data set is obtained through R/S analysis ([33] and [52]) or a variance time plot [47]. In figure 7.2, a self-similar traffic series is shown generated using fractional brownian motion. Here the Hurst parameter (H) is 0.8.

There are two primary ways of modeling traffic. A specific data source (such as MPEG or a Novell IPX workstation) can be studied and then a model produced which attempts to exhibit the same statistical properties of the model (including possibly that of self-similarity). The advantage of this approach is that the model can be shown to accurately reflect a specific traffic type. The disadvantage is that it cannot serve to test a network or switch with a traffic type that has not yet been studied and characterised. In addition, multiplexing vastly different traffic sources becomes difficult since it entails a distinct traffic model for every different source. The other approach is to produce a generic traffic model with a number of parameters, these parameters can be altered to allow the model to produce traffic that approximates a known traffic source (but never to the same degree of accuracy as a custom designed model), but more interestingly, the parameters can be varied and the switch's performance under these varying conditions can be observed. This can in fact provide one with far more insight into a switches performance with regard to future and current applications than a custom model. It is a case of "what you gain on the roundabouts you lose on the tills". The second approach is the one that is used to characterise the switches performance due to the fact that a larger variety of operating regimes can be studied.

The model chosen for generating self similar time series is fractional brownian motion.

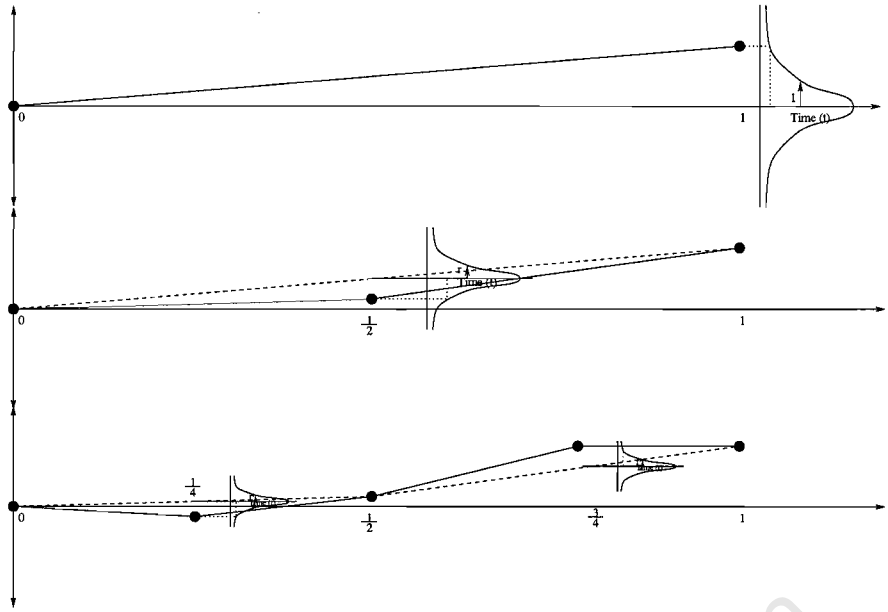


Figure 7.3: Random Midpoint Displacement Algorithm

7.3.2 Fractional Brownian Motion

A phenomenon that has self-similar properties is Fractional Brownian Motion. Following [22], the term fractional brownian motion (fBm) is used to refer to a particles position and fractional brownian noise (fBn) to the deltas of its movement (the two can easily be transformed into each other since $fBn \equiv \frac{d}{dt} fBm$). There are many possible ways of generating these time series, including Fourier transform, Displacement Process, Pareto Distribution, Markov Processes [22] and the technique used for these simulations, Random Midpoint Displacement. This process is illustrated in figure 7.3.

The algorithm conceptually consists of the following steps: First a scaling parameter, r, is calculated according to the following formula: $r = \frac{1}{2^H}$ where H is the Hurst parameter described in section 7.3.1. Then the first endpoint is chosen (conceptually at $t = 1$) on the time axis using a normal distribution curve with a mean of zero and a standard deviation of one. (The time series can of course be scaled to other values). The midpoint of this line is then deviated vertically by a normal distribution with a standard deviation of r. The two resulting line segments are then deviated at their respective halfway points ($\frac{1}{4}$ and $\frac{3}{4}$) by a normal distribution with a standard deviation of r^2. This process is continued recursively to some desired precision.

An alternative way of describing this algorithm is shown in figure 7.4 (from [22]).

This algorithm results in a data set that describes fBm. In order to calculate fBn, the difference between the succeeding data points is calculated.

The algorithm was described as a recursive algorithm in [27] and as an iterative algorithm in [22]. However both of these techniques were considered unsuitable for implementation in

$$
\begin{aligned}
r &= \tfrac{1}{2^H} \\
x[0] &= 0 \\
x[1] &= G_1 \\
x[\tfrac{1}{2}] &= \tfrac{x[0]+x[1]}{2} + rG_2 \\
x[\tfrac{1}{4}] &= \tfrac{x[0]+x[\frac{1}{2}]}{2} + r^2G_3 \\
x[\tfrac{3}{4}] &= \tfrac{x[\frac{1}{2}]+x[1]}{2} + r^2G_4 \\
x[\tfrac{1}{8}] &= \tfrac{x[0]+x[\frac{1}{4}]}{2} + r^3G_5 \\
x[\tfrac{3}{8}] &= \tfrac{x[\frac{1}{4}]+x[\frac{1}{2}]}{2} + r^3G_6
\end{aligned}
$$

Where G_i is a function which returns a normally distributed random number.

Figure 7.4: Algebraic view of the RMD algorithm

an object oriented time series methodology. In this technique, the time series is represented as an object with an operator that returns the next number in the time series. The recursive algorithm presented implementation difficulties in that the stack state represented by the recursion would effectively need to be stored as a thread within an object, synchronisation techniques would then be required to obtain the next member of the time series. The iterative solution simply produces in one pass a completely filled vector which represented the time series. Since there would be many thousands of time series in the simulation and each time series might contain many thousands of data points this one pass technique would not be feasible without recourse to secondary storage, which is slow (the paper in which it is described tellingly states that this is the slowest operation of the iterative algorithm). The solution was to use the recursive algorithm on a virtual machine stack encapsulated within an object. Here the stack was represented by an array of structures each of which stored the algorithm state at the recursion depth of the algorithm as well as the passed parameters. This technique allows the data points to be extracted one at a time. The time complexity of the algorithm is $O(1)$ per time series value extracted and $O(\log_2 N)$ in space complexity. Note that the accuracy of the RMD algorithm can be lower than other methods (see [12]), but for the purposes of these simulations computational efficiency was considered the primary concern and thus, RMD was used.

7.3.3 Time Driven Traffic Sources

Many traffic sources in an ATM network will be data. In other words, the data will be released into the network in bursts of cells that are delivered at the peak cell rate followed by a silent interval. The overall cell rate of the traffic will therefore be determined by the ratio of active bursts to inactive bursts. The reason that this type of traffic occurs is because ATM is often used to carry frame based traffic that is generated by other protocols. The process by which ATM traffic is generated in shown in figure 7.5.

An application generates data and uses it to fill a frame buffer (note that some of the classic

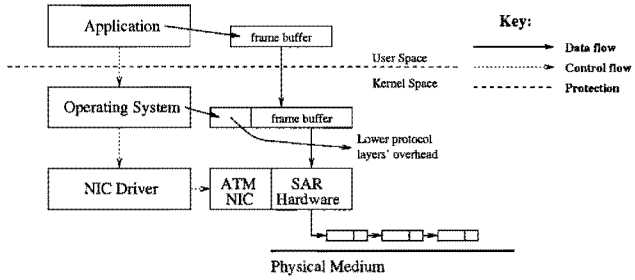


Figure 7.5: Generation of Data from Application to Physical Medium

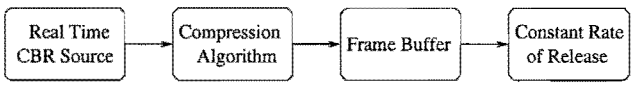


Figure 7.6: Time Driven Traffic Source

OSI protocol layers might be found in the application, especially the presentation and session layers), when the application wishes to send the data across the network it instructs the operating system that it wishes to do so. The operating system then adds lower protocol layer overhead (for example TCP and IP protocols) and then passes the resulting frame to the NIC driver, this then controls the specific ATM NIC which will normally perform the SAR function and send the cells onto the physical medium.

The overall result of this system is that the data tends to be deposited onto the network at the peak cell rate in a burst that is one frame in length. For a time driven traffic source we assume that the data is driven by the process illustrated in figure 7.6. In this model some real world data source produces data at a constant rate (for example from a digital camera or a microphone), then the data is fed into a compression algorithm. The rate at which the data is released from the compression algorithm will vary depending on how successful the algorithm is. For example, in MPEG the compression ratio can depend on the complexity of the images in every frame as well as the rate of movement of all of the figures within the frame (a sequence showing an explosion will have a far lower compression ratio than that of a presenter talking in a studio, this scene dependency is one reason that compressed video sources exhibit self-similarity), this data is written into a frame buffer. At a regular interval the data is flushed from the buffer and put into the network. The traffic generated by the source will have the appearance illustrated in figure 7.7 (here each peak of traffic represents an ATM cell). In this case, the interval between traffic bursts remain constant, however, the burst lengths themselves vary. This model can become equivalent to a cell source if the frame interval is simply set to one ATM cell interval and the peak cell rate is set to the link rate, this is illustrated in the last plot of figure 7.7 (to make the cells more clear, the time scale has been altered).

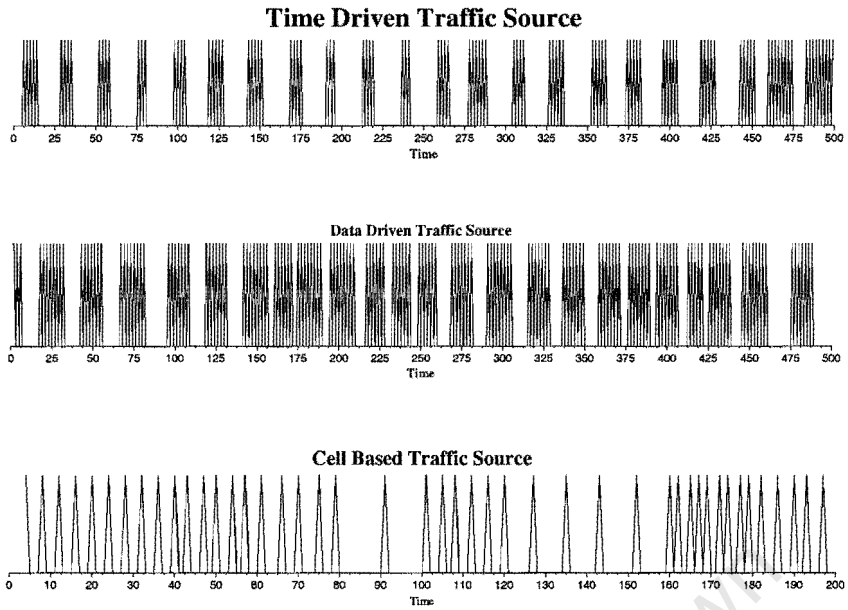


Figure 7.7: Time, Data and Cell Driven ATM traffic

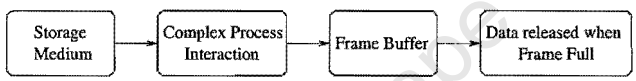


Figure 7.8: Data driven traffic source

7.3.4 Data Driven Traffic Sources

Data driven traffic sources are also assumed to be driven by the process illustrated in figure 7.5. However, the variation of their data rates is assumed not arise from varying compression ratios in a codec but from complex interactions between the processes of the operating system and its peripherals (for example, the number of competing processes and the placement of data on the disk itself). This is illustrated in figure 7.8. Unlike the time driven source in which the frame is emptied at a regular interval, in this model, the frame is only emptied when it is full. Thus, as illustrated in figure 7.7, the burst length is always constant. In this model, the determining factor of the data rate is the interval between bursts. This type of data is more likely to correspond to an FTP or HTTP session than the former data model.

In practice, real traffic is likely to feature both types, although it will have a preponderance of one characteristic. The reason for this is time outs in the data driven model will result in partial frames being placed on the network. In addition, if the frame length is insufficient in the time driven model, then constant length frames will periodically be flushed. Another consideration for VBR traffic is that the length of the frame might be restricted by the imposition of a maximum burst length in the CAC phase. These factors are not considered in the simulation or the models.

7.3.5 VBR Data Model

The VBR data model integrates the RMD algorithm with both the Time and Data driven models, a non-useful parameter that must be set for this model is the recursion depth. This is essentially a trade off since the greater the recursion depth the greater the accuracy of the source, but the larger the space requirements. It derives the traffic rate from a source which implements the RMD algorithm in order to allow this data rate to have a variable self similarity. This data source produces fBn with a setable mean and variance to the data rate. The interval at which this rate varied can be controlled. In addition, the model allows the frame size and the variance of the frame size to be specified. This frame size can be specified in term of the number of ATM cell arrival intervals that must elapse before the Frame buffer is emptied in the case of a time driven model, or it can specify the size of the data buffer in ATM cells that must be filled before it is emptied in the case of a data driven model. The other parameter that can be varied in the VBR model is the peak cell rate. This is the maximum rate that cells will be released into the network even when the model bursts. In addition, if the RMD algorithm generates a data point that exceeds the peak cell rate, the rate is clipped down to this rate. (This is assumed to be the action of traffic policing at the entrance to the network). As an illustration of the output that the model is capable of, the data rates for a time driven model are illustrated in figure 7.9. Note that a Hurst parameter of 0.5 corresponds to no self similarity and a parameter of 1.0 corresponds to complete self similarity. It is not immediately obvious from the plot, but all of the data sources have the same mean and variance of their data rate. The higher the Hurst parameter the longer the traffic must be observed for its distribution to be accurately determined (there are in addition mechanisms within the model to ensure that the correct mean and variance are maintained over a long time interval). This figure shows strikingly the difficulty that can be encountered when a highly self similar traffic source is multiplexed with other self-similar sources.

7.3.6 Cell Multiplexer

The last element of the model to be considered is the cell multiplexer. This element of the model represents the multiplexing action of an upstream ATM access switch or the internal behaviour of the operating system and ATM NIC in a host with multiple data sources. The multiplexer consists simply of a number of fixed length queues from which a scheduler retrieves cells. Any of the data sources are allowed to write to the multiplexer at any time in a cell interval, the multiplexer will always service the cells in the queue at the link rate. The scheduler selects the cells in a simple prioritised algorithm (i.e. cells are always selected from the highest priority queue if they are present, otherwise the priority falls to the next highest queue). The cell multiplexer used in the model has two priorities, one for CBR sources

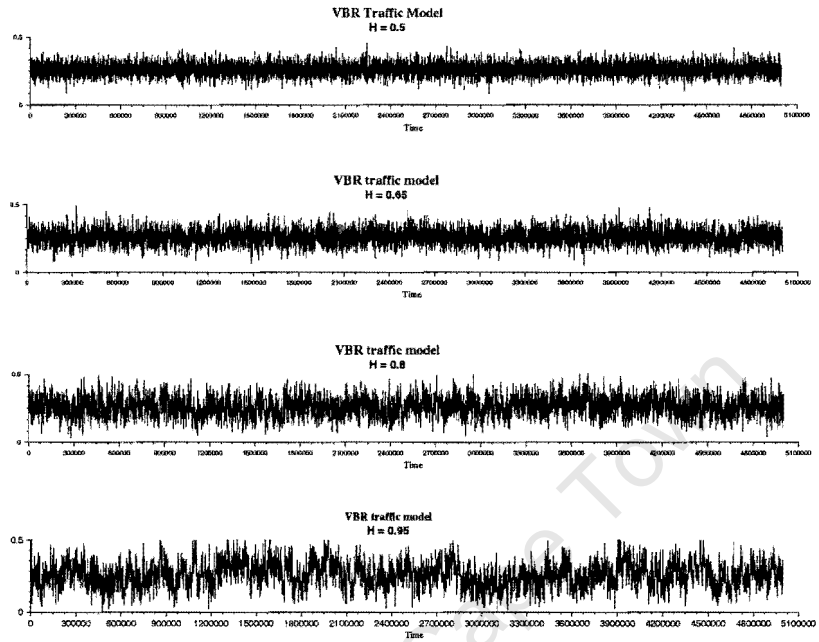


Figure 7.9: Effect of varying self-similarity parameter on Traffic Model

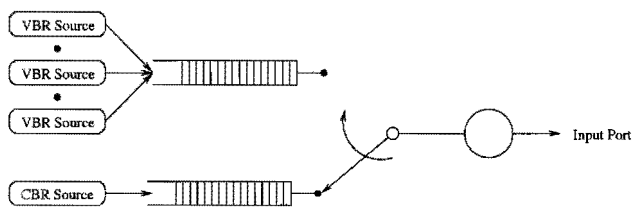


Figure 7.10: Cell Multiplexer

and the other for VBR sources. There is however only one traffic generator for all of the CBR sources, while there are many VBR traffic sources. A large number of multiplexed CBR sources can be shown to exhibit a exponential interarrival time when leaving the multiplexer, thus the CBR sources are represented as a Poisson process, whenever a cell is generated, the output port is selected randomly.

The VBR sources are, however, more difficult to characterise when multiplexed. Thus each VBR source is independently simulated, each VBR source services a set of output ports (the precise number of VBR sources that can be feasibly simulated is dependent on the amount of physical memory in the system, thus this set could be one port). The multiplexer then selects and outputs the traffic. The multiplexer losses are also considered in all of the simulations since it represents the effect of various traffic types at either the access switch or within an ATM host. The parameter that can be set for the purpose of the simulation is the buffer size per queue in the multiplexer. This process is shown in figure 7.10.

Chapter 8

Non-FIFO queueing using an Associative Memory

In order to practically implement a Large Scale ATM switching fabric it often becomes necessary to buffer ATM cells internally in the Switching Elements (SE's) to accommodate cells that must reach the same output port of a SE in the same time slot. Since the cell arrival rate in ATM is very high (one every $2.7\mu s$ on a 155Mbps link), the control function for the queuing of the ATM cells must be kept very simple in order to be practically implementable. Thus, FIFO queueing is normally used in which each output port of the switching element has a logical FIFO queue associated with it (in order to best benefit from the statistical multiplexing of cells, the cells are normally stored in a fast Random Access Memory).

Although a FIFO queue has much to commend it (it is simple and is obviously fair), it also has disadvantages. For the case of MPSR switches, the output port queue to which the cell is sent must be chosen before the cell is appended to its logical queue. However, consider the impact of a backpressure congestion signal on a switching SE. If a particular downstream node signals congestion, how are the cells in the upstream SE's FIFO queue to be handled? Although many of them might be profitably rerouted, this will in general be prohibitively expensive since the Control Unit would have to sequentially reconsider all of the cells in the logical output queue. This has two undesirable consequences; firstly, the cells destined for the downstream node will be unnecessarily delayed; and secondly, since the buffer cannot be cleared of the ATM cells, the upstream node will have an increased likelihood of in turn becoming congested. The extra delay has added implications particularly in an MPSR switch since it complicates the resequencing of the cells at the output. In this chapter an alternative non-FIFO queueing strategy using a specially designed Associative Memory will be discussed. It does not suffer from the previous problems since it compares the profitability of dequeuing every buffered cell at every cell time slot. Although this sounds improbable given the very high speeds of ATM the feasibility of such an approach will be demonstrated.

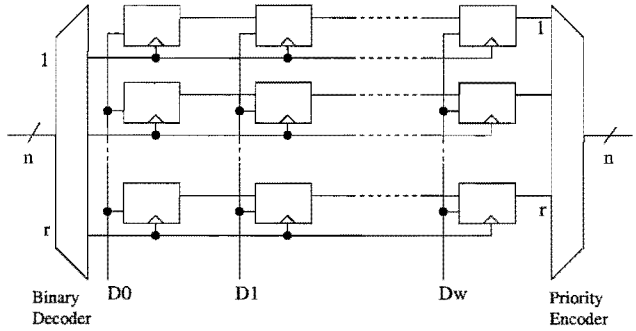


Figure 8.1: Basic Structure of the Associative Memory

The application of the memory will be in the Clos network described in section 5.4 (page 57) and the hypercube switching fabric discussed in section 5.5 (page 61). However, with simple modifications, the same basic concept could be used for other internally buffered MPSR fabrics.

8.1 Overview of Associative Memory

The associative memory replaces most of the functions of the Control Memory in a shared buffer SE, and also performs some of the Control Unit's functions. Its overall structure is as shown in figure 8.1 (Evidently $r = 2^n$). The basic operation is simple; when a write operation is performed, the Data to be written is asserted on D0-Dw. The row to be written is then given to the binary decoder and the data is clocked in . (The row would correspond to where in the Shared Memory Buffer the cell had been stored). When a read is performed, however, a pattern is asserted on D0-Dw. Each element then uses both its state and the value asserted on the Data line to determine whether it is eligible for being routed out. If it is, it passes a true signal on to the next element in its row. This element in turn performs a logic function and similarly passes its (one bit) data along. At the rightmost edge of the memory a Priority Encoder returns a binary number indicating which (if any) of the rows were eligible to be routed out.

Evidently what determines the queueing algorithm of the switch is the logic of the memory elements found in the associative memory. For the purposes of the hypercube the memory is partitioned as indicated in figure 8.2. Before the algorithm for cell selection is discussed, the following terms should be explained:

- **congested node** - a node with more than half of its buffer space occupied.
- **blocked node** - a node with all of its buffer space occupied (blocked implies congested).
- **distribution phase** - every ATM cell arriving for the first time at the fabric is distributed to any of the neighbouring nodes. This results in at worst a $d + 1$ hop count

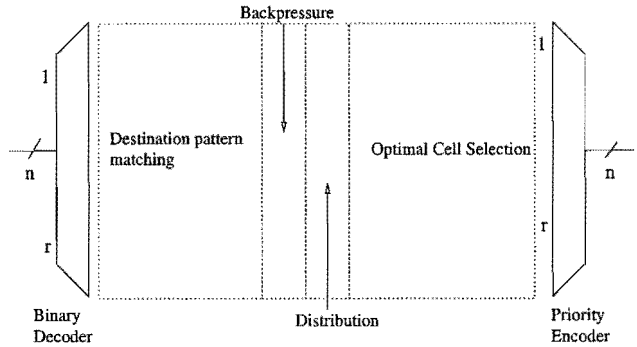


Figure 8.2: Functional partition of memory for hypercube switch

and gains a statistical multiplexing advantage in the fabric.

- **Final Hop** - a cell that is only one bit position away from reaching its final destination.
- **neighbouring node** - the node that is being considered for routing on this cycle.
- **priority** - an additional constraint used to determine which among many cells is most eligible for routing out.
- **dimension** - the hypercube dimension currently being considered for routing.

The algorithm for cell selection is:

1. Before writing the row: Determine the destination mask by XORing the cell's destination port with the node number.
2. If the neighbouring node is not congested:
   (a) The cell with the lowest priority number whose destination mask OR (1 << (leftshifted) dimension) == TRUE or which is being distributed is selected
3. If the neighbouring node is congested
   (a) The cell with the lowest priority number whose destination mask OR (1 << dimension) == TRUE is selected.
4. If the neighbouring node is blocked
   (a) The cells with the lowest priority number whose destination mask OR (1 << dimension) == TRUE and which is on its final hop is selected.

Number 4(a) of this algorithm can be justified if one considers that a cell arriving at its destination node need not be buffered and thus it is irrelevant whether the node is blocked.

The gate level diagram of the memory elements to implement this algorithm are presented next.

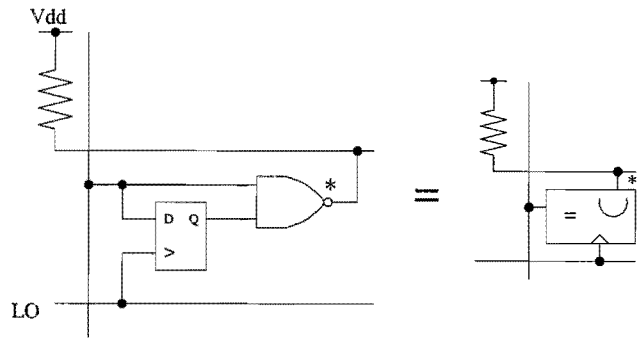


Figure 8.3: Route determining elements

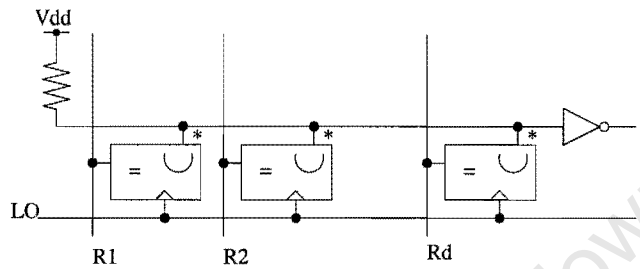


Figure 8.4: Complete route determining section

8.1.1 Route Determining Element

The route determining element is shown in figure 8.3. This element checks whether its mask bit matches the value given to it. If it does, it drives the wired-NOR chain down, the output will then be asserted from the destination pattern matching segment as shown in figure 8.4.

8.1.2 Back-pressure Element

If a neighbouring node is completely blocked, no cells should be routed to it until its congestion clears, the exception to this is that if that blocked node is an ATM cells final destination, then it will not need to be buffered and will be presented straight to the output port. Thus the element in figure 8.5. Here the element is loaded with a FALSE if it is on its final hop (i.e. when only one bit of its destination mask is set). The column line is asserted with a TRUE if the neighbouring node is blocked.

8.1.3 Distribution Element

In order to improve the statistical multiplexing of the ATM cells, all cells are distributed randomly to all the node's neighbours when they first enter the hypercube. However, if a neighbouring node is becoming congested, this additional overhead should not be imposed on it, thus the element in figure 8.6. A FALSE is loaded into the element when it should

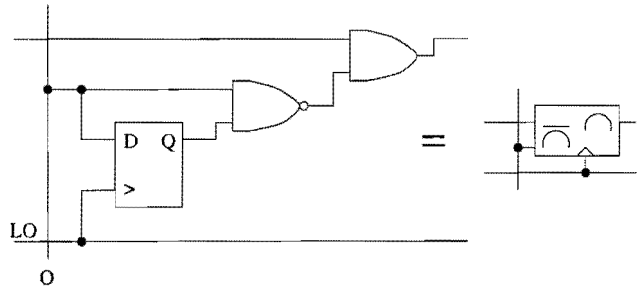


Figure 8.5: Back-pressure Element

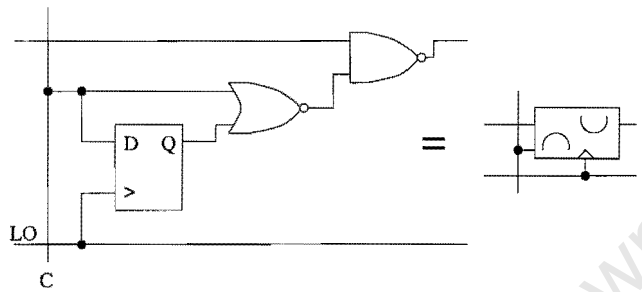


Figure 8.6: Distribution Element

be distributed and a FALSE is asserted on the line when the cell is on its final distribution phase.

8.1.4 Cell Priority

The preceding sections allow the associative memory to determine whether a cell is eligible to be routed to the next element, once this is is determined, the optimal cell to be routed must be determined. The next section of the memory performs this function. It selects the smallest number in the priority field amongst the rows of the memory that are still eligible after the first 3 stages. A suggested form for this number is shown in figure 8.7. (Note that any of the fields with the exception of MSB and NSB could in practice be absent). The cell priority would be set depending on the QoS contract associated with the virtual circuit when the cell arrives at the input port. The time field is calculated in the following manner:

- Every node in the hypercube has a clock synchronised to all the other counters. The clock is incremented at every cell arrival time.

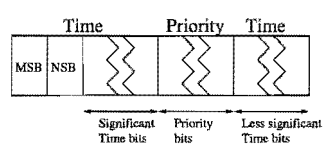


Figure 8.7: Cell Priority Field

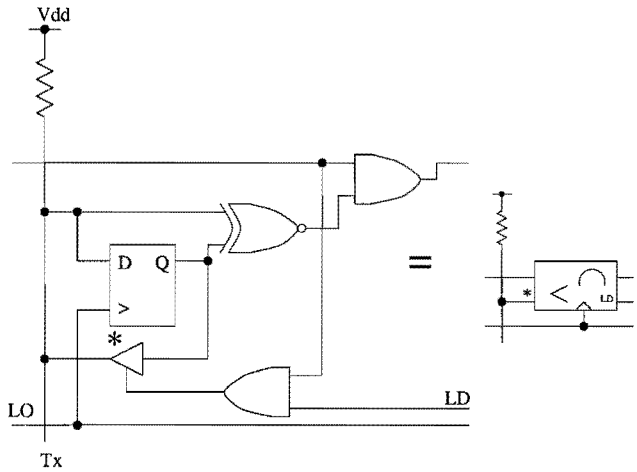


Figure 8.8: Cell Priority Comparison Element

- When a cell arrives at its input node, a fixed increment is added to the current clock counter. This value represents the time by which the ATM cell must reach its output node or be destroyed.

Thus, the time value represents the importance of routing an ATM cell to its output port before it is destroyed. A lower value of the time field indicates that there is less time for the cell to be routed and hence it should receive preference. The priority field can be used to allow certain very low latency services to receive preferential treatment regardless, so long as other lower priority cells are not becoming very critical in their timing. The memory element used to determine cell priority is shown in figure 8.8.

The memory element functions in the following way. An incoming signal arrives from the previous stage in the memory (i.e.any of the elements in sections 8.1.1 - 8.1.3) or from another cell priority comparison element. If it is TRUE and LD is TRUE then the value stored in the memory element is asserted on the open collector bus. The cell then compares its data value with the value of the bus. If its value is the same (i.e. the Exclusive NOR's output is TRUE), then it propagates a true signal out.

The effect of this element is thus that if it stores a FALSE, it will automatically win, since it will succeed in pulling down the line. If, however, it stores a TRUE and no other elements in its column either assert their own value or also have a FALSE to assert, it will also win. Thus, lower valued bits will be allowed to propagate a 'WINNING' signal onto the next element in the row. The end result of this chain is that the lowest valued row will win.

Since the number of bits that can be used to represent the timestamp is limited in practice (and since using fewer bits improves the speed of the memory), a strategy must be adopted that handles the time wraparound. In particular the strategy should ensure that no timestamp that is in the "past" should ever appear to be in the future. A simple way to handle this is to ensure that no time **difference** in the switching fabric ever exceeds a quarter of the

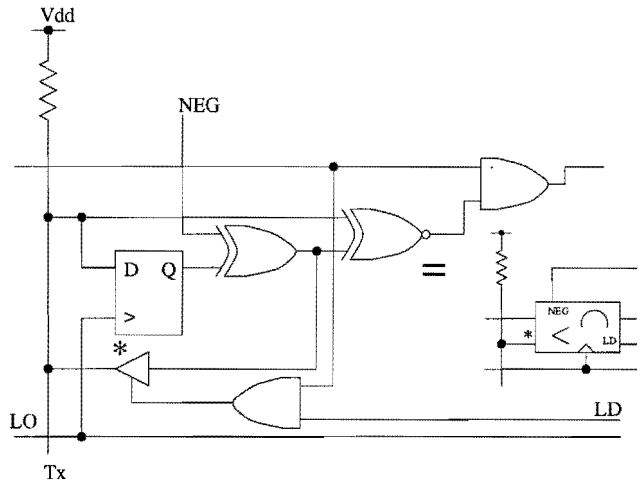


Figure 8.9: Toggled Cell Priority Comparison Element

Current Time	Toggled	Possible Values	Corrected Order
00	Y	11 00 01	01 10 11
01	N	00 01 10	00 01 10
10	N	01 10 11	01 10 11
11	Y	10 11 00	00 01 10

Table 8.1: Effect of Toggling MSB of timestamp

range expressible by the time field. When the MSB and NSB of figure 8.7 are equal to "00" or "11", the the sense of the MSB is toggled. (This requires a modification to the memory element that stores the MSB of every timestamp in the row, this modification is shown in figure 8.9.) The effect of flipping this bit is shown in table 8.1, which shows that by using this technique the relative time order is always preserved.

Finally, two rows of the associative memory are shown in figure 8.10 to indicate how the elements are connected together.

8.2 Use of the Associative Memory in the Clos Network

The associative memory has been discussed in connection with being used for routing in a Hypercube Switching fabric. As a comparison to the performance of the Hypercube Fabric a Clos network and an Ideal switching fabric have been chosen. The Ideal fabric is lossless and simply delays the cells for some small finite time, it also preserves cell order. However, this provides little information as a comparison for other switching fabrics. The Clos network was chosen largely because of its very prevalent use in both Academic and commercial large scale switching systems.

Since at every stage past the center in a Clos network the ATM cells have only one viable

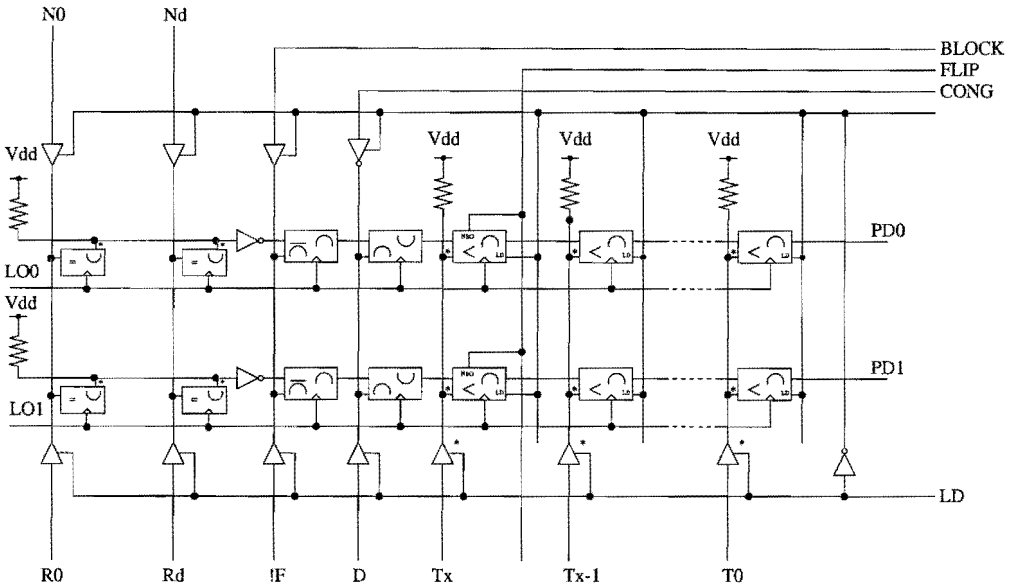


Figure 8.10: Two rows of the Associative memory

output , the route determining segment is not necessary and can be replaced by a wired AND configuration of the index of the output port of the SE where the ATM cell is being buffered. In addition, since there is no congested state in the simulation, this can be ignored. (A packpressure "blocked" signal is used in order to provide a fair comparison with the hypercube switching element). Thus, the backpressure signal and the priority determining sections of the associative memory are used unaltered in the simulation.

8.3 Implementation of the Associative Memory Elements

Before reading this section, Appendix C (page 148) can be read. This gives a very brief overview of integrated circuits and the properties of CMOS logic.

All of the elements described in section 8.1.1 (page 93) to section 8.1.4 have been tested by laying out the masks in an IC layout package [54]. The masks are then converted to a SPICE3 model and an analogue simulation of the memory is conducted to determine its performance. The masks are laid out according to the MOSIS SCMOS (Scalable CMOS) design rules [57], using *lambda* scaling (introduced by Mead and Conway in [55]). Using these design rules allows a design to scale naturally with improved lithographic processes. For the purposes of the simulation the very conservative assumption of a $2\mu m$ process was used. In practice this means that a more modern process should achieve a faster settling time. In addition, it was assumed that a 10 bit routing section was sufficient (for a 1024x1024 switching fabric) and an 8 bit cell priority section. All aspects of the memory have been designed except for the decoder and encoder (which are standard designs), and most of its functionality has been tested. All of the elements described have been tested in terms of their logical function

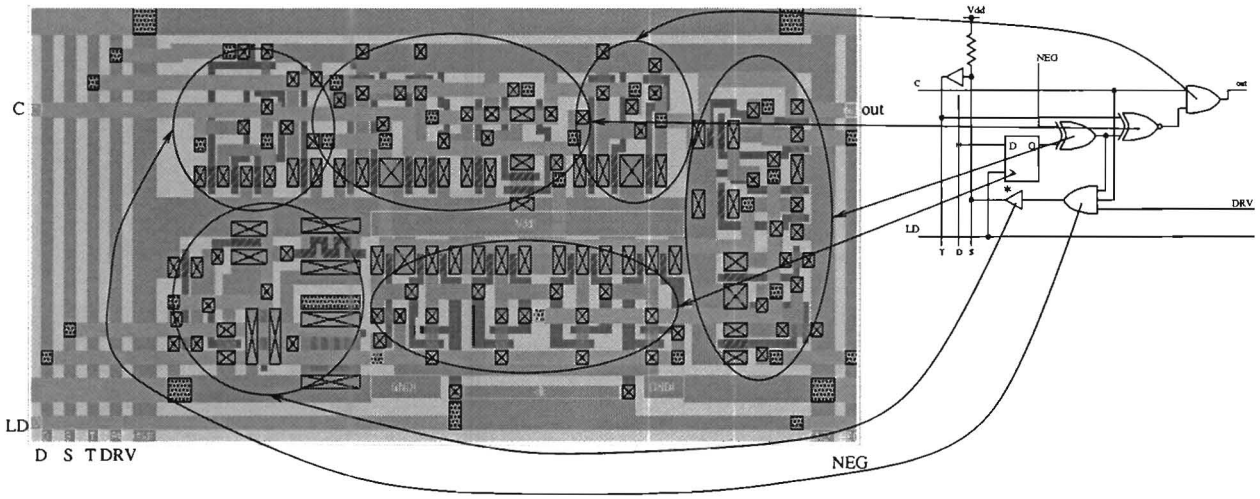


Figure 8.11: Optimised Memory Element

and have been found to perform correctly. However, problems were discovered in terms of the performance of the elements when connected together that resulted in modifications having to be made to the elements in order to attain a reasonable performance. Due to time constraints only one of the elements, the Priority Element, was optimised. The Priority Element was chosen since it was the most complex of the elements and had the greatest impact on the timing of the circuit as a whole. In other elements, the optimisations were largely performed, but some of the driver circuitry has not been optimised. This section also provides insight into some of the difficulties that can be encountered when the analogue aspects of a design must be considered outside of the domain of their digital abstraction.

In order to illustrate the optimisations that are necessary, the schematics of this element will be examined and compared to a transistor equivalent circuit in order to illustrate the necessary modifications. The schematics for all of the elements may, in addition, be found in the CD-ROM which accompanies this document.

First, the toggled time comparison element will be considered. The modified schematic and circuit layout for the memory element are shown in figure 8.11.

The most obvious addition to this design is the replacement of the one date line (in figure 8.9) with three data lines instead. There are three reasons for this. The first is to avoid fan out problems with the driver on the data line (Tx of the previous figure). If the data line is allowed to drive the inputs to all of the devices in the memory element, then the driver must be capable of driving four gates per memory element. It is immaterial how many of the gate inputs are relevant for a particular transition since the capacitance of the MOSFET transistor gates must be charged and discharged. By, dividing the input in this manner, the D line is presented with two gates per memory element. The T line is presented with one gate per memory element and the S line is presented with only one gate input regardless of the number of memory elements. Thus, the driver size per line can be dramatically reduced

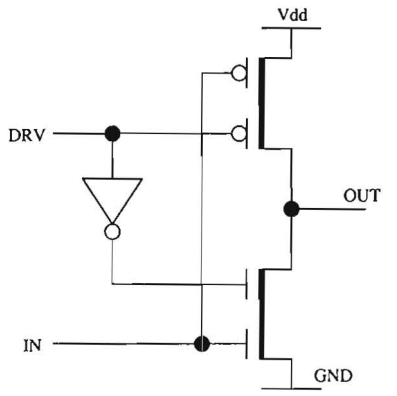


Figure 8.12: CMOS Tri-state driver

in size, or, the transitions can be made faster, resulting in a higher speed of operation.

The second reason for this design is illustrated by considering the required driver circuitry of figure 8.10, here the drivers for all of the lines must be tri-state since depending on the other control signals, different inputs can become valid at different times. Hence it is worthwhile considering the implementation of a CMOS tri-state driver, shown in figure 8.12. A CMOS tri-state driver consists essentially of two two input MOSFET transistors at the output (note that this is an inverting driver). If DRV is high, then neither of the transistors can turn on, leaving the output tristated. If DRV is low, on the other hand, then the IN signal effectively determines which of the transistors should turn on, if it is low, the upper will, thus driving the output high. If it is high, the lower will, driving the output high. Thus, this circuitry implements an inverting tri-state buffer. The essential problem with this implementation is that both of the driver transistors have two gates, this means that their effective impedance is double that of a single gated MOSFET transistor. This, expressed another way, means that each transistor must have four times the area that it would normally require. Firstly since it has two gates, and secondly, in order to present a low enough impedance to drive the load required of it. For this reason, in very high speed circuitry with high fan-outs, it can be advantageous to avoid a tri-state driver entirely. It is obvious that, if D is separated from the logic of the open collector driver (or rather, open drain driven), then it becomes entirely unnecessary to tri-state D. In addition, it turns out that it becomes unnecessary to use a tri-state driver even on the pullup line, the reasons for this will be made apparent when the driver circuitry for the S and T lines is discussed next.

The third reason why Tx has been divided into T, S and D is in order to speed up the pulldown that can be effected by each element. If each element both monitored and pulled down Tx, then every element would be required to drive one logic gate per memory element. In order to speed up the operation and to reduce the size of the pulldown transistor required by every memory element, it becomes necessary to reduce the number of inputs that each pulldown transistor drives. This is done by providing a buffer which amplifies the signal on the S line and drives it onto the T line. This buffer presents only one input to the driving

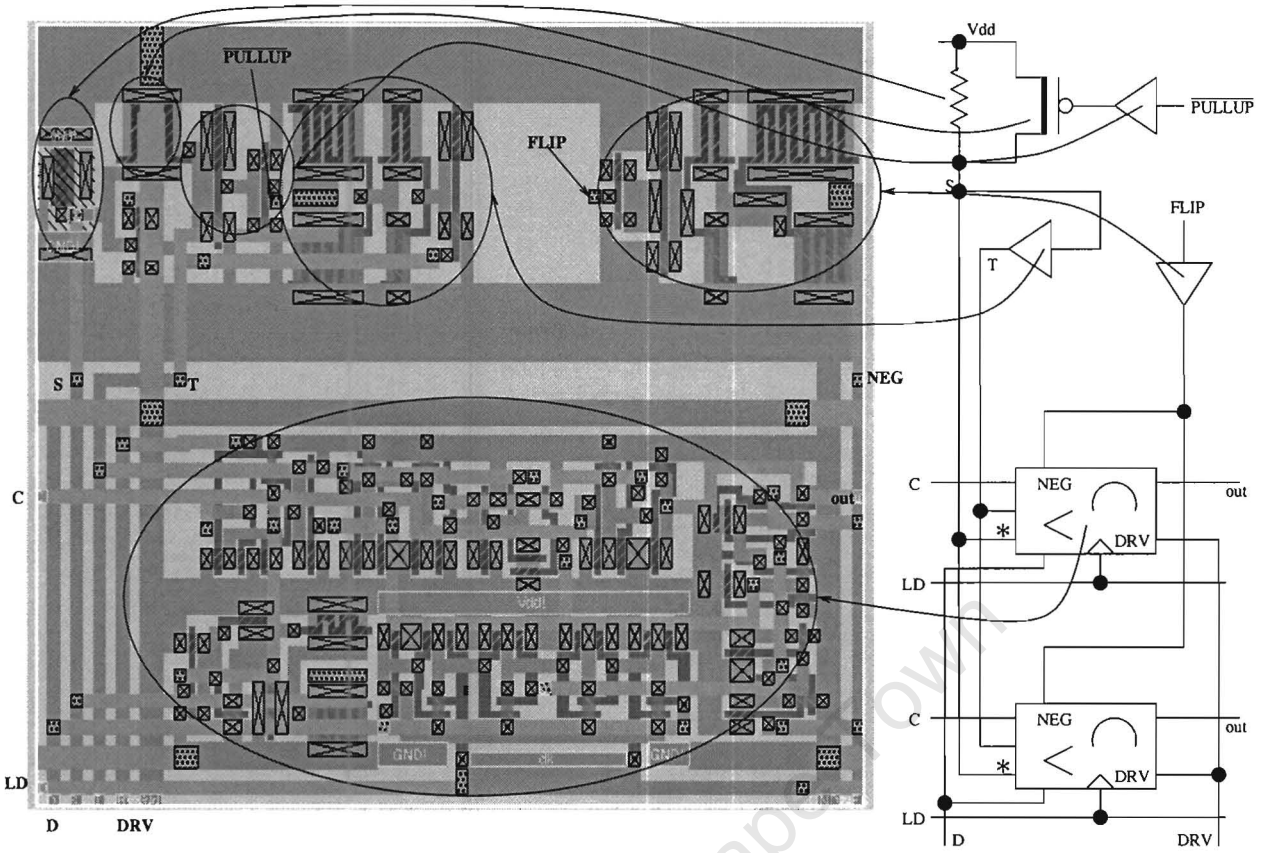


Figure 8.13: Driver Circuitry for the Optimised Memory Element

transistors in each memory element.

Figure 8.13 shows half of the driver circuitry for the element, while figure 8.14 shows the drivers responsible for driving the D and DRV lines of the memory element. Two main points arise from these figures. The one is the layout of the drivers, the other is the operation of the pullup and pulldown lines connected to the S line.

The driver circuitry as shown consists of multiple stages of inverters connected in cascade. The reason for this is that one wishes the capacitance of the input to the driver to be as small as possible, however, the impedance of the output transistors should be as low as possible (and hence the transistors must occupy a larger area of the silicon surface). However, a large transistor has a large gate area, hence the capacitance of the gate is larger. Thus, it becomes necessary to create large drivers by cascading a number of drivers together, each driver in turn becoming larger (and hence capable of driving a larger load). It was shown in [55] that the optimal speed is obtained when every stage of the driver is e larger than the previous one. This is shown schematically in figure 8.15.

The pullup and pulldown are not connected in a standard CMOS configuration. All the drivers to the line are either open source or open drain. The arrangement is shown in figure 8.16. Since a resistor is hard to fabricate on silicon and occupies a large area the pullup resistor is in fact formed by a p-channel transistor especially dimensioned (by having

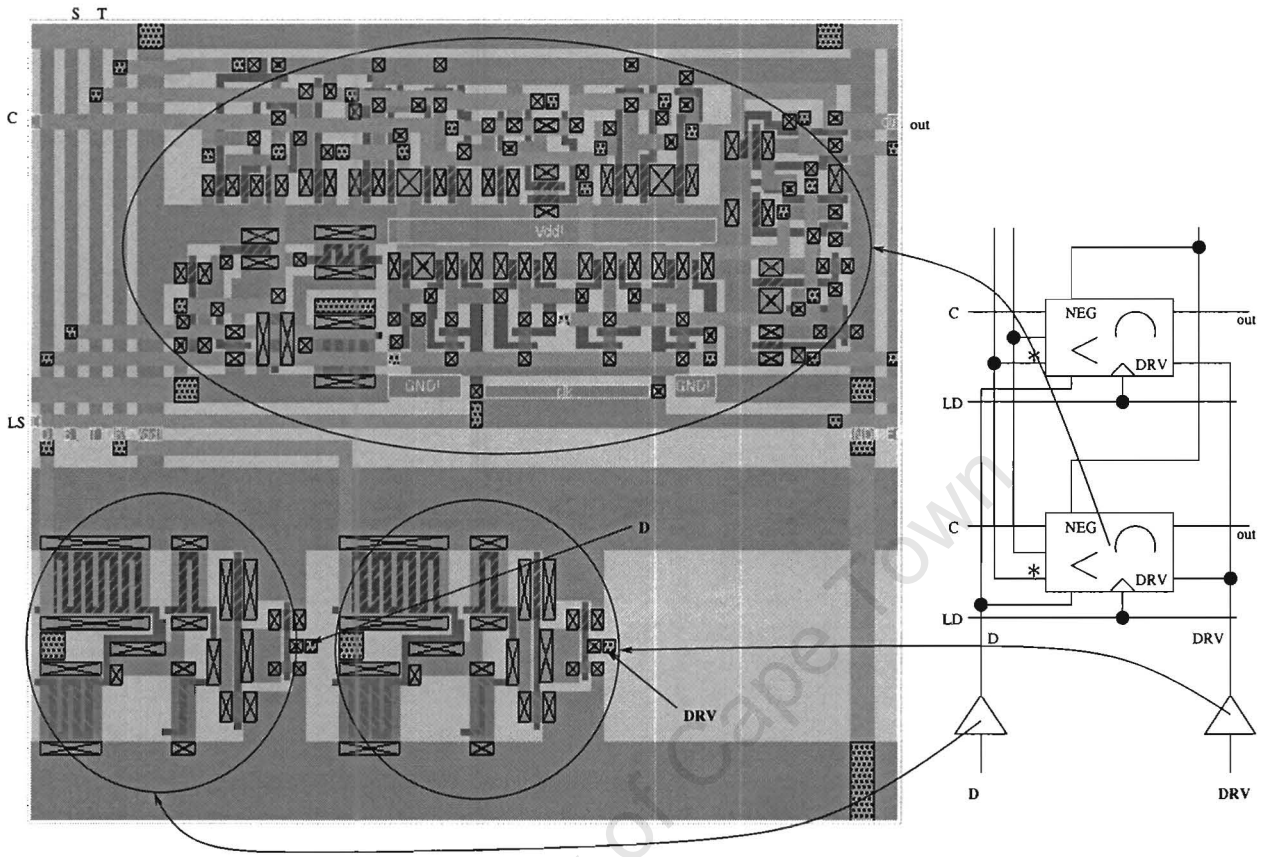


Figure 8.14: Lower Driver Circuitry for the Optimised Memory Element

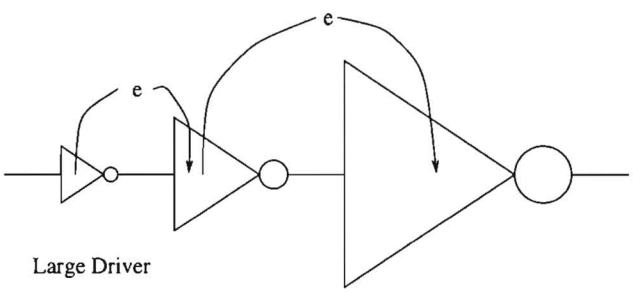


Figure 8.15: Large CMOS Driver

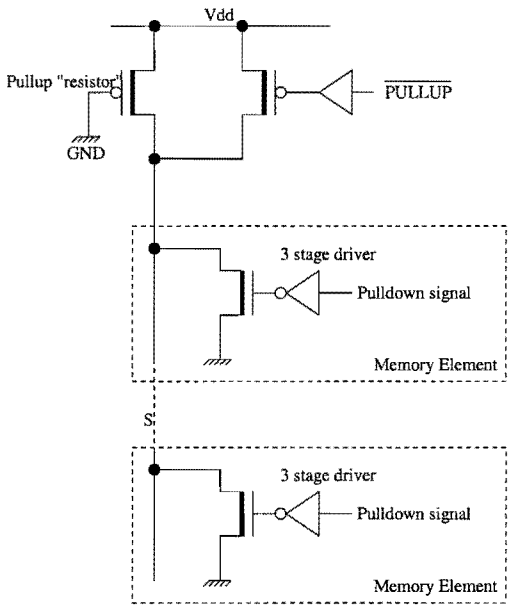


Figure 8.16: Pulldown Line on 'S'

as large and thin a channel area as possible) to have a large resistance. Since the only possible signal to pull up the line comes from a pullup drive before the commencement of the cycle to determine the highest priority element, this can be effected with a single MOSFET transistor. This also gains the advantage that a tri-state driver does not have to be used. Since the pulldown signal must still be strong (due to the length of the line and the number of connections on it), each one is formed by a large n-channel transistor that drives the line down. Each transistor is in turn supplied with a signal through a three stage driver. An advantage of using a pull down signal on the line is that an n-channel MOSFET transistor occupies half the silicon real estate of a p-channel transistor, therefore the size of the memory elements can be reduced. This arrangement neatly side-steps the need for a tri-state driver at any point in the design.

8.4 Performance of the Associative Memory

A write cycle for a particular column is shown in figure 8.17, showing a $15\eta s$ write time. The output of the memory elements, q, lags the signals because of the large internal capacitance that the line drivers are driving.

The biggest complication to the design of the associative memory entirely as shown in figure 8.10 is the slow pullup times of all the pullup resistors compared to the fast pulldown times. As the resistance of the MOS transistor used as the pullup resistor is decreased, the steady state current through the circuit on a pulldown increases, however, so, the uneven pulldown and pullup times are difficult to completely alleviate. A solution to this problem is the use of a *precharge* cycle on any line with a pullup resistor. Since an active pullup is comparatively

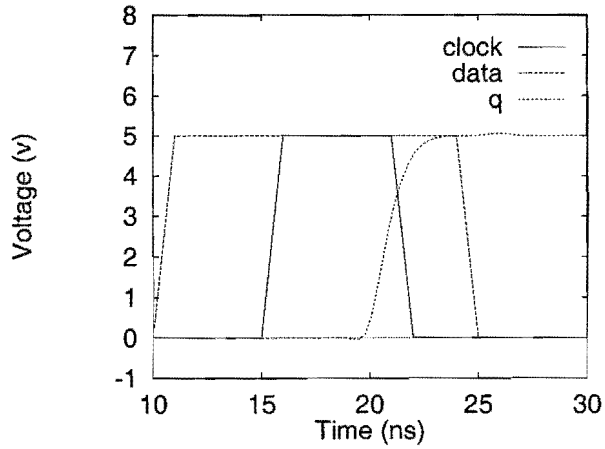


Figure 8.17: Write cycle to a memory column.

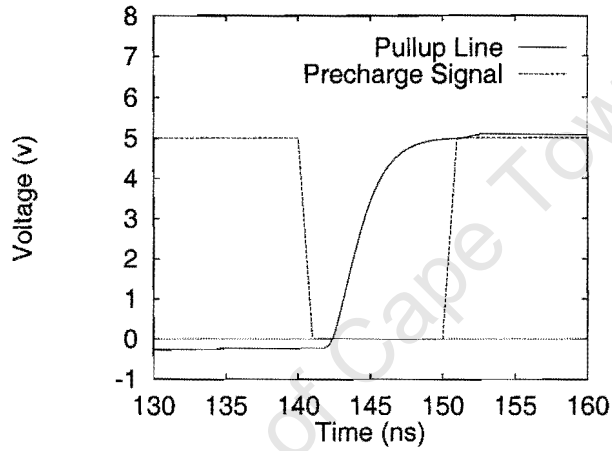


Figure 8.18: Precharge Cycles

very fast compared to the passive one, all of the pulldown transistors are disabled and an active pullup is effected on the line. Since an active pullup is used, the line can be driven high in $10\eta s$. The difference between the normal pullup time and the precharge rise time is shown in figure 8.18, the rise rate before $140\eta s$ being the pullup resistor.

The complete settling time for the destination matching, back-pressure and distribution elements combined has been simulated to be $20\eta s$. Each column of the cell priority elements has been shown to have a settling time of $15\eta s$ (see figure 8.19). However in the case of the cell priority elements, it must be ensured that successive elements in the columns do not prematurely pull down if they are not winners (since the correct value of the column might be a high voltage), thus a clocked timing chain is required to ensure that the elements don't pull down prematurely. The timing chain can be run at $66MHz$, providing a drive signal at the required $15\eta s$ intervals.

Thus, if a $10\eta s$ precharge cycle is used, followed by a $20\eta s$ settling time for all the elements except the cell priority elements, followed by an 8 bit cell priority selection (taking $120\eta s$), a read time of $160\eta s$ is possible if the priority encoder can settle in $10\eta s$.

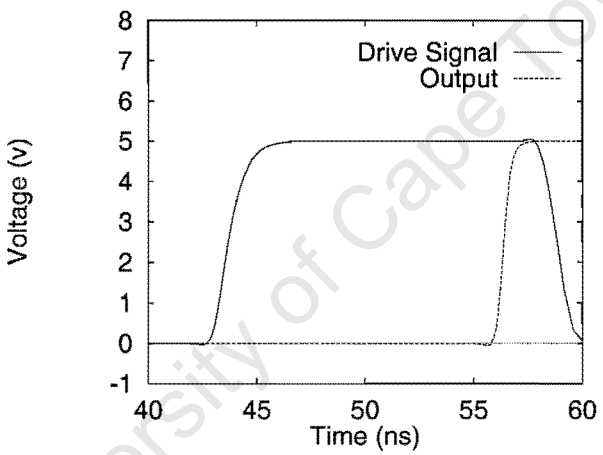


Figure 8.19: Settling Time of cell priority column

Chapter 9

Simulation Methodology

This chapter provides an overview of the simulation methodology used for constructing the simulation system used in the thesis. It is intended to be used as a companion to Appendix E and provides a detailed description of the files used in the simulation system. It is intended for those interested in extending the simulation code for their own purposes or who want to gain an in depth understanding of its operation.

9.1 Structure of the Simulation Code

The language used for the simulation was C++. This was done for a number of reasons:

- The authors considerable familiarity with the language.
- The speed of execution of C++ binaries (a considerable impact when some simulations take days or weeks to complete).
- The small size and low memory usage of well written C++ code. This is especially a consideration when the size of the switch being simulated is large since the memory usage can grow as the square of the number of ports.
- The object orientation of C++. This allows the language to be effectively extended to an arbitrary level of abstraction. This is particularly aided by the advanced features of C++ such as templates and operator overloading.
- The portability of C++. C++ compilers can be found for almost any possible architecture. Thus, if the code is sufficiently carefully written, the simulations would be able to run on any platform without modification.

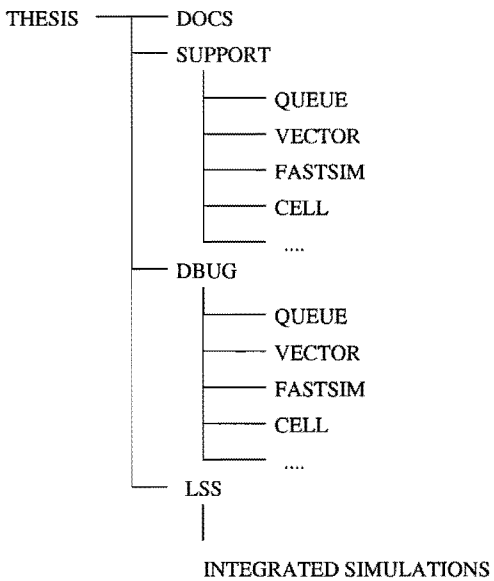


Figure 9.1: Original Project Source Tree

Other systems were considered for use. In particular a graphical system called "OPNET" [60] was investaged. It was discarded since it was primarily designed for the design of networks and not switches. Thus, the graphical interface would not allow the easy construction of very large switching fabrics. Although an extension API was provided, it was felt that the difficulty of learning the API and the slow speed of execution (and large memory usage) of the simulator would preclude its use for the task of simulating large scale ATM switches.

The methodology chosen for the simulation developed as the task progressed. The initial concept was to create a "toolbox" of objects that could be usefully and generically combined to form a simulation. This toolbox consists of classes to generate random numbers conforming to arbitrary distributions, FIFO queues, priority queues, matrices, vectors and a discrete time simulation kernel (amongst others). Thus, the design was largely bottom up. This has the advantage of generally producing efficient code and allows the functional blocks to be tested as soon as they are written. The approach can result in unnecessary classes being created where insufficient knowledge of the higher layer concerns of the simulation are encountered. (Effectively, classes can be created in the anticipation of being useful. Upon integration of the components, other classes might be discovered that are required, and other developed classes might be discovered to be superfluous). However, the technique allows the greatest possible reusability of the objects in the code.

The original code tree of the project is shown in figure 9.1. The code was developed in toolbox form in the support directory. Code fulfilling a similar function was placed in the same directory, thus for example FIFO queues and priority queues would be placed in **queue** and a templated class for a two dimensional array would be stored in **vector**. Since the compiler is unable to guarantee much more than syntactic correctness of the classes, test code is required. This is placed in the **dbug** tree. This tree exactly follows the support tree

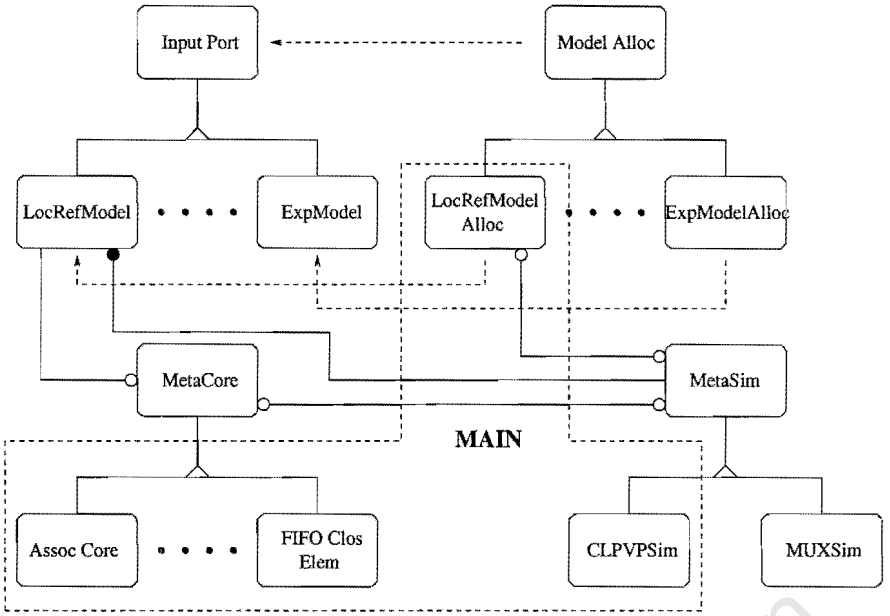


Figure 9.2: Abstract class interaction

but consists only of debug code, the object files and executables needed to test them and project files. Once the objects in the toolbox were tested, they were integrated together to form individual simulations in the **lss** tree.

The deficiency of this methodology became apparent in integration. The difficulty arose that where several variants of the same form of object were combined, a large amount of glue code had to be introduced and the amount of code was quadratic in relation to the number of objects that had to be combined. The obvious solution was to create a number of abstract classes that presented the same interface regardless of how their children implemented them. The three main abstract classes identified were the simulation (or test) being conducted, the input ports which provided the traffic for the simulation and the core, the switching system being tested. The only problematic aspect of this methodology is illustrated by the interaction of the simulation class with the input port class. The simulation class is required to allocate a large number of input ports, and these input ports require a number of parameters to classify the traffic that they are to pump into the core being tested. However, the simulation class should not be required to understand the parameters needed by the input ports since this will result in the same problem of extra glue code specific to the simulation being performed being required within the simulation. The solution chosen was to have two different abstract trees. There is the abstract interface to a logical object encapsulated through the virtual methods of the parent class and in addition, there is an abstract allocator class that has a virtual method that will allocate an instance of the derived class from the parent. The simulation is passed a reference to the allocator class and is thus able to allocate all of the input ports for the simulation. The extra parameters required for the input ports are stored in the allocator class and are provided by *external* glue code which encapsulates their

meaning (the simulation class is thus divorced from requiring to manage these parameters for the input ports). In some cases these parameters are passed to the input ports, in other cases more complex objects that require more memory to implement are stored within the allocator class and referenced by the input ports. This is used for statistical generation classes (who's random numbers are statistically independent). This technique was found to be very successful and was also used in the classes which recursively construct arbitrary Clos networks. The use of this technique is illustrated in figure 9.2 (simplified). Here a simulation encapsulating a large number of cores and a cell loss probability test and the locality of reference traffic model is illustrated.

The source directory tree was changed slightly as a result of these additions to include debug code for extra tools. The old simulations were kept in an archive directory (and modified in some cases to produce some more toolkit classes), however the toolkit classes were not modified. Thus, the simulations will no longer compile and are kept largely as an archive.

9.2 Discrete Time Event Simulation

One of the central components developed for the simulation in the toolbox was a discrete time event simulation system. The basis for this type of simulation is essentially very simple. The simulation space is divided into a number of events in time. When the time for the event is reached, an object instance is notified of this occurrence through a virtual method. The object instance then performs some appropriate action and then determines when (if ever) another time event should occur. It then re-registers itself with the simulation kernel which will wake it up at the appropriate time.

The discrete time simulation system essentially consists of a priority queue (implemented as a skew heap) of time tags. Each time tag holds a pointer to a suspended object (which must be derived from KernBase). When the time tag is removed from the heap, the object's WakeUp virtual method is called, this informs the object that it must procede with the simulation. The advantage of this approach is that where the object's next wake up time is easily calculated, no processing need occur at time slots in which nothing will take place, this results in a much higher efficiency in the case where the time density of the events are low.

This technique works very well for simulation phenomena in which the next occurrence is easy to determine. This is the case for systems with FIFO queues. If the queue is empty, no wakeup is necessary. If an element is in the queue, the object should be woken up in the next time slot. The technique becomes difficult to use when simulating a switching fabric which utilises the associative memory, however. In this case it is difficult to determine anything more than the fact that processing will probably have to occur in the next time slot.

However, this would be overkill for the discrete time simulator and can be more efficiently handled by a polling loop. For this reason, the simulation is in practice handled by using a discrete event simulator for the input models and a polling loop for the cores. Even in the case of a FIFO queued system it simplifies the handling of backpressure signals.

9.3 Distributions and Models

For a person wishing to review the code, it might be confusing to notice that there are two directories that appear to be closely related, the **dist** and **model** directories. The essential difference between these two directories can be illustrated by comparing an exponential distribution generator with the exponential model. The distribution generator simply returns random numbers that conform to a particular distribution. It makes use of the Distribution class (which uses a generic technique to produce arbitrary distributions) and extends it. The ExpModel class (Exponential Model) has a reference to this distribution and uses it to determine when next a cell should be scheduled. It thus interacts with the discrete time simulation kernel in order to determine the next time slot. In addition, it ensures that impossible traffic is not generated by the Exponential Distribution. For example, it ensures that two ATM cells are not delivered in the same time slot.

9.4 Simulation of Virtual Circuits

It was decided at an early point in the simulation that virtual circuits themselves would not be explicitely simulated. The reasons for this were that the issue of Connection Admission Control was outside the scope of the thesis (a CAC scheme would be necessary to determine whether a virtual circuit should be accepted or not). In addition, the results obtained for the performance of a switch would be take a longer time to be statistically meaningful since they would have to be amortised over a large number of virtual circuits. The approach that was taken was to ignore virtual circuits entirely when simulating only the switching fabrics. The traffic generators simply generated cells that were destined to a particular port. One particular advantage of this approach is that the simulation can be designed in such a way as to allow the steady state of a particular set of traffic conditions to be very quickly reached.

When output buffering and resequencing were added, virtual circuits were required. The approach taken here was to assume that there was a virtual circuit in place between every input port and output port per traffic class being simulated by the switch. This has some important consequences. For example, the locality of reference model also has the side effect that it simulates the effect of a large virtual circuit amongst a number of other VCs. This result is not entirely desirable but has the advantage of avoiding the issue of CAC completely.

9.5 Comparative Performance

When a simulation result is presented it aids interpretation if it can be compared to other existing fabrics. For this purpose, two competing switching fabrics were chosen, a Clos network and a perfect Cell Switch Fabric. There are many possible ways of configuring a Clos network, thus the basic approach taken was to simplify the process as much as possible by choosing round binary numbers wherever possible. The other factor was to allow the technologies used in the two simulations to be comparable, (i.e. equivalent speed up, internal buffer space and number of ports). The perfect fabric is an abstract fabric that is capable of switching any cell from the input to the output of the switch without lost and with a fixed delay (one cell time slot was chosen for the delay).

The Hypercube network has 1024 ports with 1024 elements each of which has a speed up of 11 and two inputs and outputs (to and from the SEs and to and from the IM and OM). This was the yardstick in technology used to construct the Clos network. The Clos network has 1024 ports connecting to a three layer Clos network (a speedup of two is used). This would result in 32x32 elements operating at a link speedup of 64. This is technologically far beyond that of the hypercube network. Thus, each of the Clos networks is recursively decomposed into a two layer network of 8x8 elements (with a link speedup of 16). These elements are technologically more advanced than the hypercube network elements, but do not require the intervening fabric of SEs. The total number of elements in the Clos network is 768 elements. (3 layers of 32 of 8 elements).

Chapter 10

Simulation Results

This thesis has presented the topic of large scale ATM switches by first approaching the topic from the standpoint of the fundamental requirements and limitations imposed on an ATM switch and an overview of existing research. The performance evaluation has taken place through the means of simulation rather than analysis. The primary motivations for this lie in the authors own strengths and weaknesses and also the fact that a design can be evaluated under a larger variety of traffic conditions through a simulation than is possible with analysis. For the purposes of this dissertation, the results will be presented in the order of the most traditional and conservative to the more realistic models used. (This was in fact the order in which the simulations were performed). These sections are:

- **Exponential Interarrival Model** - section 10.1 contains the most conservative model, a simple exponential interarrival model. (See Section 7.1).
- **Locality of Reference Model** - section 10.2 presents the results obtained by adding locality of reference considerations to the Exponential Interarrival Model. (See section 7.2).
- **Multiplexed Model** - section 10.3 presents the results which show the effect of altering various parameters (mean, variance, burst length and locality of reference) on the ATM switch. These results illustrate the inadequacy of traditional traffic models when faced with real ATM traffic. (See section 7.3.6).

10.1 Exponential Interarrival Model

The first stage in determining the performance of the fabric was to determine its behaviour without an output buffering and resequencing stage. In addition, the performance of the

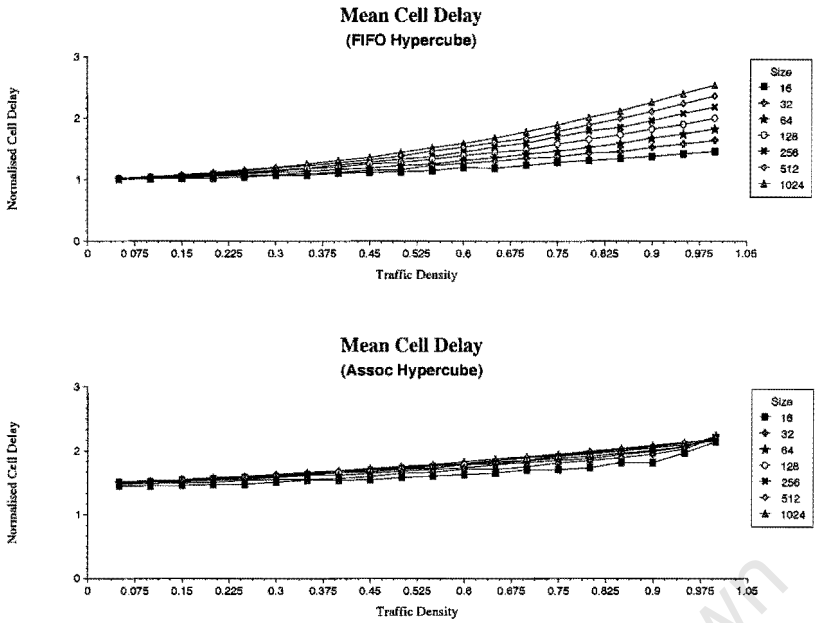


Figure 10.1: Mean Cell Delay against Hypercube Size

hypercube fabric as it scaled needed to be determined. Both the FIFO queued and Associative Memory queued versions were analysed over the complete range of traffic densities and switch sizes, (there is no output buffering stage in this simulation, thus, the results not in fact defy mathematical analysis.) The buffer size used was 32, there was no observed cell loss over any simulation period of ten million cells.

Figure 10.1 shows the means cell delay against switch size for both the FIFO queued and the Associative Memory Queued hypercubes. The charts show a very graceful performance in response to growing switch size (in the FIFO hypercube, the delay can be seen to be logarithmic with respect to the fabric size). At small switch sizes, the FIFO queued fabrics demonstrate a smaller mean cell delay than the Associative Memory fabrics. However, the performance of the Associative Memory Fabric is almost completely constant regardless of switch size and regardless of traffic density. Thus, the associative memory is shown to be superior for the queueing within the hypercube fabric .

The standard deviation of the cell delay (a measure of the burstyness of the cells being issued by the fabric) is shown in figure 10.2. The figure illustrates an interesting phenomenon. As expected, the variance of the cell delay increases gradually (and logarithmically) for the FIFO queued case. However it is almost completely constant for the Associative Memory Queued case (in fact it decreases slightly for larger fabric sizes). An interpretation of this result is the ability of the associative memory to effectively combine all of the hypercube buffers into one logical queue is more pronounced in the larger fabric sizes.

The peak cell delay against hypercube size is shown in figure 10.3. The peak cell delay is an inherently difficult quantity to measure statistically since it represents the farthest outlier

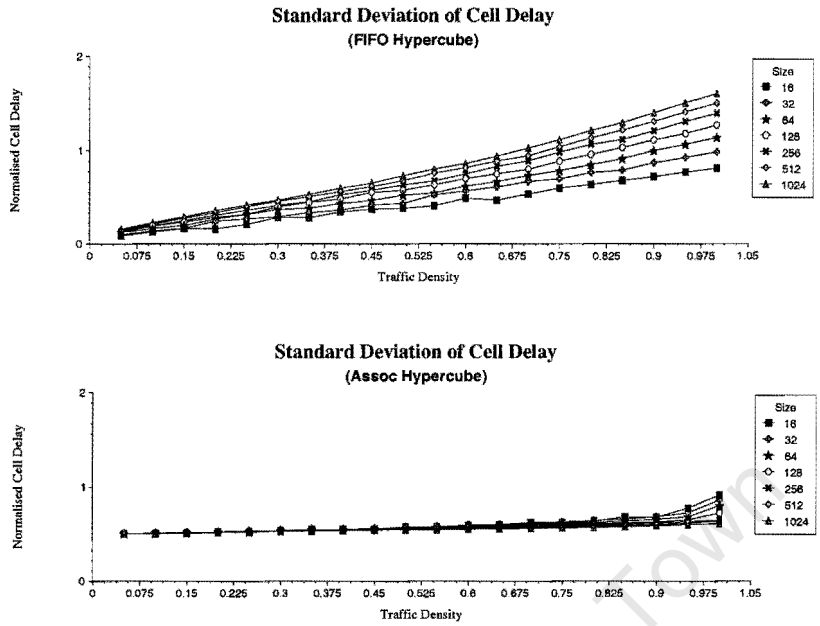


Figure 10.2: Standard Deviation of Cell Delay against Hypercube Size

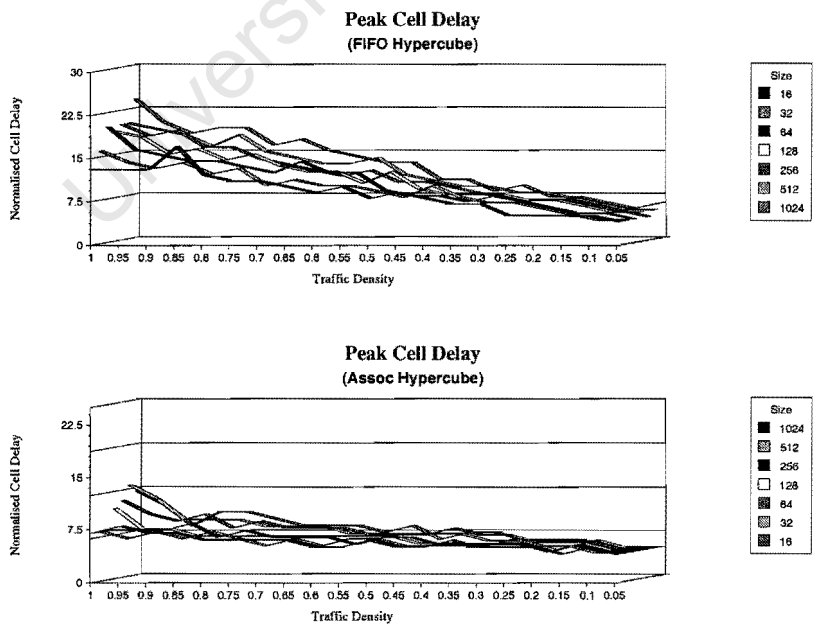


Figure 10.3: Peak Cell Delay against Hypercube Size

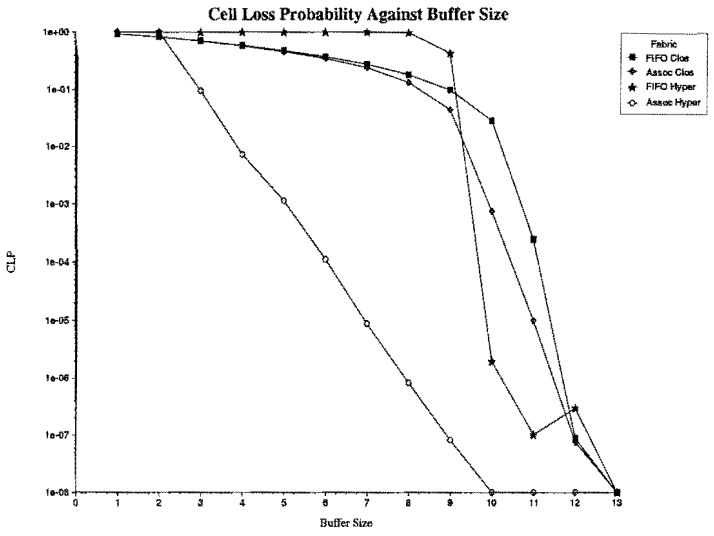


Figure 10.4: Cell Loss Probability against buffer size (Exponential Interarrival)

of a particular data set, thus a different graph format has been used. The peak cell delay is shown to increase almost linearly with increasing traffic density in the case of both fabrics. However, the gradient of this increase is far smaller in the associative memory's case. As with the Standard deviation of the cell delay, the peak cell delay increases with increasing switch size in the case of the FIFO queued hypercube and decreases with increasing switch size in the case of the Associative Memory hypercube.

The next set of tests that were performed were to determine the buffer size required for the exponential interarrival. The cell loss probability against buffer size was calculated for the FIFO queued hypercube, the Associative Memory queued hypercube, the FIFO queued Clos network and the Associative Memory queued Clos network. The results are shown in figure 10.4. This was determine by sending at least one hundred million cells through the switching fabric and ensuring that at least five cells were lost. (I.e. for some data points up to five hundred million cells were simulated). A traffic density of 1 was used. The size of the switching fabric was 1024 elements.

The figure shows the clear superiority of the Associative Memory queued hypercube in terms of its buffer usage (indeed it is an almost perfect inverse exponential relationship). Interestingly, the FIFO queued hypercube is shown to be inferior for much of the operation to the Clos networks. The Associative Memory realises a small gain over the FIFO queued scheme in the Clos networks for a portion of the buffer sizes. This shows that the Associative memory seems to have a larger impact on the hypercube than on the Clos network for this metric.

In both of the associative memory queued fabrics, a cell delay class must be determined (this is the maximum delay that the switching fabric allows a cell before it is deleted). Obviously, a larger cell delay class results in a lower cell loss probability. However, where the time

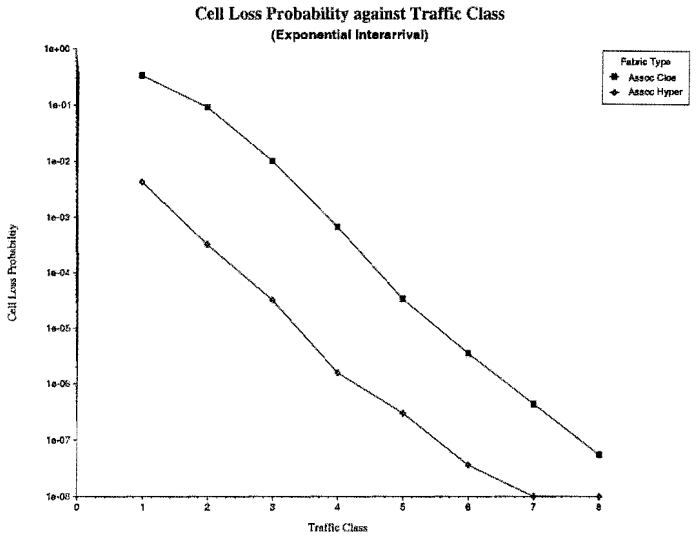


Figure 10.5: Cell Loss Probability against Delay Class

resequencer is used, this large class results in a large resequencing delay. Thus, an optimal cell delay class is one in which the overall delay is small and yet the cell loss probability is acceptable. The simulated results appear in figure 10.5.

Another influence that the buffer size has on the performance of an ATM switching fabric is the delay performance (this is mainly the case where backpressure is used). Since this is the case for the fabrics studied here, the cell delay performance against buffer size was simulated. The size of the fabric was 1024 elements. The results are shown in figure 10.6, the variance of the cell delay against buffer size is shown in figure 10.7.

These graphs show some interesting results. The mean cell delay is in fact lower for a hypercube for small buffer size than for a large buffer size, while the opposite is true for the Clos network. A similar phenomenon is shown for the variance of the cell delay. The reason for this is that a hypercube fabric is not a fair fabric. Thus, cells that are close to their destination are more likely to survive and be routed out than those that are far away. In the Clos network, the cell are delayed by HOL blocking due to the back pressure signal and their delay becomes longer. The associative memory is shown to perform very well in this environment however, particularly for small buffer capacities. In both the Clos network and the hypercube a considerable improvement in performance was observed.

As a result of figure 10.4 a very conservative buffer size of 32 was used in all subsequent simulations, thus, no cell loss was observed. One hundred million cells were passed through each cell switch fabric in order to determine its cell delay response.

Figure 10.8 shows the mean cell delay for the various fabrics. It illustrates the clear superiority in this performance that the hypercube fabric delivers. The reason for this far superior delay performance is the fact that cells in the Clos network must traverse six layers of SEs

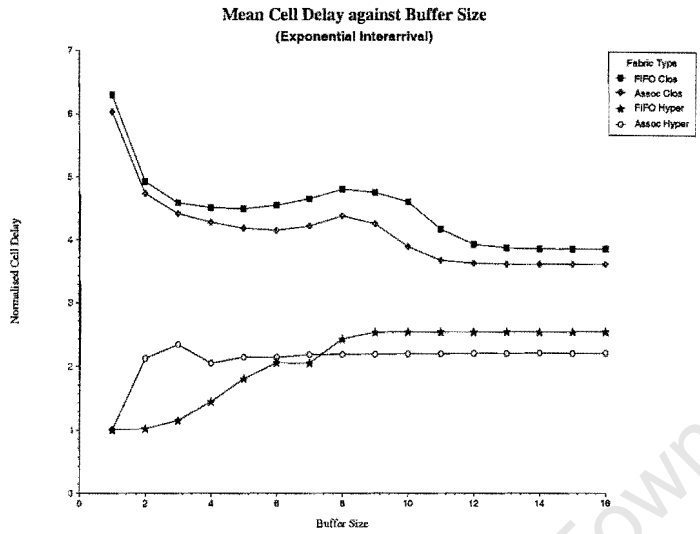


Figure 10.6: Mean Cell Delay against buffer size

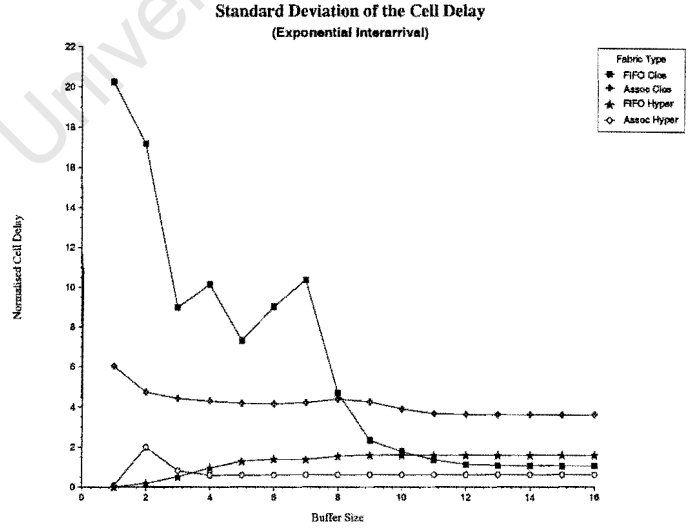


Figure 10.7: Variance of the Cell Delay against buffer size

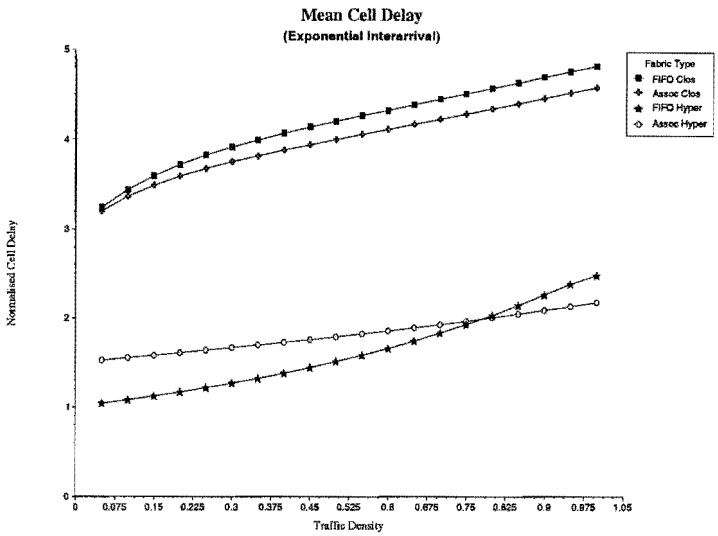


Figure 10.8: Mean Cell delay against traffic density (Exponential Interarrival)

at an effective speedup of two. The cells in the hypercube fabric need to traverse 10 REs at a speedup of 10. Thus, the delay performance of the hypercube is far superior. The Associative Memory shows some performance benefit in the Clos network. In the Hypercube, it results in a lower mean cell delay at high traffic densities. The response of the associative memory is far more linear in this case, however, which results in an easier determination of the switch's performance.

The standard deviation of the cell delay against the traffic density is shown in figure 10.9. This illustrates that the variance is most changeable in the FIFO queued hypercube, starting below all the other data points and ending far above them. Despite this result, the associative memory results in the hypercube fabric outperforming the Clos network for all of the data points. The associative memory also improved the performance of the Clos network slightly in this regard. Finally, figure 10.10 shows the peak cell delay against traffic density for the various fabrics. Here, the hypercube with associative memory far outperformed the other fabrics while the hypercube with FIFO queued discipline performed the worst. The associative memory resulted in a modest improvement for the Clos network at high traffic densities.

10.1.1 Output Buffering and Resequencing

All of the preceding simulations determine the performance of the various fabrics and schemes without an output buffer or resequencer. The following simulation results are determined for the case of output buffering and resequencing. As in the case of the fabric without an output buffer, a buffer capacity that results in a reasonable cell loss probability must be determined. An additional result of this is the inclusion of an abstract perfect switching

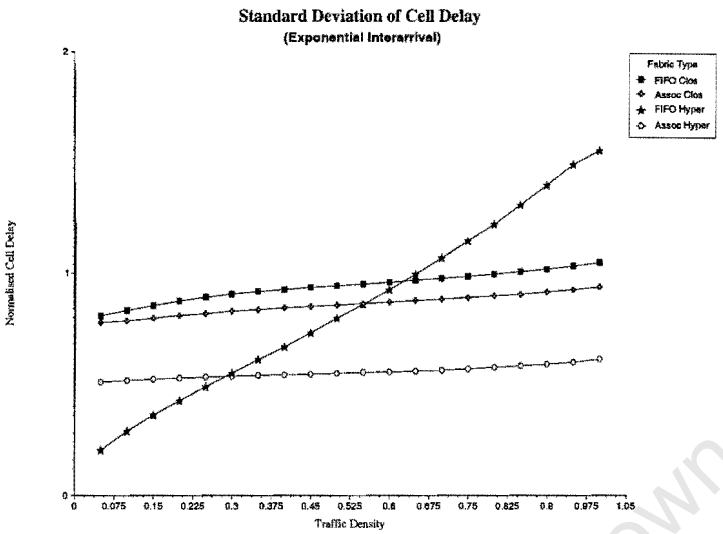


Figure 10.9: Standard Deviation of Cell Delay against traffic density (Exponential Interarrival)

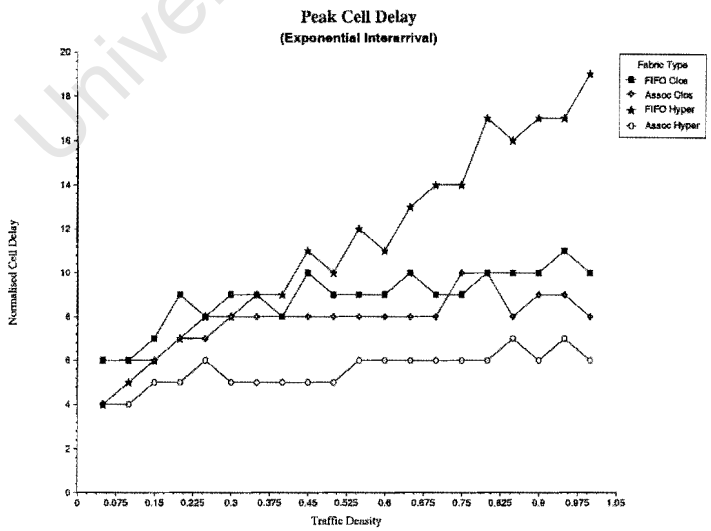


Figure 10.10: Peak Cell Delay against traffic density (Exponential Interarrival)

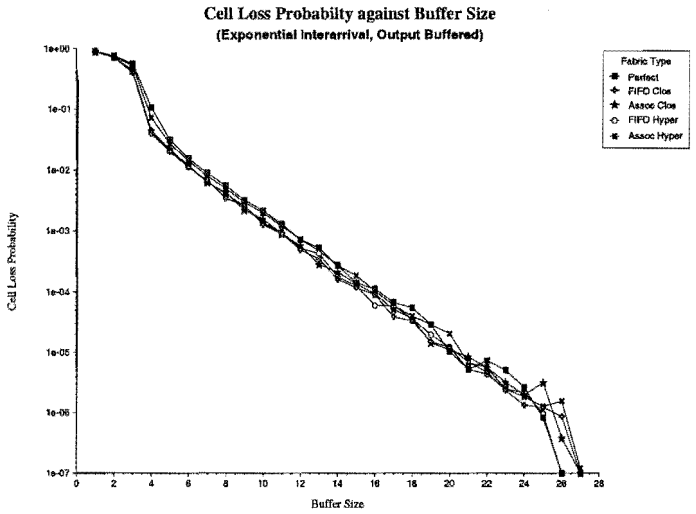


Figure 10.11: Output buffered Cell Loss Probability against buffer size (Exponential Interarrival)

fabric (simulations were not performed of this without output buffers since the results are completely invariant). Since the queue would become unstable if a traffic density of 1 were to be used in the simulation, a lower traffic density (0.8) was used to determine the buffering requirement on the output. The result of this run is shown in figure 10.11.

As a result of this simulation, an output buffer size of 32 was selected. A simulation run was then conducted in which a Traffic Delay class of 15 was used for all fabrics (this is used by the time resequencer), for a switch size of 1024 elements. Since the output buffer is only capable of supplying cells at the link rate, a cell loss probability was measured for traffic densities above 0.85. This is shown in figure 10.12. An interesting result of this simulation is that the CLP is almost completely independent of the type of fabric used.

Another interesting metric to be obtained from the fabric is those cells that are not immediately released by their sequence count number and hence do not need to be passed on to the time resequencer. This proportion does in fact vary quite considerably between the different fabrics and is shown in figure 10.13. It is interesting to note that in most fabrics this figure is very low until the higher traffic densities at which point their performance equalises. Even the perfect fabric is capable of delivering cells out of sequence. The reason for that is that if the cells are discarded by the output buffer before reaching the resequencer, the new sequence number will not be recorded. Thus, towards the higher traffic densities, the proportion not immediately resequenced is almost precisely that of the CLP. The simulation shows that both variants of the hypercube outperform the Clos network. Interestingly, the FIFO hypercube performs almost as well as the perfect fabric in this case, indicating that it releases the cells almost entirely in order. A point must be made about this simulation, since the size of the fabric is large and the cells are evenly distributed to the output, it is

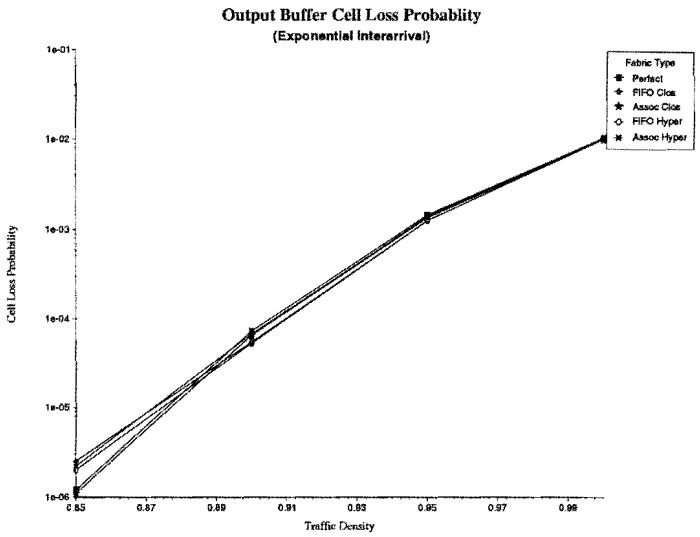


Figure 10.12: Output buffered Cell Loss Probability against traffic density (Exponential Interarrival)

in fact extremely unlikely that a cell be misdelivered regardless of the type of fabric used. More pertinent results are obtained from the locality of reference model for the case when the switch is handling large bandwidth VCs in conjunction with smaller ones.

The cell delay characteristics for the output buffered case are shown in figures 10.14 - 10.16.

In all of these figures, the profound effects of the output buffer becomes apparent and the relative unimportance of the cell switch fabric itself is shown (illustrating the important of the OM in an ATM switch). For the Mean Cell Delay, the Clos fabrics show the disadvantage interposed by the number of stages they are composed of (which adds a fixed time interval relative to the perfect and hypercube fabrics). The associative memory Clos fabric performs ever so slightly better than the FIFO buffered version. The hypercube fabric's performance is very close to the perfect fabric. A slightly higher delay is experienced in the FIFO queued version than in the Associative Memory version.

The variance of the cell delay shows almost precisely the same result. The hypercube fabrics perform very close to the perfect case while the Clos network performs slightly less well. The FIFO queued hypercube fabric performs less well than any towards the higher traffic densities but the difference is very slight. The peak cell delay shows some interesting effects. If a cell is ever not in the correct sequence when it arrives at the output port it must be resequenced in time. Thus, those fabrics which only present the cells to the output in the correct sequence (the Perfect fabric and the anomalous behaviour of the FIFO hypercube) have a much better peak cell performance than the others. Similarly to the other delay results, the differences between the fabrics is found to be small at high traffic densities.

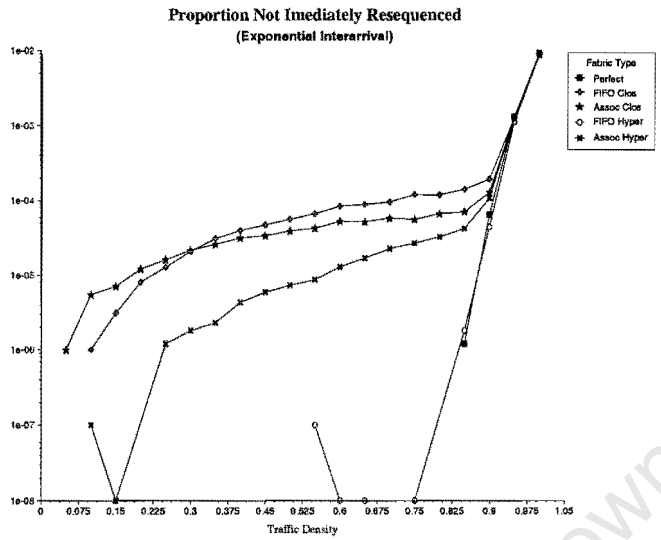


Figure 10.13: Proportion of traffic not Immediately Resequenced against traffic density (Exponential Interarrival)

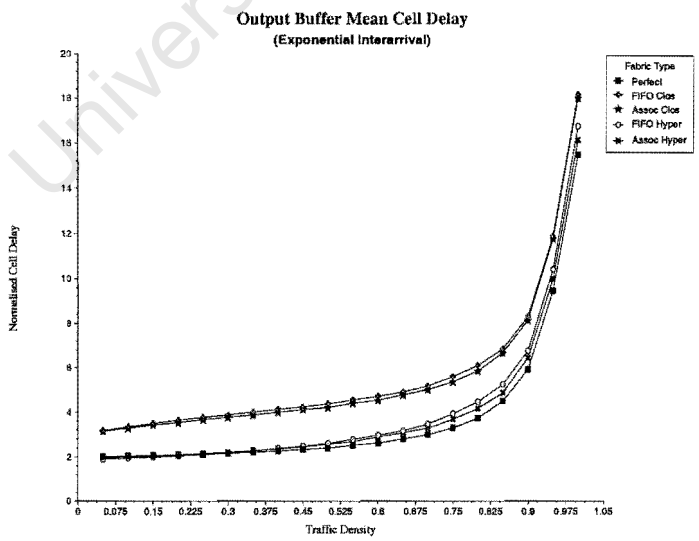


Figure 10.14: Output buffered Mean Cell Delay versus Traffic Density (Exponential Interarrival)

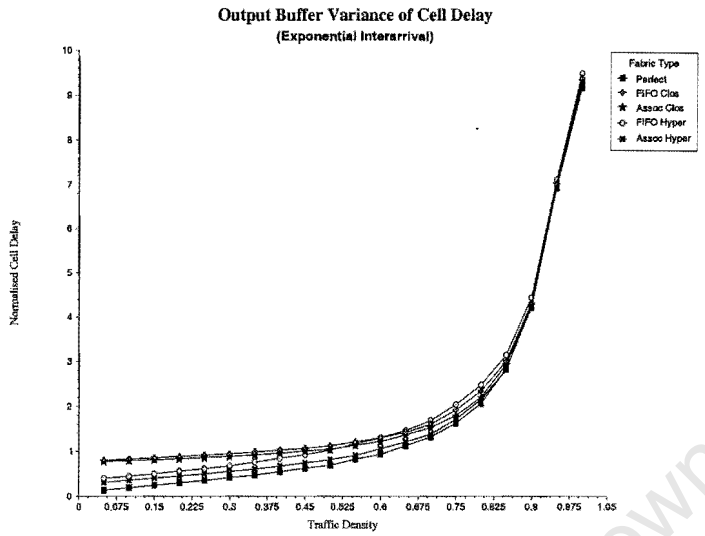


Figure 10.15: Output buffered Variance of the Cell Delay versus Traffic Density (Exponential Interarrival)

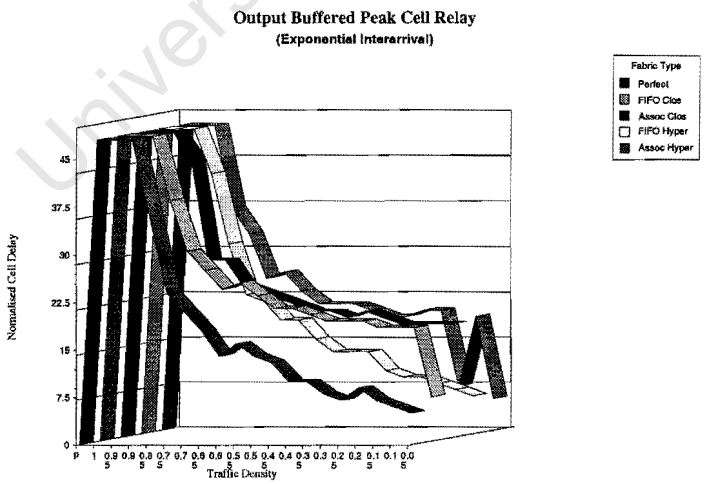


Figure 10.16: Output buffered Peak Cell Delay versus Traffic Density (Exponential Interarrival)

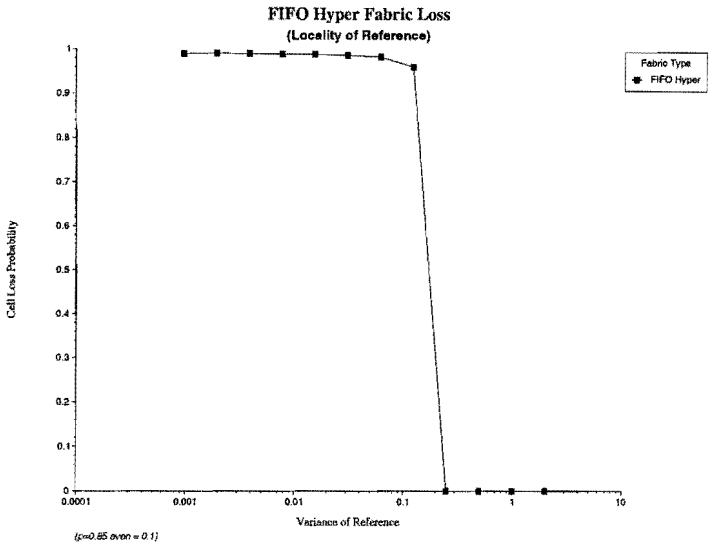


Figure 10.17: Locality of Reference Cell Loss Probability in FIFO Hypercube

10.2 Locality of Reference Model

The locality of reference model was run for a fabric of 1024 ports, a traffic density of 0.85 and a proportion of evenly distributed traffic of 10%. The variance of the reference was altered from 2 (almost completely evenly referenced outputs) to below 0.0001 (traffic that is not evenly distributed is sent to almost only one output port). There were ten million cells per run. The left hand side of the graph represents a high locality of reference (small variance of reference).

The first observed anomaly is shown in figure 10.17. The FIFO hypecube begins to perform very badly for high locality of reference (despite almost flawless performance in the Exponential Interarrival model), its CLP jumping to 95%. Interestingly all of the other fabrics have no observed cell loss probability whatsoever.

The Cell Loss Probability at the output stage is shown in figure 10.18. As shown there is little correlation except that the FIFO Hypercube loss drops to an unmeasurable quantity. This occurs at the point where the fabric itself is responsible for a large CLP. There is a slightly higher CLP for all of the fabrics for very narrow locality of references.

The next observed quantity is the proportion of cells that are not immediately resequenced at the output. This is shown in figure 10.19. The proportion is shown to degrade for all of the fabrics for increasing locality of reference. The perfect fabric's statistic probably degrades due to the fact that an increasing proportion of the traffic is being placed on VC that must be resequenced in time. The FIFO Hypercube fabric, interestingly, has the best performance of all of the real fabrics, however, some of its statistics are suspect due to the incredibly high Cell Loss Probability sustained. The Associative Memory Hypercube outperforms the other fabrics and the Associative Memory Clos fabric also shows an improvement over the FIFO

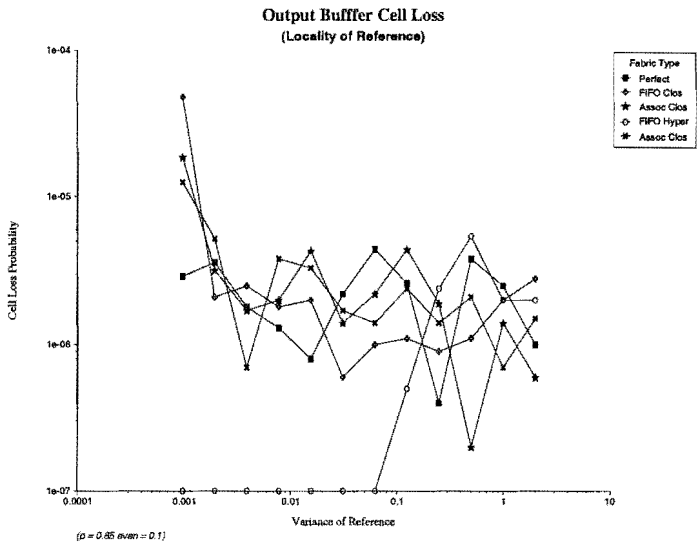


Figure 10.18: Cell Loss Probability (Locality of Reference)

queued hypercube.

The delay performance of the various fabrics is shown in figures 10.20 - 10.22. The Mean Cell Delay remains reasonably constant for the majority of the fabrics bar the FIFO Hypercube which has a fairly early rise and a drop towards the highest localities of reference (this is probably due to the high CLP suffered). The Perfect fabric in fact shows a gradual improvement in the Mean Cell Delay. Of the remaining fabrics, the Associative Memory queued hypercube shows the best performance (and the Associative Memory leverages a modest improvement in the Clos network). All of the fabrics show a very sudden degradation at the highest localities of reference, this is probably due to the fact that the majority of their traffic is now destined for only one output port. At this port the traffic must suffer resequencing in time due to the fact that a certain proportion of it will be lost.

The variance of the Cell Delay shows a similar trend. The Perfect fabric's performance is almost completely unaffected by this statistic, the FIFO Hypercube shows a very early and dramatic degradation while both the Clos networks and the Associative Memory Hypercube show a sudden degradation at high locality of references. The Associative Memory leverages a small advantage over the the FIFO queued schemes.

An interesting phenomenon is observed when the Peak Cell Delay is examined. Both the FIFO hypercube and the FIFO Clos networks show a very sudden degradation in performance. The reason for this is that cells are released from the switching fabric after the time resequencer has passed on. (The associative memory fabrics will delete the cell under the same conditions and avoid the problem, nonetheless no cells were lost in the fabrics for all the operating conditions). A zoom in of the peak cell delay shows that it degrades slightly for higher locality of references, but for lower locality of reference is essentially uncorrelated. The pecking order is hard to determine but is essentially the Perfect Fabric, Associative Memory

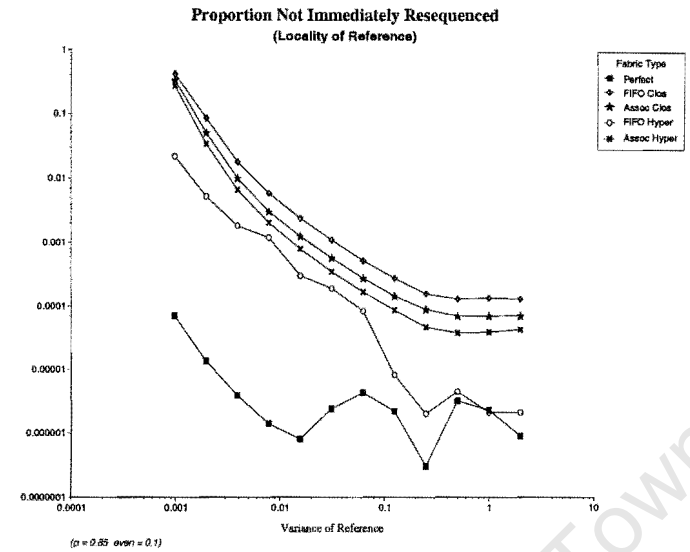


Figure 10.19: Proportion of Traffic not immediately resequenced (Locality of Reference)

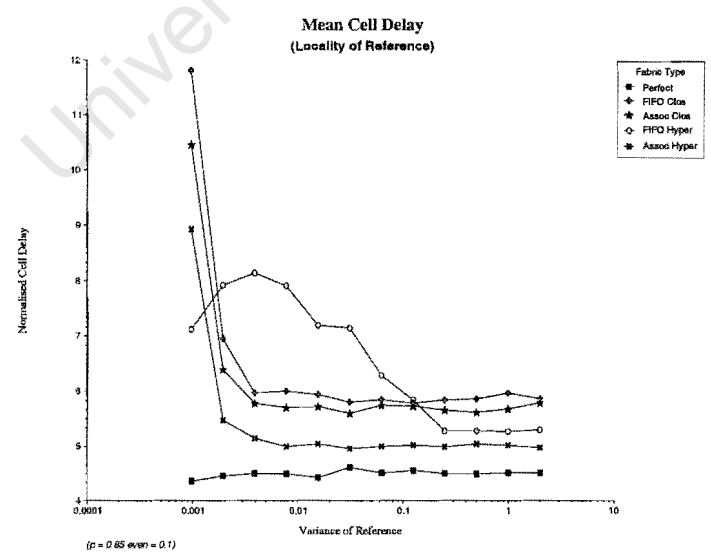


Figure 10.20: Mean Cell Delay (Locality of Reference)

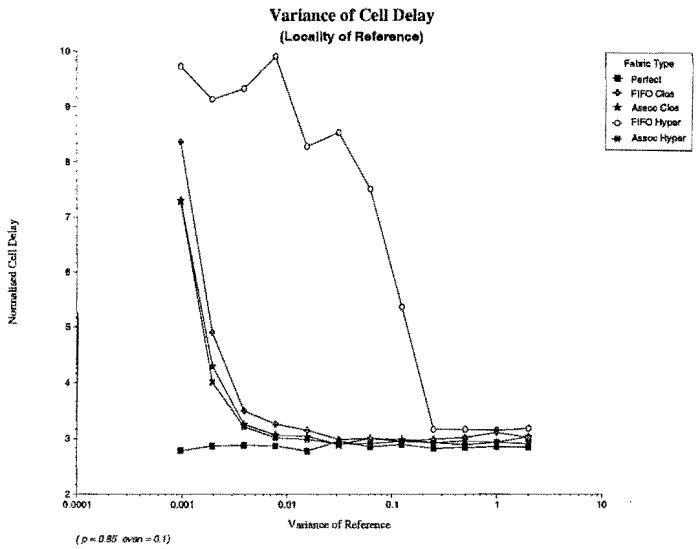


Figure 10.21: Variance of Cell Delay (Locality of Reference)

hypercube, Associative Memory Clos network, FIFO queued Clos network and FIFO queued hypercube.

Due to the loss of synchronisation in the two FIFO queued fabrics, a measurable loss of cell sequence was observed. This is shown in figure 10.23. The proportions remain reasonably small and illustrate a sudden mis-sequencing problem at a very high locality of reference.

The results of these simulations are particularly interesting since most analyses of switching fabrics assume that traffic is evenly distributed. Interesting further research would be to determine the extent to which this is in fact the case in real ATM and internet traffic. It also illustrate the difficulties inherent in the bland assumptions normally made through traditional analytical methods. Fabrics which perform perfectly acceptably under uniform traffic conditions demonstrate an unacceptable performance under higher locality of reference. An interesting extension to the research would be to investigate the effect of locality of reference patterns on other switching fabrics.

10.3 Multiplexed Model

In order to establish some baseline parameters for this model, an initially extremely simple simulation was performed. A single cell source (i.e. non-bursty) was used for the lower priority VBR traffic class. A very low variance of 0.1 was chosen. The variances used in the simulations are not absolute but are expressed as a proportion of the mean cell rate. Thus if 10% variance is chosen and the cell rate is 0.5, then the variance for the source is 0.05. This is more reasonable than choosing a fixed variance since a low volume data source is likely to have bursts that are proportional to the amount of data being sent. This cell source

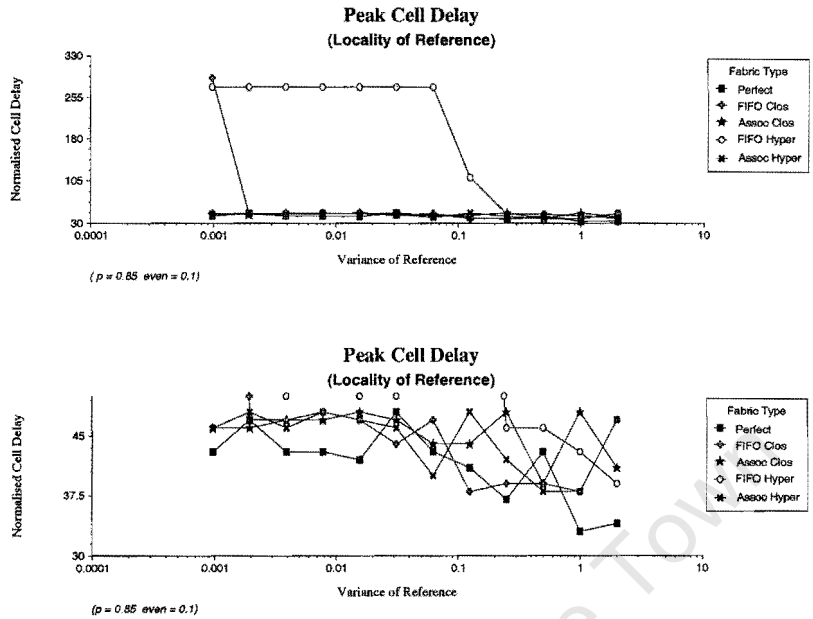


Figure 10.22: Peak Cell Delay (Locality of Reference)

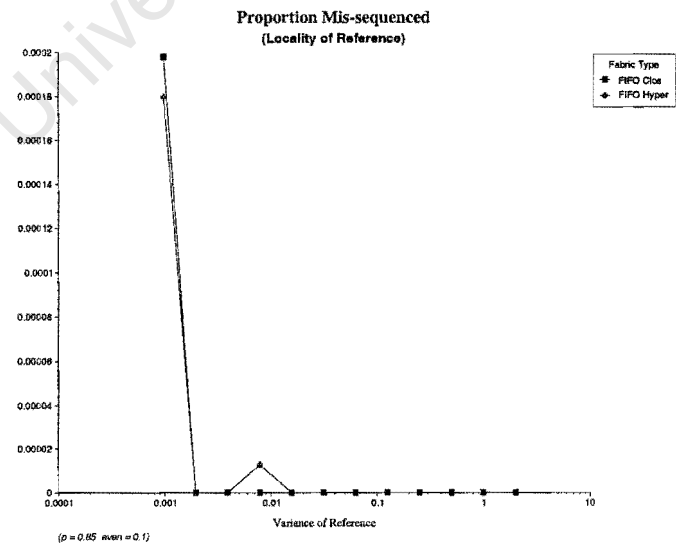


Figure 10.23: Proportion of Cells out of sequence (Locality of Reference)

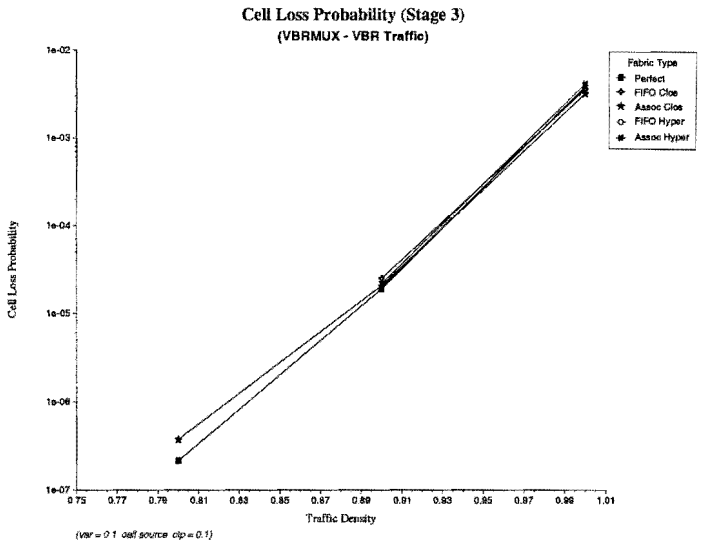


Figure 10.24: Cell Loss Probability for Multiplexed (VBR model, cell source, low variance)

was multiplexed with an Exponential interarrival model which has 10% of the bandwidth. The simulation was performed on the five standard fabrics over 20 million cells for traffic densities ranging from 0.1 to 1.0.

The first result was that, as expected, a high Cell Loss Probability was experienced at the output of the switch for very high traffic densities, shown in figure 10.24, this is only for the lowest class of traffic density. At the upstream node a uniform CLP of approximately 6×10^{-6} was obtained for the highest traffic density and the lowest traffic class. This shows the reduction in CLP obtained for a more uniform traffic source (in this case even than the exponential interarrival model). No cell loss was experienced within the switching fabrics.

Following the other analyses, the proportion of the traffic that is not immediately resequenced at the output is shown in figure 10.25. The upper graph represents the traffic results for the higher priority CBR model data while the lower figure shows the results for the lower priority VBR data. As can be seen, although both proportions are increasing, the response of the higher priority traffic is far more uniform than that of the lower priority traffic (illustrating the dramatic effect of even a simple cell output scheduling technique). The fabrics perform in more or less the following order: FIFO Hypercube, Perfect fabric, Associative Memory Hypercube, Associative Memory Clos network and FIFO queued Clos network.

The mean cell delay, variance of the mean cell delay and peak cell delay are shown in the figures 10.26 to 10.28.

The mean cell delay shows the advantages of scheduling with two different priorities. The higher priority traffic class has a much more linear (and smaller) cell delay than the lower priority class which shows an asymptotic increase towards high traffic densities. The results are otherwise similar to the Exponential Interarrival Model.

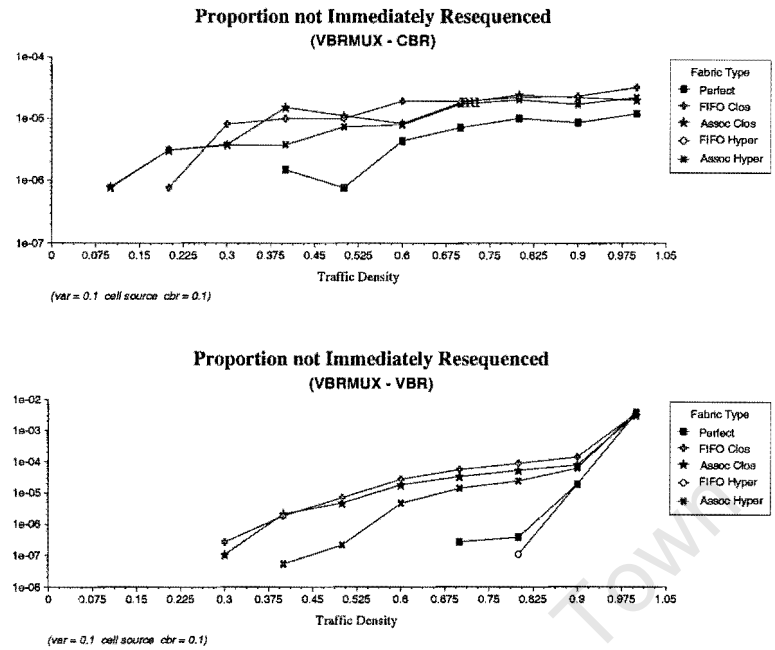


Figure 10.25: Proportion of Traffic not Immediately Resequenced (VBR model, cell source, low variance)

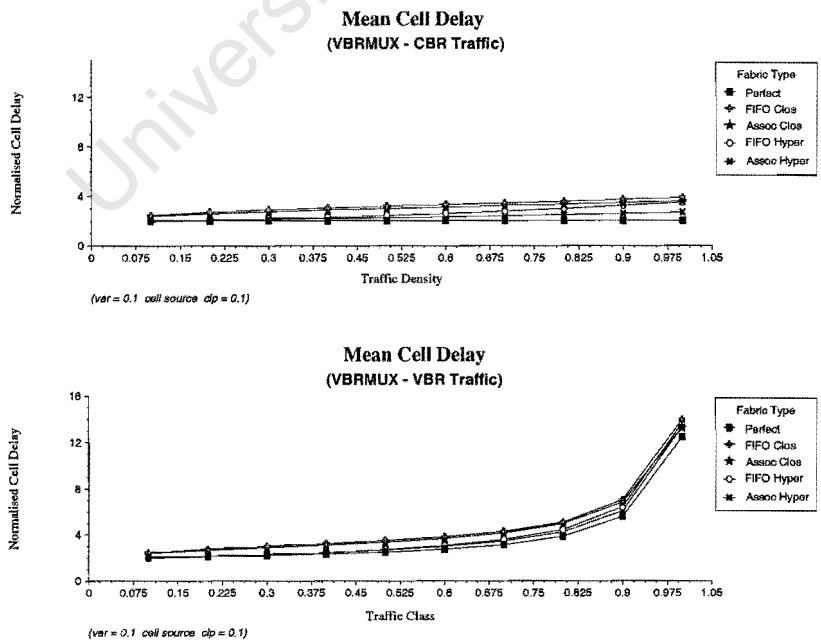


Figure 10.26: Mean Cell Delay (VBR model, cell source, low variance)

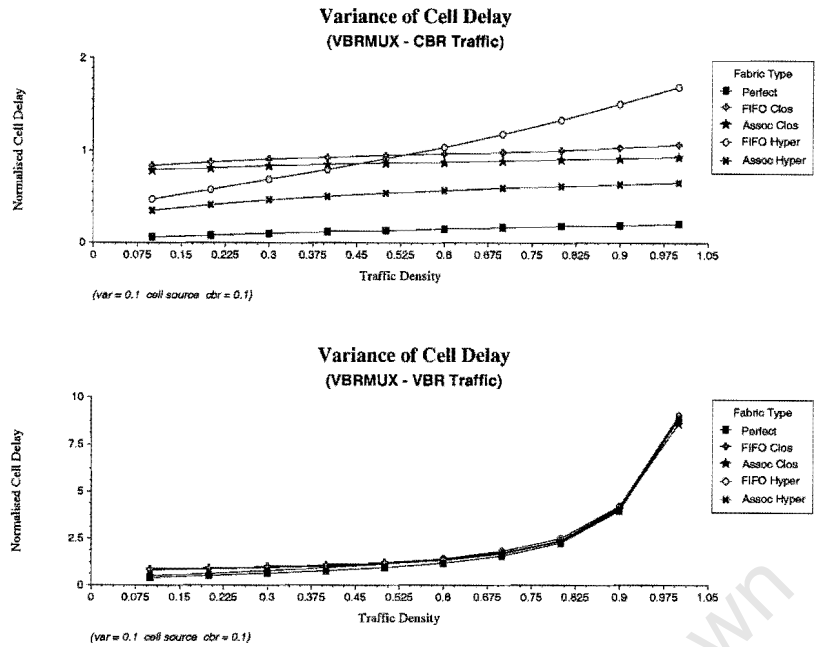


Figure 10.27: Variance of the Cell Delay (VBR model, cell source, low variance)

The Variance of the Cell Delay is more interesting. For the high priority class it is highly fabric dependent while for the lower it is largely output buffer dependent and almost invariant between the fabrics. The perfect fabric performs best followed in sequence by the Associative Memory hypercube and the Clos fabrics (the Associative Memory Clos fabric shows a slightly better response than the FIFO queued Clos fabric). The FIFO hypercube begins with almost the best performance of all of the fabrics and ends with the worst performance.

The peak cell delay shows the interesting result that time resequencing is performed more often on the Exponential Interarrival high priority cells than the lower priority cells (this is shown by the flat response at a delay of 18). The higher priority cells thus have a higher peak delay that the lower priority cells for the lower overall traffic densities. The lower priority cells are reasonably independent of the fabric type.

10.3.1 Effect of Increasing Variance on a Cell Source

After this baseline response was determined, a more vigorous exercise of the model and the fabric was necessary. However, due to time pressures the size the switching fabric was reduced to 64 ports and the Clos networks were not evaluated. The next model used a fixed traffic density of 0.5 but had a variance on this parameter varied from 0.15 to 1.5 (again, this is proportional to the traffic density). The time between changes in the variation of the rate was 200 normalised cell arrivals. The number of VBR traffic sources per input port was 16, the proportion of CBR traffic was 10%. The first result to be obtained was that at high proportional variances (1.5) a small CLP of roughly 4×10^{-6} was experienced for the VBR

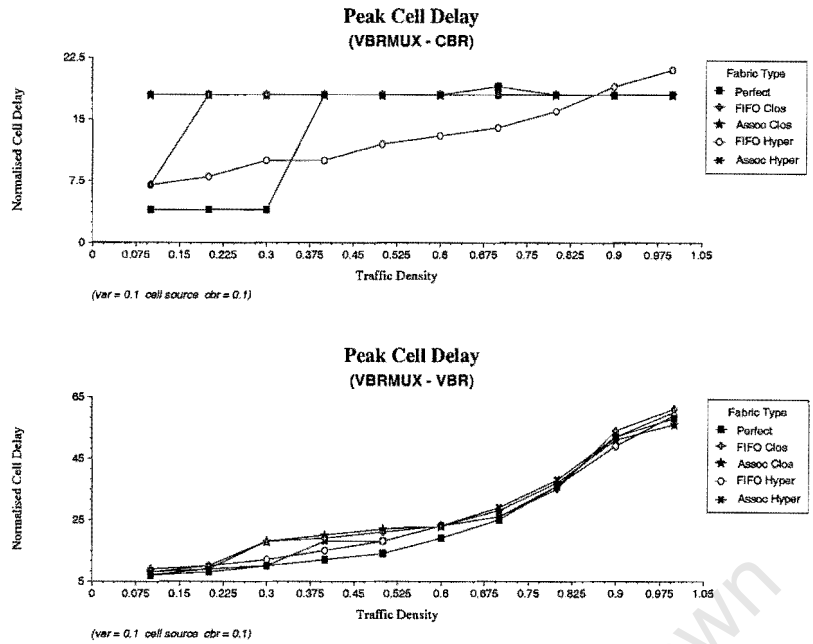


Figure 10.28: Peak Cell Delay (VBR model, cell source, low variance)

traffic class at the upstream buffer. There was no added cell loss in any of the other buffers (probably due to the low mean data rate despite the large variances) and the proportion of cells not immediately in sequence at the output was independent of the change in variance.

There was an effect in the Cell Delay parameters for the changing variances these are shown in figures 10.29 to 10.31.

The Mean Cell delay shows very little change in the high priority traffic class. In the lower priority traffic class, there is a gradual degradation in the performance. The Variance of the Cell Delay shows a similar gradual degradation in performance. The peak cell delay is not particularly influenced by the variance.

In interpreting these results, it should be born in mind that the simulation is for a simple cell based source. The behaviour of a bursty VBR data source may produce different results.

10.3.2 Effect of increasing burst length

The burst length is another parameter that has an effect on the performance of an ATM switch. Thus, a bursty data source was used with the same parameters as the variance model described in the previous section except that the proportional variance was held constant at 0.5. The burst length was varied from 4 to 40. This simulation exposed vulnerabilities in the resequencing module at the output of the switch. It has the effect of illustrating the importance of accurate traffic models in ATM switch evaluation since the performance did degrade very significantly with increasing burst length.

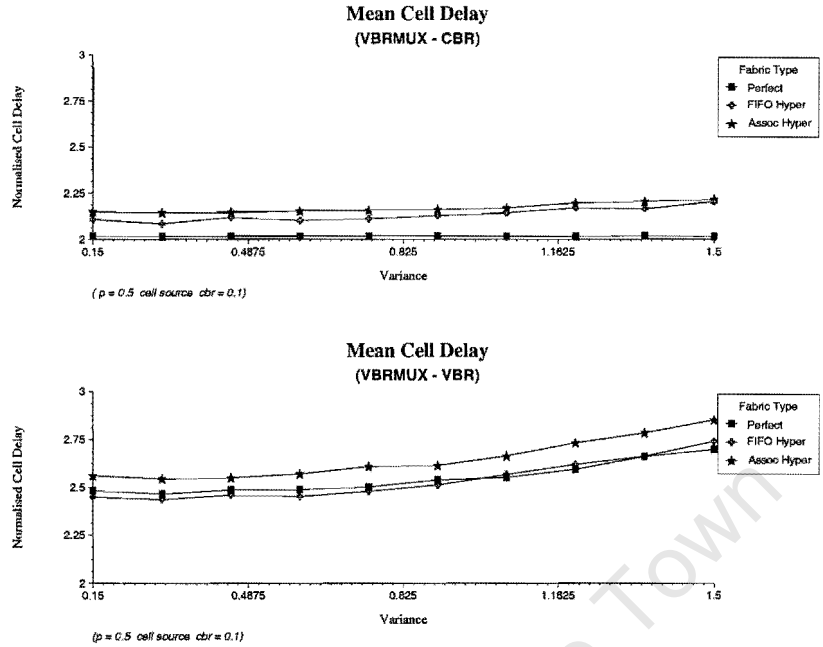


Figure 10.29: Mean Cell Delay (VBR model, cell source, p = 0.5)

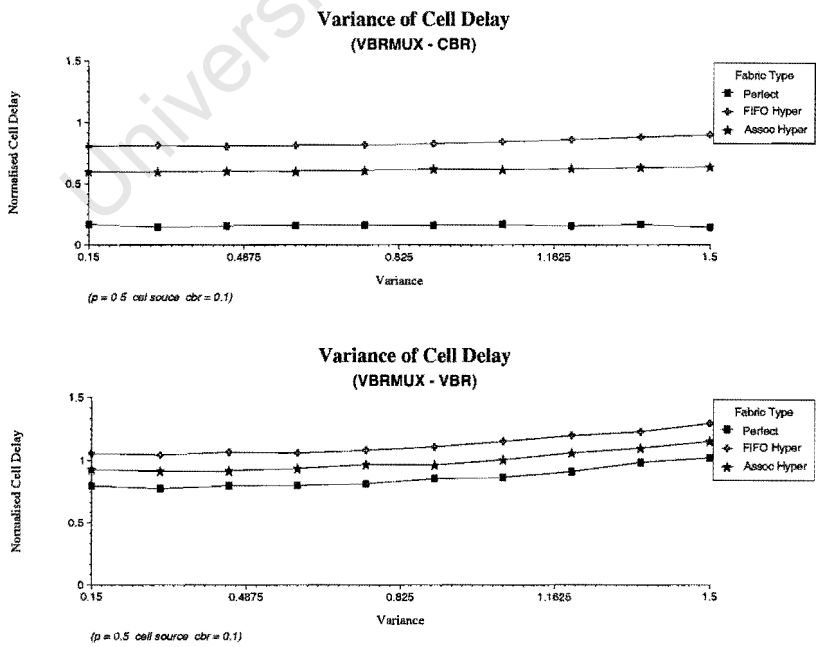


Figure 10.30: Variance of the Cell Delay (VBR model, cell source, p = 0.5)

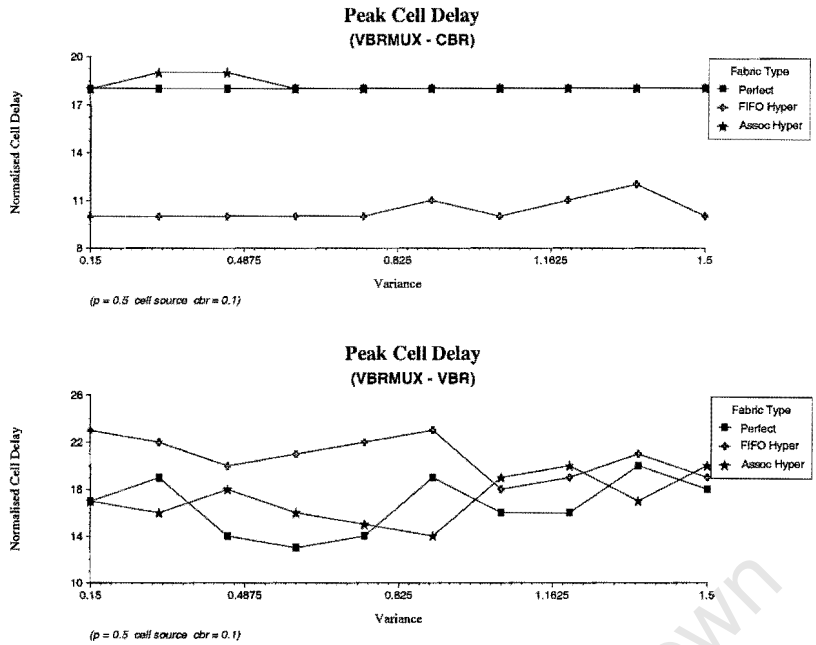


Figure 10.31: Peak Cell Delay (VBR model, cell source, p = 0.5)

	FIFO Hypercube	**Associative Memory Hypercube**
CLP Traffic	5.0×10^{-7}	4.5×10^{-7}
VBR Traffic	13×10^{-7}	4.0×10^{-7}

Table 10.1: Cell Loss Probability in Cell Switch Fabric (VBR model, bursty data source)

A large CLP was experienced both in the upstream access buffer and in the output buffers of the switch for the lower priority traffic class. This is illustrated in figure 10.32. There is only one graph for the upstream CLP since this is independent of the fabric type. An interesting result is that the CLP for the upstream network node (the access switch) is an order of magnitude higher than that of the downstream node (core switch). This illustrates the importance of the access switch in providing good performance for ATM switching. In addition, the was a CLP in both of the Hypercube fabrics shown in table 10.1 (the figures in the table are approximate).

The next graph shows the proportion of cells that have not been immediately resequenced (figure 10.33). This graph indicates that the output resequencer is unable to handle bursty traffic since the perfect model seems to be producing cells out of sequence. This could result from bugs in the simulation code or in a misdesign of the output resequencer. Unfortunately, due to time pressures the precise reasons could not be determined, it does serve to illustrate the importance of accurate traffic models in determining a switching system's performance. The statistic seems to be reasonably independent of burst length with the exception of the FIFO Hypercube which shows a dramatic degradation. However, it still performs better than the other fabrics.

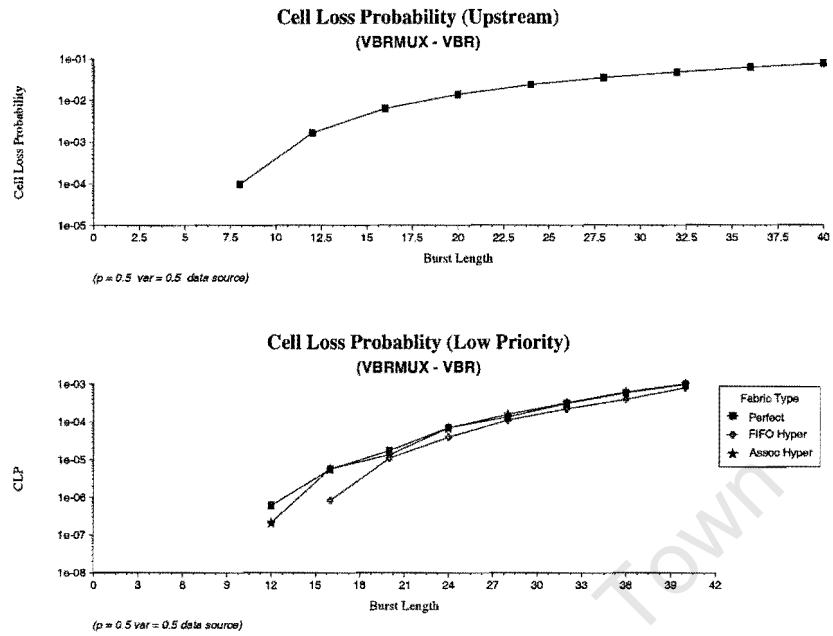


Figure 10.32: Cell Loss Probability (VBR model, bursty data source)

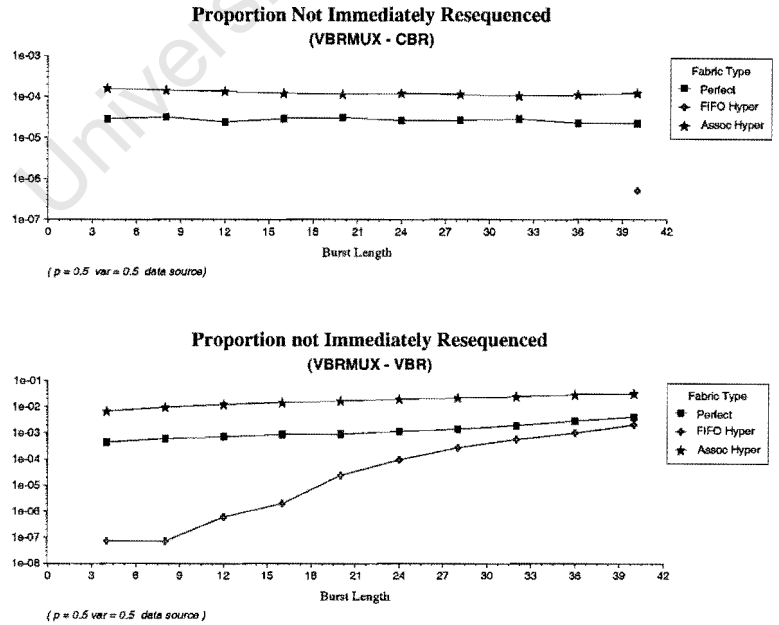


Figure 10.33: Proportion not immediately resequenced (VBR model, bursty data source)

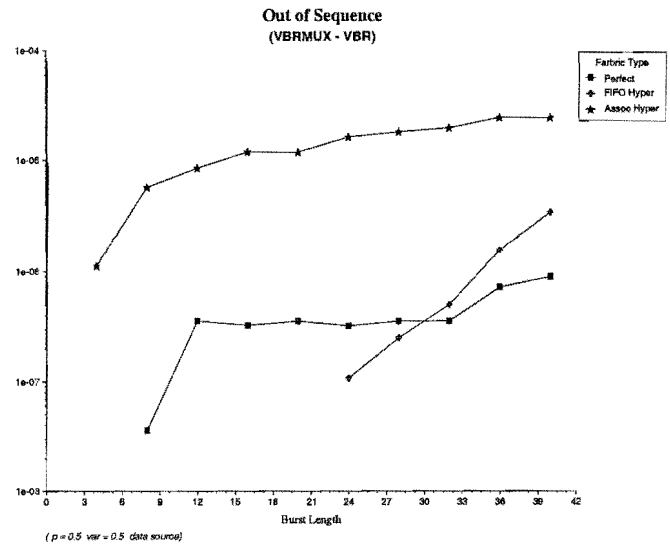


Figure 10.34: Proportion of traffic Out of Sequence (VBR model, bursty data source)

Possibly related to a misdesign in the output rescheduler is an increase in the out of sequence cells. As shown in figure 10.34 there is a steady increase in the number of cells out of sequence as a result of the increasing burst length. This result is problematic and shows that a re-evaluation of the output resequencing technique (or the simulation code that implements it) is necessary.

The cell delays show a similar degradation with increasing burst length. This is shown in figures 10.35 to 10.37. Interestingly, for the bursty case, the lower priority traffic shows a much greater dependency on the fabric type than in the simulations of simply increasing the traffic density. The FIFO hypercube and output resequencer combination produce a dramatic Peak Cell Delay due to a sudden mis-sequencing on the output (caused by the FIFO hypercube releasing a cell past its scheduled discard time). In most of the results, the higher priority traffic class is far less affected by the increasing burst length than the lower class. The exception to this occurs in the Peak Cell Delay with the FIFO hypercube.

These results serve to illustrate the extreme importance of the burstyness of traffic in relation to switch performance. It therefore also illustrates the pitfalls of making analyses without these considerations at face value.

10.3.3 Effect of Self-Similarity

The final parameter to be investigated was that of self-similarity. For this purpose, the same simulation was used as for the effect of burst length, however, the burst length was held constant at 20 and the Hurst parameter was varied from 0.5 to 0.95 in increments of 0.05. The results from the simulation were interesting in that almost no effect was found due to an increase in the self-similarity index. The only observed effect was an alteration in behaviour

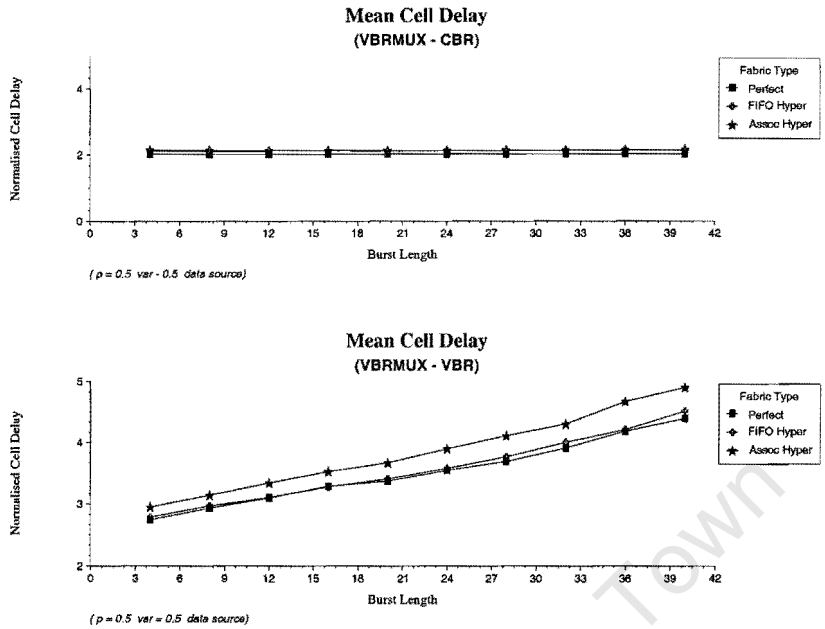


Figure 10.35: Mean Cell Delay (VBR Model, bursty data source)

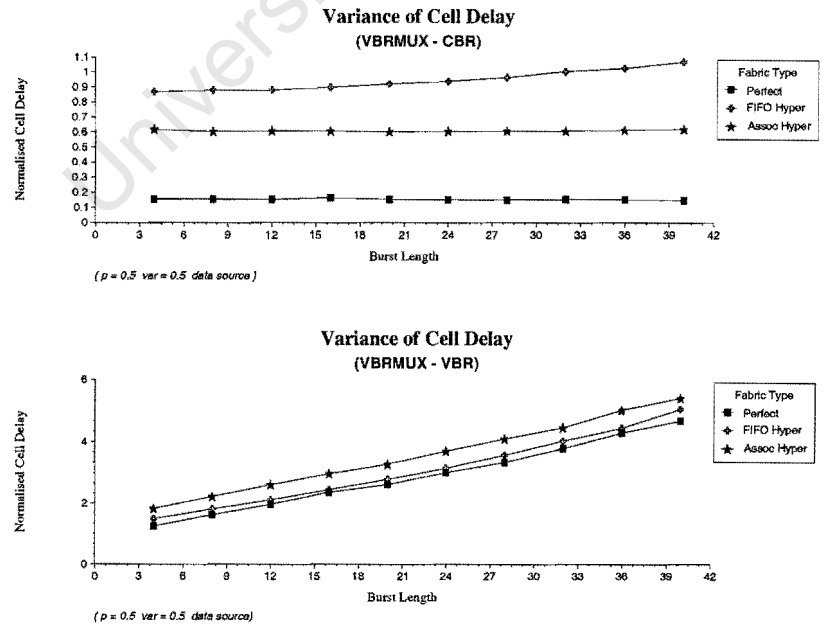


Figure 10.36: Variance of Cell Delay (VBR Model, bursty data source)

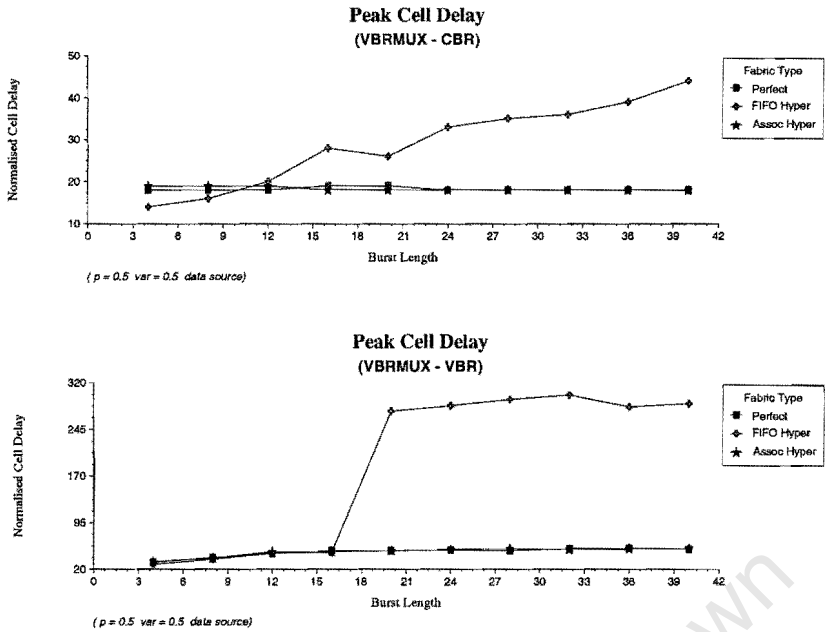


Figure 10.37: Peak Cell Delay (VBR Model, bursty data source)

of the FIFO Hypercube in the Peak Cell Delay. This is shown in figure 10.38. For the high priority traffic class the FIFO hypercube shows a sudden increase in the Peak Cell Delay at high self-similarity indices. For the low priority traffic class, all of the fabrics show a slight increase in Peak Cell Delay for very high self-similarity indices while the hypercube fabric suffers a synchronisation loss for one of its runs (resulting in a very large Peak Cell Delay that has been placed off graph).

These results for the self similar model are anomalous considering the recent emphasis placed on this aspect of ATM traffic. There are a number of possible explanations for it.

- The model's self similarity generator may not produce a sufficiently self-similar source. Or the model may in other ways be inaccurate to the real data traffic.
- The switching fabrics chosen to be studied may have a natural resistance to the effect (it would be interesting to construct other switching fabrics and determine the impact).
- The exact parameters supplied to the model may not best illustrate the effect of self-similarity.
- The effect of the upstream buffering (access switch), may be such that the self-similarity effect is minimised at the ATM core (however, no ill effects were felt at the access switch either in the simulation).
- Self similarity may be less influential on a switch's performance than believed.

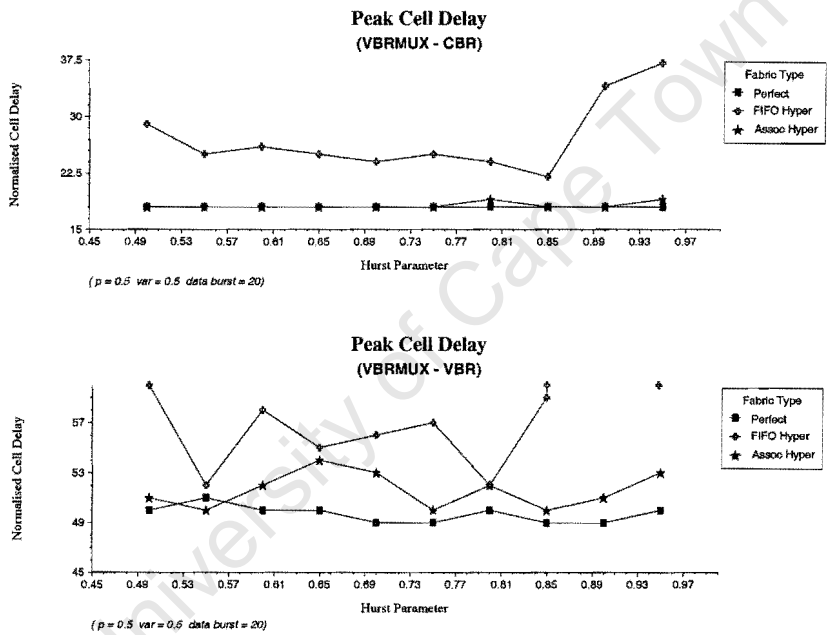


Figure 10.38: Peak Cell Delay (VBR Model, Self Similarity)

Chapter 11

Conclusions and Further Work

This dissertation has been approached by first showing the basic requirements placed on an ATM switch by the ATM architecture. An abstract model of an ATM switch was then developed which presented the functionality of an ATM switch in order to determine which parts of the switch may not be easily scaled to large ATM switch sizes. The two identified components were the switch management functions and the cell switch fabric. A simple architecture was presented which could allow the parallelisation of the switch management functions.

Aside from this consideration the cell switch fabric is the primary component that limits the scalability of large scale switches. Therefore most of this dissertation is concerned with that component. Fundamental statistical constraints imposed on ATM switches make output buffering a highly desirable feature. However, pure output buffering presents enormous difficulties when contemplating the construction of a large scale switching fabric. Input buffering results in an automatic reduction in the throughput capabilities of the ATM switch when FIFO queueing is used. In light of these constraints and after an overview of existing space division switching techniques, the hypercube switching fabric was presented. This fabric uses a novel approach to ATM switching that nearly attains the minimum complexity for space division switching. Since the techniques examined do not guarantee cell order, an output scheduler and output buffering system for the simulation were designed.

The work presented a number of different traffic models. The most simplistic was a simple exponential interarrival model with even output port distribution. Next in complexity was a locality of reference (not to be confused with hotspot) model. Lastly, a complex VBR model was presented which allowed the multiplexing of VBR and CBR sources. In addition, bursty data traffic could be modelled and the data rate envelope could be made to exhibit self-similarity by the variation of a Hurst parameter.

This work was brought together in chapter 10 in which the various models were used on various combinations of the fabric. These results showed the good scaling of the hypercube

fabric as well as the improvement in performance that resulted in using the associative memory. The associative memory leveraged a small performance benefit for the Clos network. Other results obtained in this chapter showed the dominance of the Output Module in determining the switches performance.

Interesting results were obtained from the locality of reference model. All of the fabrics except the perfect fabric experienced a degradation in performance for high locality of reference. With the VBR model similarly interesting results were observed. Increasing variance of the traffic rate envelope resulted in a decrease in performance of all of the fabrics. Similarly, increases in the burst length for the model produced a pronounced degradation in performance. Interestingly, varying the self similarity index did not produce a corresponding change. An interesting observation was that simply by providing two different classes of traffic at the output and scheduling the one in preference to the other, the higher priority traffic class (which was well-behaved), was largely insulated from the effects of the other traffic. This illustrates the importance of the cell output scheduler in ATM switch performance.

In general, the results tended to show the robustness of the hypercube fabric when using the associative memory queueing (although some problems were experienced with the output scheduler and output buffering system). The hypercube was shown to out-perform a Clos network of similar technology and indeed approached the performance of an ideal fabric for many of the measured metrics. The FIFO queued hypercube showed anomalous behaviour when confronted with certain traffic patterns, this behaviour was absent in the associative memory queued variations. All of the results tended to show the importance of realistic traffic models when evaluating an ATM switch's performance.

Possibilities for future work have also been identified:

- An ABR traffic model could be added to the input model test suite. This model could explore the relationship between the various flow control schemes and latency within the network.
- The issue of fault tolerance and redundancy in the hypercube could be explored. A brief broaching of the topic is given in appendix A.
- Multicasting functionality could be added to the hypercube model. The hypercube shows excellent capabilities in terms of its adaptation to multicast functionality. This is briefly discussed in appendix B.
- The associative memory leverages only a small performance benefit when used with Clos network. A possible reason for this is the fact that only one aspect of its behaviour was utilised (that is, the ability to give retarded cell an advantage at later points in the network). It does not take advantage of the fact that alternative routes may be

more quickly exploited if the fabric provides them. It would be interesting to use the associative memory from the first stage in the Clos network (at the moment this uses a simple random distribution mechanism) and possibly combine it with dilation in the network. The associative memory is likely to leverage a greater benefit under those conditions.

- The existing test models could be used on a larger variety of switching fabrics. For example, it would be interesting to observe the response of the knockout concentrator to the locality of reference model. In addition, the various banyan networks would be interesting to study in the light of these models.

Appendix A

Reliability and Redundancy in the Hypercube Fabric

This appendix is intended to provide an overview of the advantages of using a hypercube interconnection for a cell switch fabric. No other work or simulation has been performed on this topic in this thesis, however, since there are many advantages to this interconnection in order to enhance reliability, it is advantageous to provide a brief overview of them. This is particularly important since a large scale switch would most often be found in a Central Office environment and hence would be required to be almost completely reliable. It would, in addition, provide interesting further research.

The hypercube fabric consists of two main components, the Switching Element planes and the Routing Elements. The first proposal is to use a Hamming Code [71] pp. 590-591 on the d planes of Switching Elements. The overhead bits are placed on a separate h switching planes. Figure A.1 shows the addition switching planes added as a result. This is an acceptable overhead since if h is the number of parity bits and there are d switching planes then $h = \lceil \log_2 (d+1) \rceil$. Thus, the number of parity bits is roughly $\log_2 (\log_2 (N))$. The hamming code will ensure that if any switching plane is disabled (or any element on a switching plane), then the missing information can be reconstructed by the REs. This would also allow for hot swapability of switching planes since the fabric could continue with any arbitrary plane removed. Thus, this modification would allow for any one failure in a switching plane (or multiple failures in any one plane) at a low cost.

The next element in the routing fabric is the RE. Since every RE interfaces with an output port, internal redundancy must be provided within every RE to guard against a port failure. This is a matter of careful design and "1+1" provisioning within the RE itself and will not be discussed. However, let us assume that an RE has failed. What techniques can be used to localise this failure to the RE in question and allow the remaining REs to successfully route the cells to the correct outputs of the switching fabric? Redundant routing can be observed

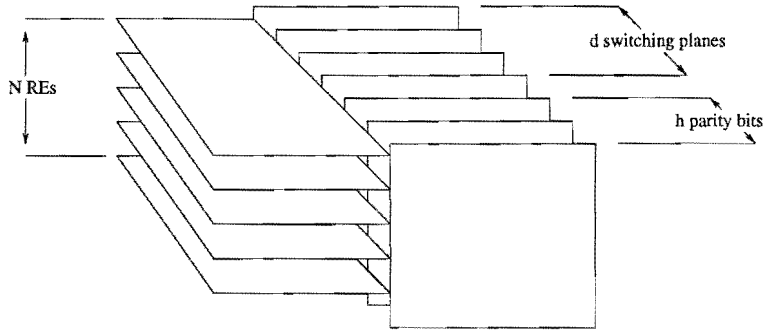


Figure A.1: Hamming Code overhead on switching planes

by noting that if the neighbouring node to a failed node receives a cell (and is aware of the node's failure), there are three possible destinations for the cell.

- The cell may be destined for the failed node as the final node. In this case the cell may be discarded since the output to which it is destined has failed.
- The cell may not be destined for the failed node, but there may be a valid shortest route through the failed node. In this case there will always be another valid route and the cell may be rerouted.
- The cell may not be destined for the failed node and there is no valid shortest route through the node. In this case the failure would not affect the operation.

The first case is hopeless since the destination itself has failed and in the last case the operation of the switch would not be affected. The second case be very easily handled by supplying a bit mask of correctly functioning SEs. When a cell is received, the bitmask determining the valid destination nodes is anded with the valid destination node mask. This technique will always work so long as no node suffers the failure of two neighbouring nodes. The probability of this is small, however, and becomes even smaller as the size of the switch grows since the number of neighbouring nodes grows only logarithmically with the number of ports in the switch. In addition, if it has been determined that a node has failed, no cells will be accepted from it and will be assumed to be suspect until such time as the node is determined to be malfunctioning.

Last to be discussed is the mechanism by which a node determines that another node has failed. The simplest technique is to provide for an indication of failure between nodes. When circuitry on a node determines a failure, it pulls all of the bits to the SE planes high for example. This signal could be used by a neighbour to determine the node's failure. This approach is functional. However, the error in the SE might be subtle enough (or catastrophic enough), that the error detecting circuitry becomes compromised. In this case a neighbouring node fails to advertise its failure, but does not function correctly.

This case can be handled through the use of the redundant cycle provided between the routing elements when a cell is received from the IM. The proposal is to have a $3(d^2-1)$ cycle at a slower rate within the redundant cycle in order to provide a technique that can detect the failure of an RE even if it is not advertised.

In the first cycle, a random sequence of bits with a parity code is sent to the neighbouring node destined for a particular second neighbour (one which can be reached through only one neighbour). The neighbouring node routes this to the second neighbour in the next cycle (preferably using as much of its normal circuitry as possible for the operation). In the last cycle, the property of the crosspoint banyan is used that allows it to interconnect any collection of nodes as long as they differ by the same bits. This allows the second neighbour to communicate directly back to the first node the original data that was sent to it in the final cycle. The sending node then inspects the data it originally sent and ensures both that the data is the same and that the parity bits are unaltered. If this is the case, then both the neighbouring node and the second neighbour are functioning correctly. If it is not, however, then either:

- The first neighbour is malfunctioning, or
- The second neighbour is malfunctioning, or
- The node itself is malfunctioning.

We assume that the switching plane cannot be malfunctioning because of the Hamming code overhead placed on it. If the node itself is malfunctioning and its own circuitry has not determined this, then the neighbouring nodes will determine its failure and neither route cells to it or accept cells from it.

Whether it is the first neighbour or the second neighbour that is malfunctioning can be determined in successive cycles. If the result from every cycle returns as a failure, then the first neighbour has failed. If the result from only one second neighbour fails, then that second neighbour has failed. However, since only the case where a first neighbour has failed is considered in the routing it would be sufficient to wait for more than one failure to occur in every cycle.

Although this suggestion has been neither simulated nor analysed, it presents an interesting approach to determining failure in the hypercube fabric and provides for a large degree of fault tolerance within the switch. It is also reasonably quick, in a 1024 port switch, the failure of a single node can be determined in $8.1ms$. It does this with the overhead of two extra buffered ATM cells and control logic. This and other methods of attaining fault tolerances in the hypercube and other fabrics provide a fertile ground for research.

Appendix B

Multicast in the Hypercube Fabric

This appendix is intended to provide suggestions for a technique whereby multicast functionality may be incorporated into the hypercube fabric. The suggestions are neither simulated nor analysed but are provided to form the basis of continuing research and to demonstrate the possibilities for the simple and natural implementation of multicasting presented by the hypercube fabric.

There are two primary techniques used to implement multicasting in an ATM switch. Once technique used is to place a copy network in the switch which precedes the cell switch fabric itself [46], [78], [50]. The network is used to duplicate cells before the switching fabric handles them. The advantage of this technique is that it requires no alteration to the cell switch fabric. The disadvantage is that an entirely new network with its own characteristics must be added to the switch. In addition, where the amount of multicast is large, the duplication performed by the copy network could inundate the switching fabric.

A preferable technique is to integrate the multicast function into the switching fabric itself [73], [11], [79]. Certain fabrics self evidently are optimal for this use. For example in a shared medium switch the multicast is simply performed by allowing more than one OM to read of the bus in the same time slot. In a shared memory switch the cell may simply remain buffered until such time as all of the copies are passed to the OMs. However, other switching fabrics also have this desirable feature. For example, the banyan network can be used to send a cell to both of its outputs [15], leading to a natural copy function (throughput may become problematic).

One of the primary problems that must be addressed in a multicasting scheme is the way that the multicast data is represented in the switch. The essential problem is, how do you represent the very large number of possible permutations that are presented by the number of output ports. A simple bit vector could be used for a small number of ports, however, this is self evidently impractical for a larger number. A potential solution to this problem is presented by the use of radix trees (presented in conjunction with their use with Banyan

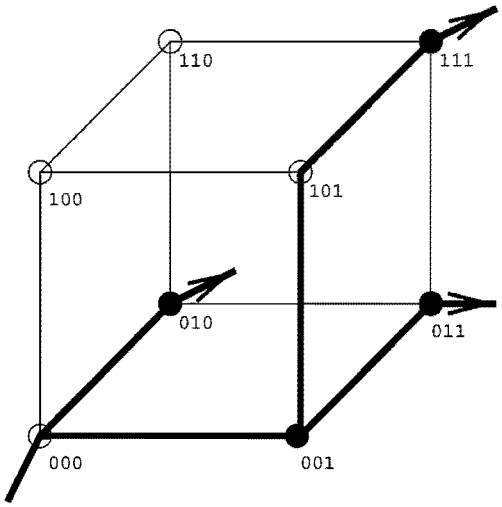


Figure B.1: A multicast connection in the hypercube

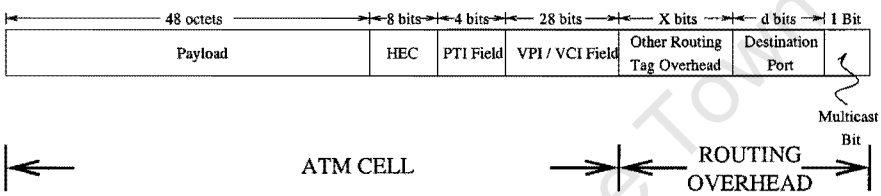


Figure B.2: Multicast overhead in the cell

networks in [41]).

The proposed scheme for ATM is to map a binary tree over the hypercube fabric to represent the multicast function (This is shown for a three dimensional cube in figure B.1). The SEs would remain unaltered in the fabric. The addition to be made would be in the hypercube REs, OMs and IMs. The proposed addressing scheme for the ATM cell is shown in figure B.2. When a cell is received by an IM it is identified as belonging to a multicast VC. Two copies of the cell are made and are sent to the first nodes in the multicast tree (this is done without any alteration to the routing mechanism). The cell VCI/VPI field is concatenated together to form an internal multicast VC number assigned during the connection admission control phase. When the cells reach their destination they are resequenced (in order to reduce the overall lack of sequence at the furthest leaves of the multicast tree) before being sent to the output to which they are destined (at this point the OM will be required to insert the correct VCI/VPI number based on the internal multicast VPI/VCI number). If the node is not a leaf node of the multicast tree, one or two copies of the cell are made, the destination port number of their output nodes are inserted and the internal multicast VC number is modified in exactly the same way that an ATM VCI/VPI value is manipulated for each cell. (The motivation for doing this is to ensure that no global multicast VC table is needed, there need only be spare multicast VC values available at every node that the multicast tree must traverse).

This shows the natural suitability of the Hypercube fabric to multicast switching. The technique of only ever making two copies of a cell at any stage of the process fulfills two objective of allowing a reasonable latency to be maintained on the individual ATM cells while spreading the multicasting load evenly throughout the ATM switch. A performance evaluation of this type of traffic and a more thorough study of this proposed scheme (as well as any other that may be applicable to the fabric) would provide an interesting basis for further research.

Appendix C

Overview of Integrated Circuits

This appendix is intended to provide a very basic understanding of integrated circuits in order to allow chapter 8 to be better understood. It is intended as a very basic review appendix only and does not necessarily absolutely accurately reflect the techniques used in Integrated Circuit manufacture (for a better overview of this see [55] and of the process used see [57].

An integrated circuit is built by a series of processes that essentially consists of first depositing (through doping or by crystalline growth) a material on the surface of a wafer of silicon. Then, the surface is treated chemically with a substance that reacts in a different manner when exposed to ultraviolet (and shorter wavelength) light. Where a combination of the surface, the chemical and ultraviolet light occurs, it produces a surface that can be removed with a chemical "etch". Thus, the features that are to be placed on the silicon surface are produced on a mask which restricts where the light can expose the wafer surface. (This is similar to the process used to produce a printed circuit board, but on a much finer scale). The design of an integrated circuit involves (either directly or through a silicon compiler) the production of a number of masks that produce, when combined with alternating sequences of doping, growing, depositing (for metal layers) and etching the required circuit. To illustrate this consider the side and mask views shown in figures C.1 and C.2 of a n-channel MOSFET

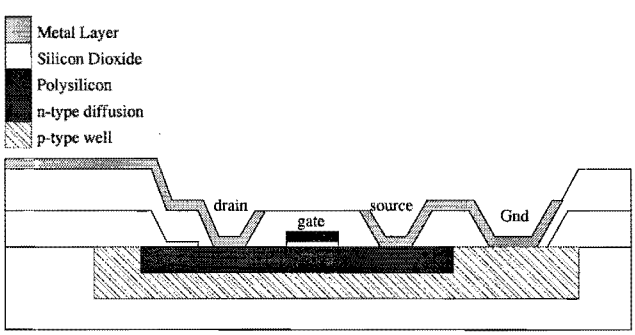


Figure C.1: Side View of an CMOS n-channel MOSFET Transistor

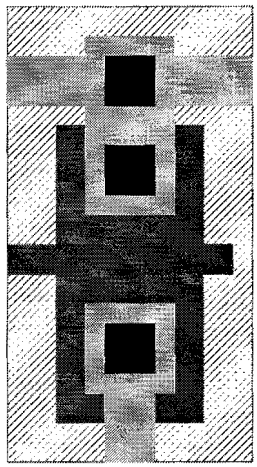


Figure C.2: Layout view of a CMOS n-channel MOSFET Transistor

transistor. The description of the process is much abbreviated, but serves to illustrate the process by which the transistor is created.

The transistor is first formed by the diffusion of n-type ions into a p-type well on a wafer surface. This produces areas of n-type channels on the silicon surface (a complementary process produces p-type channels within n-type wells). On top of this, an extremely thin layer of silicon dioxide is grown. The SiO_2 is then removed with an etch in areas where direct contact should be made with the diffusion and well areas. On top of this very thin layer of SiO_2 a layer of poly-crystalline silicon is grown and etched to form the MOSFET gates. Polysilicon is a type of silicon which is capable of conducting electricity, (however it has a reasonably high resistance compared to metal). Once the polysilicon regions have been formed, another thicker layer of SiO_2 is formed on top of which a metal conduction layer (normally aluminium) is placed. Thereafter a number of metal layers can be placed insulated from each other with layers of SiO_2. Note that wherever a connection between metal layers, or between metal and polysilicon or metal and wells and diffusion areas is desired a *contact cut* is made in the SiO_2which allows the two layers to connect. This is illustrated in figure C.1 at the position where the drain and source make contact with the n-type diffusion.

A MOSFET transistor works by restricting the flow of electrons through the channel, this restriction is generated by the electric field formed between the gate and the well. Since it is related to the electric field it is a voltage driven effect. There is no conduction between the gate and the channel layer. In addition, the gate is separated from the channel by a very thin layer of silicon dioxide. Since the capacitance of a parallel plate capacitor is given by $C = \epsilon_0 \frac{A}{d}$, (where A is the surface area of the plate, d is the distance between them and ϵ_0 is a dielectric constant) the capacitance of the gate to source of a MOSFET transistor is high. Therefore, the primary speed restriction of a CMOS circuit comes from the impedance of the output driver and the gate capacitance of the next stage (in deep submicron designs, the conductors become small enough that propagation delay becomes the constraint on circuit

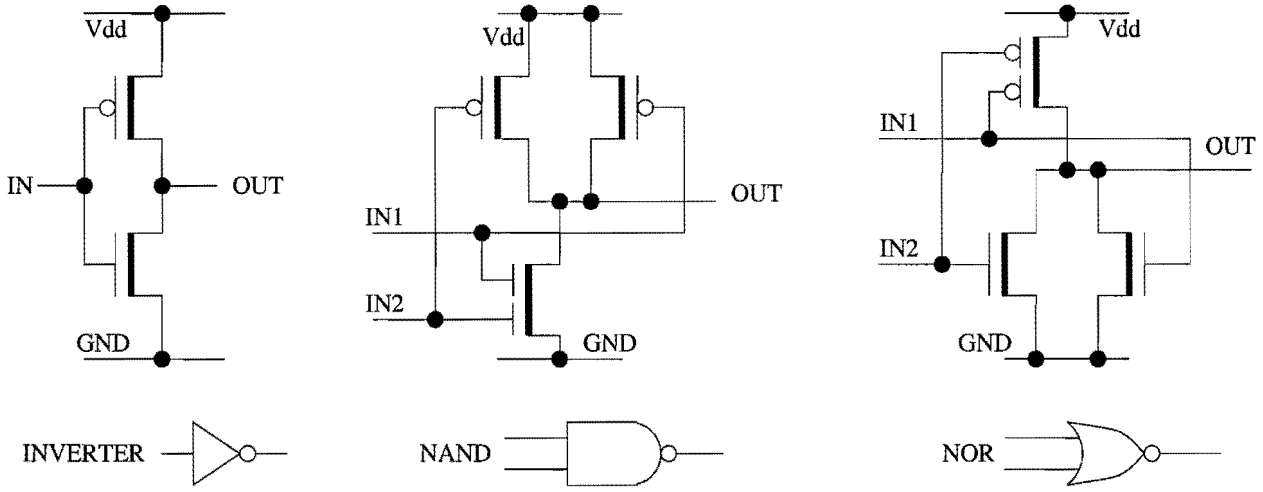


Figure C.3: Basic CMOS gates

speed, however).

In order to understand how CMOS circuitry is constructed, consider the logic gates and circuits shown in figure C.3. As shown in this figure, CMOS gates do not have a passive pullup system (such as in nMOS) and have symmetrical output drivers. This means that the pull up section of the output driver and the pulldown section operate in logical opposition (note that in the figure a slightly non-standard symbol for the p-channel MOSFET is shown for the p-channel MOSFET). As can be deduced from the figures, a useful property of CMOS circuitry is that in steady state, the output of each stage feeds directly into a pair of gates in the next. A useful result of this is that CMOS has no current draw when there is no transition on its inputs. This occurs since as the inputs to the logic elements are swinging in value, there is conduction through both sets of transistors from the positive to the negative supply rail. Thus, the faster CMOS is clocked, the higher the current drain.

C.1 Lambda Scaling Rules

A critical parameter of an silicon fabrication process is the smallest resolution that the process is capable of fabricating. The reasons for this are simple. The capacitance of a parallel plate capacitor is $C = \epsilon_0 \frac{A}{d}$. Let us call the smallest resolution that a process is capable of reaching 2λ, then $A = 4\lambda^2$. Since the primary restriction in speed of a MOSFET transistor is the gate capacitance, a linear reduction in the smallest process resolution represents a large increment in speed (Modern processes are in fact restricted primarily by the propagation delays in the chip itself due to transmission line effects). nMOS and later CMOS designers were faced with a time consuming re-design with every increase in process resolution. Thus, Carver Mead and Lynne Conway [55] introduced the *lambda* scaling technique.

In this technique, the minimum process resolution is assumed to be 2λ and the registration

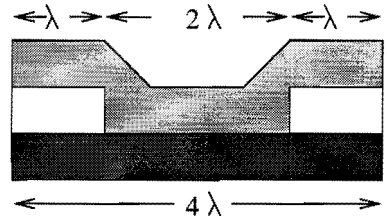


Figure C.4: Contact cut illustration of *lambda* scaling rules

resolution is λ. (Registration is the accuracy with which each mask can be placed relative to the other masks, also taking into account distortions on the masks surface). Then, a set of rules can be developed that are generic to most fabrication processes. Thus for example, for a via to be made between the metal and the polysillicon layers, one can say that the minimum length of the contact cut is 2λ in order to be guaranteed that there is in fact a break in the glass. Since the registration resolution is λ, however, it is obvious that both the glass and the silicon layers must overlap the cut by λ, thus their minimum dimensions must be 4λ on a side. This is illustrated in figure C.4. The two great advantages of lambda scaling rules is that they are generic and a design conforming to them will work with any supporting fabrication process. In addition, many automated tools will automatically enforce lambda rules when the design of an IC is undertaken. This greatly increases the probability that a mask produced for fabrication will be correct.

The rules used in this thesis are the MOSIS lambda scaling rules. The IC designs were laid out according to these rules and hence should work for any fabrication process that supports them. Thus, although the simulations were conducted for $2\mu m$ CMOS technology, a faster chip could be produced simply by changing the fabrication process.

Appendix D

Commercial ATM Switches

This appendix is intended to provide information on the functions and types of ATM switches commonly found in the market. It is by no means exhaustive or complete, but it provides an idea of how a real ATM network would be designed and allows one to speculate on the possible roles on future large scale ATM switches. Some of this material is summarised from [19] and from the authors general reading. However, before a discussion on the different types of ATM switch is entered into, some of the economics of the ATM switch market should be discussed.

ATM is a complex network architecture. In fact it is probably the most complex architecture in existence. In addition, many of it capabilities have not yet been standardised by the ATM Forum or the ITU-T, and of course released standards are often complex resulting in an appreciable lag between their specification and the necessary software upgrades appearing for a particular manufacturers switch. This has three implications. The first is that switches from different vendors are likely not to interoperate using the more attractive features of the ATM standard, the other is that many vendors implement "pre-standard" versions of software that represent their best guess as to the eventual form of the standards (or simply a quick fix way to get a feature to work). An example of this is the FORE systems SPANS protocol which allowed for ATM switches to obtain routes to each other before the advent of the PNNI protocols. Fortunately as the ATM standards mature and are released this is becoming less of a factor. The other implication of this is that owning an ATM switch requires constant upgrades and represents a large maintenance burden. In fact, it is tempting to believe that the latest push towards Gigabit Ethernet and IP switching represents an apprehensive response to precisely this factor of ATM ownership.

Another word should be said of the difference between the ATM Forum and the ITU-T standards. The ATM Forum represents the switch manufacturers and (generally private) end users, while the ITU-T represents the telcos. The ATM Forum thus tend to standardise protocols at the PNNI and Private UNI whereas the ITU-T will standardise on the Public

UNI and Public NNI. Thus, the complexity of the ITU-T protocols tends to be much higher (since it must take into account security and accounting, this is normally much less of a factor in private ATM networks) and are released much later than the ATM Forum's. Thus for example routing in private networks is by now well standardised with the introduction of the PNNI phase 1 and phase 2 protocols. However, carrier routing solutions are proprietary where they exist. Unfortunately it must also be said that the ITU-T also seems to be beset with OSIfication and does not seem to realise the impetus that is required to ensure that ATM is a market place winner.

The early adopters of ATM technology were corporations experiencing large performance problems with their LANs and research laboratories interested in experimentation with the multimedia potential that ATM alone of the asynchronous architectures is fundamentally designed to provide. Thus, by far the largest market in terms of unit volume for ATM switches is in LAN switches and Workgroup switches. (A workgroup switch is one which emphasis the reasonably priced interconnection of ATM hosts, while a LAN switch performs the switching for both ATM LAN attached hosts and legacy LAN networks). Indeed, this emphasis on LAN switching has possibly been to ATM's detriment since the ATM switches have been used to handle best effort services where in fact if it is to deliver a "killer" technology it will be in multimedia networking. In fact the emphasis on best effort traffic was so strong that many early ATM switches only supported CBR and UBR traffic. With the introduction of LANE (LAN Emulation) and MPOA (Multiprotocol over ATM) these switches have become more capable and allow easy inter-operability with existing LAN protocols (such as IPX/SPX and IP). ATM now faces a large challenge in this area due to the introduction of Gigabit Ethernet and Fast Ethernet switching. In the consumers minds, ATM is generally perceived to be a complex and expensive technology and in addition, the vast majority of LAN traffic is still best effort (users seem generally content to download their multimedia before playing it or simply inserting a CD-ROM). Thus, unless the AAL2 is standardised quickly and video on demand becomes the "killer" application for ATM, ATM is likely to lose relative market share to these technologies.

The current implementors of ATM will most likely be the telecommunications service providers who are faced with the difficulty of running separate voice, video, frame relay and SMDS networks and have much to gain by implementing these services over a core ATM backbone with access switches providing for the differentiation of these services. It is likely that if ATM cannot compete in the LAN market against the simpler best effort Gigabit Ethernet and IP switching that service providers will become the biggest implementors of ATM for the management advantages that it provides them.

D.1 ATM LAN Switches

The ATM LAN switch market was the first to mature. The essential reason for the demand for ATM LAN switching was that many corporations were directly experiencing the inadequacy of their existing Switched Ethernet and Token Ring networks and required urgent measures to provide an acceptable level of performance to the network users (their employees). This become especially critical with the advent of the client server model of computing and the widespread use of PCs and the more and more demanding nature of the applications they support.

LAN switches are presented with a far simpler task of switching than Access Switches and Backbone Switches. The reason for this is that they do in fact operate over the local area. Thus, the latencies which make flow control very difficult in a large network are less problematic. In addition, as already stated, most of the traffic that they need to carry is in fact legacy best effort traffic. This results in a number of implications, LAN switches in general implement simpler traffic management and buffering schemes than other ATM switches and also require smaller buffer sizes since the flow control across the network will generally be more accurate. Also since businesses are generally unconcerned with the exact nature of the traffic their LANs carry these switches tend to offer simple network management systems. Also LAN switches will only support the private network and user interfaces.

In terms of interface cards, the LAN switches tend to support the following:

- Legacy LAN interfaces, especially ethernet and token ring.
- ATM connected server interconnection at OC3 (155Mbps), or OC12 (622Mbps).
- Inter-switch interconnection at OC3 or OC12.
- WAN interface connections at T1, E1, E3, T3 and OC3c.

Slightly different in turn to an ATM LAN Switch is the ATM Workgroup switch. These switches are designed specifically to interconnect local ATM hosts. They thus don't support WAN interfaces and require interconnection with a LAN switch for these services. Due to the fact that fewer interfaces need to be supported, Workgroup Switches can normally be designed in a simple form factor with a fixed configuration of interface cards. This allows their price to be reduced on a per port basis.

Representative companies in the LAN switch market are FORE systems with the ASX-200WG and LE155 workgroup switches, the ASX-200BX LAN switch and the LAX20 as a bridge to existing LANs onto an ATM based backbone. Lastly they provide the ASX-1000 enterprise backbone switch. FORE tends to be a very dextrous player in the ATM market,

neatly splicing pre-standard features with quick implementation of the ATM standards. They also support SNMP based switch management as loadable modules to the HP-OpenView network management system. Another representative company is Cisco corporation. This company's traditional market is routers rather than switches so their entry into the ATM market is interesting since they run the risk of cannabalising their router market with ATM products. Their strategy it known as "CiscoFusion" which they see as a seemless integration of switching and routing. Unlike FORE they are also very active in the router market and thus their middle end solution consists of switched Ethernet and faster routers (CDDI/FDDI,100 base T, OC3). They have bought into the workgroup and edge switch market through a joint venture with NEC and BBN-Tandem). In addition, they support a hybrid ATM / Frame Relay switch, the StrataCom IGX. In terms of switch management they support SNMP with a plug in for HP OpenView and for homogeneous Cisco networks they offer CiscoView and AtmDirector management software.

In terms of ATM switches, it is unlikely that the LAN market will scale to switches offering a large number of ports. Indeed, most LAN switches on the market support sixteen ports at OC3 (155Mbps) speeds. Thus, they tend to implementable as a shared medium or shared memory type switch. And since both of these architectures are reasonably inexpensive in todays technologies and provide for nonblocking performance and almost automatic support for multicast within the switch fabric itself this trend is likely to continue. In the enterprise backbone switch (such as the ASX 1000), the switches will tend to be constructed through placing a simple non-blocking space division fabric before the shared memory architecture. As multimedia becomes more prominent in the business environment and the per-desktop bandwidth becomes appreciable (and the number of desktops increases), it may however become possible that large ATM switches might be required in the enterprise backbone.

D.2 ATM Access or "Edge" Switches

An ATM Edge switch is differentiated from an ATM LAN switch by its placement in the network. Rather than servicing a floor of a building or a Campus, an ATM Edge switch is placed at the access point to a carriers WAN ATM network. Thus, an ATM Access Switch must support both private and public network views on both the interface and signalling side. In addition it must have a larger buffering capacity in order to accommodate the high latencies of the ATM WAN. In order to make effective use of these buffers, the traffic policing and flow control functions must have a higher level of sophistication than in a LAN switch. Also adding to their complexity is that they are required to implement a more complete network management system and they will be the first ATM switches to implement usage accounting since the service provider will need to obtain this information for billing purposes. They tend to support a larger variety of WAN interfaces than LAN interfaces, in addition,

they are normally required to be more fault tolerant than a LAN or Workgroup switch. In addition to these requirements, an ATM edge switch will often include a service multiplexer for voice and video support.

It should be obvious that an access switch is far most costly than a LAN switch, both in hardware terms since the buffering and traffic management requirements are more stringent on an access switch, and in software terms since both the Private and Public network view will need to be supported. The higher demands for reliability also result in an increase in cost. In terms of the switch fabric of an access switch it is likely that it will use a similarly conservative approach to switching to a LAN switch. The reason for this is that even though the number of ports in an access switch is likely to be reasonably, the access circuit bandwidth is likely to be far more conservative (For example E2 or E3). Thus the aggregate bandwidth that the switch is likely to handle is likely to remain small. Another way of viewing this is to reallise that an Access switch is likely to have a single SDH link (likely made failsafe through the use of a SDH ring) into the service providers backbone network and thus its aggregate bandwidth will only need to scale to the capacity of this link.

Two companies that have positioned themselves in the Access Switch market include Cascade Communications Corporation and Newbridge Networks Corporation. Cascade supplies the B-STDX 9000 Edge Switch which offers Frame Relay (FR), ISDN Primary Rate (PR), SMDS and ATM capabilities. This means that they allow an evolutionary approach to the deployment of ATM. The B-STDX features separate switching planes for the various traffic types (CBR, VBR-RT, VBR-NRT, ABR and UBR) as well as large buffering capacities (128K cells for the switching plane and 96K cells per OC3 modules). Cascade also supplies the CBX 500 enterprise switch which is a newer ATM switch.

Newbridge networks presents a wide offering in the ATM market, like Cisco it must run the risk cannabalising it own FR and TDM markets with its ATM products. Its main switches include the 36150 Access Switch, the 36170 Multi-service Switch and the 36190 Core Services Switch. The 36150 can act as a LAN bridge, concentrator, switch or adaptor. It handles ATM, FR, SMDS and TDM. In addition it supports audio and video services using motion JPEG over AAL1 and VBR. It offers both local and wide area adaptors. Network management is proprietary and is performed through 4602 Mainstreet Intelligent Network Station Software on a Sun Workstation.

D.3 Large ATM Switches

There is an air of unreality associated with the use of large scale switches. It is not clear that any of the public carriers who offer data services have reached volumes where their core capacity has become an issue. The question must be asked whether they are merely a

technological capability in search of an application and whether carrier costs for the amount of bandwidth that would be required to fulfill this market simply so large that only completely cost insensitive users (the military and intelligence communities in the United States for example) will have the demand for them.

These switches are the first to become interesting in terms of their requirements for large scale switching. No longer is it possible simply use a shared memory or shared memory architecture for this class of switch, but larger switching architectures such as Clos networks and Knockout concentrators are required. These class of switches tend to have the following characteristics:

- Huge aggregate bandwidths capable of incremental deployment.
- Support OC-n media only
- Support ATM only.
- Provide redundant provisioning for fault tolerance.
- Must support UNI and NNI views.
- Support user accounting and charge back.
- Compared with Edge Switches contain fewer software lines of code with a slower release rate.

The potential market for these switches are as the backbone switch for a corporate enterprise network or as a backbone switch in a carriers central office (CO). They thus are often marketed as two different types of switch whereas in fact they are the same switch albeit with slightly different hardware and software configurations. Two vendors of large switches include Nortel with the Magellan family of switches and Fujitsu Corporation with the FETEX 150.

Nortel Magellan Range

Nortel supports three increasingly larger capacity switches, they are:

- The Magellan Passport which is a small (1.6 Gbps) ATM/FR switch positioned at the customers premises.
- The Magellan Vector - (2.5 - 10 Gbps) ATM network access concentration and multiplexing.

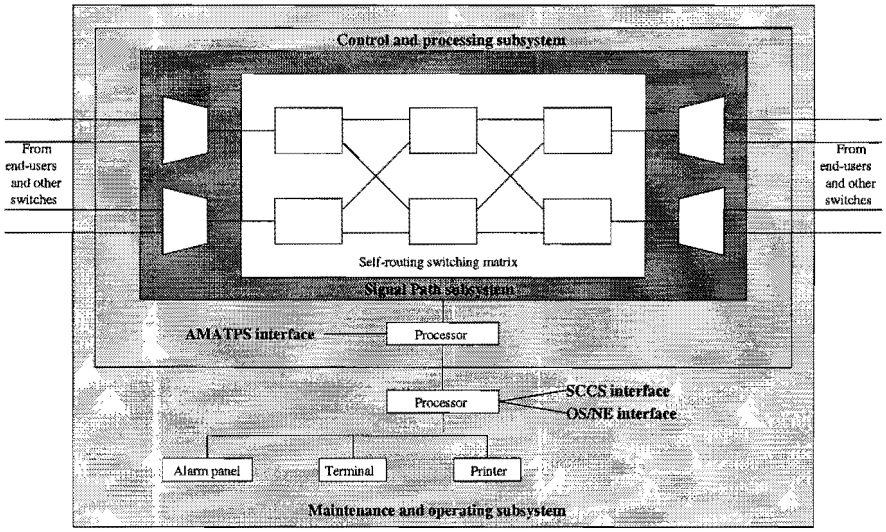


Figure D.1: Fujitsu FETEX-150 ATM Switch Schematic.

- Magellan Concorde: (40-80 Gbps) backbone switch which can be used as a CO switch or for a large video on demand system.
  This is obviously the large switch supported by this company. It more or less follows the design proposed in Chapter 3, it uses Switching Elements (SEs) to perform ATM transport while real-time controllers (RTCs) do signalling, routing and connectivity management. Since it is a large backbone switch it must have good network management support and these include the following:
  - SNMP and CMIP support
  - Remote Access
  - Auto-discovery and provisioning
  - Pre-provisioning
  - Per-connection policing and accounting
  - Performance management
  - Billing Management
  - Fault Management

Fujitsu Corporation and the FETEX 150

The latest generation FETEX 150 is known as the ESP for Enhanced Switching Platform. This is a very large capacity ATM switch supporting incremental bandwidth from a minimum of 2.5 Gbps aggregate bandwidth to a self routing configuration supporting traffic to 180Gbps. It uses 8x8 2.4 Gbps switching boards interconnected in a Clos network for its routing network, for a description of this fabric see [36]. It supports four different types of

processors, a main processor, a call processor, a signaling controller and a operations and maintenance controller. The switch performs full accounting and interfaces to a Bellcore compliant automatic message accounting system for usage based billing.

The switch's architecture is designed as shown in figure D.1 (taken from [24]). The signal path subsystem consists of a Clos network self routing cell switch fabric, the data is first multiplexed up to 622Mbps into the switching matrix via a Broadband Remote Line Concentrator. Each of the classes of ATM services is handled differently, each is assigned to a different type of buffer and the buffer size is adaptable to the type of traffic.

D.4 The place of a large scale switch

After reviewing sections D.1 - D.2 the various types of ATM switch can be considered. A simplified network utilising all of these types of switches is shown in figure D.2. The most important insight to be gained from considering all of the commercial switching designs and options is that the complexity of ATM produces a distinct set of market niches with widely varying demands in terms of cost, performance and reliability. Also, there is a definite distinction between the complexity of switching required for a high and a low latency environment (i.e. from the private to the public network). All of these various interests give rise to a large number of possible design considerations and tradeoffs. The only real way of measuring whether an ATM switch design is "good" or not is simply whether it performs its task to the users satisfaction. Another insight that can be gained from considering all of these factors is that all of the various parts of an ATM switch work together in concert. It is not sufficient to have a non-blocking switching fabric if there is insufficient output buffering to handle transient peaks in the traffic. Huge buffering capacity is to no avail if the flow control in the network is insufficient, since this will merely produce highly oscillatory traffic patterns. In addition, the cell scheduling at the outputs of the switch are critical if multimedia traffic is to be successfully carried. Thus, all of these factors together, plus the place in the network where the switch is to be placed, determine the appropriateness of an ATM switch design.

As shown in the figure, large scale switches are most likely to be found in the enterprise backbone and in the carrier backbone. However, depending on the envisaged penetration of multimedia services and ATM, another switch that could potentially become very large is the Access Switch. A possible sequence of this implementation is:

- The carrier implements ADSL to selected homes through ADSL modems connected to ATM access switches that connect to an ATM network that is run in parallel to the PSTN network.

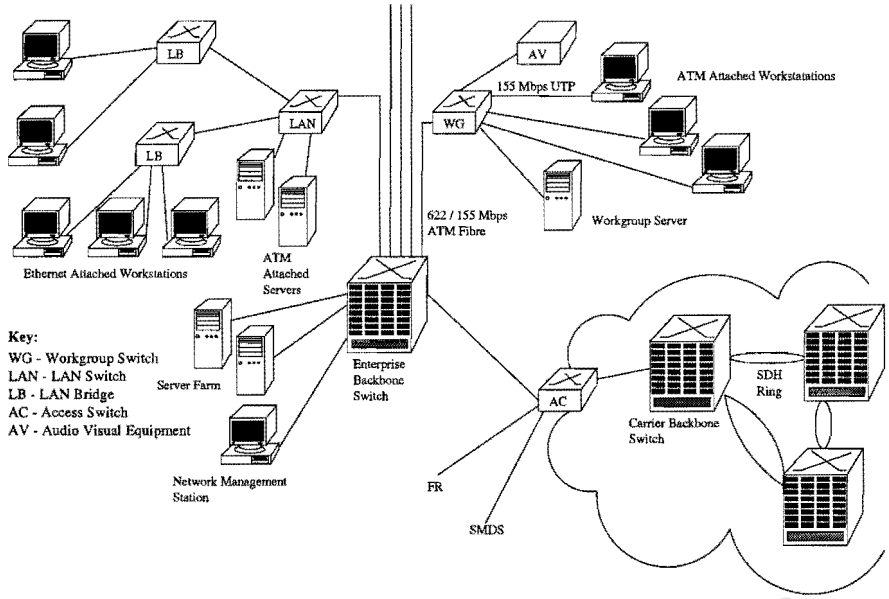


Figure D.2: Types of ATM switch and their position in the network

- The carrier moves the voice network to the ATM network, creating a single architecture backbone network
- The ubiquitousness of the internet and increasing penetration of Video on Demand (VoD) creates a large demand for ADSL. ADSL Access switches become common in Fiber to the Curb (FTTC) implementations with large port count access switches.
- The use of virtual reality, interactive 3D games and environments and other very high bandwidth applications leads to Fiber to the Home (FTTH) demand. The access switch becomes the switching bottleneck and this functionality is moved into a CO backbone switch.
- The use of Wavelength Division Multiplexing (WDM) and the increasing statistical uniformity of huge ATM bundles leads to a layer 1/2 integrated network in the carrier core based on synchronous and wavelength division optical switching.

Although the above sequence of technological milestones is almost guaranteed to be false (consider the famous IBM statement that the world demand for computers would be four, maybe five if GM wanted one), and it ignores the threat of IP to the ATM world. However, it is my belief that recent trends within IP, for example QoS routing and the use of flow labels in IPng will enable a close integration between the IP and ATM networks (unfortunately adding an extra layer to the protocol stack, but thus is flexibility and interoperability maintained). In addition, the cost of constructing very large routers is likely to far exceed that of building a large ATM switch.

Another assumption is that the core of the optical network will return to a synchronous or

WDM based switching technique. The assumption for this trend is that ATM provides three main benefits to the carrier:

- Increased effective bandwidth through the use statistical multiplexing
- Flexibility in network management, the user can signal anywhere in the network.
- Service integration, all services can be run over the same network.

However, the drawback of ATM in switch design is precisely that of the asynchronous arrival of cells. This vastly complicates the design of the switch in terms of the switching fabric, buffering and traffic policing. In addition at very fast data rates (Terrabits per second), the switching fabric must become optical, however optical buffers are slow and expensive. Another consideration is that the advantage of increased effective bandwidth through statistical multiplexing becomes less valid as the number of traffic streams is increased (in precisely the same way that the unpredictable energy usage of a single house becomes predictable when applied to an entire city), thus the chief advantage of asynchronous switching becomes less useful. The last argument for this occurrence is the relative cost of switching versus transmission. Transmission costs per bit per kilometer are continually dropping, while switch costs are currently rising, it thus makes sense to simplify the switch design to take advantage of the cheap transmission bandwidth. It is thus my belief, that just as technologies such as MPOA and CLIP provide layer 2 / 3 integration in todays network, the future trend will be to provide layer 1/2 integration in the core of the network through as yet undiscovered technologies.

In considering all of the above arguments it becomes apparent that the target switches that are likely to become very large scale in both the number of ports and the aggregate switching bandwidth is likely to be in the Access network and the core of the carrier network.

Appendix E

Simulation Classes and Code

This appendix provides information on the files used in the simulation code. Chapter 9 provides an overview of the methodology used for the simulation system and an abstract guide to the classes. This appendix is intended for people wanting to use the code developed for their own purposes or to extend its capabilities. See the accompanying CD-ROM for the electronic resources encapsulated in the thesis. It also includes a guide to where the various simulation results can be found in the thesis. Some of the directory tree is only described in very rough detail while the more important files and classes are described in a certain amount of depth. The description in the files is an extraction of the block comments from each of the source files. The structure of this appendix follows the directory structure. First the various directories or files are described and then in the same sequence each of them is examined in more detail. There is further extensive documentation in the form of block and line comments within the code itself.

The thesis contains the following high level directories:

- **dbug** - contains code that was developed for debugging the various classes. If a class is otherwise incomprehensible check that some debugging code does not exist in this tree.
- **docs** - contains all the documentation for the thesis (including this file).
- **lss** - contains the code that actually performs the simulations. This will be described in greater detail.
- **magic** - contins the layout schematics for the IC (developed in MAGIC).
- **support** - this contains the support code. This is the most important tree as almost all of the functional classes are defined here. This will be described in greater detail.
- **tests** - this has some initial exploratory code that did some basic feasibility tests.

- **tools** - this contains two miscellaneous tools that were written to help with the thesis and documentation.

E.1 support directory tree

Sets of functional classes that have similar functions are each placed in their own subdirectory within this tree.

- **alloc** - simple classes for keeping track of buffer allocation within the simulations.
- **assocmem** - the associative memory is defined in this directory.
- **cell** - the basic definition of the ATM cell used in the simulation.
- **dist** - the code for producing statistical distributions.
- **fastsim** - the later version of the simulation code.
- **graph** - code for defining a directed graph - not used.
- **heap** - a fast heap used for optimising certain of the memory allocations.
- **histogram** - a class for generation arbitrary discrete distributions - not used.
- **maths** - code that encapsulates some mathematical functions.
- **misc** - odd miscellaneous code.
- **models** - the input models.
- **queue** - the queues used within the ATM switch simulations.
- **sim** - the simulation abstract class is defined here
- **simkern** - old version of the discrete time simulator (obsoleted).
- **stats** - a technique for handling and outputting arbitrary size matrices (not used).
- **switchco** - the switch cores.
- **switchte** - the tests which can be implemented on the various core with the input models.
- **vector** - vector classes defined for the code.

E.1.1 alloc subtree

- freecnt.h - This file stores all of the procedures required to handle the free index plus a count of how many elements are free.
- freecnt.cc - This file stores all of the procedures required to handle the free index plus a count of how many elements are free.
- freeindx.h - This keeps track of which index in a structure (e.g. a buffer) is free.
- freeindx.cc - This keeps track of which index in a structure (e.g. a buffer) is free.

E.1.2 assocmem subtree

- assocmem.h - This is the associative memory for the ATM switch simulation says which ATM cell to select from the buffer storage.
- assocmem.cc - This is the associative memory for the ATM switch simulation says which ATM cell to select from the buffer storage.

E.1.3 cell subtree

- corecell.h - This defines the core cell for the switch, this by definition contains all of the possible data.
- cellalloc.cc - This provides the storage space for the Core Elem Cell this file might or might not need to be included in a project depending on whether another fast heap data object is the same size or not.
- encap.h - This just defines a simple macro that allows a cell (packet really), to encapsulate itself inside the other cell.
- rules.h - This defines more concrete rules for which elements to not include in the cell.
- stats.h- This is an abstract class that allows the elements in the simulation to post their cell losses, cell arrivals and cell departures to this class.

E.1.4 dist subtree

- dist.h - This class provides the capability to approximate a distribution given to it in an analytical form by a quadratic approximation of arbitrary accuracy and thus to produce the given distribution.

- dist.cc - This implements the standard distribution class for producing an arbitrary pdf.
- stddist.h - These classes implement standard distributions using the distribution class.
- stddist.cc.- These classes implement standard distributions using the distribution class.
- bursty.h - This defines the bursty traffic model using a two state Markov process. It is specified in terms of the mean burst length (which will also equal the variance) and the traffic density.
- bursty.cc - This implements the bursty traffic model using a two state Markov process. It is specified in terms of the mean burst length (which will also equal the variance) and the traffic density.
- markhov.h - This file performs a Markov discrete time process on a fixed size matrix that can be populated and checked (to make sure all the probabilities add up to one and to force them to do so in case of an integer round down). (It uses large integer rather than doubles to speed the computation).
- markhov.cc - This implements a nice encapsulated markhov chain.
- shuffle.h - This shuffles a set up indices to each other.
- shuffle.cc - This shuffles a set up indices to each other.
- trafalloc.h - Allocate different traffic proportions between inputs and outputs.
- trafalloc.cc - Implement the traffic source allocation class.
- trafdistrib.h - This sets up a class (TrafDistrib) that returns a random number according to some input that is distributed as given by the numbers returned by the TrafAlloc class.
- trafdistrib.cc - This implements the traffic distribution class Output shuffled and normally distributed.
- rbn.h - This describes a class that generates random brownian noise. It uses an O(log N) space and O(1) time function and produces a continuous set of values.
- rbn.cc - This implements the random brownian noise generator.
- selfgauss.h - This produces a self-similar, normally distributed set of random numbers.
- selfgauss.c - This implements the self similar normally distributed number generator.

E.1.5 fastsim subtree

- simkern.h - The simulation kernel for the project. This implements a discrete time kernel using a priority queue.
- simkern.cc - This provides for the actual simulation core (very simple as priority queue support is already there).
- time.h - This defines the unit of time to be used in the simulation kernel.
- time.cc - This provides for support of the kernel's time measure.

E.1.6 heap subtree

- fheap.h - This defines a very fast, templated, fixed size heap, useful for any fast allocation that uses small, fixed size units.
- fheap.tcc - This class keeps definitions of classes that would normally be found in the implementation section of the fast heap but must be included because it is a template.

E.1.7 maths subtree

- function.h - This stores the abstract definition of a function.
- lognormal.h - This keeps a function to produce the log normal distribution.
- lognormal.cc - This implements a lognormal function.
- normal.h - This file keeps a definition of a normal distribution.
- normal.cc - This file keeps a definition of a normal distribution.

E.1.8 misc subtree

- minmax.h - This file defines a minimum and maximum templated function for use in the C++ library.
- swap.h - This defines a templated swap function.

E.1.9 models subtree

- modalloc.h - This provides a meta class for allocating various input models so that the switch tests can be decoupled from the input models.
- modalloc.cc - This file implements various constructors and other virtual methods to alloc the model allocator class to be constructed.
- stdmodel.h - This defines the various standard models available in the system (for input traffic characterisation).
- stdmodel.cc - This implements the various projects with the standard model allocator, and the standard models.
- localref.h - This file defines the locality of reference Poisson process model.
- localref.cc - This implements the locality of reference model defined above.
- vbrmodel.h - This implements a self similar VBR model with a peak and mean cell rate.
- vbrmodel.cc - This implements the VBR model for the simulations.
- vbrmux.h - This defines the VBR multiplexing traffic source, which is also a core (it represents the action of an upstream switch). It also is a scheduler in its own right, but in turn can be scheduled by another simulation.
- vbrmux.cc - This implements the VBR multiplexer.
- vmallcore.h - (VBR mux alloc core) This implements a class that specifically integrates the vbrmux alloc and the meta meta core class through the Attach and reserve extension mechanism. It then knows how to set all the statistics for the input ports.

E.1.10 queue subtree

- priority.h - This implements a generic high-level priority queue, that will be templated at a lower level.
- priority.cc - This defines the access methods for the priority queue, the more complex stuff.
- fastq.h - This implements the parent class of the FIFO Queue with the most generic functionality.

- fastq.cc - This implements the parent class of the FIFO Queue with the most generic functionality.
- gfqueue.h - This file does the templated generic fast queue.
- gprior.h - This is a generic templated priority queue inheriting most of its functionality from its parent class.
- gqueue.h - This is the general queue, not fast, it stores anything in it.
- gqueue.cc - All that this does is create the storage space for the fast heap required for the gqueue, if needed put it here.
- nmux.h - This defines an n-way cell multiplexer with a finite buffer space. It performs simple prioritized multiplexing amongst the cells in its buffers.
- nmux.cc - This implements an n way prioritized cell multiplexer. It performs simple prioritised multiplexing amongst its cells.
- reseque.h - This defines the class which resequences cells at the output. It uses a timeline of linked lists to do time based resequencing. A VC sequence count based resequencer is also defined which slots into the resequencer and allows cells through if they are in the correct sequence before giving them to the time line resequencer.
- qreseque.cc - This is a simple output resequencer with some per circuit optimisation.
- vprior.h - This is a generic templated priority queue inheriting most of its functionality from its parent class.

E.1.11 sim subtree

- metasim.h - This contains several data members and their access that all simulations should support, so all simulations should be derived from this.
- metasim.cc - This implements the traffic classes and number of links in the meta simulation stuff.

E.1.12 switchco subtree

- metacore.h - This defines a purely virtual interface to a switching core.
- metacore.cc - This implements some default constructors and the virtual destructor.

- metaclos.h - This defines a Clos element made up of two or three layers of other Clos elements.
- metaclos.cc - This implements the two types of meta Clos element Meta2Clos, a two layer Clos network and Meta3Clos, a three layer Clos network.
- mtclosal.h - This allocates metclos elements, part of the same class sequence as in closalloc.h.
- closalloc.h - This defines the classes for allocating various types of Clos element. The size of the element can be manipulated (all other construction parameters are fixed).
- closalloc.cc - This allocates the various type of Clos element.
- closabstr.cc - This implements the virtual destructor for the Clos deallocator.
- closcore.h - This defines a core element for a Clos network. Basically a method for buffering cells, indexing them and calling a class which obtains the next element for you.
- closcore.cc - This implements the core switching Element for the Clos Elements (at least the generics of its behaviour).
- stdclos.h - This defines the various types of Clos element. Firstly a random distribution element (single FIFO queue, don't care routing), then a FIFO routed element, lastly an Associative Memory routed element.
- stdclos.cc - This implements the various queueing strategies used by the Clos switching elements.
- coreelm.h - This provides the definitions for the core switching elements.
- coreelm.cc - This implements a core switching element from other classes in the support structure.
- fifocore.h - This defines the FIFO queueing element for the hypercube switch. It uses a FIFO queueing system instead of an associative memory.
- fifocore.cc - This implements a FIFO queueing element in the switch.
- hypercore.h - This implements the FIFO and Clos fabrics as meta cores which allows them to be used as abstract elements of the traffic source classes.
- hypercore.cc - This implements an entire hypercube core element.

- metametacore.h - This defines a core that consists of a sequence of meta cores, when called upon to receive a cell it will input it into the beginning of the chain. When called upon to process, it will process all elements that it has been asked to. A certain number of the stats collectors it has are reserved and passed to a virtual method for processing (mainly used for vbrmuxalloc class).
- metametacore.cc - This implements the staged meta meta core.
- addsequence.h - This defines a core that adds a sequence count and a virtual circuit (its input port) to a cell before it passes it on to any output.
- addsequence.cc - This implements the class that adds a sequence count and VC to an incoming cell.
- perfect.h - This implements a perfect ATM switching core, represented by a single shared queue.
- perfect.cc - This implements the perfect ATM switching core. For comparison when the final output queuing and resequencing stage is tacked on.
- resequence.h - This defines the output resequencer core. This consists of a set of fixed length queues (for every traffic class), a set of sequence counters (one for every input and traffic class, should actually be one per VC) and the time resequencer for those that cannot be otherwise resequenced.
- resequence.cc - This implements the output resequencer for the metacore.

E.1.13 switchte subtree

- clpvp.h - This implements a CLP against load test for the various types of meta cores.
- clpvp.cc - This file contains the implementation of the CLP versus p test for various cores.
- muxgain.h - This file describes a test and a core that determines the multiplexing gain of a large number of sources.
- muxgain.cc - This implements a switch and a sim that calculates the multiplexing gain from a large number of sources.

E.1.14 vector subtree

- array2d.h - This file implements a simple, unchecked, dynamically allocated two dimensional templated array of a given type. (Mainly for syntactic convenience at a later stage).
- bitvect.h - This provides a dynamic, non expandable bit vector.
- bitvect.cc - This is the Bit Vector Implementation.
- vector.h - This is a templated resizing vector class.

E.2 lss/testruns directory tree

The lss directory contains two subdirectories:

- **old** - this contains the old simulation code.
- **testruns** - this contains the actual testruns performed.

The testruns directory consists of the following directories:

- **clp** - This contains the most basic code using only the exponential model.
- **locref** - This contains the locality of reference simulations.
- **mux** - This contains multiplexing gain simulations (eventually not incorporated in the thesis).
- **reseque** - This contains exponential interarrival simulations with an output module.
- **vbrmux** - This contains the multiplexed VBR model.

A further decomposition of these directories will not be entertained since every simulation has a project file which can be loaded and is reasonably self explanatory. With every simulation is an *Applixware* spreadsheet which has accumulated data from a simulation run.

Bibliography

[1] Ronald G Addie, Moshe Zukerman and Tim Neame; "Fractal Traffic: Measurements, Modeling and Performance Evaluation"; IEEE Infocom '95, pp. 977-992.

[2] M. Alimuddin, H. M. Alnuwein and R. W. Donaldson; "The Fat Banyan ATM Switch"; IEEE Infocom '95, pp. 659 - 666.

[3] Ra'ed Y. Awdeh and H. T. Moutfah; "A contention resolution algorithm for input-buffered Batcher-Banyan networks"; International Journal on Communications Systems, Vol. 7, No. 1, pp. 33-38, Jan. - Mar. 1994.

[4] Ra'ed Y. Awdeh and H. T. Moutfah; "Survey of ATM switch architectures"; Computer Networks and ISDN Systems 37 (1995) pp. 1567-1613.

[5] Sandro Bassi, Maurizio Decina, Paolo Giacomazzi and Achille Pattavina; "Multistage Shuffle Networks with Shortest Path and Deflection Routing for High Performance ATM Switching: The Open-Loop Shuffleout"; IEEE Transactions on Communications, Vol. 42, No. 10, pp. 2881-2889, October 1994.

[6] K. E. Batcher; "Sorting Networks and their Applications"; Proceedings of the Spring Joint Computer Conference, AFIPS, 1968, pp. 307-314.

[7] V. E. Benes; "Optimal Rearrangeable Multistage Connection Networks"; Bell Systems Technical Journal; 43 (July 1964), pp. 1641-1656.

[8] V.E. Benes; "Mathematical Theory of Connecting Networks and Telephone Traffic"; New York: Academic Press; 1965.

[9] B. Bingham and H. E. Bussey; "Reservation-based Contention Resolution Mechanism for Batcher-banyan packet switches"; Electronics Letters, Vol. 24, no. 13, pp. 722-723, June 1988.

[10] R. S. Bubenik and J. S. Turner; "Performance of a Broadcast Packet Switch"; IEEE Transactions on Communications, Vol. 37, No. 1, pp. 60-69, Jan 1989.

[11] H. J. Chao, B. S. Choe, J. J. Park and N. Uzun; "Design and implementation of Abacus Switch: A Scalable Multicast ATM Switch"; IEEE Globecom '96, pp. 854-861.

[12] Fred Hung-Ming Chen, John Mellor and Phillip Mars; "A Hybrid Approach for Generating Fractional Brownian Motion"; IEEE Globecom '96; p. 591.

[13] Thomas M Chen and Stephen S Liu; "ATM Switching Systems"; Artech House Inc.; 682 Canton Street; Norwood MA 02062.

[14] X. Chen, J. F. Hayes and M. K. Mehemet-Ali; "Performance comparison of two input access methods for a multicast switch"; IEEE Transactions on Communications. Vol 42, No. 5, May 1994.

[15] Xiaoquiang Chen and Vijay Kumar; "Multicast Routing in Self-routing Multistage Networks"; IEEE Globecom '94, pp. 306-313.

[16] Adhyit K. Choudhury and Ellen L. Hahne; "Dynamic Queue Length Thresholds in a Shared Memory ATM Switch"; IEEE Infocom '96, pp. 679 - 685.

[17] A. Choudhury and E. Hahne; "Buffer Management in a Hierarchical Shared Memory Switch"; IEEE Infocom '94, pp. 1410-1419.

[18] C. Clos; "A study of nonblocking switching networks"; Bell Systems Technical Journal, Vol. 32, pp. 406-424, Mar. 1953.

[19] Edwin R. Coover; "ATM Switches"; Artech House, Boston, London.

[20] Maurizio Decina, Paolo Giacomazzi and Achille Pattavina; "Multistage Shuffle Networks with Shortest Path and Deflection Routing for High-Performance ATM Switching: The Closed-Loop Shuffleout"; IEEE Transactions on Communications, Vol. 42, No. 11, November 1994.

[21] D. M. Dias and and M. Kumar; "Packet switching in NlogN multistage networks"; IEEE Globecom '84; pp. 114-120.

[22] Patrick Droz and Jean-Yves Le Boudec; "A High-Speed Self-Similar ATM VBR Traffic Generator"; IEEE Globecom '96; pp. 586-590.

[23] K. Y. Eng et al.; "A Growable Packet (ATM) Switch Architecture: Design Principles and Applications"; IEEE Transactions on Communications, Vol. 4, No. 2: pp. 423-430, Feb. 1992.

[24] Fujitsu Network Communications; "FETEX 150 ESP ATM Switching System"; August 1995.

[25] A. K. Gupta, L. O. Barbosa and N. D. Georghanas; "A 16x16 limited intermediate buffer switch with input and output buffers and speed constraints"; IEEE Infocom '91, pp. 694-700.

[26] Romano Fantacci, M. Forti and M. Marini; "A Cellular Neural Network for Packet Selection in a Fast Packet Switching Fabric with Input Buffers"; IEEE Transactions on Communications; Vol 44, No. 12, December 1996.

[27] A. Fournier, D. Fussel and L. Carpenter; "Computer Rendering of Stochastic Models"; ACM Communications 25; pp. 371-384, 1982.

[28] Koiuchi Genda, Yukihiro Doi, Kenichi Endo, Tomoaki Kawamura and Shinichi Sasaki; "A 160-Gb/s ATM Switching System using an Internal Seed-up Crossbar Switch"; IEEE Globecom '94, pp. 123-133.

[29] J. N. Giacopelli, J. J. Hickey, W. S. Marcus, W. D. Sincosjie and M. Littlewood, "Sunshine: a high performance self-routing broadband packet switch architecture"; IEEE Journal on Selected Areas in Communication, Vol. 9, No. 8, pp. 1289-1298, Oct. 1991.

[30] M. A. Henrion, G. J. Eilenberger, G. H. Petit and P. H. Parmentier; "A Multipath Self-Routing Switch"; IEEE Communications, Vol. 31, No. 4, April 1993, pp. 46-52.

[31] A. Huang and S. Knauer; "Startlite: a wideband digital switch"; IEEE Globecom '84, pp. 121-125, Nov. 1984.

[32] J. Y. Hui and E. Arthurs; "A broadband packet switch for integrated transport"; IEEE Journal on Selected Areas in Communications, Vol. 5, No. 8, pp. 264-273.

[33] H. E. Hurst; "Long-term Storage Capacity of Reservoirs"; Transactions of the American Society of Civil Engineers 116; 1951.

[34] H. Imagawa, S. Urushidani and K. Hagishima; "A new self-routing switch driven with input-to-output address difference"; IEEE Globecom '88, pp. 1607-1611.

[35] Widjaja Indra, Alberto Leon Garcia; "The Helical Switch: A Multipath ATM Switch which Preserves Cell Sequence"; IEEE Transactions on Communications, Vol. 42, No. 8, August 1994.

[36] Y. Kamigaki, T Nara, S Machida, A Hakata and Y Yamaguchi; "160 Gbit/s ATM Switching System for Public Network"; IEEE Globecom '96; pp. 1380-1387.

[37] M. J. Karol and M. G. Hluchyj; "The Knockout Packet Switch: Principles and Performance"; Proceedings of the 12th Conference on Local Computer Networks; pp. 16-22, Oct 1987.

[38] C. P. Kruskal and M. Snir; "The performance of multi-stage interconnection networks for multiprocessors"; IEEE Transactions on Computers, Vol. 32, No. 12, pp. 1091-1098, Dec. 1983.

[39] V. P. Kumar and J. R. Jump; "Performance of unbuffered shuffle-exchange networks"; IEEE Transactions on Computers, Vol. 35, No. 6, pp. 573-577, June 1986.

[40] A. La Corte, A. Lombardo, G. Schembra; "Modeling Superposition of ON-OFF Correlated Traffic Sources in Multimedia Applications"; IEEE Infocom '95; pp. 993-1000.

[41] Kay Lun Eddie Law and Alberto Leon-Garcia; "Multicast and Self-Routing in ATM Radix Trees and Banyan Networks"; IEEE Infocom '95; pp. 951 - 959.

[42] M. Lawrence and M. Ventura; "Design of a Large Scale ATM Switch"; Teletraffic '97.

[43] M. Lawrence and M. Ventura; "The use of an Associative Memory for non-FIFO Queueing in Large Scale Hypercube ATM Switches"; Satnac '98, pp. 541-548.

[44] D. Lawrie; "Access and alignment of data in an array processor"; IEEE Transactions on Computers, Vol. 24, No. 12, pp. 1145-1155, Dec. 1975.

[45] I. T. Lee and S. C. Liew; "Broadband Packet Switches Based on Dilated Interconnection Networks"; IEEE ICC '92, pp. 255-261.

[46] T. T. Lee; "Nonblocking Copy Networks for Multicast Packet Switching': IEEE Journal on Selected Areas in Communications, Vol. 6, No. 9, pp. 1455-1467, December 1988.

[47] Will. E. Leland, Murad S. Taqqu, Walter Willinger and Daniel V. Wilson; "On the Self-Similar Nature of Ethernet Traffic (Extended Version)"; IEEE/ACM Transactions on Networking, 2(1), February 1994.

[48] Raymond H. Lin, Cheuk H. Lan and Tony T. Lee; "Performance and Complexity of Multicast Cross-Path ATM Switches"; IEEE Infocom '94, pp. 947-954.

[49] Yu-Sheng Lin and C. Bernard Shung; "Queue Management for Shared Buffer and Shared Multi-buffer ATM Switches"; IEEE Infocom '96, pp. 688-695.

[50] Xinijo Liu and H. T. Mouftah; "A Dynamic Cell-Splitting Copy Network Design for ATM Multicast Switching"; IEEE Globecom '94.

[51] Benoit B. Mandelbrot; "Self-Similar Error Clusters in Communications Systems and the Concept of Conditional Stationarity." IEEE Transactions on Communications Technology COM-13; pp. 71-90; 1965.

[52] Benoit B. Mandelbrot3; "Long-Run Linearity Local Gaussian Process, H-spectra and infinite Variances"; International Economic Review; 10(1), February 1969.

[53] Benoit B. Mandelbrot; "The Fractal Geometry of Nature"; Freeman New York, 1983.

[54] "MAGIC";http://www.research.digital.com /wrl/projects/magic/index.html.

[55] Carver Mead and Lynne Conway; "Introduction to VLSI Systems"; Addison-Wesley, 1980

[56] M.K. Mehemet-Ali and M. Youssefi; "The performance analysis of an input access scheme in a high-speed packet switch"; IEEE Infocom '91; pp. 454-461.

[57] "MOSIS";http://www.isi.edu/mosis.

[58] S. Nojima, E. Tsutsui, H. Fukuda and M. Hashimoto; "Integrated services packet network using bus-matrix switches"; IEEE Journal on Selected Areas in Communications, Vol. 5, No. 10, pp. 1284-1292, Oct. 1987.

[59] M. Narasimha; " The Batcher-Banyan self-routing network: universality and simplification"; IEEE Transaction on Communications, Vol. 36, No. 10, pp. 1171-1175, October 1988.

[60] MIL3; "OPNET External Interfaces Manual"; MIL3, Inc., 3400 International Drive NW, Washington DC 20008, USA.

[61] Yoshihiro Ohba; "QLWFQ: A Queue Length Based Weighted Fair Queueing Algorithm in ATM Networks"; IEEE Infocom '97, pp. 566-575.

[62] Achille Pattavina and Claudio Bison; "Connectionless Switching by Asynchronous Banyan Networks"; IEEE Globecom '94, pp. 441-447.

[63] J. H. Patel; "Performance of of processor-memory interconnections for multiprocessors'; IEEE Transactions on Computers, Vol. 30, No. 10, pp. 771-780; Oct. 1981.

[64] Edgar J Peters; "Chaos and Order in the Capital Markets"; John Wiley and Sons Inc., 1991.

[65] G. F. Pfister and V. A. Norton; "'Hot spot' contention and combining in multistage interconnection networks"; IEEE Transactions on Computers, Vol. 34, No. 10, pp/ 943-948, Oct 1985.

[66] Martin de Prycker; "Asynchronous Transfer Mode, Solution for Broadband ISDN, Second Edition"; Ellis Horwood Limited, Campus 400, Maylands Avenue.

[67] Yoshito Sakwai, Nobuhicko Ido, Shinobu Gohara, Nabonu Endo; "Large Scale ATM Multistage Switching Network with Shared Buffer Memory Switches"; IEEE Communications, Vol. 29, No. 1, pp. 90-96, January 1991.

[68] Mischa Schwartz; "Broadband Integrated Networks"; Prentice Hall PTR; Upper Saddle River, New Jersey -07458.

[69] Yoshimitsu Shimojo; "A Fair Queueing Architecture for ATM Switches with Input Buffers"; IEEE Globecom '96, pp. 830-834.

[70] H. J. Siegel; "Interconnection Networks for Large-Scale parallel processing: Theory and Case Studies"; D. C. Health and Company, Lexington, 1985.

[71] Ferrel G. Stremler; "Introduction to Communication Systems, Third Edition";Addison-Wesley Publishing Company.

[72] Subhash Suri, George Varghese and Girish Chandremenon; "Leap Forward Virtual Clock: A New Fair Queueing Scheme with Guaranteed Delays and Throughput Fairness"; IEEE Infocom '97, pp. 557-565.

[73] Pierre U. Tagle, Neeraj K. Sharma; "Multicast Packet Switch based on Dilated Network"; IEEE Globecom '96. pp. 849-853.

[74] A. S. Tannenbaum; "Computer Networks, 3rd Edition"; Prentice Hall Inc., 1996.

[75] 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 Communication, Vol. 9, No. 8, pp. 1173-1193, Oct 1991.

[76] J. S. Turner; "Design of an Integrated Services packet network"; IEEE Journal on Selected Areas of Communications, Vol. 4, No. 8, pp. 1373-1380, Nov. 1986.

[77] J. S. Turner; "Queueing analysis of buffered switching networks"; IEEE Transactions on Communications, Vol. 41, No. 2, pp. 412-420, Feb 1993.

[78] J. Turner; "A Practical Version of Lee's Multicast Architecture"; IEEE Transactions on Communications; Vol. 41, No. 8, August 1993, 1166-1169.

[79] Kuochen Wang and Ming - Howe Cheng; "Design and Performance Analysis of a Growable Multicast ATM Switch"; IEEE Infocom '97, pp. 932-938.

[80] S. Wei, E. Coyle, M. Hsiao; "An Optimal Buffer Management Policy for High-Performance Packet Switching"; IEEE Globecom '91, pp. 924-928.

[81] C. L. Wu and T. Y. Feng; "On a class of multistage interconnection networks"; IEEE Transactions on Computers, Vol. 29, No. 8, pp. 692-702.

[82] Peter Y. Yan, Kyeong Soo Kim and Paul S. Min; "Multi-Channel Deflection Crossbar (MDCD): A VLSI Optimized Architecture for Multi-Channel ATM Switching"; IEEE Infocom '97, pp. 12-19.

[83] Y. S. Yeh, M. G. Hluchyj, A. S. Acampora; "The knockout switch: a simple modular architecture for high performance packet switching"; IEEE Journal on Selected Areas in Communications; Vol. 5, No. 8, pp. 1274-1283, Oct 1987.

[84] Ellen Witte Zegura; "Architectures for ATM Switching Systems"; IEEE Communications, Vol. 31 No. 2, pp. 28-37, February 1993.

[85] Tianming Zhang and Arun K. Somani; "DIRSMIN: A Fault-Tolerant Switch for B-ISDN Applications using Dilated Reduced-Stage MIN"; IEEE Infocom '95; pp. 643-650.

Index