MODELING, ANALYSIS AND DESIGN OF THE INPUT CONTROLLER FOR ATM SWITCHES

DONGMEI WU

Modeling, Analysis and Design of the Input Controller for ATM Switches

By

© **Dongmei Wu, B.Sc.**

A thesis submitted to the
School of Graduate Studies
in partial fulfillment of the
requirements for the degree of
Master of Engineering

Faculty of Engineering and Applied Science
Memorial University of Newfoundland

August 2001

St. John's Newfoundland Canada

Abstract

In broadband communication networks, commonly used traffic rates are of the order of gigabits per second, or even terabits per second. The nodes of the networks, also known as switches or routers, are among the primary technology barriers that hinder the deployment of fast speed networks, while the modern optical fibre technology allows the transmission media to meet the application requirements. Within the switch itself, routing table lookup is the worst bottleneck. Among the proposed Multistage Interconnection Network (MIN) architectures for ATM (Asynchronous Transfer Mode) switch fabric, the Balanced Gamma (BG) network has been shown to be reliable, fault-tolerant, efficient, scalable and superior in performance when compared with other MINs with similar hardware complexity. In this thesis, we provide the modeling, analysis and design of the input controller (IC) for ATM switches using BG networks.

The IC temporarily stores the incoming cells in input buffers, performs routing table lookup, and forwards them to the switch fabric that delivers cells to outgoing lines. A cache-based IC architecture improves the efficiency by locally storing the frequently used forwarding information. We realize this purpose by high-speed cache attempts followed by slower routing table lookups, if necessary.

We have developed a simulator to evaluate different schemes to construct the IC. The simulator has the capability of generating traffic following uniform random traffic (URT) and bursty traffic models. Simulation results show that the IC system works well and the system performance can be improved as cache hits occur most of the time.

Encouraged by the good performance shown, we have developed the hardware implementation for the proposed IC system using Very High Speed Hardware Description Language (VHDL). This is simulated and synthesized using design tools supplied by Model Technology and Synopsys.

Acknowledgement

First of all, I am thankful to my family, especially my parents, my husband and my daughter for their continuous support and great sacrifices that were the major factors in making this work a reality.

I would especially like to express my sincere thanks to my supervisor Dr. Venkatesan for his academic guidance and financial support. Without his supervision I could not have finished this work. I also do not forget the support and help of Dr. Howard Heys and Dr. Paul Gillard.

I would like to thank the School of Graduate Studies at the Memorial University of Newfoundland for the financial support it provided during my Master's Program.

Also, Mr. Nolan White, systems administrator in the Department of Computer Science, deserves special thanks for his true cooperation in fixing the problem related to the Synopsys CAD tools.

Thanks also to Ms. Moya Crocker for her help during my study at the Faculty of Engineering, Memorial University of Newfoundland.

Finally, I would like to thank all my friends and colleagues who sincerely helped during my Master's study. I specially thank Y. E. Sayed, P. Mehrotra, Cheng Li, Lu Xiao, J. Deepakumara, Qiyao Yu, Wei Song, Ji Xie, and K. Kannan.

Table of Contents

Abstract i

Acknowledgement iii

Table of Contents iv

List of Figures vii

List of Tables ix

List of Abbreviations xi

Chapter 1 1

INTRODUCTION 1

1.1 Background on Broadband Communication Networks 1

1.2 ATM Switching 3

1.3 Motivation for This Thesis 5

1.4 Thesis Organization 6

Chapter 2 8

INPUT CONTROLLER FOR ATM SWITCHES 8

2.1 Introduction 8

2.2 Input Controller Functions 8

2.3 Input Controller Architectures 12

2.3.1 Input Buffer 12

2.3.2 Routing Table Memory 16

2.4 Input Controller in Commercial Switching Products 21

2.5 Summary 24

Chapter 3 25

BALANCED GAMMA NETWORK-BASED ATM SWITCHES 25

3.1 Introduction 25

3.2 Topology 26

3.3 Routing Algorithm 28

3.4 Properties 31

3.4.1 Hardware complexity 31
3.4.2 Fault Tolerance Properties 32
3.4.3 Reliability Analysis 35
3.5 Summary 36
Chapter 4 37
INPUT CONTROLLER ARCHITECTURE DESIGN 37
FOR BG ATM NETWORKS 37
4.1 Introduction 37
4.2 System Description 37
4.3 Input Buffer Module 39
4.3.1 Input Buffer Operations 40
4.3.2 Input Buffer Structure 41
4.4 Cache Memory Module 42
4.4.1 Cache Structure 42
4.4.2 Cache Operation 45
4.5 Routing Table Module 48
4.5.1 Routing Table Structure 48
4.5.2 Routing Table Operation 50
4.6 Arbiter Logic Module 51
4.7 Summary 52
Chapter 5 54
PERFORMANCE ANALYSIS OF THE INPUT CONTROLLER 54
5.1 Introduction 54
5.2 Traffic Modeling 54
5.2.1 Traffic Patterns in Broadband Communication Networks 54
5.3 Simulator Software 58
5.4 Simulation Results and Analysis 68
5.4.1 Input Buffer Number 68
5.4.2 Cache Performance 74

5.5 Summary 76
Chapter 6 78
HARDWARE IMPLEMENTATION OF THE IC 78
6.1 Introduction 78
6.2 Hardware Implementation Methodology 78
6.3 Input Controller Architecture 81
6.4 System Components 86
6.4.1 Input Buffer 86
6.4.2 IC Center 87
6.5 Simulation and Synthesis 100
6.6 Summary 105
Chapter 7 106
CONCLUSION 106
7.1 Contributions in this Thesis 106
7.2 Recommendations for Future Work 108
7.3 Summary 110
REFERENCES 111
APPENDICES 114
A Module Structure of the IC System 114
B Behavioral Simulation Result of the IC System 114

List of Figures

FIGURE 1.1 BASIC IDEA OF BROADBAND COMMUNICATION NETWORKS 1

FIGURE 1.2 GENERIC MODEL OF SWITCH 4

FIGURE 1.3 THE ROLE OF ATM SWITCHES IN AN INTERNETWORK 5

FIGURE 2.1 CELL STRUCTURE AT THE UNI/NNI 9

FIGURE 2.2 CELL HEADER STRUCTURE AT UNI 9

FIGURE 2.3 ATM VP/VC SWITCHING 10

FIGURE 2.4 VARIOUS BUFFERING STRATEGIES 13

FIGURE 2.5 CACHE ORGANIZATIONS 19

FIGURE 2.6 THE THREE PORTIONS OF AN ADDRESS IN A CACHE 20

FIGURE 3.1 8 X 8 BG NETWORK 27

FIGURE 3.2 SE OUTPUT LINK NOTATION 28

FIGURE 3.3 ROUTING IN AN 8 X 8 BG NETWORK 29

FIGURE 3.4 CROSSPOINT COMPLEXITY IN A SE 31

FIGURE 3.5 DYNAMIC REROUTING IN AN 8 X 8 BG NETWORK IN CASE OF LINK FAULTS 33

FIGURE 3.6 DYNAMIC REROUTING IN AN 8 X 8 BG NETWORK IN CASE OF SE FAULTS ... 35

FIGURE 4.1 INPUT CONTROLLER ARCHITECTURE 38

FIGURE 5.1 THE ON/OFF SOURCE MODEL 62

FIGURE 5.2 NUMBER OF INPUT BUFFERS UNDER URT TRAFFIC FOR DIFFERENT SIZES OF SWITCHES 71

FIGURE 5.3 NUMBER OF INPUT BUFFERS UNDER BURSTY TRAFFIC WITH MEAN BURST LENGTH = 15 73

FIGURE 5.4 CACHE HIT RATES UNDER VARIOUS TRAFFIC TYPES AND N 74

FIGURE 5.5 CACHE HIT RATES UNDER VARIOUS VP/VC NUMBER AND N 75

FIGURE 6.1 DESIGN FLOW RECOMMENDED BY CMC 79

FIGURE 6.2 UART AND THE INPUT CONTROLLER (IC) 83

FIGURE 6.3 INPUT CONTROLLER (IC) SYSTEM BLOCK 84

FIGURE 6.4 IC CENTRE ARCHITECTURE 85

FIGURE 6.5 INTERFACE OF THE IB 86
FIGURE 6.6 LOOKUP CACHE ARCHITECTURE 87
FIGURE 6.7 INTERFACE OF THE C_REGFILE_0 88
FIGURE 6.8 24-BIT COMPARATOR 89
FIGURE 6.9 COMPARISON PORTION IN THE LOOKUP CACHE 90
FIGURE 6.10 INTERFACE OF THE C_REGFILE_1 91
FIGURE 6.11 INTERFACE OF THE CACHE_MEM 92
FIGURE 6.12 TRI-STATE SYMBOL AND TRUTH TABLE 93
FIGURE 6.13 TRI-STATE BUS 93
FIGURE 6.14 ROUTING TABLE (RT) ARCHITECTURE 94
FIGURE 6.15 INTERFACE OF THE RT_REGFILE 95
FIGURE 6.16 COMPARISON PORTION IN THE RT ARCHITECTURE 96
FIGURE 6.17 INTERFACE OF THE RT_MEM 97
FIGURE 6.18 INTERFACE OF THE ARBITER 98
FIGURE 6.19 PSEUDO RANDOM NUMBER GENERATOR (PRG) STRUCTURE 99
FIGURE 6.20 A PART OF THE SIMULATION REPORT 102
FIGURE 7.1 IMPROVED PSEUDO RANDOM NUMBER GENERATOR (PRG) 109

List of Tables

TABLE 4.1 INPUT BUFFER ORGANIZATION 41
TABLE 4.2 CACHE ORGANIZATION 45
TABLE 4.3 CACHE BLOCK ACCESS NUMBERS UPDATING PROCESS 47
TABLE 4.4 ROUTING TABLE (RT) ORGANIZATION 50
TABLE 5.1 NUMBER OF INPUT BUFFERS AND OUTPUT BUFFERS UNDER DIFFERENT URT TRAFFIC LOADS 70
TABLE 5.2 NUMBER OF INPUT BUFFERS AND OUTPUT BUFFERS UNDER DIFFERENT URT TRAFFIC LOADS 70
TABLE 5.3 NUMBER OF INPUT BUFFERS AND OUTPUT BUFFERS UNDER DIFFERENT URT TRAFFIC LOADS 70
TABLE 5.4 NUMBER OF INPUT BUFFERS AND OUTPUT BUFFERS UNDER DIFFERENT URT TRAFFIC LOADS 71
TABLE 5.5 NUMBER OF INPUT BUFFERS UNDER URT FOR DIFFERENT SIZES OF SWITCHES 71
TABLE 5.6 NUMBER OF INPUT BUFFERS AND OUTPUT BUFFERS UNDER BURSTY TRAFFIC WITH MEAN BURST LENGTH = 15 UNDER DIFFERENT TRAFFIC LOAD, FOR SWITCHES WITH N = 8 72
TABLE 5.7 NUMBER OF INPUT BUFFERS AND OUTPUT BUFFERS UNDER BURSTY TRAFFIC WITH MEAN BURST LENGTH = 15 UNDER DIFFERENT TRAFFIC LOAD, FOR SWITCHES WITH N = 16 72
TABLE 5.8 NUMBER OF INPUT BUFFERS AND OUTPUT BUFFERS UNDER BURSTY TRAFFIC WITH MEAN BURST LENGTH = 15 UNDER DIFFERENT TRAFFIC LOAD, FOR SWITCHES WITH N = 32 72
TABLE 5.9 NUMBER OF INPUT BUFFERS AND OUTPUT BUFFERS UNDER BURSTY TRAFFIC WITH MEAN BURST LENGTH = 15 UNDER DIFFERENT TRAFFIC LOAD, FOR SWITCHES WITH N = 32 72

TABLE 5.10 NUMBER OF INPUT BUFFERS UNDER BURSTY TRAFFIC FOR DIFFERENT SIZES OF SWITCHES 73
TABLE 6.1 PATTERN – GENERATOR OUTPUTS 99
TABLE 6.2 SUMMARY OF AREA AND TIMING REPORT FOR THE COMPONENT AND THE SYSTEM 103

List of Abbreviations

ABR:	Available Bit Rate
ALFSR:	Autonomous Linear Feedback Shift Register
ASIC:	Application-Specific Integrated Circuit
ASSP:	Application Specific Standard Products
ATM:	Asynchronous Transfer Mode
BG:	Balanced Gamma
B-ISDN:	Broadband Integrated Service Digital Network
BR:	Broadcast Reliability
CAM:	Content Addressable Memory
CBR:	Constant Bit Rate
CLP:	Cell Loss Priority
CMC:	Canadian Microelectronic Corporation
FIFO:	First – In – First – Out
FPGA:	Field Programmable Gate Array
FT:	Fault Tolerance
GFC:	Generic Flow Control
HC:	Hardware Complexity
HEC:	Header Error Control
HOL:	Head Of Line
IB:	Input Buffer
IC:	Input Controller
IP:	Internet Protocol
IPv4:	IP Standard version 4
IPv6:	IP Standard version 6
ITU-T:	International Telecommunications Unit-Telecommunications Standardization Sector
LAN:	Local Area Network

LANE:	LAN Emulation
LFSR:	Linear Feedback Shift Register
LIFO:	Last In First Out
LRU:	Least-Recently Used
MIN:	Multistage Interconnection Network
MSB:	Most Significant Bit
MTTF:	Mean Time To Failure
NR:	Network Reliability
OC:	Output Controller
OSI:	Open Systems Interconnection
PRG:	Pseudo Random-number Generator
PT:	Payload Type
QoS:	Quality of Service
RAM:	Random Access Memory
RT:	Routing Table
RTL:	Register Transfer Level
SE:	Switch Element
SF:	Switch Fabric
SIN:	Single-stage Interconnection Network
SONET:	Synchronous Optical NETwork
TGD:	Truncated Geometric Distribution
TP:	Terminal Path
TR:	Terminal Reliability
UART:	Universal Asynchronous Receiver Transmitter
UBR:	Undefined Bit Rate
UNI:	User Network Interface
URT:	Uniform Random Traffic
VBR:	Variable Bit Rate
VLSI:	Very Large Scale Integration

VPI:	Virtual Path Identifier
VCI:	Virtual Channel Identifier
VHDL:	Very-high-speed Hardware Description Language
WAN:	Wide Area Network

Chapter 1

INTRODUCTION

1.1 Background on Broadband Communication Networks

Modern telecommunication networks are evolving at a fast pace. In today's broadband communication networks, many newly emerging applications have diverse service features and require huge bandwidths. These include conventional telephony services, computer data transfer services, applications for audio- and video-communication, audio and video broadcast, games and interactive multimedia applications, etc. The word "broadband" here refers to the wide bandwidths these services require.

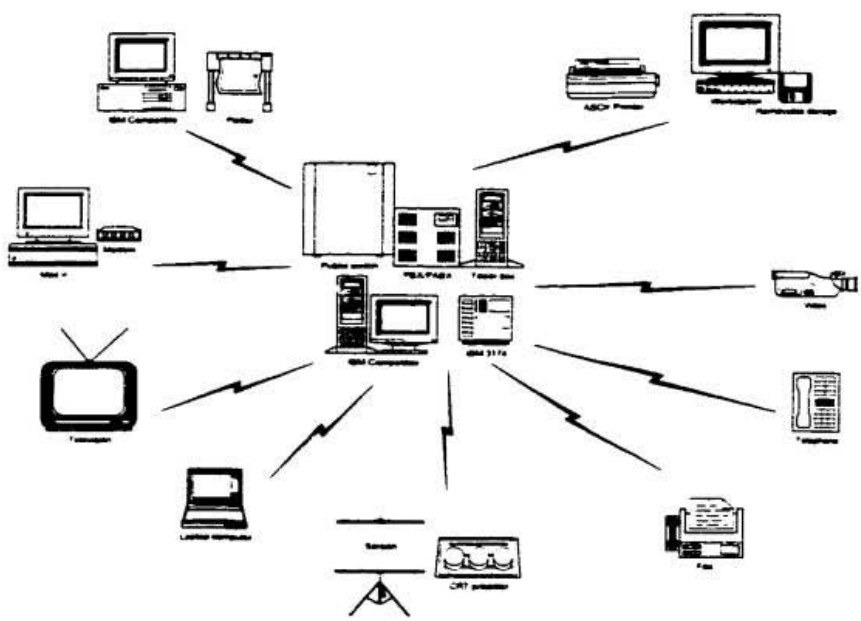


Figure 1.1 Basic Idea of Broadband Communication Networks

Figure 1.1 depicts the basic idea of broadband communication networks. For decades, great efforts have been made to develop broadband communication networks that are flexible enough to accommodate continual changes in service mixes, easy to install and maintain, and efficient on resource utilization, while providing user-friendly access. ATM (Asynchronous Transfer Mode) has been identified by the ITU-T (International Telecommunications Unit – Telecommunications Standardization Sector) as the most suitable protocol to integrate all services over a unified network. ATM is a standard for cell relay, where data, such as voice, data or video, are all converted into small cells of fixed size. ATM technology combines the benefits of circuit switching and packet switching. Combining these two technologies brings the advantage of guaranteed capacity and constant transmission delay (circuit switching) with the efficiency and flexibility for increasing data traffic (packet switching). One of the main benefits of ATM is that it offers assured quality of transmission with service-level agreements. ATM provides QoS (Quality of Service) to every user, congestion control, delivery classes, and dynamic bandwidth allocation capabilities. Currently, the most popular protocol for broadband communication networks is IP (Internet Protocol). IP was invented to be the network layer (layer 3 in the OSI model) protocol running on the top of Ethernet and Token Ring LANs (Local Area Networks). IP is an unreliable protocol, offering services on a best-effort basis, and discarding messages when necessary without attempting retransmission and without warning the transmitting program [1].

1.2 ATM Switching

In this thesis, among many aspects of ATM and IP protocols, we will only discuss aspects regarding ATM switching. This is based on the following observations.

Irrespective of the choice of the protocol, ATM or IP, the nodes of the networks, or switches, have essentially the same functional and performance requirements. Besides, ATM will play an important role in the future IP core networks. In the WAN (Wide Area Network), ATM is expected to carry IP traffic. IP gives us specific attributes, including efficient multiplexing, internetworking and multicasting. But, as mentioned above, ATM supports QoS while IP does not. ATM is providing the transitional solution for IP's limitations in quality and legacy service adaptation. IP core platforms are utilizing ATM backplanes and IP QoS schemes are mimicking the capabilities of ATM as IP continues to evolve as a future core network protocol.

Research on ATM switching has been developed worldwide for several years and many ATM switching architectures have been proposed. Throughout this thesis, we will divide the switching architecture, or the switch, into three function blocks: input controller (IC), switch fabric, and output controller (OC). We discuss only cell forwarding operations, namely, operations related to the transfer of cells from the inputs to the outputs of the switch. Thus, other functionalities relevant to the set-up and tear-down of the virtual connections are not discussed here. Figure 1.2 illustrates this generic switch model. As shown in Figure 1.3, these switches are geographically distributed over a communication network. A user-to-user connection typically goes through several

nodes (switches), and thus each cell belonging to such a connection experiences many hops.

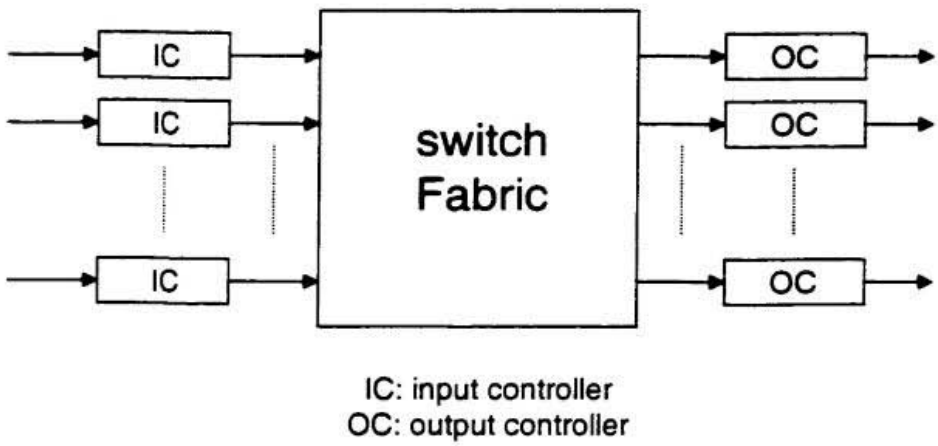


Figure 1.2 Generic Model of Switch

The switch fabric (SF) forms the core of a switch. The SF is primarily responsible for transferring cells between the other functional blocks, routing, signaling and managing cells.

The output controller (OC) prepares the ATM cell streams for physical transmission by removing and processing the internal tags, managing output cell buffers, etc.

The input controller (IC) performs some important functions for each ATM cell:

- temporarily buffers the incoming cell streams
- performs table lookups to translate the cell VPI/VCI (virtual path identifier/virtual channel identifier) values
- determines the destination output port

- rewrites the ATM cell headers
- generates the internal routing tag for use only within the switch

The following chapters are devoted to exploring the input controller in considerable detail.

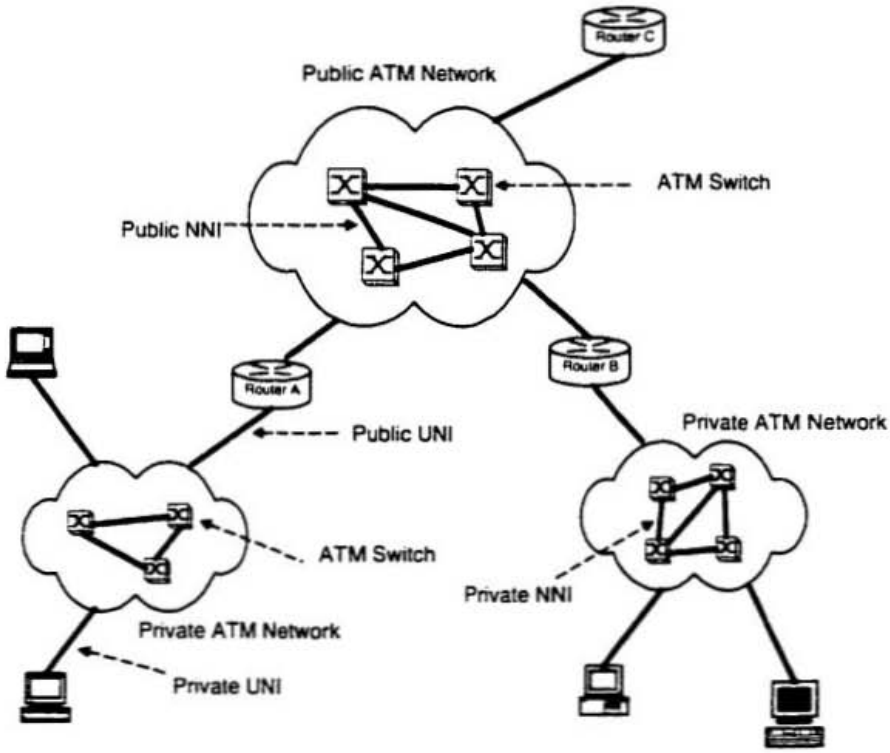


Figure 1.3 The Role of ATM Switches in an Internetwork

1.3 Motivation for This Thesis

The standards for ATM are relatively mature, but the actual hardware designs and implementation are still in progress. Tremendous research has been conducted to develop more efficient and better-performing ATM switch architectures.

As we know, in very high-speed networks, the technology deployed in the nodes of the network, or switches, influences the speed of the networks greatly, and the routing table lookup is the worst bottleneck in a switch. There have been many attempts to overcome the problem of speeding up the routing table lookup techniques. Our work here is an effort in this direction.

We propose a hardware implementation of a routing table lookup scheme. At very high line rates, this process must be executed as rapidly as possible, which is why we consider the design of an ASIC (application-specific integrated circuit) dedicated to this process. Hardware can greatly speed up the system process compared with software implementation. Besides, a cache-based architecture of input controller (IC) is motivated here. This is realized by high-speed cache attempts followed by slower table memory lookups. We expect to gain better system performance through caching the frequently used forwarding information.

Our input controller solution in this thesis would be applicable to any switch or router that handles fixed-sized packets, not just to ATM switches. For those routers that have to deal with long, variable-length packets, the packets are chopped into small equal-sized chunks, switched using an ATM switch-like router, and then reassembled to get back original packets. Such router designs are commonplace now.

1.4 Thesis Organization

The remainder of the thesis is organized into seven chapters.

In Chapter 2, we emphasize the input controller functions in ATM switches and investigate the various categories of input controller architectures in terms of the input buffering strategies, routing table hierarchy design, etc. To clarify the different schemes, we also show some examples of input controllers in commercial switching products. We end this chapter with a detailed commercial example.

In Chapter 3, we depict a clear picture of the Balanced Gamma (BG) network-based ATM switch. This is necessary before we start discussing our design because our IC is designed for BG ATM networks and we will take some advantage of the characteristics of the BG network. In this chapter, we include the topology, the routing algorithm employed and various properties of this network.

In Chapter 4 and 5, we discuss our IC design in detail. We divide the whole system into several modules and analyze the functions and structures of the individual modules. To verify our model, we build up a software simulator. The traffic patterns used in the simulation are uniform random traffic (URT) and bursty traffic. After examining the simulation results, we make sure that the system modeling is correct and thus we can prepare for the hardware design process.

In Chapter 6, we present the IC hardware design. The design is carried out using Synopsys CAD tool supported by the Canadian Microelectronic Corporation (CMC). The design is described at the Register Transfer Level (RTL) using Very-high-speed Hardware Description Language (VHDL). We adopt the 0.18 μm CMOS technology, also supported by CMC, to achieve a high-speed design.

Chapter 7 concludes our work.

Chapter 2

INPUT CONTROLLER FOR ATM SWITCHES

2.1 Introduction

In this chapter, we present a survey of several input controller schemes for ATM switches. Firstly, we explain the input controller main functions that are of interest in our work. Then, we discuss various input controller architectures commonly used for switches in broadband communication networks. We mainly focus on two issues. One is the input buffering strategies. The other is the memory hierarchy for the routing table. Finally, we show some input controller examples in commercial switching products.

2.2 Input Controller Functions

Before we talk about input controller functions, it is useful to examine the ATM cell and cell header structure, as well as ATM VPI/VCI switching.

ATM is a cell-based switching technology. The cell consists of a 5-octet (i.e. 5-byte) header and a 48-octet information field as shown in Figure 2.1 [2]. Some conventions used are [2]:

- bits within an octet are sent in decreasing order, starting with bit 7;
- octets are sent in increasing order, starting with octet 1;
- for all fields, the first bit sent is the most significant bit (MSB).

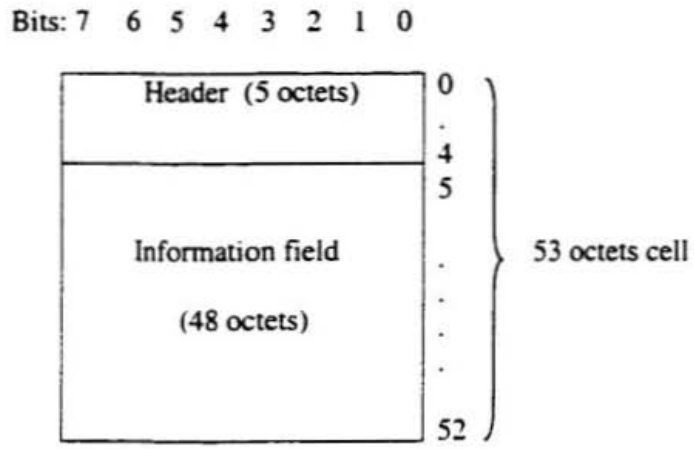


Figure 2.1 Cell Structure at the UNI/NNI

The structure of the header is shown in Figure 2.2 [2]. The fields contained in the header and their encoding is described below.

7	6	5	4	3	2	1	0	Byte
GFC				VPI				0
VPI				VCI				1
VCI								2
VCI				PT			CLP	3
HEC								4

Figure 2.2 Cell Header Structure at UNI

CLP Cell Loss Priority

GFC Generic Flow Control

PT Payload Type

HEC Header Error control

VPI Virtual Path Identifier

VCI Virtual Channel Identifier [2]

The 24-bit routing field consists of 8 bits for virtual path identifier (VPI) and 16 bits for virtual channel identifier (VCI). ATM is connection-oriented and the header values, VPI/VCIs, are assigned to each section of a connection for complete duration of the connection. VPI and VCI are unique for cells belonging to the same virtual connection on a shared transmission medium.

There are basically two types of ATM switching used in an ATM network (see Figure 2.3) [3]:

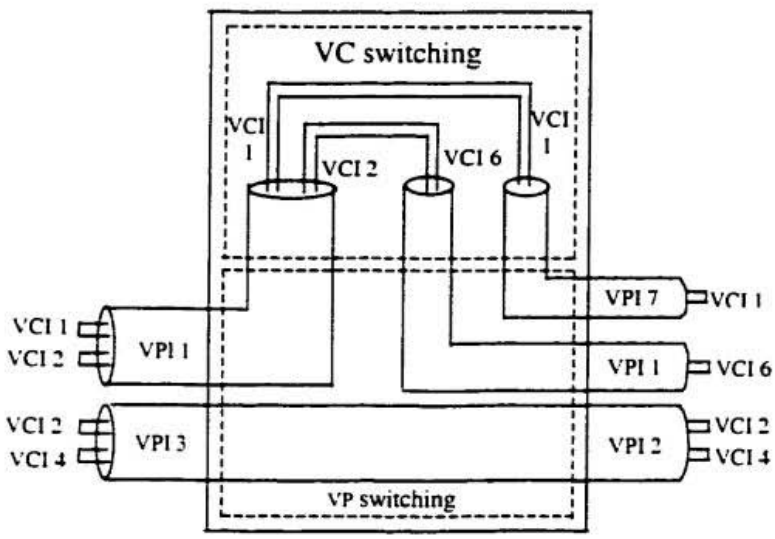


Figure 2.3 ATM VP/VC Switching

1) the virtual path switch

This switch routes cells based only on the VPI value within the routing field of the cell header. This means a low switching overhead and efficiency. This type of switch is known as an ATM cross-connect and is frequently used as a network trunk-switching device [3].

2) the virtual channel switch

This switch routes cells based on the value of the whole of the routing field within the cell header, that is, using the VPI/VCI combination. This type of switch has a higher overhead than a cross-connect type switch [3].

The input controller is located in front of the switch fabric. Figure 1.2 in Chapter 1 depicts the input controller location in the ATM generic switch. Input buffers are the interface of a switch with the incoming signals. When reaching a switch, the incoming signal is first terminated and the ATM cell streams are extracted. These involve signal conversion and overhead process, and cell delineation and rate decoupling [4]. After that, for each ATM cell, several important processes are done. These important processes are the main focus of our work here and are explored in great depth in this thesis. They include:

- temporarily buffering the incoming cell streams
- routing table lookup to translate the cell VPI/VCI values
- determining the destination output port
- rewriting the ATM cell headers

- generating the internal routing tag for use only within the switch

2.3 Input Controller Architectures

Because the switch design is not part of the ATM standards and research in this field is still continuing, vendors utilize a wide variety of techniques to build their switches. A lot of research has been carried out to explore the different switch design alternatives. In this section, we review two issues related to performing the input controller functions in the IC: the input buffering strategy and the memory hierarchy of the routing table (RT).

2.3.1 Input Buffer

2.3.1.1 Blocking in Multistage Interconnection Networks

In multistage interconnection networks (MINs), the packet inputs are unpredictable. When paths of two cells addressed to two different output lines might conflict before the last stage, the **internal blocking** occurs. Only one of the two cells contending for a link can be passed to the next stage. A fabric is said to be internally blocking if a set of N cells addressed to N different outputs can cause conflicts within the fabric. When cells wait for a delayed cell at the head of the queue to go through, even if their own destination output ports are free, **head-of-line (HOL) blocking** occurs. This also reduces the overall throughput.

2.3.1.2 Buffering Strategies

To reduce the above cell loss and improve the throughput, packet switches need buffers to hold packets as they wait for access to the switch fabric or the output trunk.

There are several basic approaches to the placement of buffers. These basic approaches are illustrated in Figure 2.4 [4] [5]:

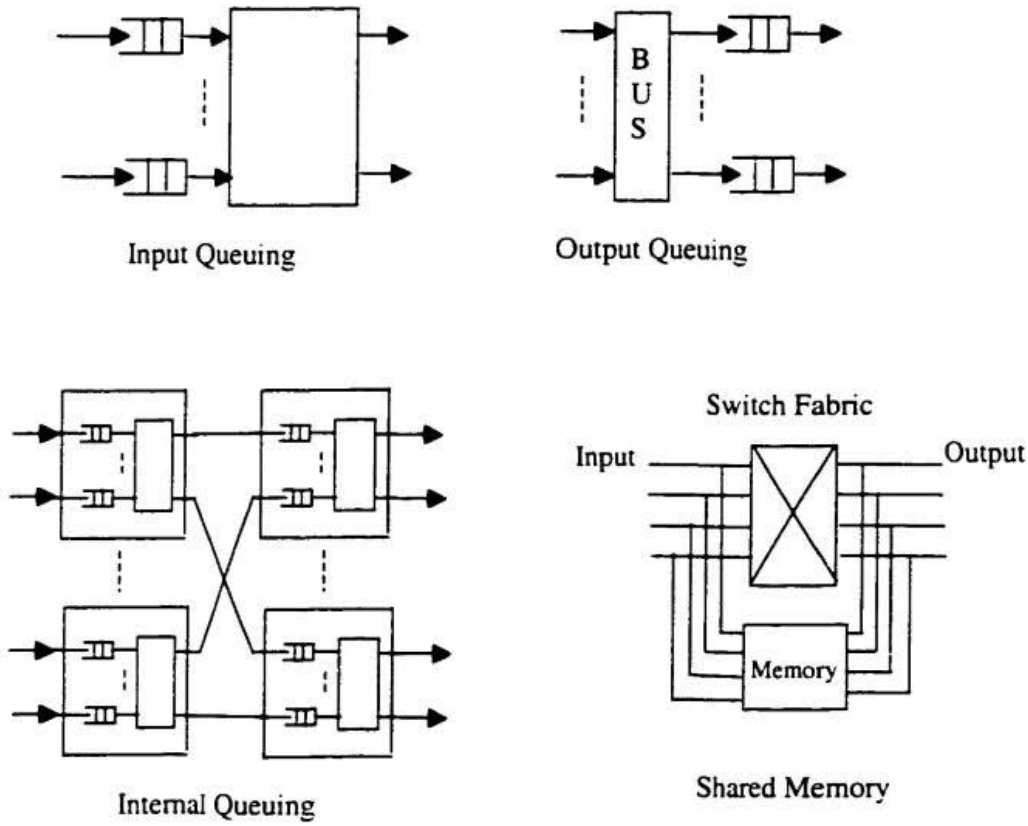


Figure 2.4 Various Buffering Strategies

Input Queuing

In a purely input-queued switch, packets are buffered at the input and released when they win access to both the switch fabric and the output trunk. This approach suffers from head-of-the-line (HOL) blocking. When two cells arrive at the same time

and are destined to the same output, one of them must wait in the input buffers, preventing the cells behind it from being admitted. Thus capacity is wasted [4].

Several methods have been proposed to tackle the HOL blocking problem, but they all exhibit complex design. Increasing the internal speed of the space division fabric, or changing the first-in-first-out (FIFO) discipline are two examples of such methods [4].

Output Queuing

A pure output queued switch buffers data only at its outputs. This approach is optimal in terms of throughput and delays because the output queues do not suffer from HOL blocking. However, in the worst case, cells at all N input ports may be destined to the same output port. If the switch has no input buffers and we want to avoid cell loss, the switch fabric must deliver N cells to a single output, and the output queue must store N cells in the time it takes for one cell to arrive at an input. This makes the switch fabric and queues more expensive than with input-queued switching [5]. Besides, in both cases, the throughput and scalability are limited.

Internal Queuing

One kind of switch fabric is called **space division switch fabric**, in which there is more than one path between each input and output port. These paths work in parallel so that many cells can be sent at the same time [28]. Buffers can be placed within the switching elements in a space division fabric. Again, HOL blocking might occur within the switching elements, and this significantly reduces throughput, especially in the case of small buffers or larger networks. Internal buffers also introduce random delays within

the switch fabric, causing undesirable cell delay variation [4]. Getting out of sequence of cells is another problem in this kind of multi-path switches.

Shared Memory

In a shared-memory switch, input and output ports share a common memory. Cells are stored in the common memory as they arrive, and the cell header is extracted and routed to the output port. When the output port scheduler schedules a cell for transmission on the trunk, it removes the cell from the shared memory. Since the switch fabric only switches headers, it is easier to build. However, an N x N switch must read and write N cells in one cell arrival time. Because memory bandwidth is usually a highly constrained resource, this restricts the size of the switch [5].

Studies show that various combinations of the above, not one of the pure queuing strategies, would lead to better compromises [6]. We also need to notice that buffering cannot totally eliminate cell loss, but it can reduce it to acceptable levels. Adding buffers is not a penalty-free solution towards minimizing cell loss for two reasons. One is that buffering requires adding extra hardware to the system in the form of memory elements and control circuits. The other is that cells that are stored within the buffers suffer from delay. In fact, the larger the buffer sizes, the bigger is the delay, which imposes another limit on the buffer size used. For instance, when we try to improve the level of cell loss, which is normally achieved by increasing buffer sizes, we end up experiencing poor levels of cell delay, and vice versa [7].

2.3.2 Routing Table Memory

The routing table (RT) memory is a very important memory system in an ATM switch. In practice, the existing memory system can be classified into two categories, the non-cache scheme and the cache-based scheme. In this section, we focus on the description of each choice.

2.3.2.1 Non-cache Scheme

This category can be subdivided into centralized RT memory and distributed RT memory. In the centralized RT memory system, a single RT receives lookup requests from all input ports of the switch. An arbitration mechanism must be introduced to coordinate the requests from different input ports. A centralized RT can be a performance bottleneck if it is overloaded by processing demands. Distributed RT memory can solve the bottleneck, but, since every input port has a RT, huge amounts of memory may be needed. The hardware complexity may increase since coordination may be required.

2.3.2.2 Cache-based Scheme

This category usually has two main levels of memory hierarchy, small cache memory and large RT memory. These two levels of memory may be further subdivided into multiple levels.

2.3.2.2.1 Motivation for Using Cache

One of the most important memory properties that can be exploited is **locality** of reference: information that has been used recently tends to be reused. This says that every item of the data or code is not accessed with the same probability. Two different types of locality have been observed. **Temporal locality** states that recently accessed items are

likely to be accessed in the near future. **Spatial locality** says that items whose addresses are near one another tend to be referenced close together in time [8].

Cache is used in the RT memory system because of the presence of a strong temporal locality, not spatial locality, displayed by the nature of data forwarding in a communication network. This will be explained in Section 4.4.1. The input controller performs the lookup at line speeds, or wire speeds, that is, at the rate of incoming cells on the input lines. To keep up the speed, given a large routing table, it is obviously useful to cache recently used routing entries.

2.3.2.2.2 Cache Attributes

Before we consider the application of a cache structure in the ATM switch design, it is necessary to exploit cache attributes briefly.

Cache is the name generally given to the first level of the memory hierarchy encountered once the address leaves the CPU [8]. Since the principle of locality applies at many levels, and taking advantage of locality to improve performance is so popular, the term 'cache' is now applied whenever buffering is employed to reuse commonly occurring items.

A **block** is the minimum unit of information that can be present in the cache (**hit** in the cache). **Miss rate** is the fraction of accesses that are not in the cache. Cache organization can be classified into three categories based on where a cache block is placed [8]:

Direct mapped cache

If each block has only one place it can appear in the cache, the cache is said to be direct mapped. The mapping is usually

(Block address) MOD (Number of blocks in cache)

Fully associative cache

If a block can be placed anywhere in the cache, the cache is said to be fully associative.

Set associative cache

If a block can be placed in a restricted set of places in the cache, the cache is said to be set associative. A set is a group of blocks in the cache. A block is first mapped onto a set, and then the block can be placed anywhere within that set. The set is usually chosen by bit selection, that is,

(Block address) MOD (Number of sets in cache)

If there are n blocks in a set, the cache placement is called **n-way set associative**.

Direct mapped is simply one-way set associative and a fully associative cache with m blocks could be called m-way set associative; alternatively, direct mapped can be thought of as having m sets and fully associative as having one set.

Figure 2.5 [8] depicts the different cache organizations. Caches can be further sub-classified into multi-level caches, that is, main cache, secondary cache, even tertiary cache. Secondary cache is widely used nowadays in computing systems.

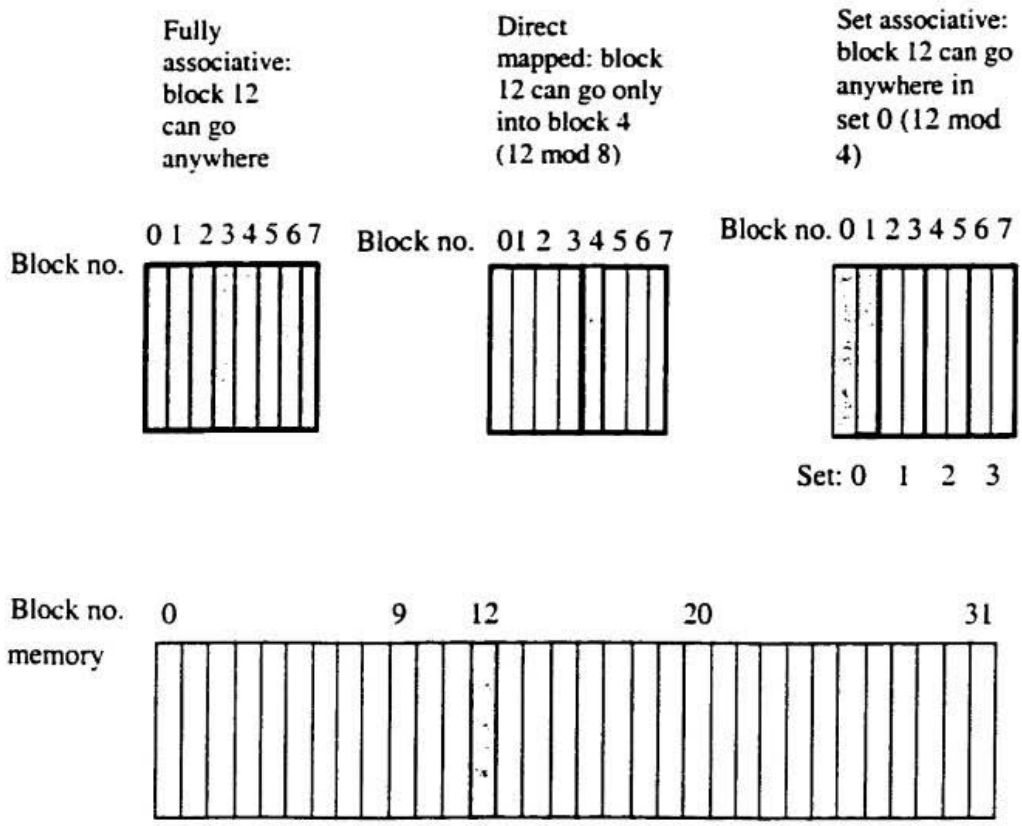


Figure 2.5 Cache Organizations

(This example cache has eight block frames and memory has 32 blocks.)

Caches have an **address tag** on each block frame that gives the block address. The tag of every cache block that might contain that desired information is checked to see if it matches the block address. All possible tags are searched in parallel because speed is critical. A **valid bit** is added to the tag to say whether or not this entry contains a valid address. If the bit is not set, there cannot be a match on this address. The cache block also includes the index and block offset. The block offset field selects the desired data from

the block, the index field selects the set, and the tag field is compared against block address for a hit [8]. Figure 2.6 [8] shows the cache block structure.

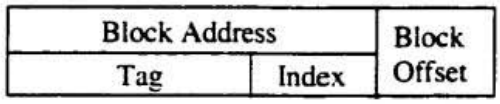


Figure 2.6 The Three Portions of an Address in a Cache

When a miss occurs, the cache controller must select a block to be replaced with the desired data. Several algorithms are used to select the victim block:

Random

To spread allocation uniformly, candidate blocks are randomly selected. Some systems generate pseudorandom block numbers to get reproducible behavior, which is particularly useful when debugging hardware [8].

Least-recently used (LRU)

To reduce the chance of throwing out information that will be needed soon, accesses to blocks are recorded. The block replaced is the one that has been unused for the longest time. LRU marks use of a corollary of locality: If recently used blocks are likely to be used again, than the best candidate for disposal is the least-recently used block.

Some other replacement algorithms are: **FIFO** which replaces the oldest block, **LIFO** which replaces the newest block, as well as **IDEALLY** which replaces the block

that will not be used for longest time (optimal) [9]. Considering the feasibility and additional control bits, some of the above algorithms are not suitable in practice.

Besides the various schemes for the input controller, there also exists the trade-off between software implementation and hardware implementation. While some people prefer software's flexibility, hardware has the benefit when speed is still a main concern.

2.4 Input Controller in Commercial Switching Products

One of the most attractive aspects of technology is how the technology would be used. Looking back over the communications products of the last three decades, cost/performance is probably the most important of the three attributes – the others being standards and time to market – that determine product success.

Today's vendors of switching equipment include some traditional telecommunication equipment-makers, such as Nortel Networks, Lucent Technology, Fujitsu, etc., some data/networking companies, such as Cisco, Juniper Networks, etc., and others like IBM, 3Com, PMC-Sierra, MOSAID, etc. Some small companies also have leading edge solutions for switching products.

We now review some commercial switching products.

Cabletron Systems has introduced a new product called Smart Switch Router, which is based on YAGO Systems, Inc. router development. There is a 16 Gbps non-blocking switching fabric in the backplane and the full routing table of maximum of 250,000 routing entries. Each port can switch 50,000 MAC addresses [10].

BBN Corporation is part of GTE Internetworking, a unit of GTE Corporation. The research group of BBN has published a design for "A Fifty Gigabit Per Second IP Router" called MGR. MGR is built around a high-speed custom design 15-port switch. Forwarding engine cards, having a complete set of routing information, are separate from the line cards. All the line cards are able to translate the link-layer header to an abstract link-layer header having information required for IP forwarding. The forwarding engine comprises three levels of caches: the most used part of the code fits into the first level 8KB Instruction cache (Icache), the on-chip 94KB secondary cache is filled with 12000 cached route entries, and a third level 16MB off-chip cache is reserved for the complete forwarding table. This memory is divided into two 8 MB memory banks. The route updates are made by copying a new table to one bank and then disabling the previously used memory bank. The switch is input-queued. All inputs have a separate FIFO for each output. The scheduling guarantees that there is no HOL blocking and the switch operates with full throughput [10].

The Cisco 12000 series switch is optimized for performing routing and packet forwarding functions to transport IP datagrams across a network. This product is based on a high speed distributed routing architecture. The packet forwarding functions are performed by each of the line cards. A copy of the forwarding tables computed by the GRP (Gigabit Route Processor) is distributed to each of the line cards in the system. Each line card performs independent lookup of a destination address for each datagram received on a local copy of the forwarding table. The routing table has up to 1 million route entries [33].

The next example is the SiberCore Technologies SiberCAM Ultra-2M device [11], a product of the third generation packet forwarding engine. The packet forwarding solutions have progressed through three generations already. In the first generation, the lookups was largely done in software. Most systems today, including the above discussed commercial examples, are the second generation products which migrate lookup and forwarding functions to hardware solutions, especially using ASICs with an on-chip engine and SRAM. The third generation forwarding engines are tending towards using application specific standard product (ASSP) based on high-performance, high-capacity ternary CAM technology. CAM is a type of specialty memory device that is accessed in a fundamentally different way than RAMs. While a CAM is written and read exactly as a RAM, it may also be searched. In a search, all data stored in the memory is simultaneously compared to the search key in a single operation. The result of the search is the physical address at which the matched entry is found. This operation inherently accomplishes a table look-up in hardware, and requires that dedicated comparison circuitry be co-located with every bit of storage. The CAM-based forwarding engine uses this input to search the route look-up table to determine the appropriate address for the data. This, in turn, is output from the CAM to a context memory to determine the appropriate routing, switching or policy to be applied for that particular packet. The table update and maintenance operations are performed out-of-band, that is, they can occur at the same time as look-up operations, without either operation being slowed down [11].

2.5 Summary

Many proposals have been made on ATM switch architecture design. Among these, various input controller schemes show up. These choices, including the examples we discussed in this chapter, have pros and cons in terms of hardware complexity, material cost, management, etc. Some of them can no longer meet the speed requirements in the broadband applications in the gigabit range. Nowadays, researchers still work hard to look for other solutions to high performance ATM switching. Our design also works in this direction. In the later chapters, we will discuss our input controller architecture in detail.

Chapter 3

BALANCED GAMMA NETWORK-BASED ATM SWITCHES

3.1 Introduction

Before we forward to our input controller architecture design, we discuss the Balanced Gamma (BG) network-based ATM switch, which is one of the applications our input controller is designed for. As we mentioned in Chapter 1, our design of input controllers are not only applicable to BG ATM switch, but also any switch or router that handles fixed size packets.

Many multistage interconnection networks (MINs) for ATM switch architectures have been reported in the literature, but most of these architectures are not efficient due to their high cost/performance ratios, inefficient performance, or lack of scalability. The Balanced Gamma network has been recognized to be reliable, fault-tolerant, efficient, scalable and has much better performance than the other MINs with same hardware complexity [7].

The BG network is a MIN based on 4 x 4 switch elements (SEs). It is called "balanced" because the BG network is developed from Gamma network which is based on 3 x 3 SEs. The Gamma network is unbalanced in the sense that, among the three

output links of each SE, one of the nonstraight links may be considered as an alternative to the other nonstraight link and a rerouting scheme may be evolved to exploit this inherent redundancy. But the straight link would become the critical element without an alternative. The BG network adds the 4^{th} link to each SE, making it 4 x 4. Hence each output link can have an alternative and it is balanced.

In the following sections, many aspects of the BG network, including the topology/structure, routing algorithm, hardware complexity, fault tolerance, reliability, and performance analysis, will be summarized. In Section 3.2, the topology/structure is firstly introduced. In Section 3.3, the routing algorithm is discussed. Then, in Section 3.4, the BG network properties are shown, including the hardware complexity (HC), fault tolerance (FC), and reliability analysis. At last, based on some simulation results, the BG network's performance is analyzed.

3.2 Topology

BG network is a MIN based on 4 x 4 SEs. For an N x N BG network, where N is the size of the network, there are $\log_2 N+1$ stages. The 1st stage (Stage 0) consists of 1 x 4 crossbar SEs, i.e., each SE has one input link and four output links. Each of the following $\log_2 N-1$ stages are based on 4 x 4 crossbar SEs each of which has four input links and four output links. The last stage is a special buffer stage.

In each of the first $\log_2 N$ stages, there are N SEs numbered from 0 to N - 1. The i^{th} SE in the j^{th} stage is connected to four SEs in the $(j + 1)^{th}$. The four SEs are [14]:

i, $(i + 2^j) \bmod N$, $(i - 2^j) \bmod N$, and $(i + 2^{j+1}) \bmod N$.

The pseudo code showing the connection scheme can be found in [17].

The buffer stage is a special stage. It is used to collect the outgoing cells from the switch fabric and feed them to the respective destination output port. From the structure, there are four input links for each output buffer. The output buffer is designed to accept up to 4 cells in each switch cycle.

Figure 3.1 [7] depicts an 8 x 8 BG network, where N = 8, totally $\log_2 8 + 1 = 4$ stages including the buffer stage.

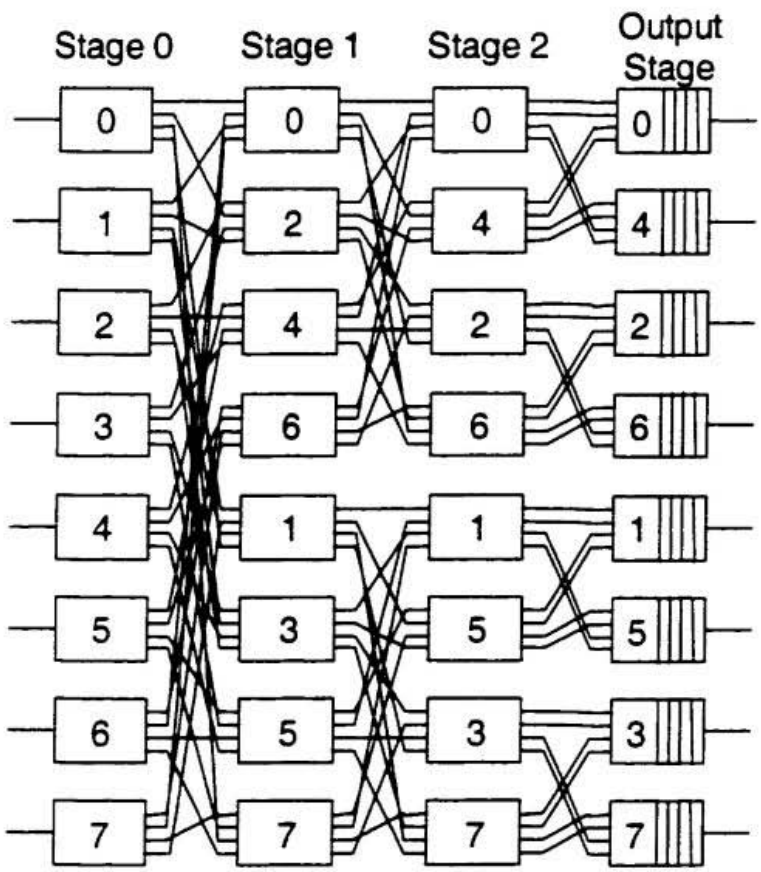


Figure 3.1 8 x 8 BG Network

3.3 Routing Algorithm

One of the attractive characteristics of BG network is its simple routing algorithm called Reversed Distance Tag Routing algorithm. It is explained in detail in the following.

Each cell from source S to destination D has been assigned a routing tag which represents D, i.e. the output port number, in binary: $d_{n-1}\, d_{n-2} \ldots d_0$, where $n = \log_2 N$;

SEs interpret the tag in reverse order, i.e., SEs in Stage 0 switch a cell based on bit d_0, SEs in Stage 1 switch a cell based on bit d_1, ..., and so on, until SE in Stage n-1 which switches on bit d_{n-1}.

For each of the SEs, the four output links can be divided into two groups. The upper two links are used for switching a cell with routing tag bit 0; the lower two links are used for switching a cell with tag bit 1. In either the upper or the lower group, the first output link is considered normal. When the normal link is faulty or busy directing a cell, then the second one – the alternative link will be chosen. Figure 3.2 [14] shows this rule.

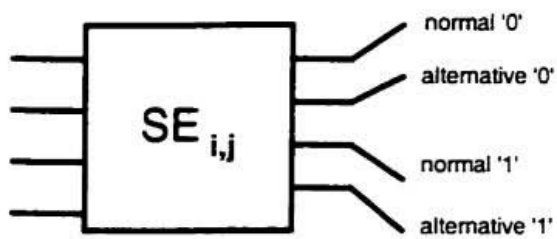


Figure 3.2 SE Output Link Notation

When more than one cell with same tag enter a SE, up to 2 cells with same tag can be routed without any cell loss. If three or four cells with same tag bit enter a SE, then two will be routed and the rest are discarded.

Figure 3.3 shows an example of routing in an 8 x 8 BG network.

A cell with routing tag 110 comes into the network from input port #3. Stage 0 switches it based on '0' and chooses one of the two upper links; when it reaches SE #2, Stage 1, it is switched based on the second '1' in 110 and routed to one of the two lower links; when it reaches SE #2, Stage 2, it is switched based on the first '1' in 110. Thus it gets to the output port #6.

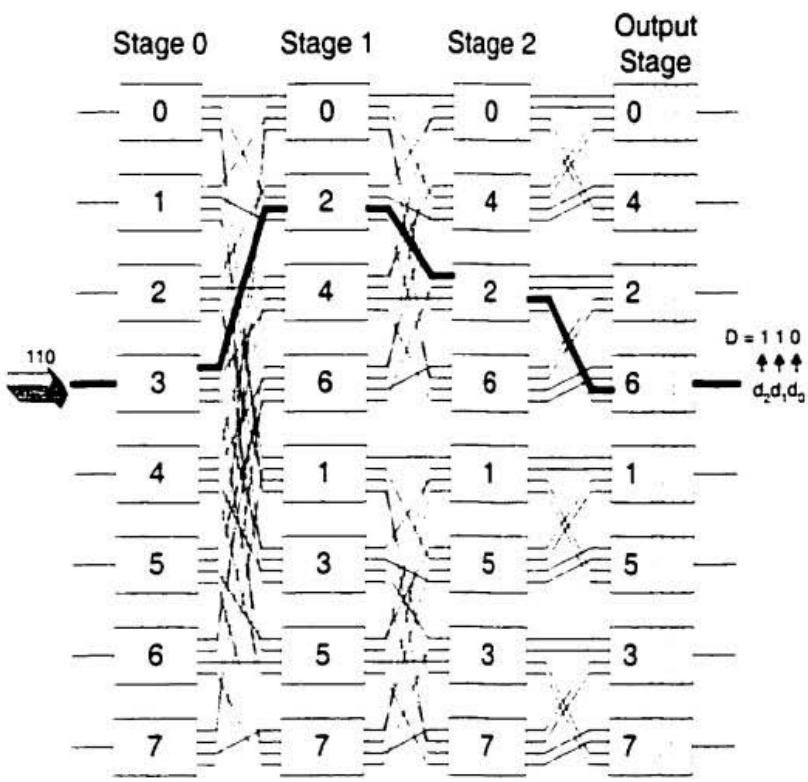


Figure 3.3 Routing in an 8 x 8 BG Network

Here we inspect the switching mechanism that contributes to low cell loss and high performance in BG networks, the **backpressure** strategy. The Backpressure (BP) strategy is a technique in which by means of a suitable backward signaling the number of packets actually switched to each downstream queue is limited to the current storage capability of the queue and the collisions are avoided during the middle stages in a MIN; in this case all the other HOL packets remain stored in their respective upstream queue [3]. The BG network is self-routing, that is, the cells have prepended routing tags that tell the switch fabric which output links the cells should be sent along. According to the BG network switching mechanism, the self-routing tag specifies the required output port of the switch. It is the inverse order of the output port numbers. The bits are used, one per stage, by the switching elements. In BG networks, a switching cycle is composed of two periods, a reservation period (the routing period) and a relaying period. In the reservation period, each input controller sends the self-routing tag representing the cell at the HOL of the buffer that belongs to that input controller. The routing unit in each switching element (SE) uses the arriving self-routing tag to set up a routing pattern for the main cell to be relayed. This process continues until the tag reaches the output controller. The output port acknowledges the arriving self-routing tag. A tag is positively acknowledged if 1) it successfully reaches its output destination, and, 2) there is room for the cell represented by self-routing tag in the buffer of its destination. The acknowledgment signal is returned to the input controller by the switching mechanism. In the relaying period, the input controller passes the body of the cell that was positively acknowledged during the reservation period [7]. We will discuss this in Chapter 4.

3.4 Properties

3.4.1 Hardware complexity

To increase the speed is the main purpose of research on ATM switch fabrics. Therefore, minimum hardware complexity (HC) is desired in the hardware implementation.

The HC of a network is the sum of the HCs of the SEs of the MIN. The HC of a SE depends on the total number the connections between the input ports and the output ports.

In a BG network, there are four connections in a SE in the first stage and there are 4x4 connections in a SE in the intermediate stages. Figure 3.4 [14] shows these connections.

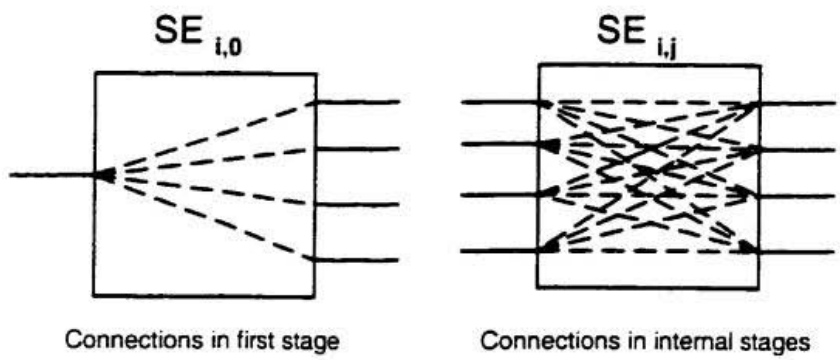


Figure 3.4 Crosspoint Complexity in a SE

So, the HC for a BG network is [14]:

$$HC_{(BG)} = 1 \times 4 \times N + 4 \times 4 \times N \times (\log_2 N - 1)$$

In the above HC calculation, the complexity of the output stage is ignored, because this HC is given as the HC for the switch fabric and the complexity of the buffer stage, that is, the output stage, is usually not counted. Here, connections are the main concern about the HC; in practice, some other factors are also taken into account. These factors may include the number of I/O pins, the interconnection lines, and environmental parameters.

3.4.2 Fault Tolerance Properties

A fault-tolerant MIN is a network that is able to route packets from input ports to the requested output ports, in at least some cases, even when some of its network components (SEs, links) are faulty.

There are mainly two network components in the BG networks: the SEs and the links. The fault tolerance performance is discussed for these two components.

When a link fault occurs, the fault will be notified to the corresponding SEs to which the links are connected. The SE will route cells through the alternative link. If both normal and alternative links for a switching bit from a SE are faulty, the SE informs SEs in Stage (j-1), to which it is connected, to modify the routing tables.

Figure 3.5 shows the dynamic rerouting in an 8 x 8 BG network in case of link faults. For the SE #1 in Stage 1, the first link to route cells with tag bit 0 is broken. So, SE #1 will route cells with tag bit 0 to the alternative link. In the SE #0 in Stage 1, both links routing cells with tag bit 0 are broken. Then, SE #0 notifies the four SEs to which it is connected in the Stage 0 to change their routing table and re-route cells.

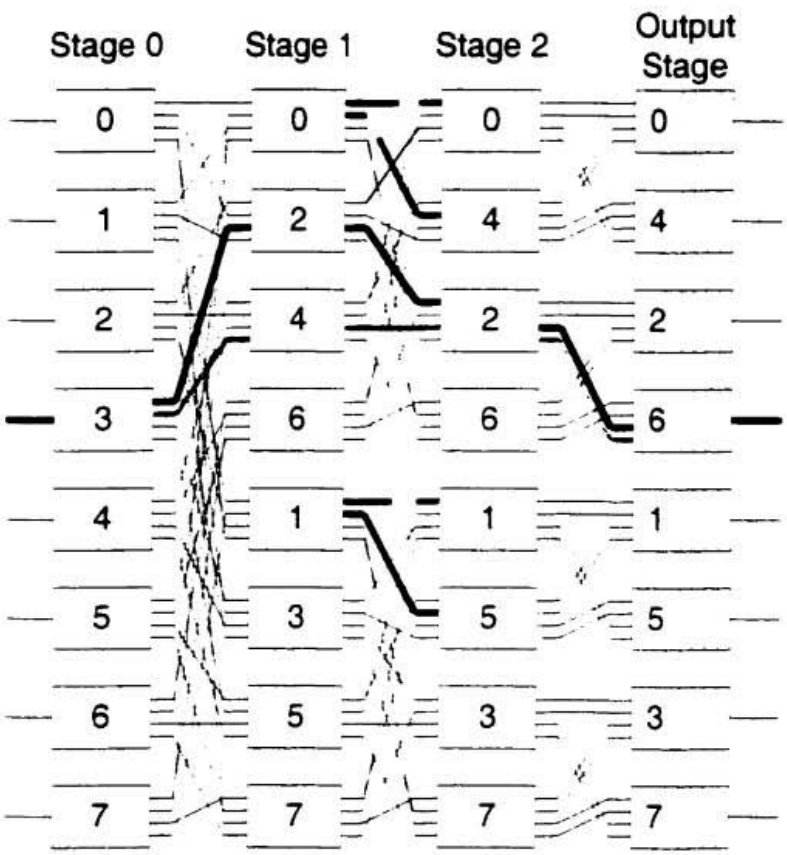


Figure 3.5 Dynamic Rerouting in an 8 x 8 BG Network in Case of Link Faults

We will now consider SE faults; these are similar to link faults with some minor differences in terms of manifestation ad correction. When an SE fault occurs, the fault will be notified to the four SEs connected to it in the previous stage to change their routing tables.

The SEs in previous stage to the fault will route cells through others SEs in the same stage as that of the faulty SE. But, there exist some SE pairs that the failure of both two will lead to loss of full access property which means that there exists at least one path

between any input-output pair in case of faults. These pairs are called critical pairs. For each SE, $SE_{i,j}$ forms critical pair with exactly two other SEs, $SE_{i+2^j,j}$ and $SE_{i-2^j,j}$, for $0 < j < n-1$.

It is obvious that, for a BG network, it can tolerate up to N/2 SE faults in each stage, excluding the first and output stage, and it can tolerate up to N/2 x ($\log_2$N-2) SEs in the whole network. Figure 3.6 shows the dynamic rerouting in an 8 x 8 BG network in case of SE faults. The SE #1 in Stage 2 is faulty. This SE will notify the SEs to which it connects in the previous stage, here, SE #1, #3, #5, #7. These SEs will change their routing table. For instance, if SE #1 in Stage 1 wants to send out a cell with tag bit 0, it will route it to the alternative link that is connected to SE #5 in Stage 2, instead of the normal output link which is connected to SE #1 in Stage 2. Another case: SE #0 and #4 in Stage 2 is a critical pair. If they are both faulty, then the network loses the full access property. For instance, if SE #0 in Stage 1 wants to route a cell with tag bit 0, it cannot choose either of the two links since both SE #0 and #4 are broken. So, the cell cannot be routed.

The grey line in Fig.3.5 shows an example of rerouting for the routing path from SE #3 (Stage 0), SE #2 (Stage 1), SE #2 (Stage 2), to SE #6 (Output Stage).

So, it can be concluded that BG network is a single fault-tolerant MIN and robust. (A network is called single fault-tolerant if it can function as specified by its FT criterion despite any single fault conforming to its fault model. If a network can tolerate any set of i faults, then it is said to be i-fault tolerant. A network that can tolerate some instances of i faults is said to be robust although it is not i-fault tolerant).

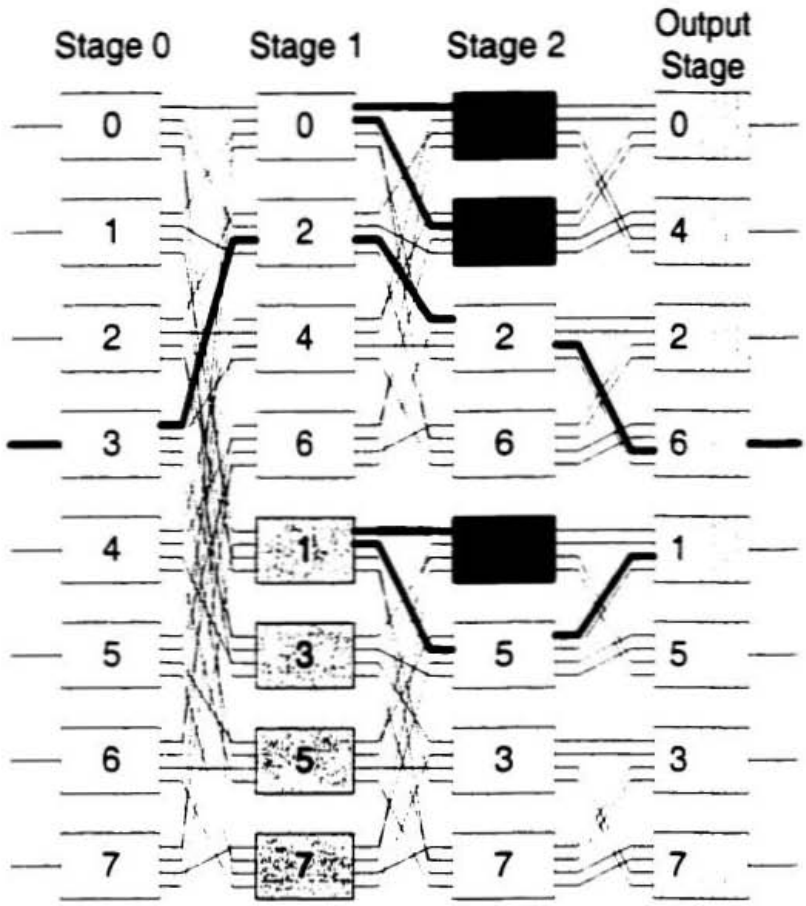


Figure 3.6 Dynamic Rerouting in an 8 x 8 BG Network in Case of SE Faults

3.4.3 Reliability Analysis

High reliability is important for real-time communication systems. There are three main measures that are used to evaluate the reliability performance of MINs. These are terminal reliability, broadcast reliability and network reliability [14].

Terminal reliability (TR) is the probability that there exists at least one fault-free path from a particular input port to a particular output port [14]. TR is always associated

with a terminal path that is one-to-one connection between an input port (the source) and an output port (the destination) [7].

Broadcast reliability (BR) is the probability that at least one fault free path exists from a particular input port to all the output ports [14]. BR is always associated with a broadcast path that is a connection from one source to all destinations in the network [7].

Network reliability (NR) is the probability of maintaining full access capability throughout the network. The BG network loses full access property only when one or more critical pairs fail [14].

Mean time to failure (MTTF) specifies the quality of a system and is the expected time that a system will operate before the first failure occurs. The failure rates from MTTF of the BG network are used to evaluate the above three metrics. The obtained results demonstrate that the BG network is robust and reliable. The detailed reliability analysis is presented in [7].

3.5 Summary

Most of the aspects of BG network are briefly discussed: topology/structure, routing algorithm, hardware complexity, fault tolerance, and reliability. The BG network is a MIN based on 4 x 4 SEs. It adopts a distributed Reverse Distance Tag routing algorithm. It is a single fault-tolerant MIN and robust under multiple faults. It exhibits good reliability. In summary, the BG network is an excellent candidate for switches in the broadband communication networks.

Chapter 4

INPUT CONTROLLER ARCHITECTURE DESIGN FOR BG ATM NETWORKS

4.1 Introduction

In Chapter 2, we reviewed various popular schemes of input controllers for switches. In this chapter, we discuss in detail our input controller (IC) design for BG networks. We firstly give readers an overall picture of the entire IC architecture. Next, we illustrate several key portions of the IC, namely, the input buffers, the routing table (RT) memory, the cache, and the arbiter. For each we describe things such as the structure, the algorithm, the operation, and the reasons why we choose such kinds of structures, etc. At the end of this chapter, we conclude the IC architecture design.

4.2 System Description

As we discussed in the previous chapter, the IC processes the look-up operation and maps the VPI/VCI in the header of the incoming cell into the corresponding output VPI/VCI. At this stage of the switch an internal routing tag is added to each cell. Buffering of cells may also be provided by the IC to prevent cell collisions.

A simplified outline of the IC design for an N x N ATM switch is shown in Figure 4.1, which illustrates the data processing path for a stream of cells entering from the IC left and exiting from the IC on the right.

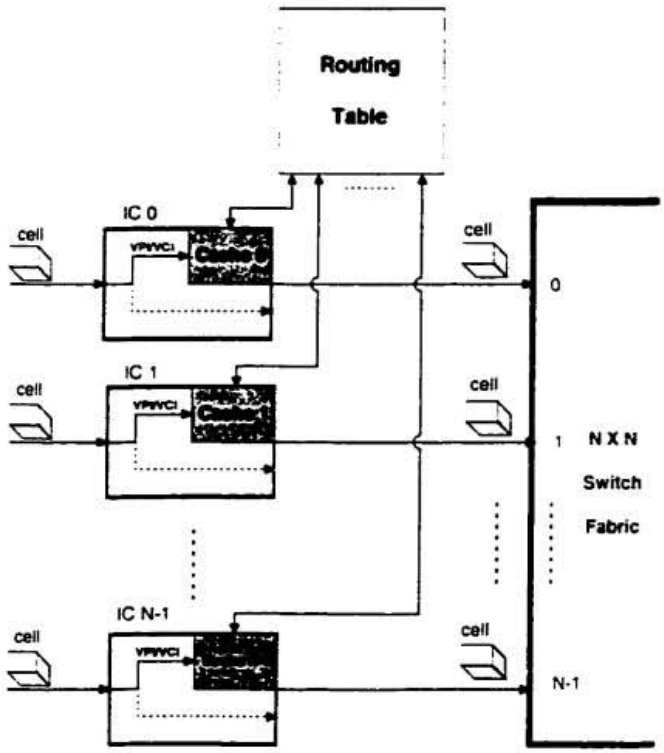


Figure 4.1 Input Controller Architecture

As we know, ATM is a cell-based technology. Data is segmented into 48-byte payloads. A 5-byte header is attached to this payload to form a cell. The header has fields that determine routing, flow control, error control, and other functions. The field that is of interest here is the VPI/VCI (Virtual Path Identifier/Virtual Channel Identifier). These VPI/VCI values are assigned to each section of a connection for the complete duration of

the connection. In the IC architecture, there is a central master routing table (RT) and the satellite caches of modest size with recently used routes. An arbiter logic is used to coordinate the requests from each input port cache to RT. Each input port keeps an input buffer. When a cell arrives at IC, it is temporarily stored in the corresponding input buffer. The header information is retrieved into the cache to look up all information necessary to forward a cell towards its destination. If a route is not in the satellite cache, it would request the relevant route from the central table, RT. IC then rephrases the VPI/VCI and other routing information in the cell header, and forwards the cell to the switch fabric that delivers cells to outgoing lines. In the following sections, we describe each portion of the IC in details.

4.3 Input Buffer Module

In Chapter 2, we mentioned that there are several buffer strategies for ATM switches. It is verified that a pure output buffering is clearly the best solution, but it would require a prohibitively complex switch fabric. A pure input buffered strategy is not acceptable as performance will be throttled due to the HOL blocking problem. Instead, the hybrids of those buffering strategies would lead to a better performance. This conclusion can be drawn from several studies reported in [6]. In BG ATM networks, we use input-output buffering strategy. We gain the capability of the input buffers to combat internal and output blocking. We also gain the reduction of the HOL blocking enjoyed by the output buffering strategy [7]. Detailed information about the input-output buffering for BG ATM switches can be found in [7]. The design of the output buffer portion of this

input-output buffering is also described there. In this section, we discuss the design and the operation of the input buffer portion of the input-output buffering.

4.3.1 Input Buffer Operations

In the N-port switch that we describe here, each of the input ports has an input buffer. It is a first-in-first-out (FIFO) queue. The input buffer interfaces to the transmission system. When a cell arrives, the corresponding input buffer first checks if the buffer is full. If the buffer is full of cells, then the incoming cell is rejected; in a well-designed system, this occurs very rarely, for example, once in ten million times. Otherwise, the cell is injected into the buffer. Meanwhile, a flag is prepended to the cell. When this flag is set, the cell has gone through the routing table (RT) lookup; when the flag is cleared to 0, it indicates that the RT lookup is under service or waiting for the service.

The HOL cell is ready to be sent to the cache related to this input port to lookup routing information. When the lookup is done, along with the updated VPI/VCI value, a self-routing tag is added to the cell. The self-tag values are also stored in the RT with the updated VPI/VCIs. Meanwhile, the flag is set to indicate that the HOL cell has completed the routing information lookup.

The HOL cell with the updated header is ready for picking up by the switch fabric. The input buffer waits for the acknowledgement from the switch fabric to send out the HOL cell. Once the HOL cell is sent out, the cell next to the HOL cell in the FIFO buffer is forwarded to the HOL and would be ready for lookup in the next cycle.

4.3.2 Input Buffer Structure

The input buffers for an N x N switch are n FIFO queues, each with a fixed length of 10 entries. This number has been determined based on our software simulation results that are discussed in the next chapter. Buffering requires adding extra hardware to the system in the form of memory elements and control circuits. The larger the buffer size, the bigger the delay. In next chapter, we will explain the simulation results of buffer length. Each entry consists of a 1-bit flag, a 10-bit self-routing tag, 40 bits cell that includes 24 bits VPI/VCI and 16 bits payload. Here, to simplify the hardware complexity, we use a 2-byte payload instead of the standard 48-byte payload: that is, we use a 40-bit cell, instead of a 53-byte cell. However, this simplification does not affect the performance characteristics we attempt to evaluate in our simulation. The important information, the VPI/VCI, in a cell still remains as 24 bits. The input buffer structure is shown in Table 4.1

	1 bit	10 bits	24 bits	16 bits
Entry	Valid bit	Self-routing tag	VPI/VCI	Cell payload
0				
1				
.				
.				
.				
9				

Table 4.1 Input Buffer Organization

4.4 Cache Memory Module

Cache memory is the key component in our IC. We expect an efficient cache structure optimized for routing. Based on our study of the ATM/Internet traffic attributes and our simulation experiment (reported in the next chapter), we build our cache as a one-level 32-block fully associative cache. Each input port has such a cache and totally there are N caches for an N x N BG switch.

4.4.1 Cache Structure

To build up our cache, we need to address several issues:

- One-level or multi-level cache?
- Separated or distributed cache?
- Cache organization: direct mapped cache? Fully associative cache? Or set associative cache?
- Decide the number of cache entries, and the mapping scheme.

We only consider one-level cache rather than a multi-level cache memory which leads to more searching time and higher hardware complexity. We choose distributed caches rather than a commonly shared cache for a whole switch. This is based on the observation that it will cause severe bottleneck if a commonly shared cache is built since cache is a frequently accessed component. Also the entries in a cache are naturally associated with connections which correspond directly to input ports.

To decide the cache organization, we have carried out much study. Firstly, how well the cache works depends on the localization of the data. We have introduced the concepts of spatial locality and temporal locality in Chapter 2. The study on the Internet traffic streams indicates that they tend to exhibit a low degree of spatial locality, especially when they are occurring on as large and diverse a medium as an Internet router. Once a switch is determined to be on the virtual path or virtual channel (that is, on the route of a session), several hundreds or thousands of cells belonging to that connection will pass through this switch for the duration of the connection. However, if one particular VPI/VCI field is currently accessed, there is no greater probability for the adjacent VPI/VCI to be accessed in the near future than any other VPI/VCI; that is, spatial locality is not present. However, there is a degree of temporal locality. This observation tells us that cache entry size should be small. Since all of the blocks in an entry are referenced by the same tag value, they rely on spatial locality to be useful; low spatial locality means we should give up large entry size. Here, we consider only one block in a cache entry. A cache is valuable provided it achieves at least a modest hit rate. To compare among different organizations, we build up a software simulator to evaluate cache performance of various cache organizations. We test fully associative cache, directly mapped cache, two-way set associative cache and that combined with a victim cache, and four-way set associative and that combined with a victim cache. The victim cache contains only blocks that are discarded from a cache because of a miss – "victim" – and are checked on a miss to see if they have the desired data before going to the next lower-level memory. If it is found there, the victim block is swapped. The simulation

results (given in Chapter 5) show that a fully associative cache gains much better performance in terms of cache hit rate, hardware complexity and time-consumption. Besides, a fully associative cache has the most flexibility in mapping cache blocks. Detailed performance analysis can be found in Chapter 5 of this thesis. Caches are costly. So, caches should be small yet efficient. We have tested different entry numbers, such as 16, 32, 64, 128, 256, etc. and compared the results of cache hit rates. Finally, we decided each cache to have 32 entries.

Based on our study and analysis, we finally conclude that, for an N x N switch, there are N caches, each for an input port. They are all fully associative with 32 entries each. Each entry has only one block. The fields in a cache entry are described as follows:

- Valid_bit (1 bit)

  It indicates an entry in the cache is valid or not in these pre-setup connections.

- tag (24 bits)

  It is a cell's incoming VPI/VCI and serves as a tag to search for cache entries.

- VPIVCI_o (24 bits)

  It indicates a cell's outgoing VPI/VCI related to the tag in the same entry.

- output_num (10 bits)

  In the performance analysis of switch fabrics with port number from 8 to 1024, output_num is 10 bits. In the hardware implementation of an 8 x 8 switch fabric, output_ num is 3 bits.

- access_num

5 bits. It serves as an index of access number of the given entry. Since there are 32 entries in a cache, 5 bits are needed.

So, one cache has 8 bytes x 32 = 256 bytes. In an 8 x 8 switch fabric, the total cache size is 8 x 256 bytes. Table 4.2 depicts the cache organization in the IC.

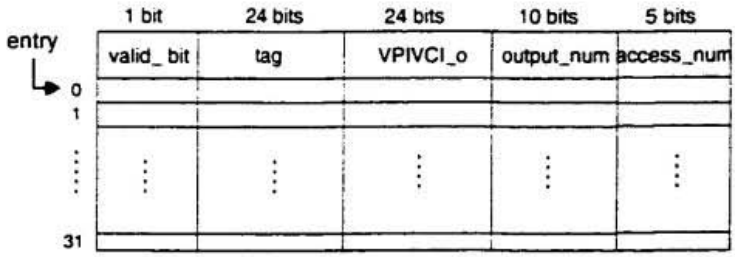
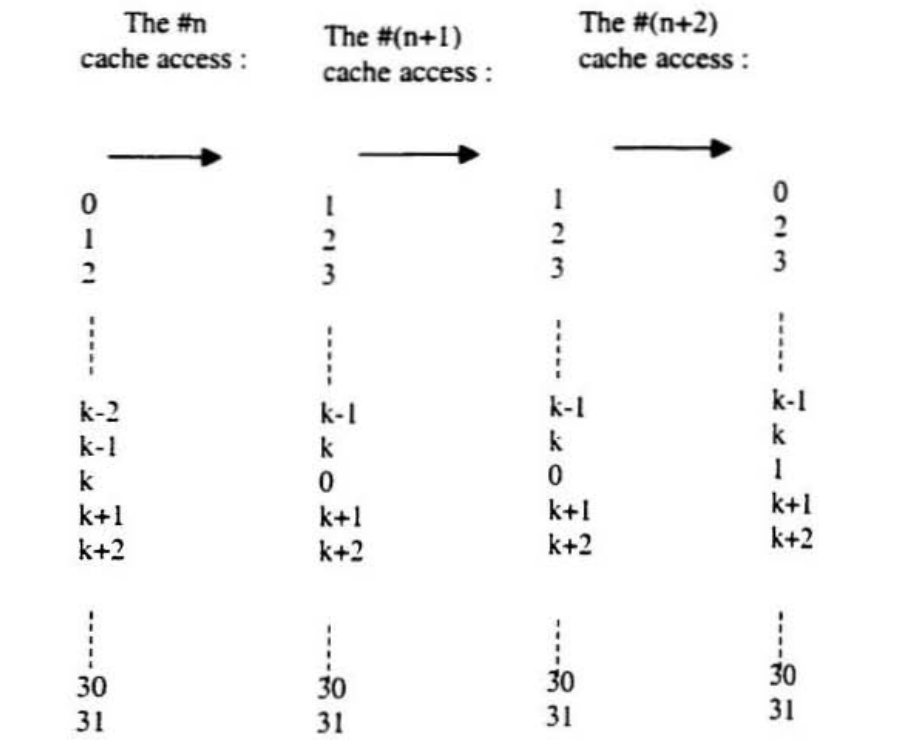


Table 4.2 Cache Organization

4.4.2 Cache Operation

The caches are initialized at the network pre-setup stage. All valid bits are cleared to 0. During the switching cycles, the incoming VPI/VCI from the input buffers are used as tags to match entries. All the n caches of the n input ports work in parallel. If there is a cache hit, then the data in that matched entry can be accessed. The updated VPI/VCI, as well as the output port number, is returned to the related input buffer. If there is a cache miss instead, we start the main RT lookup operation. The RT lookup operation will be explained in the next section. The returned data from RT, the updated VPI/VCI and the output port number, are sent to the input buffer. Meanwhile, the returned data from RT replaces one of the cache blocks.

With respect to the above mentioned "replace", among the various replacement algorithms reviewed in Chapter 2, what we use in the IC design is the least-recently used (LRU) algorithm. In LRU, to reduce the chance of throwing out information that will be needed soon, accesses to blocks are recorded. The block replaced is the one that has been unused for the longest time. It can be seen here that LRU makes use of a corollary of locality: If recently used blocks are likely to be used again, then, the best candidate for disposal is the least-recently used block. Obviously this is consistent to the Internet traffic attribute, the temporal locality. The core of LRU algorithm is to record the cache block access numbers. In a practical hardware implementation, the access number must be finite and reasonably small. A dynamically calculated value is assigned to the access number. Let us firstly assume the access numbers of the block currently accessed to be k. Next we set the access numbers of this block to be 0 and increment access numbers of all other blocks that currently have an access numbers less than k. Those blocks whose access numbers are equal to or greater than k remain. In this way, we keep the access numbers always less then 32, that is, the cache block size. This process is shown in Table 4.3.



- 32 cache blocks listed in access numbers order	- the cache block with access number 'k' is accessed - after operation, k changes to 0, 0 – k-1 increase 1, k+1 – 31 not change	- once again that cache block is accessed - all the access numbers keep the same	- the cache block with cache access number '1' is accessed

Table 4.3 Cache Block Access Numbers Updating Process

4.5 Routing Table Module

The RT data structure contains all the information necessary to forward cells toward their destination. There is a single RT in each ATM switch. When the RT receives the lookup requests, it processes the lookup operation and returns the result.

4.5.1 Routing Table Structure

Our analysis shows that complicated RT structures, such as multi-level memory or sorted memory, are not required for this primary storage. Thus, memory access times and processing time are minimized when the RT structure is as simple as possible, that is, the RT has a low depth in its storage structure. Instead, an efficient cache is the main concern with respect to the performance gain.

To determine the RT size, let us do some calculation first:

For a 32 X 32 ATM switch, if all traffic is voice traffic (bandwidth 64kbps) coming at the line rate of 2.5Gbps, and a traffic load of 0.7, then, the number of VP/VCs is defined by:

$$\frac{2.5Gbps \times 32}{64kbps} \times 0.7 = 875.000 \quad (VP/VCs)$$

Another example: *For an 8 X 8 ATM switch, if video traffic with bandwidth 1Mbps is coming at the line rate of 2.5Gbps, and a traffic load of 0.8, then, the total VP/VCs set up is given by:*

$$\frac{2.5Gbps \times 8}{1Mbps} \times 0.8 = 16.000 \quad (VP/VCs)$$

From above two examples, we can see that, theoretically, for the 24-bit index VPI/VCI, there should be 2^{24} = 16777216 entries in the RT; however, the VP/VCs setup in a given time for a particular switch are very small amount in practice. The 2^{24} VP/VCs denote the traffic all over the world. We also learn from above examples that the VP/VC number is defined by the traffic type, line rate, traffic load, as well as the switch fabric size N. Another point we should note is that the speed of memory degenerates as the size increases, placing another limit on the maximum memory size. In other words, the switch capacity limits the traffic volume passing through. Therefore, keeping a routing table of 2^{24} entries is impractical and prohibitive in terms of memory size. We only need consider those VP/VCs that are set up for a particular switch. Based on a study of the industry experience, for example, Lucent Cajun A500 ATM Switch and Cisco LightStream 1010 Multiservice ATM Switch can cope with 32,000 VCs, respectively [30] [31]; Nortel Networks Centillion 100 ATM-LAN Switch can support greater than 10,000 VCs [32]. We have decided to support maximum 20,000 VP/VCs at any given time. So, the designed maximum RT entries are 20,000.

Obviously, RT memory has almost the same fields as cache memory except no 'access_num' field that indicates the cache block access numbers. The fields of an entry in the RT are described as follows:

- valid_bit (1 bit)

  It indicates an entry in the RT is valid or not in these pre-setup connections.

- VPIVCI_I (24 bits)

  It indicates a cell's incoming VPI/VCI value.

- VPIVCI_o (24 bits)

  It indicates a cell's outgoing VPI/VCI value related to the incoming VPI/VCI in the same entry.

- output_num (10 bits)

  In the performance analysis of switch fabrics with port number from 8 to 1024, output_num is 10 bits. In the hardware implementation of an 8 x 8 switch fabric, output_ num is 3 bits.

Thus, the RT is estimated to occupy around 160k bytes. Table 4.4 depicts the organization of the RT.

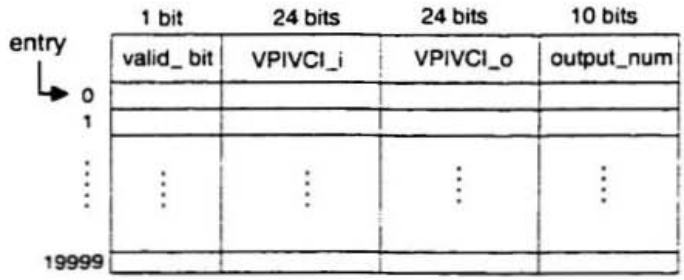


Table 4.4 Routing Table (RT) Organization

4.5.2 Routing Table Operation

The RT is initialized at the network pre-setup stage. The data is updated and valid bits are set for new connections and cleared for previous connections.

During switching cycles, if RT receives a lookup request, the field "VPIVCI_i" of all the RT entries with valid bit '1' are compared with the incoming VPI/VCI. The RT finds the matched entry and returns the result to the relevant cache that issued the request. If no matched entry is found, then the request is discarded.

4.6 Arbiter Logic Module

One concern with our IC architecture is the coordination of a set of lookup requests at the same time. Thus, an arbitration logic, or arbiter, is needed to determine which one of the requests from the caches should be granted to access the RT. There are several possible arbitration mechanisms, such as simple round robin, a specified port assigned, and others. Studying the traffic attributes and switch architecture, we have chosen Random Port Number Select algorithm for the reason of fairness and simple hardware implementation. The arbitration algorithm is described using an example of an 8 x 8 switch as follows:

1) Since there are 8 caches, there can be maximum of 8 simultaneous cache misses. So the maximum of RT lookup requests is 8. $C_0 - C_7$ denote the eight caches, and $R_0 - R_7$ denote the eight requests.
2) Now, let us assume that in a given cycle, C_0, C_2, and C_6 are the only three caches that have cache misses and issue RT lookups. The requests are R_0, R_1, and R_2, respectively.
3) Generate a random number between 0 and 7; Suppose the random number is $110_2 = 6_{10}$.
4) Then, 6 mod 3 = 0. So, select R_0 that corresponds to the C_0 to grant in this cycle.
5) In the next cycle, following the same rule, the arbiter selects another request to grant issued by C_2, C_6; and so on.

4.7 Summary

The IC architecture is a cache-based memory hierarchy. The key function of the IC is to process the RT lookup operation. The design proposed in this chapter combines many beneficial considerations in terms of switch fabric architecture, broadband network traffic characteristic, memory attributes, speed, hardware complexity, etc.

In the end, we make a rough comparison of the IC with one commercial product discussed in Chapter 2, the BBN MGR:

1) The MGR switch is input-buffered. All inputs have a separate FIFO for each output. A scheduler is dedicated to guarantee that there is no HOL-blocking and the switch operates with full throughput [10]. The BG switch utilizes input-output buffering with mainly output buffering. In the IC for the BG network, every input port has a small FIFO buffer. HOL blocking is not a major concern because of the largely output buffering strategy.

2) The MGR is pipelined. So a switch allocator is dedicated to form a switch pattern. The allocator has to choose the way in which N x N possible input-output pairings are to be connected to serve all connections effectively. A limitation of this is that it cannot make one-to-many transfers. Thus, for multicast switches, multicast packets are copied to each line card separately. The IC of this thesis is a non-pipelined structure. So no hardware of allocator or scheduler is needed. This largely simplifies the hardware. Multicast can be realized flexibly, either making cell duplication inside switch elements in the switch fabric, or inserting a copy network before the switch fabric.

3) The MGR is implemented using a RISC processor and features three level caches, the code, the cached route entries, and the complete forwarding table, respectively. This structure is similar to the 2-level cache-memory hierarchy of the IC that follows ASIC design flow. But no cache performance result of the MGR is reported. So we cannot continue the comparison.

In the next chapter, we will show some simulation results to assess the IC design.

Chapter 5

PERFORMANCE ANALYSIS OF THE INPUT CONTROLLER

5.1 Introduction

In this chapter, we discuss the performance of our Input Controller (IC), the design of which was described in Chapter 4. The parameters used to measure the performance are cell loss ratio, cache hit rate, cache miss rate, input buffer requirements, etc. We use both uniform and non-uniform traffic patterns to study the IC performance. Among these patterns, we choose URT and bursty traffic loads. The organization of this chapter as follows: we first provide a survey of traffic patterns in broadband communication networks. We mainly focus on the URT and bursty traffic loads. Next, we describe our simulator software built for the performance analysis. The simulator also includes the traffic load generator module. Then comes the simulation results and analysis under the URT and bursty traffic loads. We finally conclude with the results of the performance analysis for the IC.

5.2 Traffic Modeling

5.2.1 Traffic Patterns in Broadband Communication Networks

In the communication systems, **throughput** is a key parameter in performance analysis. It is defined as [14]:

$$throughput = \frac{total \cdot number \cdot of \cdot delivered \cdot cells}{total \cdot number \cdot of \cdot inputted \cdot cells} \quad \text{(in a given period of time)}$$

Since an ATM network is a unified network integrating various services, such as video, voice, data, and other payloads. So, the performance analysis should be under different traffic types. Studying the traffic patterns used in the switch fabric of broadband packet switch architectures could give a better understanding of the performance of the IC system. It may not be possible to exactly simulate realistic traffic loads. However, some certain traffic patterns may help in understanding performance under realistic traffic patterns.

The traffic patterns expected in broadband packet switch architectures can be classified into two categories, uniform traffic patterns and non-uniform traffic patterns [14]. We present several popular traffic patterns under these two categories.

5.2.1.1 Uniform Traffic Patterns

The two main uniform traffic patterns that are being studied are permutation traffic and uniform random traffic.

Permutation Traffic

The permutation traffic pattern refers to the case where the output ports requested by packets arriving to the switch in the same time slot are distinct from one another. It is referred to as permutation traffic because, at full load, the list of requested destinations ordered by input lines forms a permutation of the set (0, 1, ..., N-1).

The amount of internal blocking in the MIN is distinctly visible under permutation traffic pattern. In the case of permutation traffic, at full load, only one packet is destined

to each destination. So packets will be lost only due to internal blocking in the network and not due to output contention at a particular destination.

Uniform Random Traffic

The uniform random traffic (URT) pattern refers to the case where each output port has equal probability of being requested by packets arriving at the input ports. As destination addresses can be repeated, it is referred as uniform random traffic.

The performance of MINs used in multiprocessor interconnections and in the switch fabric of broadband packet switch architectures is usually studied under uniform random traffic. It is a much simpler traffic pattern to be analyzed as compared to the non-uniform traffic types, yet much more realistic traffic pattern when compared to the permutation traffic pattern described above.

5.2.1.2 Non-uniform Traffic Patterns

Much research has been carried out and is still ongoing to determine the traffic types closer to the real traffic. Although it is quite difficult to model, simulate and analyze the exact traffic expected in broadband communication networks, researchers have come up with certain tractable non-uniform traffic types and these patterns are more realistic than simple permutation traffic and uniform random traffic.

The most widely studied non-uniform traffic types are hotspot traffic, community of interest traffic and bursty traffic.

Hotspot Traffic

Hotspot traffic is defined as the traffic type in which one or more nodes of the switch fabric or of the destinations receive a given percentage of the incoming packets. The rest of the incoming traffic is of the type uniform random. These nodes or destinations are referred to as hotspots. Hotspot traffic is also referred to as output-concentration traffic.

Community of Interest Traffic

The traffic pattern of Community of Interest Traffic is described in such a way that a certain percentage of the traffic arriving at a certain input(s) is always directed to certain output(s). The remaining traffic originating at these inputs are of the uniform random traffic type. Uniform random traffic type is expected at all other inputs.

In this traffic type, certain output request patterns cause degradation of throughput primarily due to contention on internal links as opposed to output conflicts. This occurs when the number of community of interest input – output pairs increases. Due to this, more path conflicts occur in the intermediate stages of the MIN. The throughput degradation is more pronounced in case of large sized networks.

Bursty Traffic

Bursty traffic is a traffic type in which the inputs of the switch receive sudden bursts of packets. A popular method to approximate bursty traffic is as follows: the traffic source at each input port alternates between active and idle periods with two independent geometrical distributions of predefined average burst lengths. The active period has cells for m continuous cycles is called the **burst length**. The idle period is also referred to as a **burst gap** as it occurs between two active periods that have bursty traffic. Cells arriving

at an input port within the same burst are always directed to the same output port and are separated by fixed or random spacing. It is assumed that the active periods have a probability p and thus the idle periods a probability of $1-p$. The burst length for an input port is not constant because the burstiness is assumed to be caused due to different payloads that are to be integrated in broadband communications.

Here are some attributes of bursty traffic. The gap between bursts under full load bursty traffic is assumed to be zero. It is also assumed that the distribution of burst lengths is the same for all bursts arriving on any given input port, and burst lengths and gaps between bursts are drawn independently from geometric distributions with mean L packets/burst and L packet slots/idle period . The output port requested by a burst is assumed to be uniformly distributed over all the output ports independent of all other bursts entering the switch.

5.3 Simulator Software

To evaluate the performance of the designed IC, we have developed a simulator software using the C++ language. The simulator has two main capabilities, traffic generation and IC function simulation. The goal of the simulation is two. One is to ascertain that the designed architecture has good performance; the other is to gain a clear understanding of the relationship between the various modules in the IC structure that facilitates hardware implementation. We now discuss the simulation in detail.

5.3.1 Traffic Generation

Among the traffic patterns mentioned in Section 5.2, we select URT as the representative of the uniform traffic patterns and bursty traffic as the representative of the non-uniform traffic patterns to evaluate the IC performance. We select URT because it is simple, feasible to model and more realistic than permutation traffic pattern. It can be a good start point to study the effect of traffic patterns. As for the bursty traffic, since the expected traffic loads in broadband communication networks may mix voice, video and data, etc., the bursty traffic is flexible enough to accommodate most of the existing traffic sources with a reasonable accuracy.

It is obvious that the variability of the bit rate and the connection mode (connection-oriented or connectionless) are important characteristics of an end-to-end path. Besides, timing information is another important factor that needs to be preserved from source to destination. Different traffics have various requirements of the above three factors. ATM has the ability to guarantee class of service to different applications, that is, to map different information into cells. This means that with decent planning, users with very different application needs (voice, video, several kinds of computer data) can use the same network links and be satisfied with the service, and thus save money. This is why ATM is referred as the 'unification technology'. According to ATM Forum, which is a commercial association of ATM equipment manufacturers and researchers, and the Internet Engineering Task Force or IETF, which is the Internet's standardization body, the ATM traffic can be subdivided into four classes in terms of the three criteria [15] [29]:

(1) **Constant Bit Rate (CBR)** sources.

This type consists namely of voice and some video sources, e.g. telephony, voice mail, and some telemedicine applications. CBR sources require a sustained amount of bandwidth, low latency, and a low cell delay jitter.

(2) **Variable Bit Rate (VBR)** sources.

This type is mainly the result of multimedia applications. It models applications that generate traffic in bursts, rather than in smooth stream. The peak bit rate and the allowed 'burstiness' of the traffic will have been prearranged between the network provider and the user. VBR application can tolerate higher delays and higher delay variations than can CBR sources.

(3) **Available Bit Rate (ABR)** sources.

These mainly include computer related data sources, such as file transfer, electronic mail, terminal emulation, or LAN emulation (LANE) over ATM. These sources do not require any specific bandwidth and have widely varying latency requirements and cell delay jitter. Instead of defining in advance the bandwidth that a customer can use, the system provides flow control from the network to limit the information flow from the customer's terminal to a rate that the network can accept. The bandwidth available to the customer is therefore subject to change as the network congestion changes. This allows the network provider to exploit otherwise unused bandwidth on the network.

(4) **Unspecified Bit Rate (UBR)** sources.

A UBR source neither specifies nor receives a bandwidth, delay, or loss guarantee. This corresponds to the 'best effort' traffic. Currently, the Internet provides only this type of traffic, and does not facilitate quality guarantees.

Traffic modeling is the first step of traffic generation. It should be noted that by the term 'model' for a traffic source we shall be referring to an algorithm giving the generation time X_i of the ith cell, for i=1, 2, ...; almost always, the X_i's will be taken as random variables [16].

For URT generation, the only parameter we need to know is the traffic load rate. This traffic load rate determines whether or not there is a cell at an input line in a given cycle.

Many models exist to describe bursty traffic. The most general model of an ATM traffic source would be the one that takes into account the entire history of cell generation so as to determine the time for the next cell to be generated. Such a model would be very complicated to describe, as well as very hard to fit to actual sources; it would also be analytically intractable. The model we use is the popular ON/OFF model.

According to the ON/OFF model [17] [18], during the lifetime of a virtual connection, the cell stream from a single ATM source is modeled as a succession of active and silence periods. Cell generation occurs, or the source transmits cells, during the active periods, i.e. the ON periods, as opposed to the OFF (silence/idle) period, which involves no cell generation. Different active and/or silence periods are assumed independent of each other. The cells generated during the same ON period form a **burst.**

In almost all of the related work, each of these periods is taken either as an exponential or a geometric random variable, depending upon the choice of the time axis as either continuous or slotted. More general distributions can also be assumed. The length of the active period is denoted by L_{on}, while that of the silence period is denoted by L_{off}. During the active period, one cell is generated every cycle time T, with the first cell emerging T after the beginning of this period. A scenario of cell generation pertaining to an ON/OFF traffic source is depicted in Figure 5.1 [16].

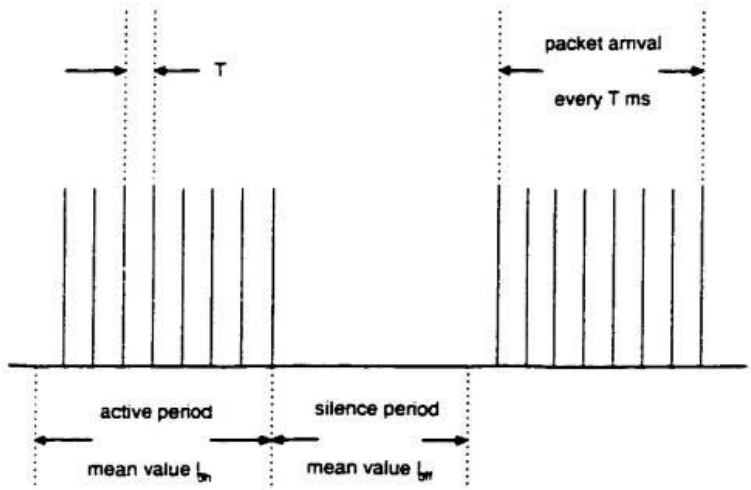


Figure 5.1 the ON/OFF Source Model

In our work, the lengths of ON and OFF periods are independently evaluated from geometric distributions given by [17] [7]:

$$L_{on} = 1 + \lceil \frac{\ln(1-R)}{\ln(1-p)} - 1 \rceil, \quad (1)$$

$$L_{off} = \lceil L_{on} \left(\frac{1-\rho}{\rho} \right) \rceil . \quad (2)$$

Where: L_{on} is the ON period length in cells,

L_{off} is the OFF period length,

$0 \leq R < 1$ is the random number generated,

$0 < p < 1$ is the inverse of the average period length in cells, and

$0 < \rho < 1$ is the network traffic load.

It is obvious from the above equations L_{on} and L_{off} are equal to or greater than 0. But the lengths should be equal to or greater than 1, that is, $L_{on} \geq 1$ and $L_{off} \geq 1$. In the experiments, we found that the ON-OFF model could be strictly followed only if $L_{on} \geq 3$, $L_{off} \geq 1$. Also in order to regulate the generated traffic as close to the assigned traffic load as possible, the above equations should be modified as follows:

$$L_{on} = 4 + \left\lceil \frac{\ln(1-R)}{\ln(1-p)} - 1 \right\rceil, \tag{3}$$

$$L_{off} = \left\lceil 1 + L_{on} \left(\frac{1-\rho}{\rho} \right) \right\rceil, \tag{4}$$

Therefore, the bursty model used here is a variation of the conventional ON-OFF model.

The burst length for an input port is not constant because the burstiness is assumed to be caused due to various payloads that are to be integrated in broadband communications. The cells arriving at each input port in a burst are destined to the same output port. The output is selected randomly.

Despite its simplicity, the ON/OFF model can be used for traffic modeling of broadband communication networks. There is a general belief that, with the exception of

CBR sources, all other traffic sources exhibit this ON/OFF behavior. For CBR sources, there is no idle period [18]. Also, given its analytical tractability, the ON/OFF source model is by far the most popular in performance evaluation work [16].

Input Controller Simulation

We now proceed to outline the IC simulation software.

The main data structures used are:

- ATM cell structure

```
struct cell_struct{
        long VPIVCI;    //the 24_bit VPI/VCI in ATM header
        int routingTag; //(logN)_bits for self-routing tag, the reverse of the
                        // output port number of the switch fabric
};
```

- Cache structure

```
struct cache_struct{
        bool valid_bit;     //1_bit valid bit
        long tag;           //24_bit tag
        long VPIVCI_o;      //24_bit updated VPI/VCI
        int output_num;     //10_bit output port number
        int access_num;     //5_bit record of cache accessing times
}
```

- Routing table structure

```
struct memory_struct{
        bool valid_bit;     //1_bit valid bit
        long VPIVCI_i;      //24_bit incoming VPI/VCI
        long VPIVCI_o;      //24_bit updated VPI/VCI
        int output_num;     //10_bit output port number
}
```

Since this simulator is developed in modular fashion, the IC functions are simulated in several main procedures:

- void cache_initial (cache_struct** cache)

  //This procedure is to perform the cache initialization at the IC system start. The valid bit and cache access number are reset.

- void RT_initial (long *header_in)

  //This procedure is to initialize the RT at the IC system start. The values of the pairs of the incoming VPI/VCI and the updated VPI/VCI are assigned to the RT entries, as well as the output port numbers. All this routing information is assumed to be given by the higher network-level routing mechanism. The valid bit is set.

- bool lookup (long addressy, const int portNum, cache_struct *Pcache, long

*VPIVCI_o, int *output_num)

//This procedure is to implement the cache lookup during the IC system running. The incoming VPI/VCI is used as the tag to lookup in the cache. If cache hits, that is, the tag matches the VPI/VCI in a certain cache block, and the valid bit of the corresponding cache block is set, the updated VPI/VCI and the output port number in that cache block are returned. Meanwhile, the cache access numbers of the whole cache block are updated following the given rule. If cache misses, that is, either the tag does not match any VPI/VCI in the cache, or the corresponding valid bit is not set, though the tag may match the VPI/VCI. When cache miss happens, the IC system first uses LRU algorithm to find the cache block number to be replaced later. Then, the incoming VPI/VCI is used as tag to process lookup in RT. The found updated VPI/VCI and the output port number are returned. At the same time, this found content is written back to the cache block whose block number is calculated previously using LRU algorithm. The cache access times of the whole cache blocks are updated following the given rule.

- bool memory_op (long addresssy, long* VPIVCI_o, int* output_num)

//This procedure is to implement RT lookup function during IC system running. When IC cache miss happens, the system converts to RT lookup procedure. The incoming VPI/VCI is used as the tag to check through all the RT entries. If one entry matches, the updated VPI/VCI value and the output port number in the

corresponding RT block are returned. Otherwise, invalid VPI/VCI message is output.

- int LRU (cache_struct * Lcache)

  //This procedure is to implement the LRU algorithm when cache miss happens during IC system running. This is done by searching for the least-recently used cache block number based on the records of the cache access times. The corresponding cache block number is returned.

To evaluate the IC performance, some important parameters are necessary to model the traffic and simulate the IC using the developed simulator software. They are:

- Switch fabric size N:

  The simulator can handle switch fabric size N from 8 to 1024.

- VC/VP numbers:

  The simulator can handle a maximum number of VC/VPs 20,000 set up at any given time.

- Network traffic load ρ:

  The simulator can handle network traffic load ρ from 0.1, 0.2, ..., to 0.9.

- Switching cycle number;

- Traffic type:

  Currently the traffic types that the IC simulator uses are URT and bursty traffic.

- Mean burst length, if bursty traffic:

This parameter is expected to be a positive integer.

5.4 Simulation Results and Analysis

In this section, we analyze the performance of the input buffer and the cache-memory hierarchy described in Chapter 4 based on the simulation results. The simulation of the designed input buffer is done in conjunction with the BG switch fabric simulator, which has been developed at Memorial University of Newfoundland to evaluate the performance of the BG switch fabric. The criteria are the input buffer number, the output buffer number, and the cell loss rate, under different traffic loads. The simulation of the IC is done under the simulator mentioned in Section 5.3. The main evaluation criterion is the cache hit rate.

5.4.1 Input Buffer Number

According to the description in Chapter 4, the input buffer number is expected to be as small as possible. So the hardware cost can be decreased while the high system performance can still be gained.

Table 5.1 – 5.4 depict the number of input buffers and output buffers tested under 100.000 switching cycles, under different URT traffic loads for a switch fabric with N ranging from 8 to 64. The results imply that, for BG networks, only small numbers of input buffers would be sufficient to achieve high system performance if traffic can be assumed to be URT. Even through the number of input buffers increase a little while the traffic load and/or the size N grow, the number of the input buffers is still very small.

In the above simulations, we do the following procedures to try to find the proper number of input buffers, meanwhile keeping the cell loss rate as small as possible, and even towards to zero. We explain it by giving an example of 0.5 URT traffic load for an 8 x 8 switch. First, we choose 5000 input buffers, 5000 output buffers and 1000 cycles. We then run the simulation for five times to find out the output buffer numbers needed from the simulation reports. Next, still using 1000 cycles, we run the simulation for five times with selected output buffer numbers in above step. By so doing, we can determine the input buffer numbers needed. Finally, we run the simulation at 100,000 cycles, with the selected input buffer number and output buffer number in the above two steps. We check the cell loss ratio. We apply proper adjustments to both the input buffer number and the output buffer number, if necessary, and repeat the simulations until the cell loss ratio is as small as required. We repeat this procedure for 0.6, 0.7, 0.8, and 0.9 URT loads for an 8 x 8 switch fabric. Then we can conclude the input buffer number needed for an 8x8 switch fabric under URT traffic. We can also conclude the input buffer numbers needed for 16 x 16, 32 x 32, and 64 x 64 switch fabric under URT traffic in the same way. Table 5.5 and Figure 5.2 show the conclusion of the number of input buffers under URT for different sizes of switches.

Using the simulation method introduced in Table 5.1 – 5.4, we measure the number of input buffers and output buffers tested under 100,000 switching cycles, under different bursty traffic loads, with mean burst length of 15, for switches with N ranging from 8 to 64. Tables 5.6 – 5.9 show the corresponding results. Table 5.10 and Figure 5.3 summarize the results. The simulation results imply that only a small number of input

buffers is needed for BG switches such that a small cell loss ratio can be obtained when the network traffic loads are 0.8 or lower. Traffic loads of 0.9 and above require inordinately high numbers of input buffers, as HOL blocking dominates in these cases. Therefore, under bursty traffic conditions, the load must be kept at 0.8 or lower so that reasonably sized input buffers can be employed.

Load	No. of Input Buffer	No. of Output Buffer
0.5	1	9
0.6	2	9
0.7	2	14
0.8	3	20
0.9	3	41

Table 5.1 Number of Input Buffers and Output Buffers under Different URT Traffic Loads for Switches with N = 8

Load	No. of Input Buffer	No. of Output Buffer
0.5	2	9
0.6	2	14
0.7	3	17
0.8	3	26
0.9	6	47

Table 5.2 Number of Input Buffers and Output Buffers under Different URT Traffic Loads for Switches with N = 16

Load	No. of Input Buffer	No. of Output Buffer
0.5	2	11
0.6	2	21
0.7	3	20
0.8	4	50
0.9	7	92

Table 5.3 Number of Input Buffers and Output Buffers under Different URT Traffic Loads for Switches with N = 32

Load	No. of Input Buffer	No. of Output Buffer
0.5	2	16
0.6	3	22
0.7	4	30
0.8	5	42
0.9	9	100

Table 5.4 Number of Input Buffers and Output Buffers under Different URT Traffic Loads for Switches with N = 64

Load / Switch Size	0.5	0.6	0.7	0.8	0.9
8x8	1	2	2	3	3
16x16	2	2	3	3	6
32x32	2	2	3	4	7
64x64	2	3	4	5	9

Table 5.5 Number of Input Buffers under URT for Different Sizes of Switches

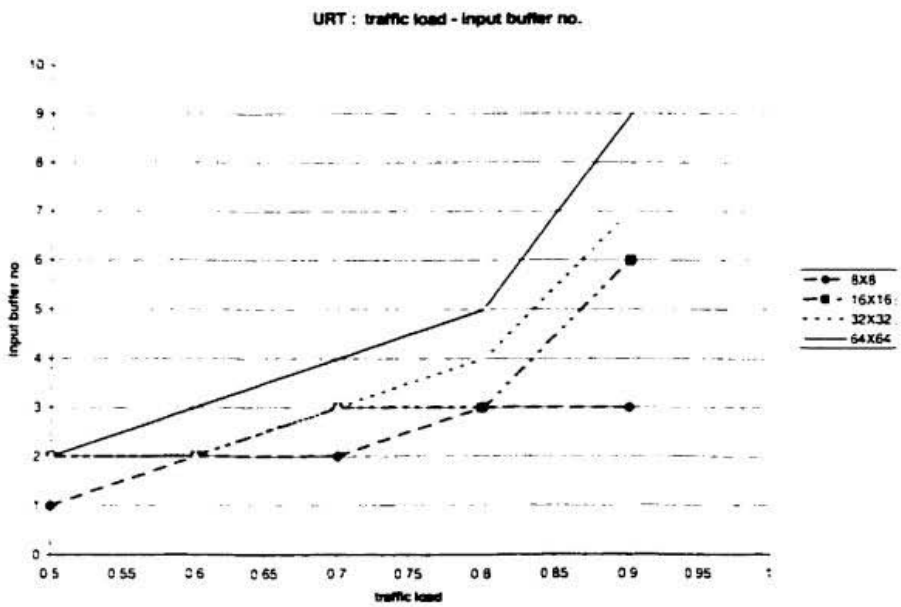


Figure 5.2 Number of Input Buffers under URT Traffic for Different Sizes of Switches

Load	No. of Input Buffer	No. of Output Buffer
0.5	13	89
0.6	11	139
0.7	12	131
0.8	14	211
0.9	1502	1309

Table 5.6 Number of Input Buffers and Output Buffers under Bursty Traffic with Mean Burst Length = 15 under different Traffic Load, for Switches with N = 8

Load	No. of Input Buffer	No. of Output Buffer
0.5	14	94
0.6	12	111
0.7	15	171
0.8	27	298
0.9	1534	2690

Table 5.7 Number of Input Buffers and Output Buffers under Bursty Traffic with Mean Burst Length = 15 under different Traffic Load, for Switches with N = 16

Load	No. of Input Buffer	No. of Output Buffer
0.5	13	89
0.6	16	119
0.7	15	141
0.8	29	292
0.9	1703	4077

Table 5.8 Number of Input Buffers and Output Buffers under Bursty Traffic with Mean Burst Length = 15 under different Traffic Load, for Switches with N = 32

Load	No. of Input Buffer	No. of Output Buffer
0.5	14	94
0.6	13	139
0.7	16	147
0.8	32	317
0.9	1908	4500

Table 5.9 Number of Input Buffers and Output Buffers under Bursty Traffic with Mean Burst Length = 15 under different Traffic Load, for Switches with N = 32

Load / Switch Size	0.5	0.6	0.7	0.8	0.9
8x8	13	11	12	14	1502
16x16	14	12	15	27	1534
32x32	13	16	15	29	1703
64x64	14	13	16	32	1908

Table 5.10 Number of Input Buffers under Bursty Traffic for Different Sizes of Switches with Mean Burst Length = 15

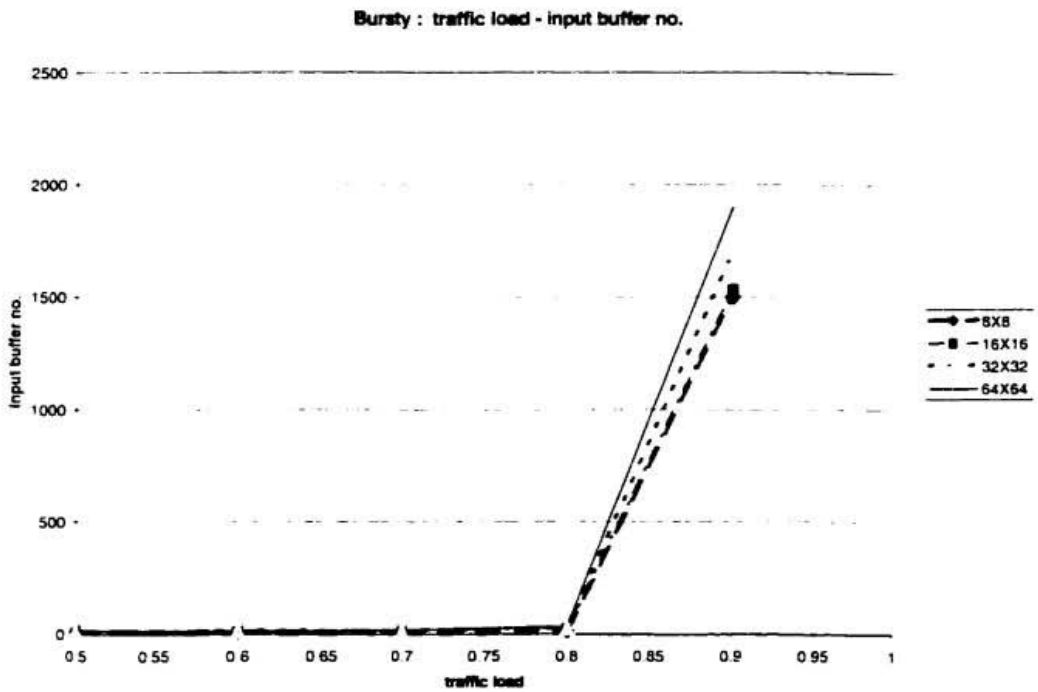


Figure 5.3 Number of Input Buffers under Bursty Traffic with Mean Burst Length = 15 for Different Sizes of Switches

5.4.2 Cache Performance

As discussed in Chapter 4, we aim at building high performance caches. The higher the cache performance is, the better the IC system performance is. Cache hit rate is the major criterion to evaluate the cache performance. Figure 5.4 depicts the cache hit rate comparisons under various traffic types and switch fabric sizes N, given a traffic load of 0.8 and the number of VP/VCs of 9000.

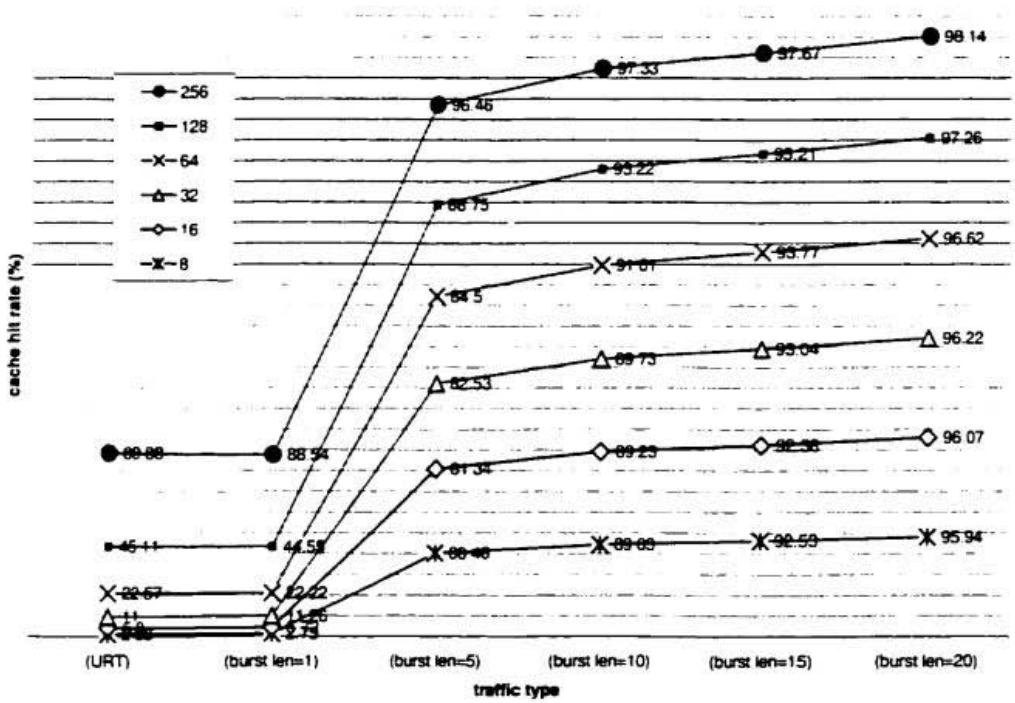


Figure 5.4 Cache Hit Rates under Various Traffic Types and N

For example, for a 32 x 32 switch fabric, when bursty traffic comes, the cache hit rate is 22.22% if the mean burst length is 1, 82.53% if the mean burst length is 5, 96.22% if the mean burst length is 20. When URT traffic comes, the cache hit rate is 22.57%.

Theoretically, URT is a special case of bursty traffic with mean burst length 1. But in traffic modeling, we have modified the bursty traffic model to strictly follow the ON-OFF model. Therefore, we can see the little difference between the hit rate 22.22% of bursty traffic with mean burst length 1 and the hit rate 22.57% of URT. From Figure 5.4, we can conclude that, for a given switch, for the bursty traffic, the longer the burst length is, the higher the cache hit rate is. This is logical since all the cells in a certain burst have the same output VPI/VCI.

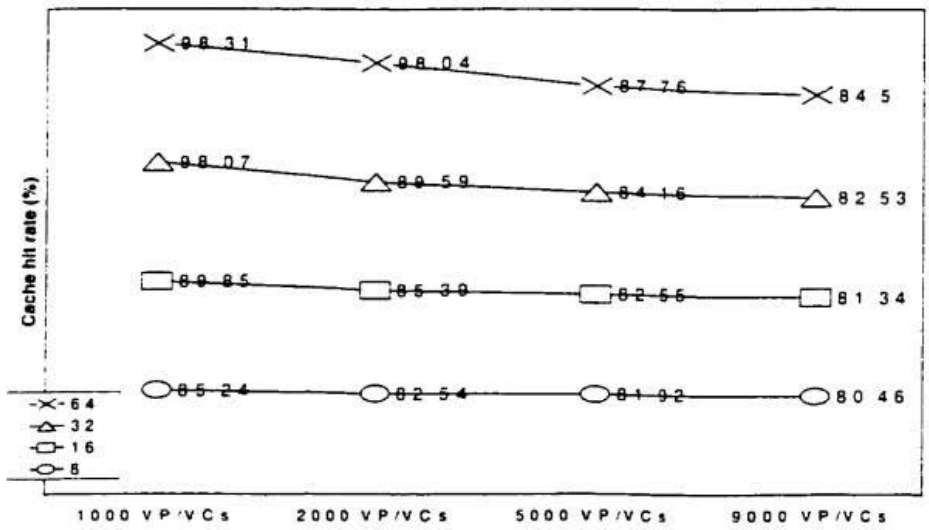


Figure 5.5 Cache Hit Rates under Various VP/VC Number and N

Figure 5.5 depicts the cache hit rate comparisons under various VP/VC numbers and switch fabric sizes, given bursty traffic with mean burst length 5 and traffic load 0.8. The curves in this figure imply that, for a given switch fabric under certain bursty traffic, if the numbers of VP/VCs increase, the cache hit rates decrease. Nevertheless, the cache

hit rates are kept very high at more than 80%, for all sizes N and for as many VP/VCs numbers as 9000, even for burst length 5.

From Figure 5.4 and Figure 5.5, it can also be concluded that the cache performs better in larger switches because, for the same number of total VP/VCs, larger N means fewer VP/VCs at each port, and temporal locality is more strongly obeyed.

According to the studies on Fix West (a commercial US core router, a major inter-exchange point in the Internet) core routers, the hit rate for the 12000-entry centralized cache under realistic traffic can reach 95% [19], and in the industry, the hit rate of 50% is still an optimistic rate, e.g., in Ascend GRF 400 Multi-gigabit IP Switch [20]. From our results, when under bursty traffic with burst length = 10 which is close to realistic traffic, a hit rate 95% can be achieved. The cache hit rate cannot reach 100% due to the cache compulsory miss which defines system cold start misses or first reference misses. These results show that the designed cache architecture works well. The Input Controller performance should be improved when cache hits occur most of the time. It is also interesting to note that the cache performance improves with increase in burstiness. This is reasonable because temporal locality is followed better in the bursty case. Therefore, the IC design proposed here works especially well if the anticipated traffic is bursty.

5.5 Summary

Evaluation of performance is very time-consuming because it requires huge data processing to simulate the realistic situation. The numerical results in this chapter have demonstrated the efficient performance of the IC. The simulation results show that, with

very small numbers of input buffers, the BG networks can still achieve low cell loss rates. The simulation also shows that the designed cache-based IC architecture works efficiently and the system performance can be improved when the cache performance is good. The simulator not only helps us figure out the details of the interactions between different pieces of the design, but works as a tool to decide the cache and the memory size. Only with a working simulation can a full hardware design be undertaken.

Chapter 6

HARDWARE IMPLEMENTATION OF THE IC

6.1 Introduction

In this chapter we introduce the hardware implementation of the input controller (IC) for BG ATM networks. We first describe the hardware implementation methodology recommended by Canadian Microelectronic Corporation (CMC). Then we illustrate the architectural design of the IC in detail. Following the design flow, we describe the overall picture of the IC architecture and then the structure of each module in the design hierarchy. Finally, we talk about the simulation and synthesis for the individual modules and for the whole IC system.

6.2 Hardware Implementation Methodology

In broadband communication hardware designs, achieving high speeds is among the primary missions. So, the design method of application-specific integrated circuits (ASIC) is applied to the design process. A chip designed for a particular product or application are called an ASICs. An ASIC has many advantages that make it a good fit in our design: reducing the total component and manufacturing cost of a product by reducing chip count, physical size, and power consumption, and the higher performance, etc.[21]. Field programmable gate arrays (FPGAs) are another integrated circuit technology which features lower quantity production costs and faster turnaround design

time. However, an FPGA cannot meet the speed requirements in some broadband communication applications.

The ultimate goal of our work is to allow a seamless transition from algorithm design and architecture specification to the final IC implementation. Our approach to achieve this goal is following the design flow recommended by Canadian Microelectronic Corporation (CMC) for deep submicron (DSM) technologies. As depicted in Figure 6.1 [22], using Synopsys VSS (VHDL System Simulation), this first simulation is performed to verify that the function of the very-high-speed integrated circuits hardware description language (VHDL) behavioural register transfer level (RTL) model matches the specified requirements. If the requirements are met, then the VHDL RTL model can be synthesized to produce a VHDL gate-level circuit description. If the requirements are not met, then it will be necessary to modify the RTL model, and/or relax the specifications, and then repeat the simulation. The modification loop must be repeated as many times as it is necessary to acquire a satisfactory simulation.

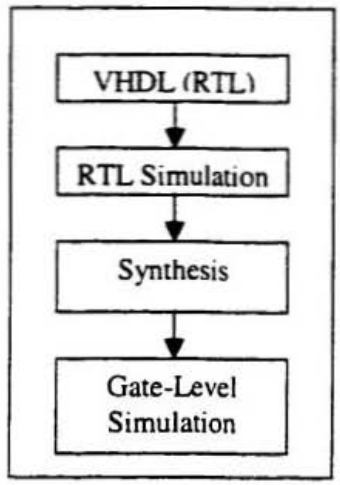


Figure 6.1 Design Flow Recommended by CMC

The IC system is described and modeled using VHDL at RTL level. VHDL is well established in the hardware design community and is known as a strong specification language. Initially targeted to circuit-level synthesis, tools and methodologies have evolved to synthesize designs from the behavioural level [23]. Simulation and synthesis tools, VHDLAN and DESIGN_ANALYZER, supplied by Synopsys, are powerful tools used in the implementation. The VHDL simulator in V-System, supplied by Model Technology, is another helpful tool we use.

In the IC implementation, we put great efforts on modeling of the system and individual partitions. This is based on the observation that, although highly significant, synthesis has played a secondary role to the description language definition. It has very powerful constructs for simulating the hardware; however, in part because of its low-level modeling features, automatic refinement of models has been proven to be elusive at the very high levels of abstraction that are needed for the analysis of cost/performance trade-offs [23]. Additionally, it is reported in [24] that, in the ATM switch design that was performed, almost all of the errors recorded occurred during behavioral modeling of the ASIC.

In the following sections in this chapter, we will reflect how we use these state-of-the-art design language and tools in the IC implementation.

6.3 Input Controller Architecture

As noted before, one of the most important aims of ATM switch design is to find a fast routing table lookup technique for the high-speed communication networks to increase speed, capacity and overall performance. So, we choose hardware implementation for the IC design instead of software implementation. Obviously, hardware implementation has the advantage in terms of speed compared with software implementation.

We choose a non-pipelined architecture for the IC. The principle of using pipelined architecture is that the whole system can be nearly evenly divided into several small stages; all these stages occur in one clock cycle. In the IC, the most time-consuming operations are the cache lookup and the RT lookup. Other operations consume much less time. If we choose a pipelined architecture for the IC, the divided stages will not be balanced in terms of timing. The small stages will spend the same working clock cycle as the "heavy stages". Therefore, the whole system's processing time might not improve. Besides, the system controller will be complicated. Rather than risking this mistake, we choose non-pipelined architecture.

The design also features a modular style. It is recommended in the Synopsys documentation that large designs not be imported directly to the synthesis tool. Instead, a hierarchical bottom-up approach should be followed. This is because importing large designs leads to crashing the synthesis tool and in some cases may result in an unoptimized design [7]. Accordingly, we partition the architecture of the IC into modules (see Appendix A). Each module is designed, tested, simulated, and synthesized

individually. These modules then are used as building blocks in forming the final IC design. Besides, the modular style has many other merits. First, the debugging process is simpler when building high-level modules from low-level ones. Second, it simplifies hardware, thus reducing its cost and increasing its speed. Third, the modules can be reused in the future design [7]. Thus, a top-down approach is used in design, but a bottom-up approach in simulation and synthesis.

Figure 6.3 shows the IC system diagram. Before continuing this discussion, we present some background information. In Figure 6.3, the cell streams come into the IC system from the left side. In commercial networks, serial data arriving at fiber-optic transmission lines is converted from light signals to electrical signals using readily available equipment. Another kind of equipment, the Universal Asynchronous Receiver-Transmitter (UART) is used to convert the serial data stream into a parallel data stream on a certain bit wide bus, which then feeds the data into the corresponding input port in the IC. These processes are shown in Figure 6.2.

Here, as discussed in Chapter 4, for simplicity, we assume a 40-bit cell. In practice, a cell contains 53 bytes. The UART is usually not designed to send out 53 bytes at a time, but 8/16/32/64 bits in parallel, with some Start, End, and Parity check bits. Thus, the input port side needs to deal with these control bits and collect all 53 bytes in a cell.

Figure 6.3 shows the IC hardware implementation for an 8 x 8 ATM switch. It is composed of two groups of modules. One is the input buffer module group. There are eight identical Input Buffer (IB) modules, which receive incoming cells, temporarily store cells, and send out updated cells to the switch fabric; the other is the input controller

center (IC Center) which performs the cell header lookup function. The IC Center can be further divided into several sub-modules. The IC Center architecture is shown in Figure 6.4. The main sub-modules in IC Center are a set of Lookups, routing table (RT), and Arbiter. The Lookups are cache memory performing routing information lookup for the incoming VPI/VCI. The RT performs the RT lookup for the VPI/VCI whose Lookup request is granted. The Arbiter coordinates the RT lookup requests from multiple Lookups. The input signals are system clock, system preset, and cell streams to eight input ports. The output signals are updated cell streams to the switch fabric.

In the next section, we will present each component in detail.

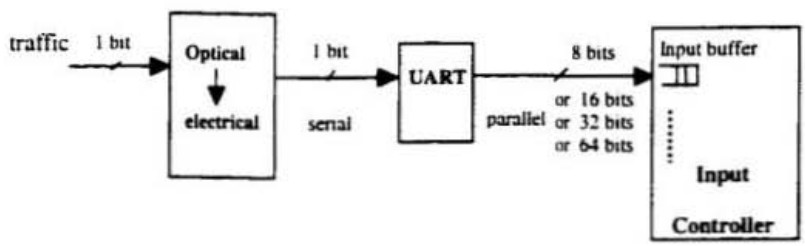


Figure 6.2 UART and the Input Controller (IC)

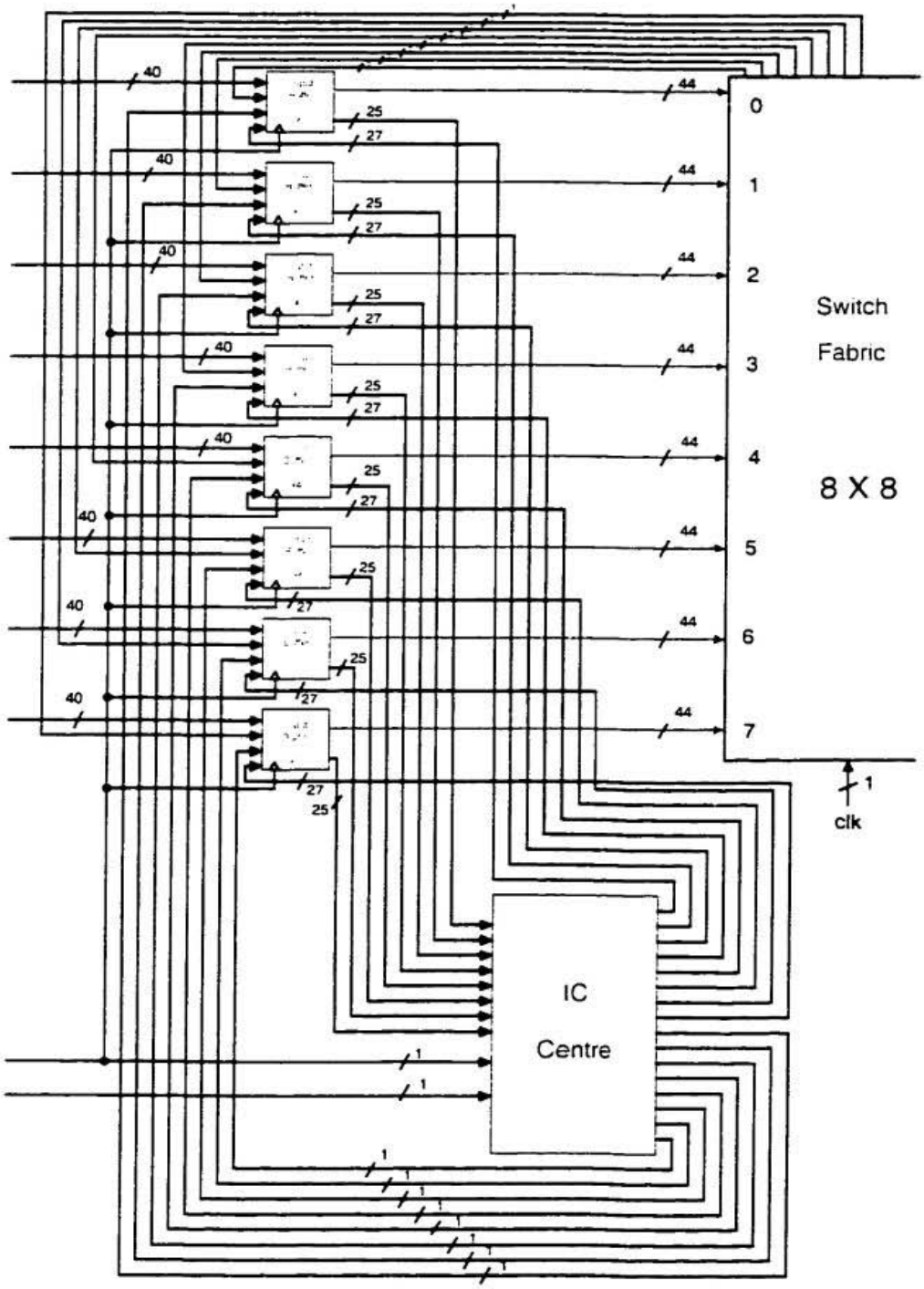


Figure 6.3 Input Controller (IC) System Block

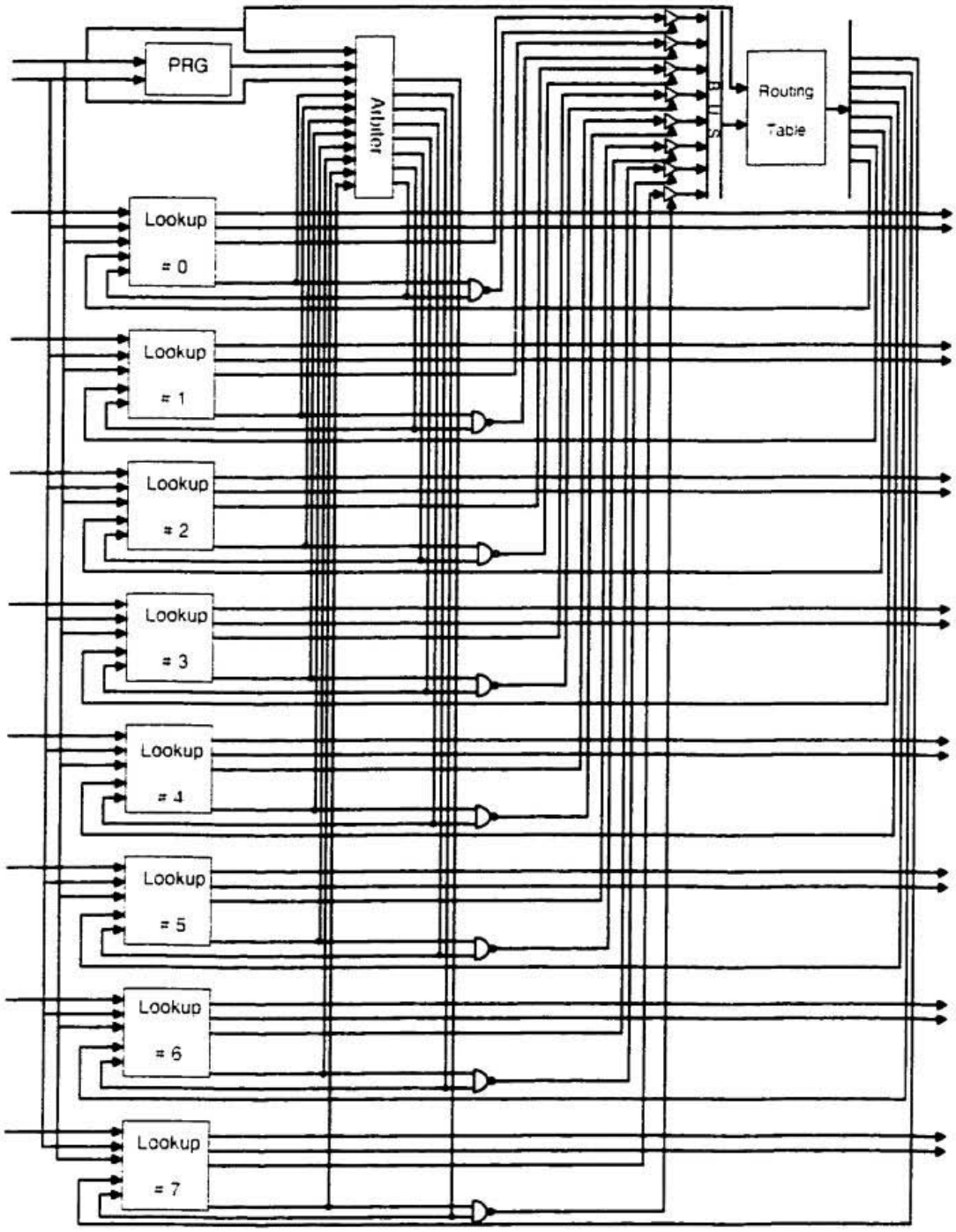


Figure 6.4 IC Centre Architecture

6.4 System Components

In this section, we present each module in the IC system.

6.4.1 Input Buffer

The Input Buffer (IB) receives the cells, sends the VPI/VCIs to be looked up, packs the cells again with the returned VPI/VCIs, waits for the acknowledgements from the switch fabric to process the next cell. It is a FIFO queue with length 10. The interface of IB is shown in Figure 6.5. An updated cell in IB composes of 44 bits, including 1 bit flag, 3 bits self routing tag, 24 bits output VPI/VCI, and 16 bits payload.

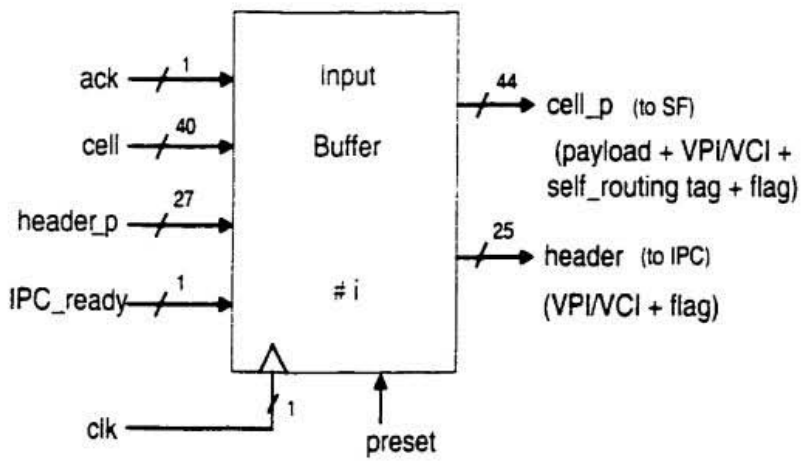


Figure 6.5 Interface of the IB

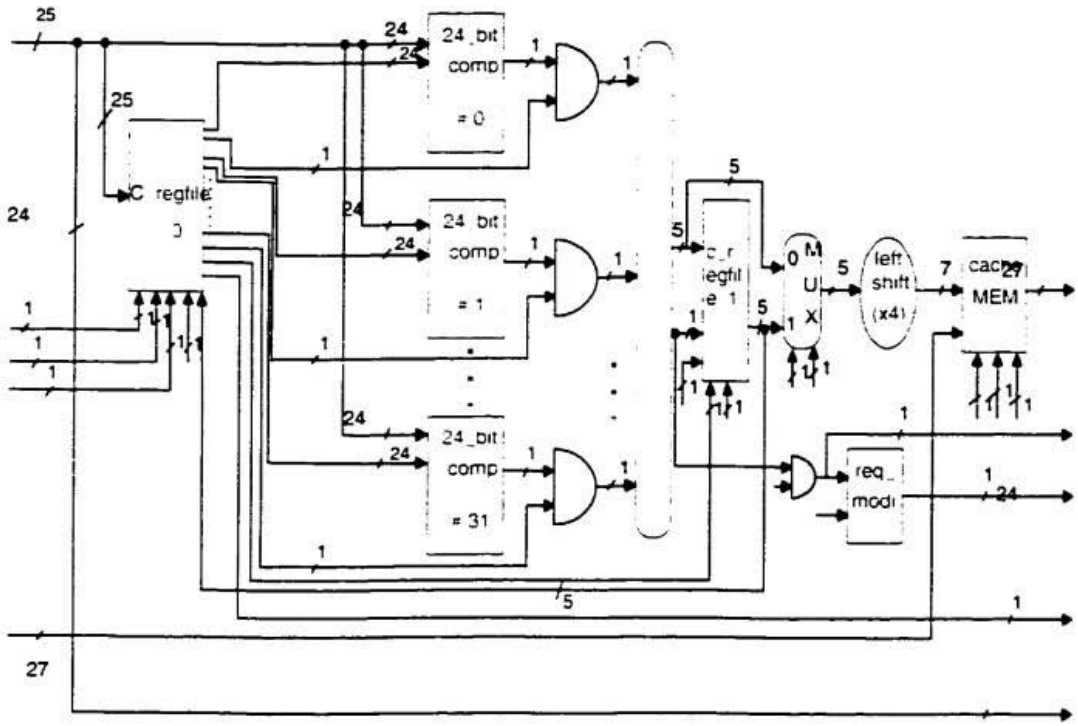


Figure 6.6 Lookup Cache Architecture

6.4.2 IC Center

Figure 6.4 illustrates the IC Center architecture.

6.4.2.1 Lookup

The Lookup performs mapping the cache blocks to find the routing information and sending it back to the IB. If cache misses occur, Lookup requests the routing information from the RT, loads in the returned content from the RT and sends it back to the IB.

Figure 6.6 depicts the Lookup cache architecture.

The Lookup is composed of several components. The main components are:

c_regfile_0

It holds incoming VPI/VCIs for mapping process. It also has valid bits to combine with VPI/VCI in the mapping. The replacement in the cache miss is under the control signal of 'rt_req' and 'grant'. The written back block is indexed by the signal 'hitBlockNum'. Figure 6.7 depicts the interface of the **c_regfile_0.** The 1-bit flag indicates whether or not the signal 'vpivci_i' is looked up.

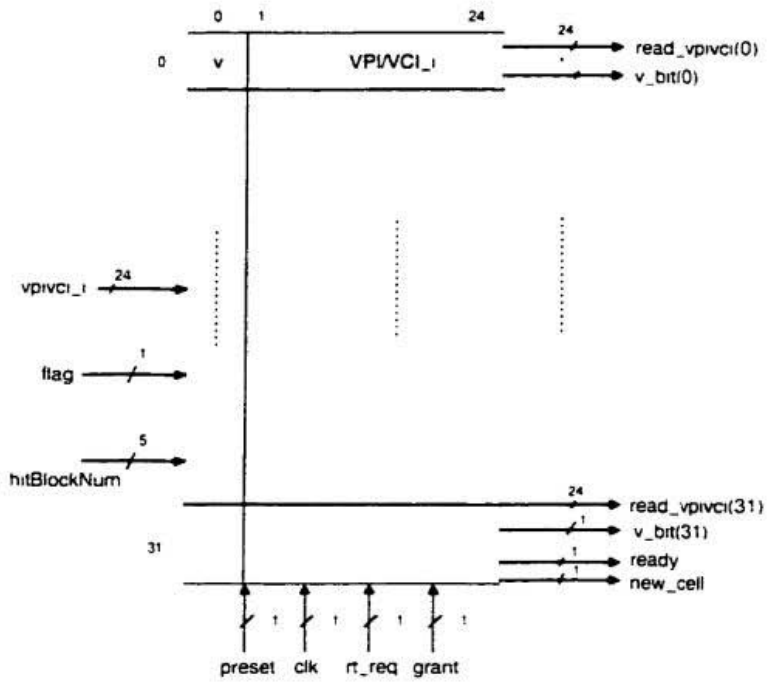


Figure 6.7 Interface of the c_regfile_0

24_bit_comp

This is the core component in the mapping process. The incoming 24-bit VPI/VCI is compared with the stored VPI/VCIs. Figure 6.8 shows the wired up 24 bits comparator. The application of the 24-bit comparator, the comparison portion in the IC, can be seen in Figure 6.9.

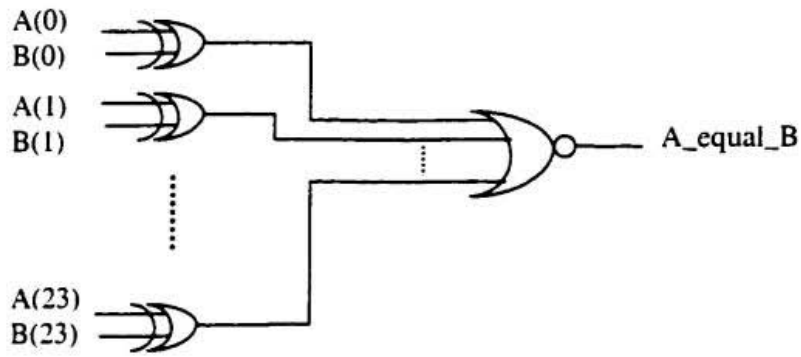


Figure 6.8 24-bit Comparator

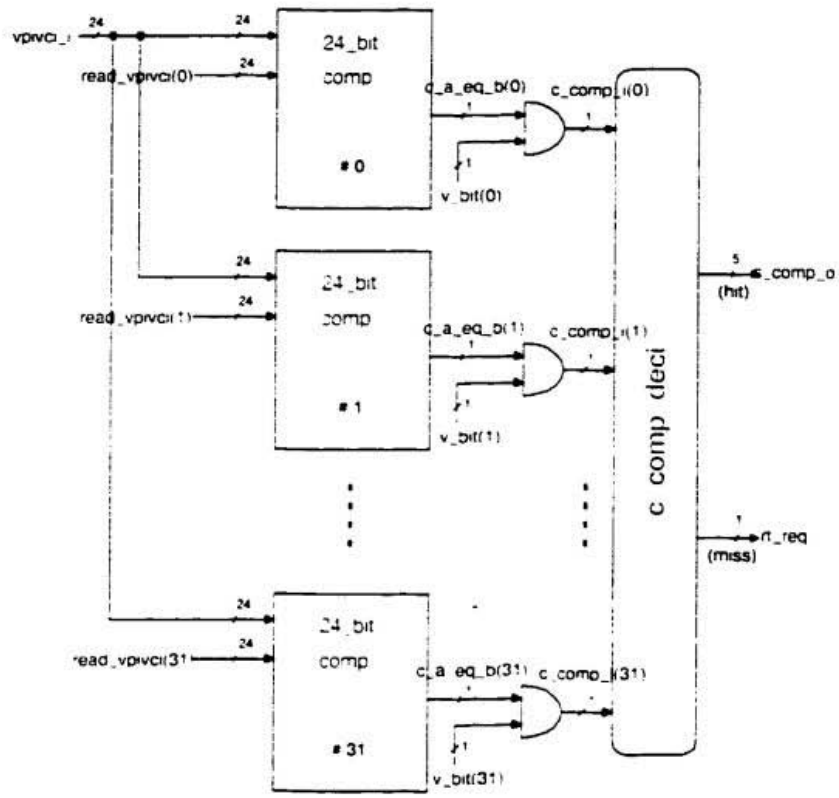


Figure 6.9 Comparison Portion in the Lookup Cache

c_comp_decision

This component makes the decision if there is a cache miss or hit. If the valid bit is set and the 24-bit comparator sends '1' then the Lookup cache hits at that block. **c_comp_decision** outputs that 5-bit block number: if the AND of valid bit and the result of the 24-bit comparator is '0', then a cache miss occurs. A 1-bit cache miss signal is sent out. The interface of the **c_comp_decision** can be seen in Figure 6.9.

c_regfile_1

This component is used in the LRU algorithm. It records and updates the access numbers of the blocks in the Lookup cache. It updates the access numbers based on the hit block number if miss signal is cleared to 0; it calculates the replace block number if miss signal is set. Figure 6.10 depicts the interface of the **c_regfile_1.**

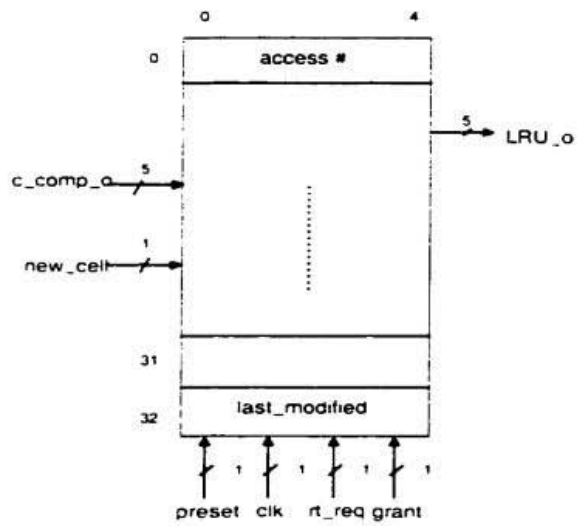


Figure 6.10 Interface of the c_regfile_1

cache_MEM

This component is the cache memory housing the 24-bit updated VPI/VCIs and 3-bit output port numbers. Since there are 32 cache blocks, a 5-bit address is applied. In the cache memory, (24 + 3) = 27 bits occupy 4 bytes. So, totally there are 32 x 4 = 128 bytes, which results in 7-bit address for the **cache_MEM**. So, a 2-bit shifter component is added on the path before the hit block number or the replace block number is loaded as address into the **cache_MEM.** Under the control signals, if cache hits, the memory content indexed by the signal 'addr' is read out; if cache misses, the content of signal

'matched_data' is loaded into the cache memory where signal 'addr' points to. Figure 6.11 depicts the interface of the **cache_MEM.**

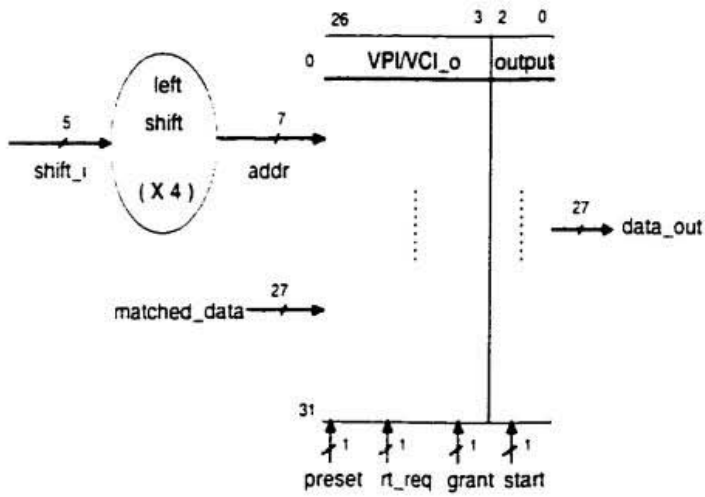
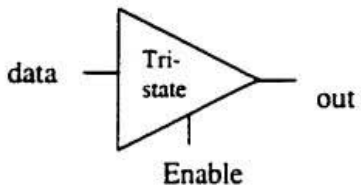


Figure 6.11 Interface of the cache_MEM

6.4.2.2 Routing Table

The requesting VPI/VCIs from eight Lookups are tied together and connected to RT through a tri-state bus. In this structure, only one VPI/VCI at a given cycle can be gated onto the bus and forwarded to the RT for the lookup process. Figure 6.12 gives the tri-state symbol and the truth table [25]. Figure 6.13 depicts the tri-state bus we use in the design. Figure 6.14 depicts the RT architecture. In the hardware implementation, we simplify the RT entry number to 50 in order to complete the synthesis step within a reasonable time.



Enable	data	out
0	0	Hi-Z
0	1	Hi-Z
1	0	0
1	1	1

Figure 6.12 Tri-state Symbol and Truth Table

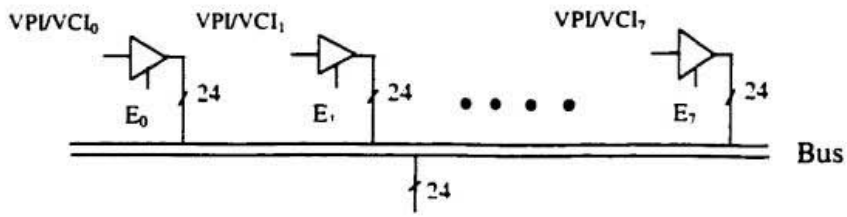


Figure 6.13 Tri-state Bus

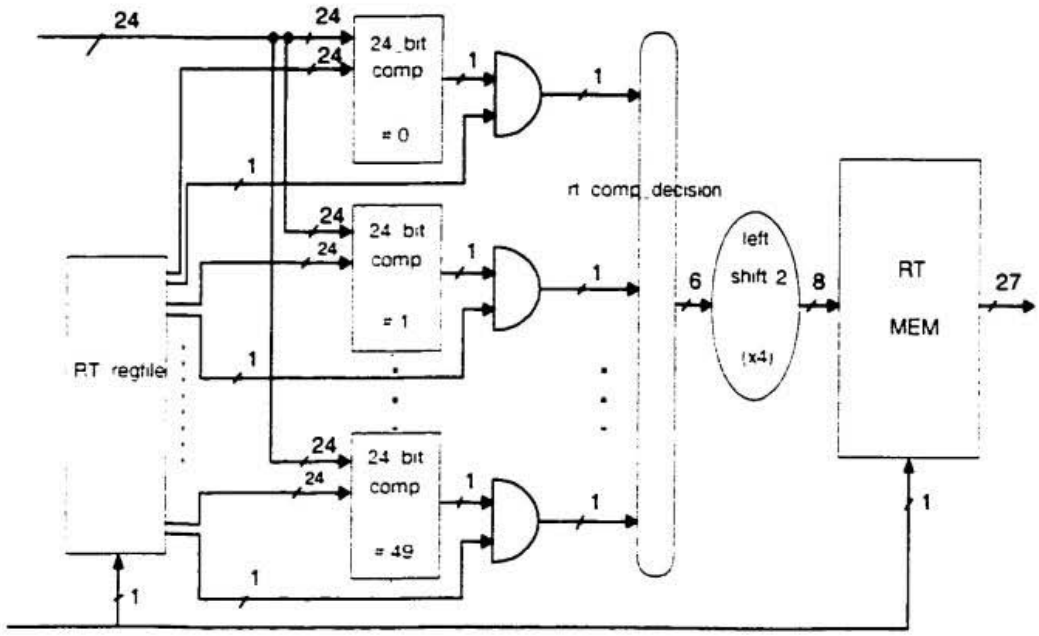


Figure 6.14 Routing Table (RT) Architecture

The RT is partitioned into several components. The main components are:

RT_regfile

It holds incoming VPI/VCIs for mapping process. It also has valid bits to combine with VPI/VCI in the mapping. Figure 6.15 depicts the interface of the **RT_regfile.**

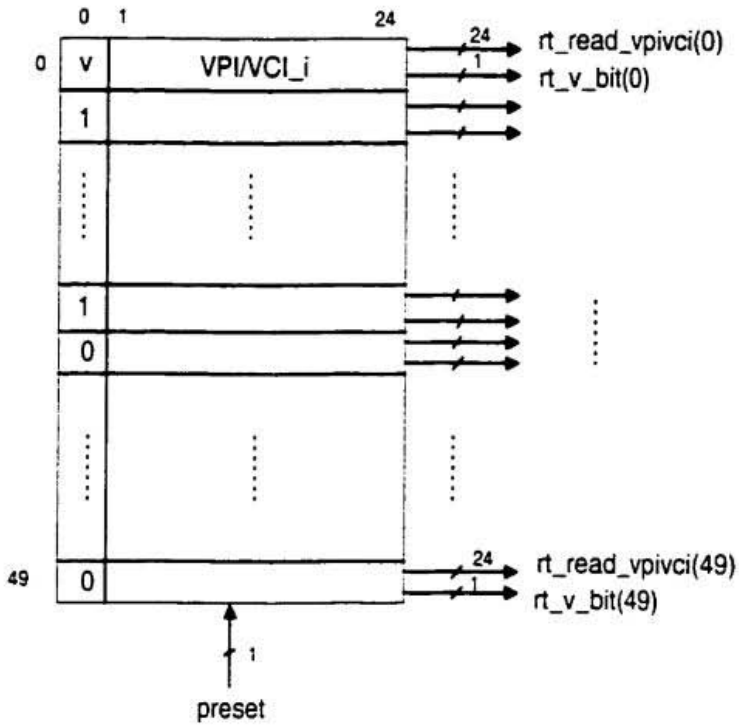


Figure 6.15 Interface of the RT_regfile

24_bit_comp

This core component is exactly the same as the one in the IC Center. Refer to the circuit diagram of the 24-bit comparator in Figure 6.8. Figure 6.16 shows the comparison portion in the RT architecture.

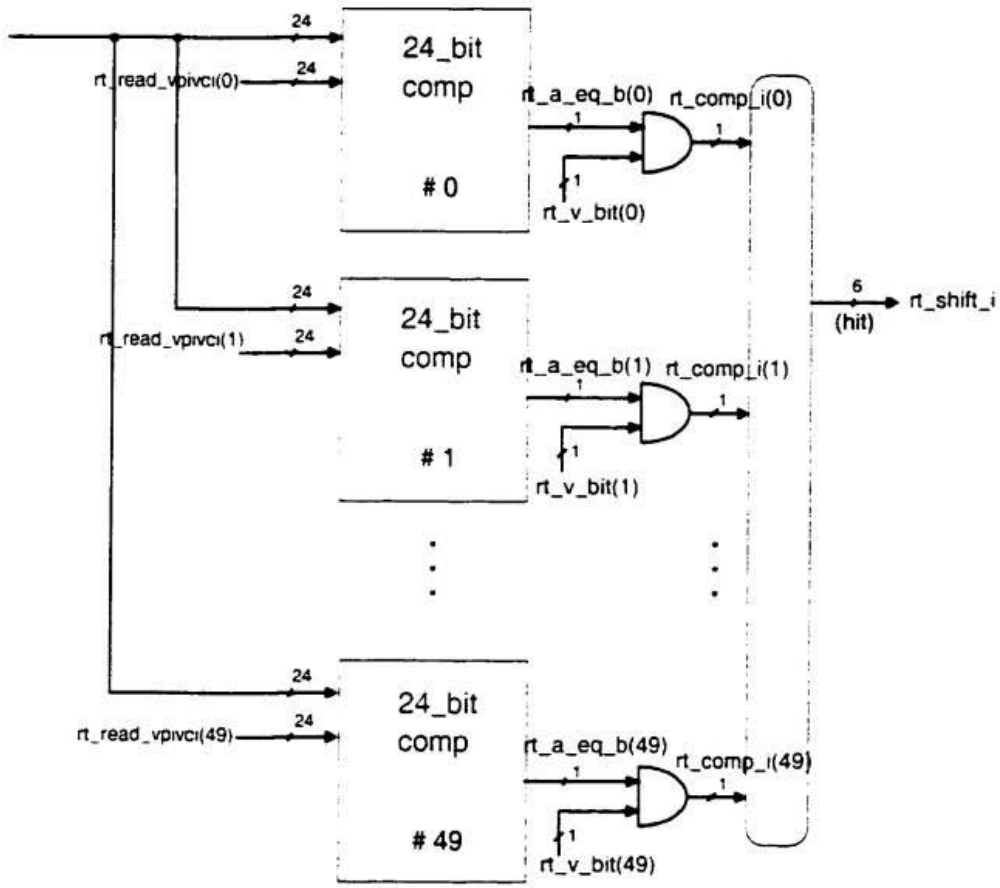


Figure 6.16 Comparison Portion in the RT Architecture

RT_comp_decision

This component makes decision as to which RT entry is mapped. If the valid bit coming from the RT_regfile is set and the result of the 24_bit_comp is '1', then that entry is chosen. The entry number is output. Figure 6.16 illustrates the component interface.

RT_MEM

This component is the RT memory housing the 24-bit updated VPI/VCIs and 3-bit output port numbers. Since there are 50 RT entries, a 6-bit address line is applied. In the RT memory, (24 + 3) = 27 bits occupy 4 bytes. So, totally there are 50 x 4 = 200 bytes, which results in 8-bit address for the **RT_MEM**. So, a 2-bit shifter component is added on the path before the entry number is loaded as the address into the **RT_MEM.** The memory content indexed by the signal 'rt_addr' is read out. Figure 6.17 depicts the **RT_MEM** interface.

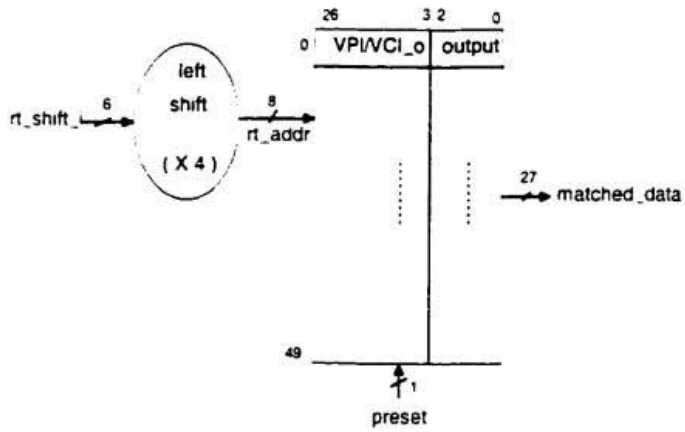


Figure 6.17 Interface of the RT_MEM

6.4.2.3 Arbiter

The job of the Arbiter is to coordinate the Lookup cache miss requests and to grant only one request at a given clock cycle to the RT memory following a specified rule. Figure 6.18 shows the interface of the Arbiter.

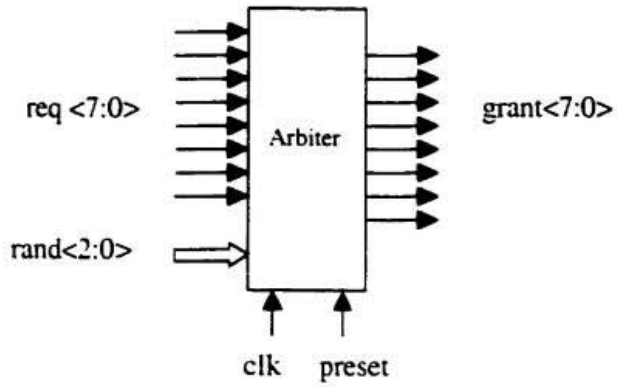


Figure 6.18 Interface of the Arbiter

At any given cycle, the Arbiter checks the eight 'req' signals from the Lookup cache. If no 'req' signal is set, then none of the eight 'grant' signals is issued. If only one 'req' is set, then the Arbiter sets the relevant grant signal. If more than one 'req' are set, then the Arbiter picks signal 'rand' and uses this number to decide which one to grant. At the next cycle, the Arbiter first sets the 'grant', which was set in the last cycle, to 0 and then continues the arbitration corresponding to the current cycle. The 'preset' signal is used to invalidate the internal variables and signals in the Arbiter at IC reset stage.

A pseudo random number generator (PRG) serves in the arbitration as the pseudo random numbers provider, using a linear feedback shift register (LFSR). The use of LFSR to generate pseudo random numbers is well documented in general literature. An LFSR is a shift register that, when clocked, advances the signal through the register from one bit to the next most-significant bit. Some of the outputs are combined in exclusive-OR configuration to form a feedback mechanism. An LFSR can be formed by performing exclusive-OR on the outputs of two or more of the flip-flops together and feeding those

outputs back into the input of one of the flip-flops [26]. Provided a suitable feedback connection is used, an LFSR produces a pattern count equal to $2^n - 1$, where n is the number of register elements in the LFSR [27]. In the Arbiter, the maximum number of requests is 8, so, n = 3. The generated pattern count is $2^3 - 1 = 7$. Figure 6.19 [26] shows the PRG, that is, a 3-bit LFSR. Table 6.1 lists the patterns produced by the LFSR in Figure 6.19, assuming that a pattern of 111 is used as a seed.

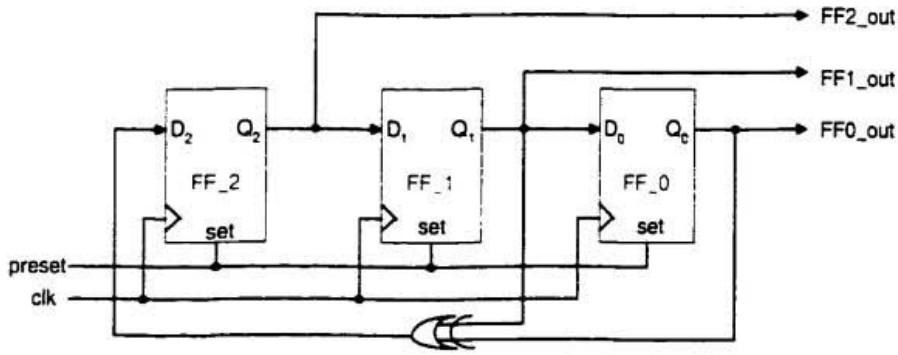


Figure 6.19 Pseudo Random Number Generator (PRG) Structure

Clock Pulse	FF1_out	FF2_out	FF3_out	Comments
1	1	1	1	Seed value
2	0	1	1	
3	0	0	1	
4	1	0	0	
5	0	1	0	
6	1	0	1	
7	1	1	0	
8	1	1	1	Starts repeat

Table 6.1 Pattern – Generator Outputs

LFSRs make extremely good pseudo random generators. The seed value can be anything except all 0s, which would cause the LFSR to produce all 0 patterns. The only signal necessary to generate the pattern is the clock and hence, it is self-generating. So, an LFSR is also called autonomous linear feedback shift register (ALFSR) [27].

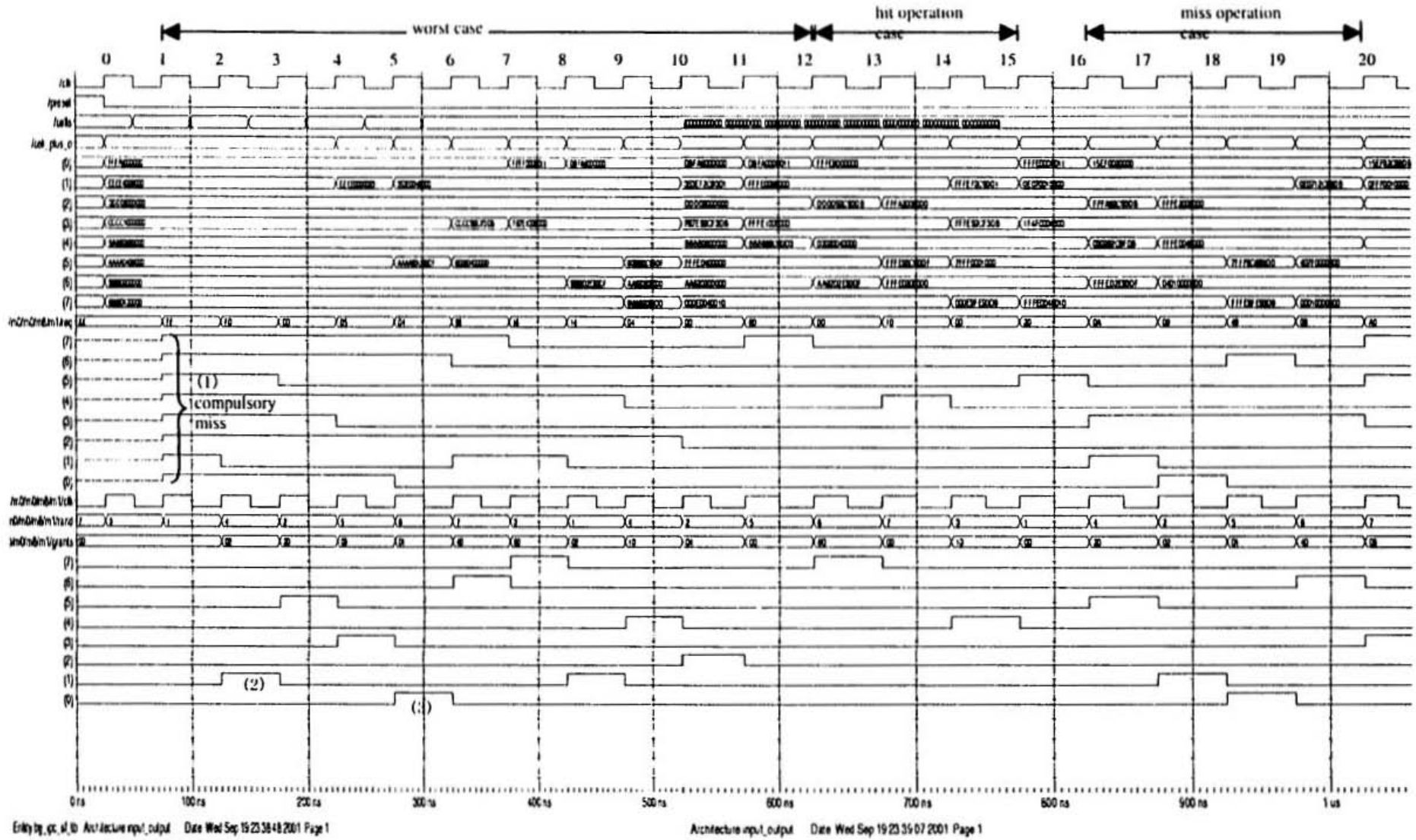
6.5 Simulation and Synthesis

Verification of the IC implementation has been carried out on different levels of the design hierarchy. Testbenches have been developed to check the functionality of individual components and the whole IC system. The powerful and convenient tools we use in simulation are **VHDLAN** by Synopsys, and **V-System/Windows** by Model Technology. Simulation waveforms are shown and explain the design correctness. The simulation results are included in Appendix B in this thesis.

The simulation results demonstrate the correct functionality of the IC system designed here. Figure 6.20 shows a part of the simulation results. It can be seen that the cache hit operation takes three clock cycles. The cache miss operation takes at least one cycle more than the hit operation. This best case happens if that cache is granted access to the RT at once. If more cache misses occur in a given cycle, waiting is needed. Seven cycles more than the best case cache miss will be required in the case that all eight caches miss. In this case, it will take eleven cycles to complete the cache miss operation. Cache compulsory miss is one of the worst cases. The figure shows the compulsory miss case. At the beginning of the system running, at cycle #1, all eight caches miss and issue the RT lookup requests simultaneously (see (1) in Figure 6.20). In this example, the arbiter

judges that the random number is 1 at that cycle and there are 8 RT requests. 1 MOD 8 = 1. So cache #1 is granted access to the RT (see (2) in Figure 6.20). Let us take another sample point in this example. At cycle #4, the random number at this cycle is 5 and there are 5 RT requests. 5 MOD 5 = 0. Cache #0 is granted access to the RT (see (3) in Figure 6.20). The simulation results verify our objective of cache usage: in Figure 6.20, the lookup clock cycle numbers have been saved when cache hits occur which would be the case most of the time. Thus, the IC performance should be improved.

The **Design Compiler** is the core synthesis engine of the Synopsys synthesis product family. It has two user interfaces, the graphical user interface (GUI) and the command line interface. We use the GUI, that is named **Design Analyzer**. The synthesis process is to analyze our RTL files, the abstract descriptions of the circuits, described using VHDL and to produce a gate level net-list that would be ported to the target library. The synthesis is based on the 0.18 μm CMOS technology. We synthesize the IC system from the bottom to the top. A report is obtained from the tools for each synthesized component and the integrated system. In Table 6.2, we summarize the synthesis report of the main components and the integrated system. We include two main parameters in the table, the area and the timing. A 2-input AND gate requires a cell area of 70 units. The IC system is comprised of the ipc_center and the input buffer. The total cell area for these two modules as can be seen from Table 6.2 would be (0.5M x 8 input buffer modules) + 16.7M ipc_center) = 20.7M units of cell area, that would be approximately equivalent to 0.3M gates. Thus, to build the IC system, 0.3M gates are expected.


worst case
hit operation case
miss operation case
(1) compulsory miss
(2)
(3)

Component	Cell Area	Timing (ns)		Component	Cell Area	Timing (ns)	
		Delay	Slack			Delay	Slack
and2	70	0.31	-	cache_mem	822920	0.81	-
mux	1102.5	2.19	-	rt_mem	50680	17.86	-
xor_struct	122.5	0.54	-	rt_comp_deci	8540	0.75	-
not_or_24	910	1.38	-	rt	664457.5	17.06	-
leftshift_0	262.5	0.26	-	c_regfile_0	487952.5	-	-4.12
leftshift_1	315	0.26	-	d_ff	367.5	-	49.5
tri_state_buff	3010	1.19	-	arbiter	37765	-	25.83
bit_comp_24	3850	2.16	-	prg	1225	-	47.76
c_comp_deci	5967.5	0.84	-	input_buffer	496282.5	-	20.54
c_regfile_1	504402.5	0.83	-	ipc	1948957.5	-	-4.12
req_modi	542.5	0.7	-	ipc_center	16700000	-	-4.12
rt_regfile	408905	0.66	-				

Table 6.2 Summary of Area and Timing Report for the Component and the System

The timing analysis looks at the structure of the circuit and measures the delays along the datapath. In the timing report, one important item is the **slack**. The amount of slack (time before the next clock edge) indicates the **required delay** minus the **actual delay**. The slacks corresponding to most of the components reported in Table 6.2 were shown as **MET**, i.e. having positive values. So, these designs have met the timing constraints. We select the clock speed at 20MHz for the synthesis. But the **c_regfile_0,**

the **ipc** and the **ipc_centre** report **VIOLATED**. They have negative slacks. Hence, the circuit cannot run at the clock speed selected for the synthesis. However, the reported discrepancy is small and the worst-case assumptions made by the timing model are normally too strict for the target library. We selected a clock frequency of 20MHz and a low-effort synthesis algorithm. We chose low-effort since the design is large and time-consuming to synthesize even at this setting. This may also affect the timing and cause longer delay in the circuit. In the real hardware implementation, the delay in the circuit is usually much smaller than the synthesized delay value. Therefore we expect this circuit to function correctly but if the failure is detected during the testing, this timing issue would be worth re-addressing. Note that the **c_regfile_0**, the **ipc** and the **ipc_centre** have the same negative slack. The **c_regfile_0** is one of the subcomponents of the system. So, the negative slack of the system is caused by the slack of the **c_regfile_0.** So, we might be able to improve the design of the **c_regfile_0** to improve timing. To help improve the circuit performance is the purpose of our timing analysis.

The above synthesis report is for our simplified IC system. In practice, to design a realistic 8 x 8 ATM switch, the cell size is 53-byte and the RT memory has 10,000 entries. The above timing will almost not change since it is still the 24-bit VPI/VCI which is retrieved to process lookup, not the whole 53-byte cell, and the extended RT will not affect the delay on the datapath. But the number of gates will definitely be changed. The input buffer will enlarge ((53 bytes x 8 + 3 bits) / (40 bits + 3 bits)) ≈ 10 times (see table 4.1 for the input buffer structure). The eight input buffer modules will have 40M units of cell area. The RT would be 10,000 entries which is 200 times the simplified 50 entries.

So, the totally cell area of the RT will be 200 times of the cell area of the simplified RT: 200 x 16.7M = 3340M. Hence, the whole IC system will be 3340M + 40M = 3380M units of cell area, which is approximately equivalent to 48M gates.

The schematics generated by Design Compiler are not provided here for lack of space.

6.6 Summary

Based on the IC design presented in Chapter 4, and encouraged by the performance analysis results reported in Chapter 5, we carry out the hardware implementation of input controller (IC) for BG ATM networks. Following the design flow recommended by CMC, we design, test, simulate and synthesize each component and then assemble the whole system. The results of simulation and synthesis demonstrate that the designed IC performs correctly. Besides, the IC is easy to implement, practical and cost efficient.

One problem we should mention here is about the pseudo random number generator (PRG) design. In Section 6.4.2.3, we describe that the PRG may have seven possible outputs, 1, 2, 3, 4, 5, 6, and 7, respectively. Note that there is no 0. Thus, if all eight caches miss in a cycle, though it is not very probable, there is no chance for cache #0 to be granted to access RT in that cycle. This affects the performance to some degree. In Chapter 7, we will briefly discuss a solution to this problem.

Chapter 7

CONCLUSION

7.1 Contributions in this Thesis

The main contributions of this work can be summarized as follows:

• **Modifying the traffic models and using these models in the performance analysis of the IC system for BG ATM networks.**

To design the IC system, traffic models are needed in performance analysis before implementation. URT and bursty traffic are the two traffic types we choose. The former is the most commonly used traffic model when analyzing the performance of switches. The latter is flexible enough to model most of the existing traffic sources. We follow the same modeling means set up for BG networks, but modify and apply to the IC system.

• **Demonstrating the outstanding performance of cache structure in the IC**

After studying the various cache structures, we finally decide that, for an N x N switch, there are N caches, each for an input port. They are fully associative with 32 blocks each. Fully associative cache has the most flexibility in mapping cache blocks. It has higher hit ratio than other organizations. The possible concern for this kind of cache is that, since the whole block address is used as tag to match the entries, it increases the hardware complexity. Thus, a fully associative cache has been implemented only in moderate size. A 32-block cache is a reasonable choice. The designed cache-memory IC

system yields a hit rate well in excess of 80% under most bursty traffic and achieves above 95% under bursty traffic with mean burst length of 15. This compares favorably with the results reported for the commercial devices.

- **Developing a practical and high-performing IC architecture.**

A cache-based IC architecture is motivated in this thesis. The IC performs routing table lookup, rewrites the processed ATM cells and forwards them to the switch fabric that delivers cells to outgoing lines. The cache holds the frequently used forwarding information. The IC looks up all the information necessary to forward a cell toward its destination from a high-speed cache. If cache misses occur, a slower routing table lookup follows the cache operation. The key issue of this architecture is to find an appropriate cache-memory hierarchy in terms of cache hit rate, speed, cost, etc. Based on our design experience, the whole IC performance should be improved when cache hits occur most of the time.

- **Carrying out software simulation and further hardware implementation.**

The use of a C++ based programming enables the efficient modeling of switch hardware components in order to perform cost/performance trade-off, and the successive refinement of interfaces and behavior. Then, following a smooth design flow, the hardware components are built to a detailed level that contains all necessary information about timing and structure in order to support functional verification capabilities. This combination of software analysis and hardware implementation guarantees the IC design operates clearly, correctly and efficiently.

7.2 Recommendations for Future Work

This section summarizes several missing pieces because of the limited time:

- **Routing table (RT) refresh issue**

One of the most difficult decisions in ATM switch design is to determine how to handle the routing table memory refresh. The designed RT works fine if no dynamic changes occur. However, in a dynamic routing algorithm, we need to update the routing table periodically or frequently (whenever we receive a routing update message). In this case, we need to refresh the RT. A simplistic, but wasteful, solution would be to keep two copies, and alternately use one and refresh the other. One is hot. The other is standby. This will need more complex controller logic. How to keep the Lookup caches and RT memory consistent is a key topic. In practice, if one VP/VC is torn down and is made invalid in the RT, even though the same VP/VC cannot be marked invalid simultaneously in Lookup caches, it seldom causes trouble, because incoming cells are much less likely to be assigned that VPI/VCI by the network operator in immediate future. Cache coherence is a well-studied issue in parallel processing, and some of the innovative ideas from this domain may be considered for adaptation to this problem.

- **Pseudo Random Number Generator (PRG) improvement**

In the PRG designed in Chapter 6, a problem exists. If eight cache misses happen simultaneously, since there is no 0 in the PRG outputs, cache #0 cannot be issued a grant to access the RT. This happens rarely, but this would still affect the system performance. However, this is not a major problem since in the subsequent cycle, there will be fewer caches misses and thus cache#0 will get a chance. One solution is to use four D-flipflops

instead of three D-flipflops in the design of the PRG since $2^4 - 1 = 15$. Thus, the outputs of the PRG will be 1, 2, 3, ..., 14, 15. This makes it possible for cache #0 to be granted access if all eight caches miss, since 8 MOD 8 = 0. In this way, the PRG is improved because every cache has a chance to be granted access to the RT when all caches miss. Although this improved PRG is still unfair (as there is only a 1/15 chance for cache #0 to access the RT while 2/15 for all others), this bias is insignificant because that the event of all eight caches missing in the same cycle is rare. Figure 7.1 shows the improved PRG circuit.

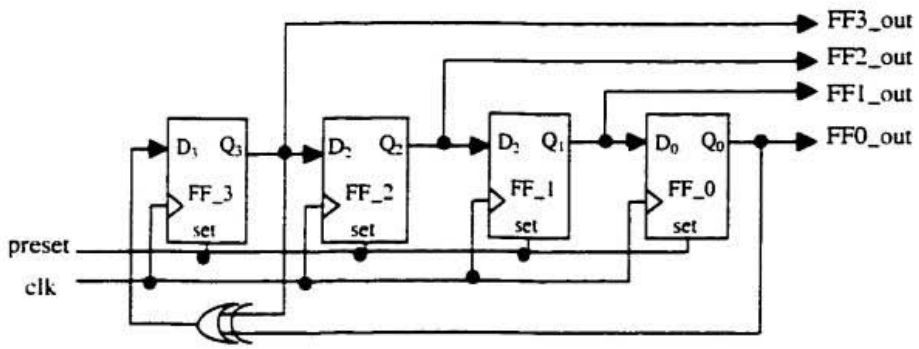


Figure 7.1 Improved Pseudo Random Number Generator (PRG)

- **Next-to-HOL Cell Lookup Improvement**

As mentioned in Chapter 4 and 6, the system always looks up the HOL cell in the input buffer (IB) which is a FIFO queue. Only after the HOL cell lookup is completed and the HOL cell is picked up by the switch fabric, the next to HOL cell can start lookup. This strategy works well in most situations except when all eight caches miss at a cycle. For the latter situation, the cell waiting behind the HOL cell may have to wait for several

cycles to be served, for seven cycles in the worst case. A possible improvement to address this issue is to add logic to check the next-to-HOL cell and look it up after the HOL cell has completed the lookup. Thus, the waiting time for the lookup is saved and the system performance is improved. Lookup for more cells than the first two cells in the IB may not be practical because the hardware complexity is increasing greatly.

- **More traffic loads should be tested when investigating the relationships between the number of input/output buffers and the traffic load**

In the system performance analysis, when we investigated the relationships between the number of input/output buffers and the traffic load, traffic loads of 0.5, 0.6, 0.7, 0.8, and 0.9 were tested and the results showed clear trend of the performance. However, obviously, we missed some points among this range, especially, we were expected to investigate by executing cases of loads between 0.8 and 0.9, say, 0.82, 0.84, 0.86, 0.88. This future work may help us find more information about the jump in number of input/output buffers for load of 0.9.

7.3 Summary

This chapter summarizes the contributions of this thesis, and also states the work in the future.

REFERENCES

[1] M. Bentall, C. Hobbs, and B. Turton. *ATM and Internet Protocol: A Convergence of Technologies*. Arnold Inc., 1998.

[2] B-ISDN ATM Layer Specification, Telecommunication standardization sector of ITU (ITU-T) I.361(11/95).

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

[4] "A Survey of ATM Switching Techniques", http://www.cis.ohio-state.edu/~fahmy/cis788.08Q/atmswitch.html.

[5] S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*, Addison Wesley, 1997.

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

[7] Y. E. Sayed, "Performance Analysis, Design and Reliability of the Balanced Gamma Network", Ph.D thesis, Memorial University of Newfoundland, December 1999.

[8] D. A. Patterson, J. L. Hennessy, *Computer Architecture: A Quantitative Approach*, Second Edition, Morgan Kaufmann Publishers, Inc. 1996.

[9] http://phoenix.goucher.edu/~kelliher/cs26/nov1414.html.

[10] K. Lindberg, "Multi-gigabit Routers", http://www.csc.fi/lindberg/tik/paper.html.

[11] K. Schultz and A. Sorowka, "High-Performance CAMs for 10Gb/s and Beyond", http://www.sibercore.com/profile.html.

[12] "SiberCAM Application Note", SiberCore Technologies Inc.

[13] D. Wu, "The Balanced Gamma Network: A Prospective ATM Switch Fabric", course project report, Memorial University of Newfoundland, December 1999.

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

[15] M. Bentall, C. Hobbs and B. C. Turton, *ATM and Internet Protocol: A Convergence of Technologies*, Arnold, 1998.

[16] G. D. Stamoulis, M. E. Anagnostou, and A. D. Georgantas, "Traffic source models for ATM networks: a survey", *Computer Communications*, Vol. 17, number 6, June 1994.

[17] J. Banks, J. S. Carson, B. L. Nelson, *Discrete-Event System Simulation*, Second Edition, Prentice Hall, 1996.

[18] M. Al-Mouhamed, H. Youssef, and W. Hasan, "A Fast Parallel-Tree Switch Architecture for ATM Networks", *International Journal of Communication Systems*, Vol. 11, pp. 59-77, 1998.

[19] C. Partridge, et al., "A 50-Gb/s IP Router", *IEEE/ACM Transaction on Networking*, Vol.6, No.3, pp.237-248, Jun. 1998.

[20] "GRF 400: A Practical IP Switch for Next-Generation Networks" http://apac.ascend.com/1680.html.

[21] J. F. Wakerly, *Digital Design: Principles & Practices*, Third Edition Updated, Prentice Hall, Inc. 2001.

[22] Royal Military College of Canada and Canadian Microelectronic Corporation, Instruction on Basic Digital IC Design Flow From RTL Description to Completed CMOS Design Using Cadence (97A) and Synopsys, November 1998. Document ICI-089.

[23] "Design of ATM Switch Hardware", http://www.ert.rwth-aachen.de/Personen/post/node2.html.

[24] A. Silburt et.al., "Accelerating Concurrent Hardware Design with Behavioural Modelling and System Simulation", in Proc. of DAC'95.

[25] V. P. Heuring, H. F. Jordan, *Computer Systems Design and Architecture*, Addison Wesley, 1997.

[26] "What's an LFSR", Texas Instruments Inc. document, http://www-s.ti.com/sc/psheets/scta036a/scta036a.pdf.

[27] E. J. McCluskey, "Built-in Self-Test Techniques", *IEEE Design & Test*, Vol. 2, No. 2, pp. 21-28, April 1985.

[28] M. Guizani, A. Rayes, *Designing ATM Switching Networks*, McGraw-Hill, 1999.

[29] E. R. Coover, *ATM Switches*, Artech House, 1997.

[30] "Cajun A500 ATM Switch", Lucent Technologies, www.lucent.com/products/a500/.

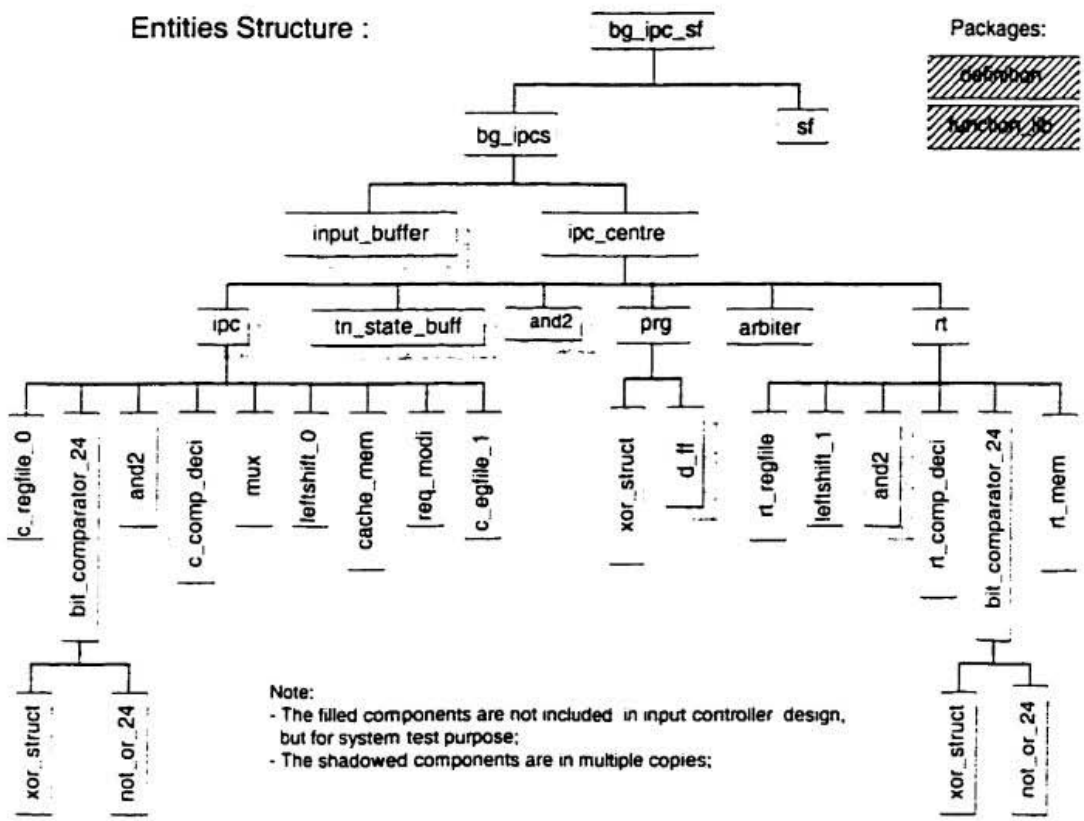
[31] "LightStream 1010", Cisco, www.cisco.com/warp/public/cc/cisco/mkt/switch/ls1010/prodlit/lantm_ai.pdf.

[32] "Centillion 100", Nortel Networks, www.nortelnetworks.com/products/.

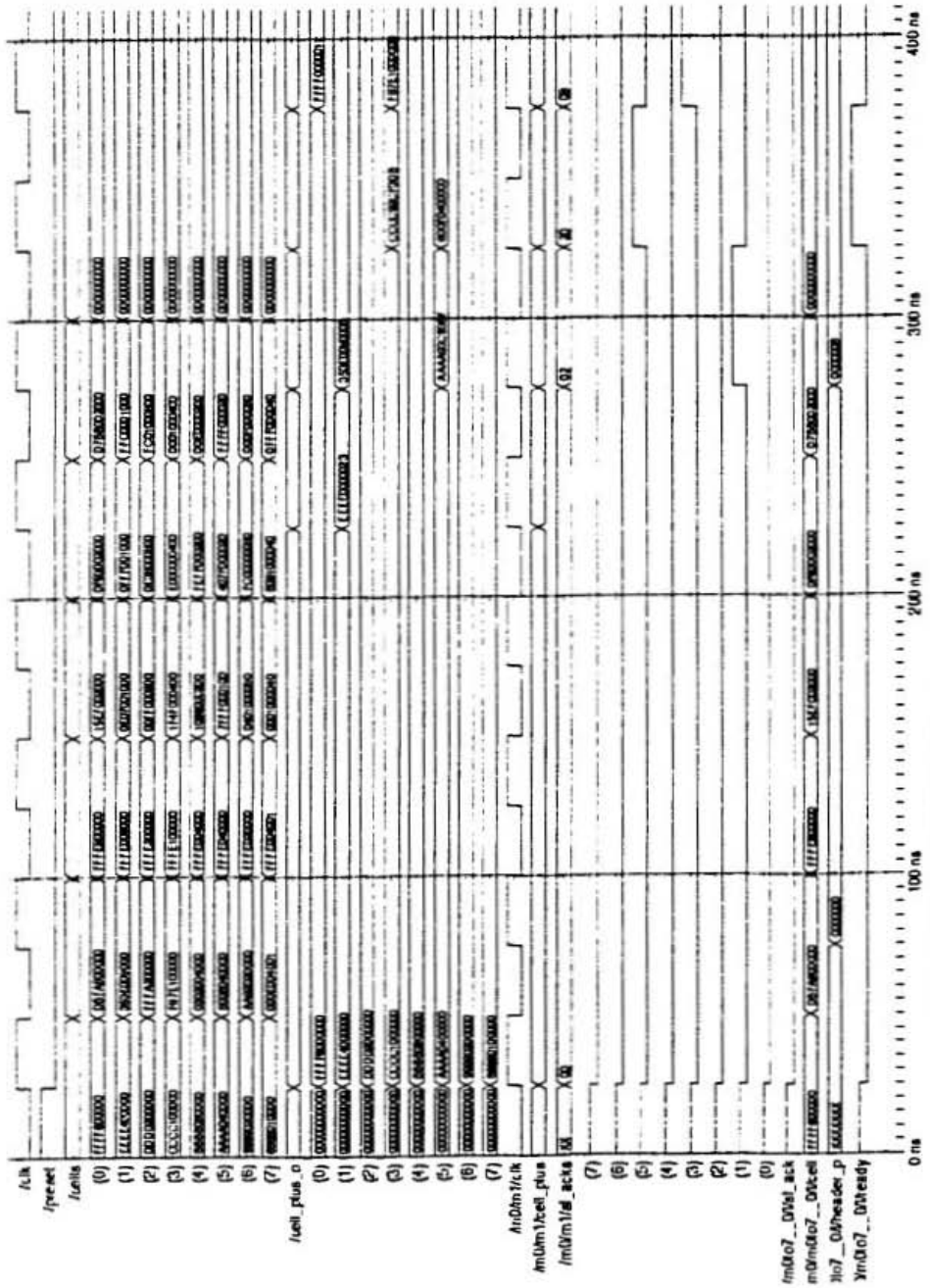
[33] "Cisco 12000 Series", Cisco, www.cisco.com/warp/public/cc/cisco/mkt/switch.

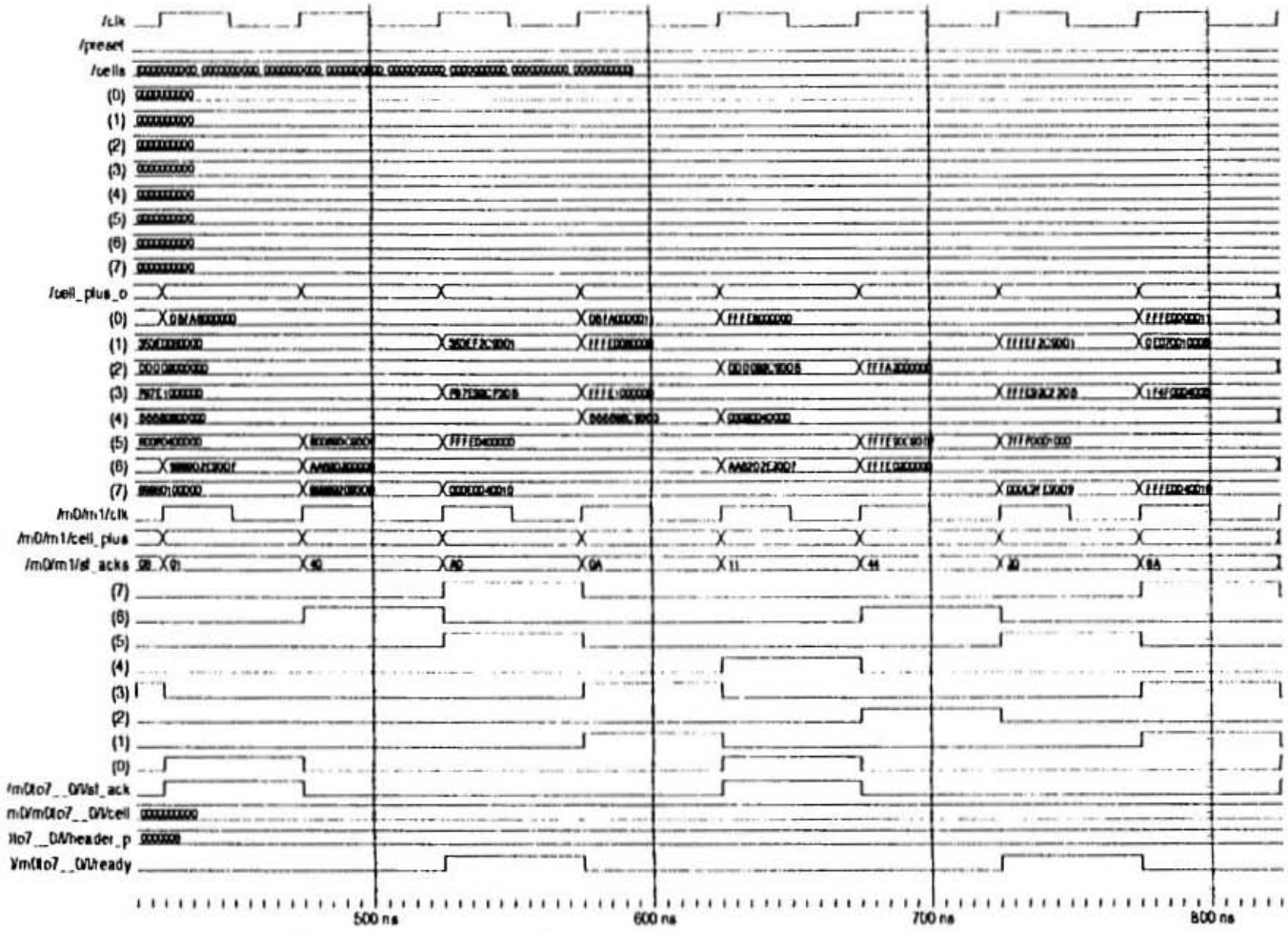
APPENDICES

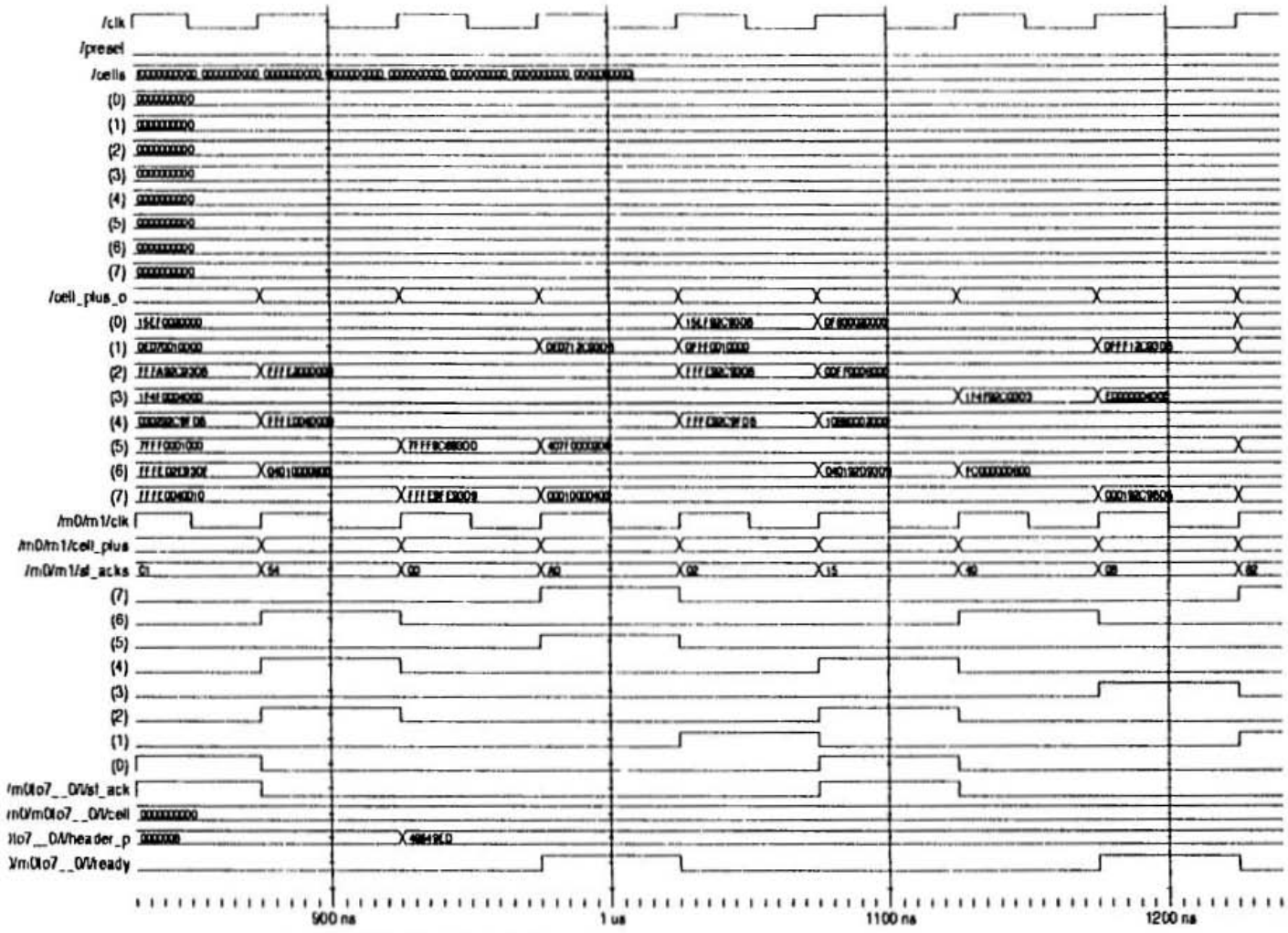
A Module Structure of the IC System

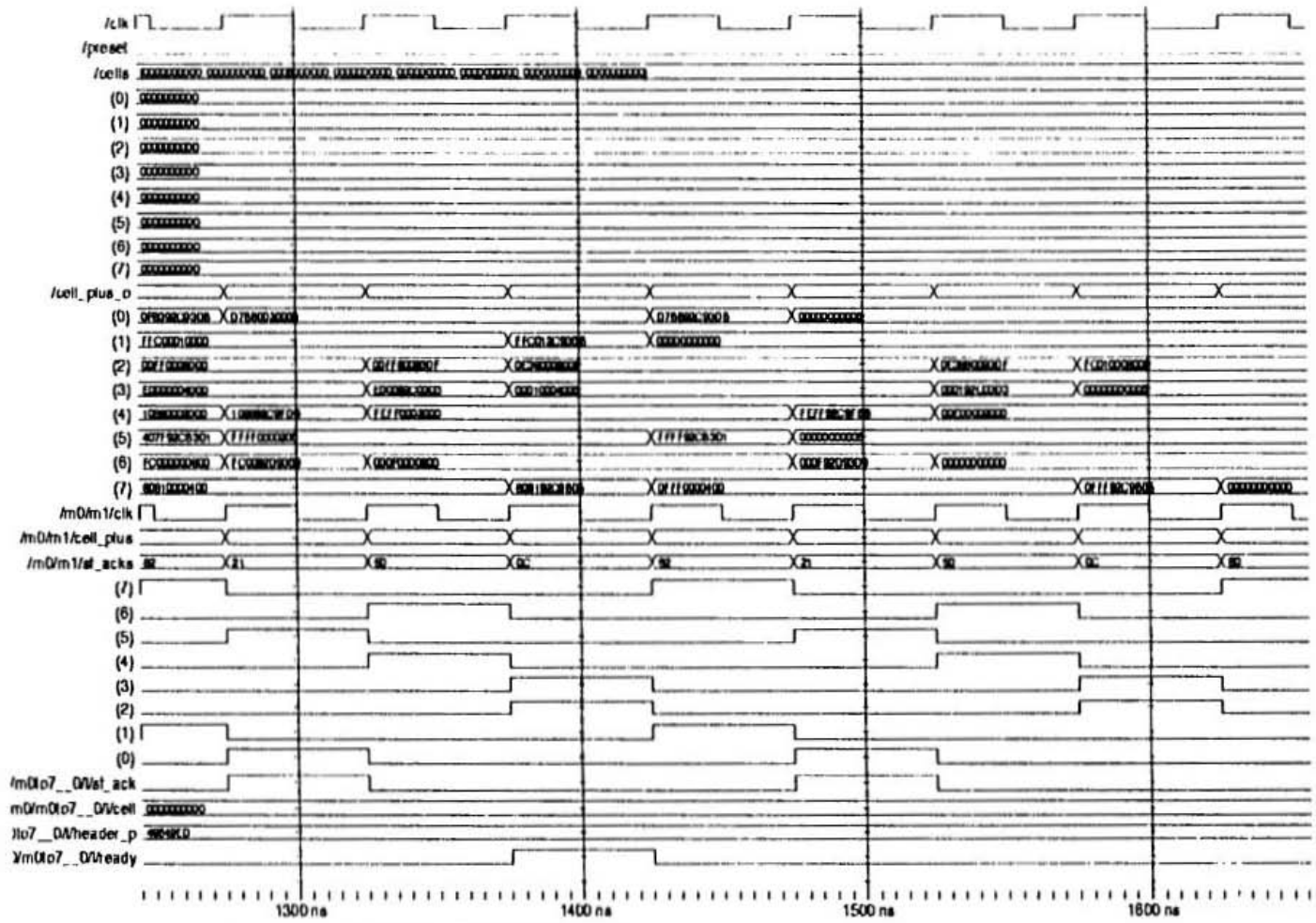


B Behavioral Simulation Result of the IC System

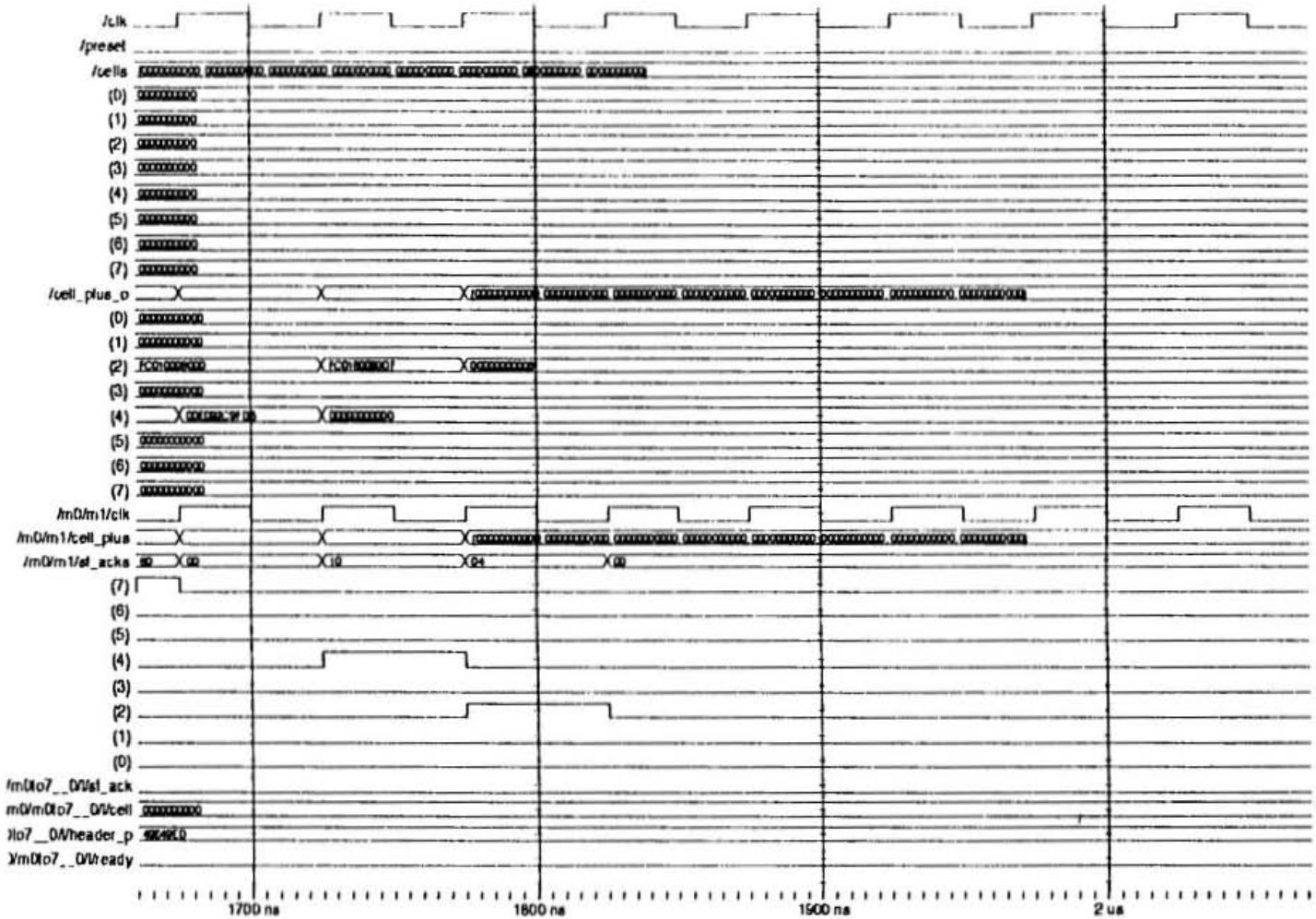



/clk
/preset
/cells
(0)
(1)
(2)
(3)
(4)
(5)
(6)
(7)
/cell_plus_o
(0)
(1)
(2)
(3)
(4)
(5)
(6)
(7)
/m0/m1/clk
/m0/m1/cell_plus
/m0/m1/sl_acks
(7)
(6)
(5)
(4)
(3)
(2)
(1)
(0)
/m0/o7__0/sl_ack
m0/m0/o7__0/cell
/o7__0/header_p
/m0/o7__0/ready
500 ns
600 ns
700 ns
800 ns
Entity bg_ipc_sl_tb Architecture input_output Date Wed Aug 22 16:57:15 2001 Page 1


/clk
/preset
/cells
(0)
(1)
(2)
(3)
(4)
(5)
(6)
(7)
/cell_plus_o
(0)
(1)
(2)
(3)
(4)
(5)
(6)
(7)
/m0/m1/clk
/m0/m1/cell_plus
/m0/m1/sl_acks
(7)
(6)
(5)
(4)
(3)
(2)
(1)
(0)
/m0/o7__0/sl_ack
/m0/m0/o7__0/cell
/o7__0/header_p
/m0/o7__0/ready
900 ns
1 us
1100 ns
1200 ns
Entity bg_ipc_sl_tb Architecture input_output Date Wed Aug 22 16 57 33 2001 Page 1



Entity bg_ipc_sl_tb Architecture input_output Date Wed Aug 22 16:59:05 2001 Page 1



Entity bg_ipc_sl_tb Architecture input_output Date Wed Aug 22 16:59:26 2001 Page 1