INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

A 155-Mbit/s SONET/ATM Interface for a Photonic Backplane

Michael C. Kim

McGill

Department of Electrical Engineering
McGill University, Montreal, Canada

August 1997

A Thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfilment of the requirements of Master of Engineering.

Abstract

McGill University is near completion on a five year Major Project in Photonic Devices and Systems funded by the Canadian Institute for Telecommunication Research (CITR). The objective is to build and demonstrate a representative subset of an electro-optical "Intelligent Optical Backplane" system capable of handling terabit data rates. This thesis describes the construction and demonstration of a prototyping printed circuit board (PCB) known as the *ATMCard*. The goal of the *ATMCard* is two-fold. The first is to develop an electronic PCB supporting ATM switching which can be connected to the 1998 CITR optical backplane demonstrator with minimal adjustments. The second goal is to serve as an initial step towards the specification and development of a more advanced electronic PCB designed for an optical backplane capable of supporting a bandwidth on the order of 10 Gbit/s. *ATMCard* functions include the extraction of ATM cells from SONET frames, ATM cell header translation, cell queueing, access to the CITR optical backplane, and the extraction of ATM cells from the backplane. The *ATMCard* is also a reconfigurable and reusable hardware prototyping platform, enabling rapid testing and verification of digital systems.

The first objective of this thesis, the successful demonstration of an operational *ATMCard* which can be connected with minimal adjustments to the 1998 CITR demonstrator, has been achieved. Furthermore, the detailed descriptions of the *ATMCard* design and implementation included in this thesis provide the framework for the construction and demonstration of more advanced second generation boards. Thus, the second objective of this thesis has also been achieved.

Résumé

Un projet majeur dans le domaine des composantes et des systèmes photoniques sera complété sous peu à l'Université McGill. Ce projet est subventionné par l'Institut canadien de recherche en télécommunications (ICRT). Le but de ce projet est la construction et la démonstration d'une portion représentative d'un système électro-optique nommé "Intelligent Optical Backplane". Ce système est capable de supporter un débit de transmission un terabit/s. La présente dissertation décrit la construction et la démonstration du *ATMCard*, une plaquette de circuits imprimés. Le but du *ATMCard* est de démontrer une interface entre le domaine SONET/ATM et le domaine optique à l'air libre. L'interface traduit les préfixes des cellules ATM, mise en attente des cellules, permet l'accès au fond de panier optique à l'air libre et extrait les cellules ATM du fond de panier. Le *ATMCard* constitue également un outil réutilisable qu'il est possible de reconfigurer selon les besoins pour la production rapide de prototypes. Ceci permet de vérifier rapidement des conceptions numériques. En outre, cette dissertation dévoile les bases nécessaires à la prochaine génération du système qui à son tour permettra un débit de transmission d'un ordre encore plus grand.

Le *ATMCard* a été démontré avec succès, démontrant sa capacité à pourvoir une interface fonctionelle entre les domaines electronique et optique à l'air libre. Au meilleur de la connaissance de l'auteur, cette dissertation constitue la première description détaillée d'une architecture opérant l'interface entre le domaine électronique conventionnel et un fond de panier optique à l'air libre.

Acknowledgments

First and foremost, I would like to express my gratitude to my supervisor, Ted Szymanski, for his support, guidance, and patience during the past three years. Without Ted's supervision and insight, I wouldn't have a thesis and all the valuable hardware experience that resulted from it. I would also like to thank H. Scott Hinton of the University of Colorado at Boulder and Dave Plant of McGill University for their help in the earlier stages of this thesis.

I would like to thank the Microelectronics and Computer Systems (MACS) Laboratory, especially Jacek Slaboszewicz and Alain Magloire, for keeping the network up and running and also for retrieving the files I would always accidentally delete. John Walker of McGill University also helped me immensely when I was trying to get an earlier version of the board running. Thank you very much, John!

During my three years as a graduate student, I have been fortunate enough to meet many people who have shared their time and knowledge with me. From the MACS laboratory, I would like to thank and acknowledge Boonchuay, Palash, Stephane, Sabeen, Victor Zia, Nilanjan, Myriam, Victor Tyan, Albert, Winnie, Xavier, Benoit V., Benoit, D., James, John, Loai, Choon, Mourad, Thomas, Guoying, Arman, Ara, and Manoj. From the Photonics laboratory, I would like to thank David Rolston, Nam, Guillaume, Alain, Marcos, Brian, Wayne, and Yongsheng. Finally, several additional individuals I would like to thank include Rick, Remi, and Majd. I'd like to single out Rick, Boonchuay, Palash, and Stephane for putting up with my constant chatter over the last three years! Also, special thanks are directed to Myriam for translating my abstract at the last moment. I apologize if there is anyone that I have omitted.

During my third year as a graduate student, I spent most of it working at Nortel. I made a few friends who made my stay in Ottawa worthwhile. From Nortel, I would like to thank Eric, Marco, Sherif, and Roger.

Most importantly, I would like to express my deepest gratitude to my parents, Chris, Julia, and her parents. Julia, especially, has enriched my life and made my most trying moments bearable. Thank you.

Funding Acknowledgments

The Canadian Institute for Telecommunications Research (CITR) is a member of the Canadian Networks of Centres of Excellence (NCE). The CITR is currently funding a Major Project in Photonic Devices and Systems [CIT95]. The Major Project consists of four projects located throughout Canada, including the *Optoelectronic Devices*, *Optoelectronic Packaging Concepts*, *Optical and Optomechanical Hardware*, and *Large ATM Architectures* [CIT95].

One of the goals of the Major Project in Photonic Devices and Systems is to construct a free-space optical backplane system using smart pixel arrays. In order to meet this objective, a free-space "Intelligent Optical Backplane" architecture has been proposed [SZY96]. The backplane architecture is currently under development at McGill University in which printed circuit boards will be optically interconnected.

The research presented herein was partially funded by CITR through project number 94-3-4 entitled *Large ATM Architectures*. Additional funding was provided by the Natural Science and Engineering Research Council of Canada grant number OGP0121601. This research was performed in the Microelectronics and Computing Systems Laboratory (MACS Laboratory) at McGill University. The computing resources are on loan to the MACS Laboratory from the Canadian Microelectronics Corporation (CMC) through the 1993-1997 Equipment Loans.

Table of Contents

CHAPTER 1
INTRODUCTION **1**

1.1 MOTIVATION 1
1.2 AUTHOR'S CONTRIBUTIONS 2
1.3 IMPACT OF THE *ATMCARD* 3
1.4 THESIS OVERVIEW 4

CHAPTER 2
ATM NETWORK FUNDAMENTALS **6**

2.1 INTRODUCTION 6
2.2 TRANSFER MODES - WHY ATM? 6
2.3 ASYNCHRONOUS TRANSFER MODE 9
2.3.1 Introduction 9
2.3.2 ATM Protocol Reference Model 11
2.3.3 ATM Traffic Management 24
2.4 BROADBAND ATM SWITCHING 26
2.4.1 ATM Switching Fabrics 28
2.4.2 ATM Cell Queueing 32
2.5 SUMMARY 33

CHAPTER 3
FREE-SPACE OPTICAL INTERCONNECTS **34**

3.1 INTRODUCTION - THE NEED FOR OPTICAL INTERCONNECTS 34
3.2 *HYPERPLANE* SMART PIXELS 37
3.3 THE MCGILL *HYPERPLANE* ARCHITECTURE 41
3.4 SUMMARY 43

CHAPTER 4
SYSTEM SPECIFICATION **44**

4.1 INTRODUCTION 44
4.2 SMART PIXEL ARRAY SPECIFICATIONS 44
4.3 *ATMCARD* SPECIFICATIONS 46
4.4 *ATMCARD* FUNCTIONS 47

4.5 UTP-5 CABLES 48
4.6 SUMMARY 48

CHAPTER 5
ATMCARD OVERVIEW 49
5.1 INTRODUCTION 49
5.2 ATMCARD SYSTEM DESIGN GOALS 49
5.3 COMPONENTS OF THE ATMCARD 51
5.3.1 Xilinx FPGAs 52
5.3.2 Texas Instruments' SONET-ATM Transceiver: The TNETA1500 54
5.3.3 MUSIC Semiconductor's Content-Addressable Memory 55
5.3.4 Memory 57
5.4 SUMMARY 57

CHAPTER 6
ATMCARD DESIGN IMPLEMENTATION 58
6.1 INTRODUCTION 58
6.2 *ATMCARD* DESIGN OVERVIEW 60
6.3 THE *TNET_INTERFACE* 66
6.3.1 Introduction 66
6.3.2 The *Main_Control_Unit* 70
6.3.3 The *IO_Control_Unit* 77
6.3.4 The *CAM_Init_Unit* 82
6.4 THE *HYPERPLANE_INTERFACE* 86
6.4.1 Introduction 86
6.4.2 The *SRAM_Control_Unit* 88
6.4.3 The *SPA_TX_Control_Unit* 91
6.5 THE *TNET_TX_INTERFACE* 95
6.5.1 Introduction 95
6.5.2 The *TNET_TX_FSM* 96
6.6 SUMMARY 97

CHAPTER 7
ATMCARD EVALUATION 98
7.1 INTRODUCTION 98
7.2 VHDL SYNTHESIS RESULTS 101
7.3 SUMMARY 104

CHAPTER 8
CONCLUSION 105

REFERENCES 107

RELATED ARCHITECTURAL PUBLICATIONS AT MCGILL UNIV. 113

List of Figures

Figure 1.1: A pictorial representation of a free-space optical backplane....4

Figure 2.1: Synchronous transfer mode (STM). (Not all of the 24 time slots available in the North American standard are shown)....8
Figure 2.2: Traffic characteristics of broadband services [HIN93]....10
Figure 2.3: The ATM Protocol Reference Model (PRM)....11
Figure 2.4: The SONET STS-1 frame structure....15
Figure 2.5: The SONET STS-*N* frame structure....16
Figure 2.6: (a) Negative STS-1 pointer adjustment operation and (b) positive STS-1 pointer adjustment operation [BAL89]....17
Figure 2.7: ATM cell format for the UNI (a) and NNI (b)....18
Figure 2.8: Virtual path switching [HAN94]....20
Figure 2.9: Virtual channel/virtual path switching [HAN94]....21
Figure 2.10: ATM B-ISDN service classes....22
Figure 2.11: Model of an ATM switch....27
Figure 2.12: Shared media switch....28
Figure 2.13: An *N*x*M* crossbar switch ($N = 4$ and $M = 4$)....29
Figure 2.14: (a) An 8x8 delta-2 network with (b) internal link blocking and (c) output port blocking [AHM89]....30

Figure 3.1: Free-space optical backplane....36
Figure 3.2: Smart pixel arrays [HIN95]....37
Figure 3.3: Typical *HyperPlane* smart pixel [SZY96]....39
Figure 3.4: *HyperPlane* node connectivity [SZY94]....41
Figure 3.5: *ATMCard* and the *HyperPlane* system....43

Figure 4.1: Block diagram of an 8x4 *HyperPlane* smart pixel array used by the *ATMCard*....45
Figure 4.2: High-level block diagram of the *ATMCard*....46

Figure 5.1: Simplified block diagram of the *ATMCard* and the *HyperPlane* showing the major components and *incoming* ATM cell data flow and *outgoing* ATM cell data flow....51
Figure 5.2: Generalized FPGA architecture [ROS93]....53
Figure 5.3: Simplified functional block diagram of the TNETA1500 [TEX94]....55
Figure 5.4: Simplified functional block diagram of the LANCAM [MUS94]....56

Figure 6.1: Equivalent representations of the *ATMCard* and the McGill *HyperPlane* system....59
Figure 6.2: *ATMCard* block diagram (all data path widths are 8-bits except data path connecting *TNET_Interface* can CAM which is 16 bits)....60
Figure 6.3: Operation and ATM cell data flow in the *ATMCard*....62
Figure 6.4 (a): Block diagram of the *ATMCard* components involved in the transmission of ATM cells from the TNETA1500A to the *HyperPlane*....64
Figure 6.4 (b): The *HyperPlane*-to-SONET Interface block diagram....65
Figure 6.5: VHDL hierarchical organization of the three main *ATMCard* VHDL components. The three top level entities - *TNET_Interface*, *HyperPlane_Interface*, and *TNET_TX_Interface* - are shown in **bold**....66
Figure 6.6: Block diagram of the *TNET_Interface*....68
Figure 6.7: Block diagram of the *Main_Control_Unit*....71
Figure 6.8: Functional flow chart for the *Main_Control_FSM*....72
Figure 6.9: *ATMCard* CAM write cycle timing diagram (signals named according to Figure 5.7)....74
Figure 6.10: *ATMCard* CAM read cycle timing diagram (signals named according to Figure 5.7)....75
Figure 6.11: Functional flow chart for the *CAM_Write_FSM*....76
Figure 6.12: Functional flow chart for the *CAM_Read_FSM*....76
Figure 6.13: Block diagram of the *IO_Control_Unit*....78
Figure 6.14: Functional flow chart for the *IO_Control_FSM*....80
Figure 6.15: Linear feedback shift register (LFSR) counters with maximum sequence length of $(2^n - 1)$. (a) A 3-bit LFSR counter (*LFSR3*) and (b) a 6-bit LFSR counter (*LFSR6*) are shown (clock and reset signals are not shown) [ALF94]....82
Figure 6.16: Block diagram of the *CAM_Init_Unit*....83
Figure 6.17: Block diagram of the *HyperPlane_Interface*....87
Figure 6.18: Block diagram of the *SRAM_Control_Unit*....89
Figure 6.19: Functional flow chart for the *SRAM_Control_Unit*....90

Figure 6.20: Block diagram of the *SPA_TX_Control_Unit*. 91
Figure 6.21: Functional flow chart for the *SPA_TX_Control_Unit*....92
Figure 6.22: Timing diagram showing the transmission of an ATM cell to the *HyperPlane* [SZY94a]. 93
Figure 6.23: Block diagram of the *TNET_TX_Interface*....95
Figure 6.24: Functional flow chart for the *TNET_TX_FSM*. 96

Figure 7.1: *ATMCard* evaluation setup showing two *ATMCards* connected together.. 98
Figure 7.2: Photograph of the *ATMCard*. 100

CHAPTER 1
Introduction

1.1 Motivation

Asynchronous Transfer Mode (ATM) is a network protocol that has been introduced to meet the growing bandwidth demands over telecommunication and computer networks. ATM is revolutionary because it aims to merge diverse forms of traffic such as voice, cable, and data onto a single universal network. Presently, there are several specialized networks such as the cable and telephone networks. Each network incurs its own maintenance, administration, and deployment costs. The economic advantages of having a single multi-service network based on ATM are significant.

As bandwidth demands continue to grow, future ATM switching networks may require novel architectures and technologies since physical limits such as crosstalk make it increasingly difficult to build systems entirely in the electrical domain. McGill University, with the Canadian Institute for Telecommunications Research (CITR), has undertaken a multi-year project to develop a reconfigurable free-space optical backplane capable of supporting one trillion bits of data per second. A reconfigurable optical backplane architecture called the *HyperPlane* has been developed. An ATM switching fabric is one possible configuration of the *HyperPlane*. The goal of this thesis is two-fold. First, it is to develop an electronic PCB supporting ATM switching which can be connected with minimal adjustments to the 1998 CITR optical backplane demonstrator. Second, it is to serve as a first step toward the specification and development of a more advanced electronic PCB for an optical backplane capable of supporting a bandwidth on the order of

10 Gbit/s. In particular, this thesis will describe the design and implementation of a transmit/receive PCB used to transport ATM cells across the *HyperPlane*.

The ATM transmit/receive card - also known as the *ATMCard* - utilizes various *Integrated Circuits* (ICs). The main control unit of the *ATMCard* is implemented in a specialized IC known as a *Field-Programmable Gate Array* (FPGA). FPGAs are programmable logic devices which are ideal in rapid prototyping applications. Design entry of the *ATMCard* control components are accomplished using VHDL (*Very high speed integrated circuit Hardware Description Language*). Once the VHDL design code is compiled and synthesized, the design can be programmed into the FPGA.

Another IC central to the *ATMCard* is a SONET/ATM transceiver called the TNETA1500A. The TNETA1500A device is a single-chip implementation capable of transporting ATM cells over a SONET network [TEX94]. In the *ATMCard*, ATM cells are received from the SONET/ATM transceiver by the FPGA and transmitted to the *HyperPlane*. Conversely, cells received from the *HyperPlane* are transported by the FPGA to the SONET/ATM transceiver. Since the *HyperPlane* is currently under development and will not be operational until 1998, a VHDL hardware emulation of the *HyperPlane* was implemented on the FPGA.

Finally, before ATM cells are transmitted over the *HyperPlane*, ATM header address translation occurs in order to switch the cell to its correct destination. The FPGA controls the ATM header translation process. Headers are translated using look-up tables that are implemented in a *Content-Addressable Memory* (CAM) and SRAM. The devices that are used in the *ATMCard* will be discussed in detail in subsequent chapters.

1.2 Author's Contributions

The author's contributions are four-fold. The first contribution is the research and specification of an ATM hardware system which can be connected to the 1998 CITR optical backplane demonstrator with minimal adjustments. This thesis also serves as a first step toward the specification and development of a more advanced board. The third contribution involves the design and implementation of the ATM hardware system. This hardware system incorporates various aspects of ATM technology and integrates them for

use in a free-space optical backplane. The final contribution is the system evaluation of the ATM hardware system.

1.3 Impact of the *ATMCard*

As a fully functional prototype, the *ATMCard* will aid in the development of second generation *HyperPlane* PCBs. Figure 1.1 illustrates a typical free-space optical backplane supporting ATM/SONET traffic. If the backplane is able to support an aggregate bandwidth of 1 Tbit/s, then each of the PCBs are envisioned to support tens of gigabits of data per second. Hence, a second generation *ATMCard* could support a bandwidth on the order of 10 Gbit/s. The architecture of the current *ATMCard* will be used as a foundation for these second generation PCBs. Furthermore, new students working on these future PCBs will benefit tremendously from the lessons learned in the design and implementation of the *ATMCard*.

The *ATMCard* currently contains a single SONET OC-3 data stream. Second generation *HyperPlane* PCBs may contain multiple SONET streams at rates higher than OC-3 (Figure 1.1). Since the long-term objective of the *HyperPlane* project is to have each PCB supporting a bandwidth in the neighbourhood of 10 to 50 Gbit/s, these multiple SONET streams will be necessary [SZY96]. Other possible enhancements for these future PCBs include software-controlled traffic management and monitoring, buffer management, the use of faster FPGAs, and the possible replacement of FPGAs with *Application Specific Integrated Circuits* (ASICs).

Demonstrating a more immediate use, the *ATMCard* may be used with a *HyperPlane* demonstrator due for completion in 1998. The *ATMCard* was designed for use with a *Smart Pixel Array* (SPA) designed in 1994 [SZY94a]. SPAs are optoelectronic devices composed of optical input and output windows along with associated electronic processing circuitry. The 1998 *HyperPlane* demonstrator will use a SPA very similar to the 1994 SPA. Hence, the *ATMCard* may be incorporated within the 1998 *HyperPlane* with little or no modifications. The impact of the *ATMCard* in this system would be substantial since it would allow ATM switching to occur over the 1998 *HyperPlane* demonstrator. Currently, no other research team has demonstrated ATM switching over a free-space optical backplane.

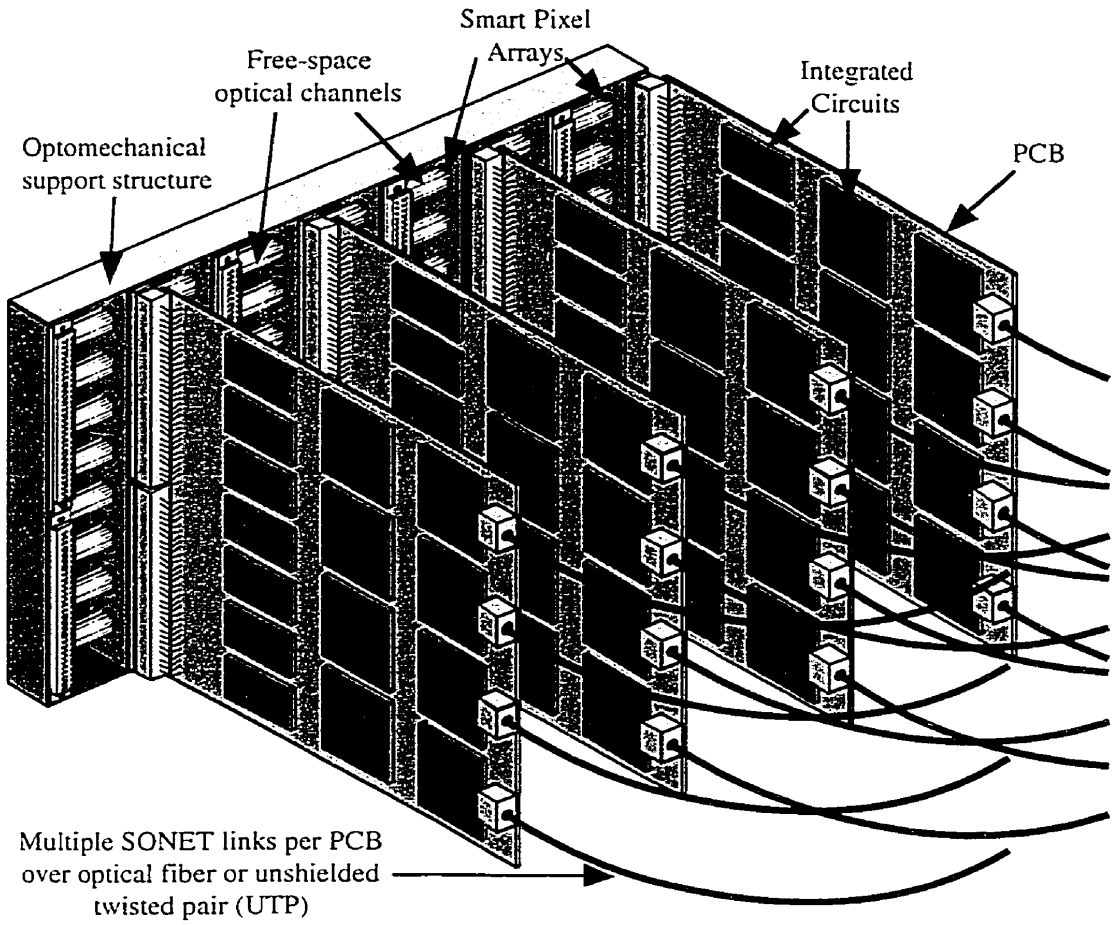


Figure 1.1: A pictorial representation of a free-space optical backplane.

1.4 Thesis Overview

Chapter 2 presents a broad overview of ATM. It depicts the technological evolution towards ATM and follows with a description of ATM including protocol layers and ATM traffic management. Chapter 2 concludes with a discussion of ATM switching fabrics and queueing. ATM is a broad area of research that is difficult to fully grasp in one reading. Therefore, it is not necessary to obtain a detailed understanding of Chapter 2 in order to continue on to subsequent chapters.

Chapter 3 discusses free-space optical interconnects and reasons for their possible use in future high-bandwidth digital systems. One particular project at McGill University,

the smart pixel based *HyperPlane*, is presented. The *HyperPlane* is the target system for the *ATMCard*.

Chapter 4 presents the system specification of the *ATMCard*. Included in this chapter is a specification of the optical backplane.

Chapter 5 presents an overview of the *ATMCard*. It begins with an explanation of the system design goals used for the *ATMCard* and continues with a description of the hardware components used by the *ATMCard*.

Chapter 6 provides a detailed top-down hierarchical description of the *ATMCard*. Each of the VHDL components designed for the *ATMCard* are discussed here. The functional operation of the *ATMCard* is also described.

Chapter 7 discusses the system evaluation of the *ATMCard*. FPGA area and timing statistics are provided as well as a brief description of an ATM cell generator used to source ATM cells for the *ATMCard/HyperPlane* system.

Chapter 8 concludes with a summary of the design and implementation of the *ATMCard*. The *ATMCard* was designed to be compatible for use with the CITR 1998 demonstrator with minimal adjustments. Also, with the knowledge obtained from the design and implementation of the *ATMCard*, the foundation has been made enabling faster development of second generation *HyperPlane* PCBs.

CHAPTER 2
ATM Network Fundamentals

2.1 Introduction

ATM is a novel network protocol promising integration of diverse broadband multimedia services over a high-speed network. The complexity of such a network makes ATM a vast area of research. For completeness, this chapter will attempt to provide a general overview of the key aspects of ATM. The reader does not require a detailed understanding of Chapter 2 to comprehend subsequent chapters. Hence, a "light review" of Chapter 2 is recommended. Advanced readers requiring additional information are referred to the references.

The reasons behind the introduction of ATM are discussed in the first section of this chapter (2.2 Transfer Modes - Why ATM?). The following section (2.3 Asynchronous Transfer Mode) describes the ATM network layer model. Section 2.3 includes a discussion of the *Synchronous Optical Network* (SONET), ATM header translation (VCI/VPI switching), and ATM traffic management. A description of the 53-byte ATM cell is also provided. The final section (2.4 Broadband ATM Switching) introduces ATM switching fabrics and ATM cell queueing. Again, the reader is reminded that it is not a requirement to have a detailed knowledge of this chapter in order to understand subsequent chapters. Sections that are needed, however, include the description of the 53-byte ATM cell, ATM header translation (also known as VCI/VPI translation), and a rudimentary knowledge of ATM switches.

2.2 Transfer Modes - Why ATM?

Current and future telecommunication networks will require an increasing amount of integration of high-bandwidth services. Existing single-traffic networks such as voice and cable can handle service integration only in a limited capacity. The reason for the recent focus on service integration is the economic advantage that arises from having a single network serving the demands of a wide variety of existing traffic with the capability of supporting future applications [VIC93]. *Narrowband-Integrated Services Digital Network*

(N-ISDN or just ISDN) was a first step at integrating voice and data traffic over a single network [DEP93]. The transfer mode used for ISDN is the Synchronous Transfer Mode (STM), otherwise known as circuit switching.

The *International Telecommunications Union - Telecommunications Standardization Sector* (ITU-T) defines a transfer mode as the method by which a telecommunication network handles the transmission, multiplexing, and switching of traffic. STM has long been used in telephone networks. The fundamental transmission rate of STM is based on the bandwidth required to effectively transmit the human voice over a communication channel.

The human voice contains frequencies from 0 to about 4 kHz. From a theorem developed by Harry Nyquist, it is known that discrete-time sampling must occur at least twice as fast as the highest frequency produced by an analog signal - the human voice, in this case - in order to accurately reproduce that analog signal [KES93]. Thus, the sampling frequency required for the human voice is at least 8 kHz corresponding to a time interval of 125 μs. Each sample is converted into a digital signal using *Pulse Code Modulation* (PCM) in which a sample's amplitude is quantized into 256 levels (8 bits). Therefore, the human voice requires 64 kbit/s (8 bits every 125 μs is equal to 64 kbit/s) of bandwidth.

Information carried on an STM network is based on these 64 kbit/s signals. In North America, 24 of these voice signals are *Time-Division Multiplexed* (TDM) together with framing bits to form what is known as a *frame* (Figure 2.1). Each of the 24 voice signals resides in its own *time slot*. The resulting bandwidth of these 24 64 kbit/s voice signals, also called a T1 or DS1 frame, is 1.544 Mbit/s. A single telephone connection ("circuit") will always use the same time slot for the duration of the call. Other users cannot use this time slot even during periods when the two users are silent. ATM overcomes this inefficient use of bandwidth by setting up what is known as virtual circuits. In an ATM network, a connection is made between end users that will be used for the duration of the call. However, in ATM other users will be able to share any unused bandwidth through a scheme called statistical multiplexing. Statistical multiplexing takes advantage of the fact that voice packets belonging to a particular connection will not generally follow one after the other. In other words, they will not utilize the entire channel bandwidth. Any remaining bandwidth will then be free for use by other connections [VET95].

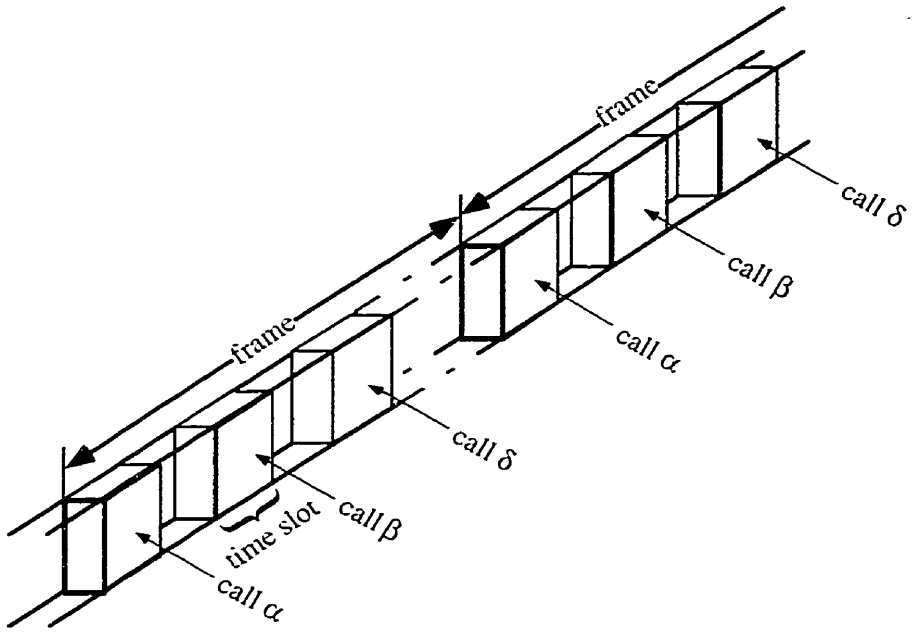


Figure 2.1: Synchronous transfer mode (STM). (Not all of the 24 time slots available in the North American standard are shown.)

In addition to inefficient bandwidth utilization, pure circuit switching is inappropriate for an integrated services network since all services are offered the same fundamental bit rate (64 kbit/s). Services requiring bit rates below 64 kbit/s will be allocated the entire bandwidth leading to an inefficient use of resources. On the other hand, very high bandwidth services will not be allocated sufficient bandwidth. This inflexibility in time slot bandwidth becomes even more troublesome when considering future applications whose bandwidth characteristics are presently unknown. Therefore, STM's potential for scalability appears limited.

One proposed solution was to allow a connection to utilize several basic rates (in STM's case this is 64 kbit/s) simultaneously [DEP93]. This transfer mode is called *Multirate Circuit Switching* (MRCS). A problem arises when trying to choose the size of this basic rate. If the basic rate is chosen to be small then applications requiring large bit rates become difficult to manage and correlate since they will use a large number of these basic rates. If a larger basic rate is chosen so that high bit rate applications such as digital video become easier to manage, then bandwidth will be wasted for low bit rate applications such as voice and facsimile.

In order to overcome the problem of optimal base rate selection, another solution was proposed whereby multiple basic rates would be used. This is the solution currently

employed in ISDN. Here, the basic rates are chosen so that they more closely match the known bit rates of available services. For example, ISDN includes as one of its basic rates the B channel which has a bit rate of 64 kbit/s. For applications such as medical imaging and high-definition television (HDTV), it also offers higher bit rate channels: the H_0 primary rate access channel running at 384 kbit/s, the H_{11} primary rate access channel running at 1.536 Mbit/s, and the H_{12} offering a bit rate of 1.920 Mbit/s [HAN94].

Despite these improvements, difficulties arise when these networks try to support several integrated services whose traffic characteristics differ greatly. Furthermore, since the numerous basic rates were chosen with existing applications in mind, future applications may not be serviced efficiently with the current partitioning of bandwidth. Finally, STM networks in general are unable to cope with traffic that is bursty in nature. Bursty traffic is characterized by having a large peak-bandwidth-to-average-bandwidth ratio. In circuit switching, once a connection is made between two or more parties, the connection is maintained and the bandwidth is retained for the duration of the call, whether that bandwidth is being utilized or not. Inefficient bandwidth usage is quite common for bursty traffic since the selected channel rate must be at least as large as the peak bit rate of the bursty source.

Although circuit switching networks may have suited the limited requirements demanded by users in the past, telecommunication groups now realize that in order to cope with diverse high-bandwidth applications in single digital network, a new transfer mode was required. In addition, this new transfer mode must allow for the integration of future applications whose traffic characteristics are presently unknown. The transfer mode chosen for this new network, the *Broadband Integrated Services Digital Network* (B-ISDN), is the *Asynchronous Transfer Mode* (ATM) [MIN89].

2.3 Asynchronous Transfer Mode

2.3.1 Introduction

Asynchronous Transfer Mode (ATM) is emerging as the base technology for the next generation of broadband communication networks. Diverse applications such as medical imaging, high-speed data transfer, digital television, voice, and the *Global*

Positioning System (GPS), with their widely varying traffic characteristics, will benefit from the flexibility and speed that ATM provides (Figure 2.2). ATM's versatility supports the integration of future applications whose traffic characteristics are currently unknown. In terms of economics, having one universal broadband network catering to the demands of a multitude of services is much more efficient than having an overlay of several networks each specialized for a particular traffic type. In addition to telecommunication related applications, another application that will benefit from ATM is distributed computing [VET95], making ATM a truly universal broadband network.

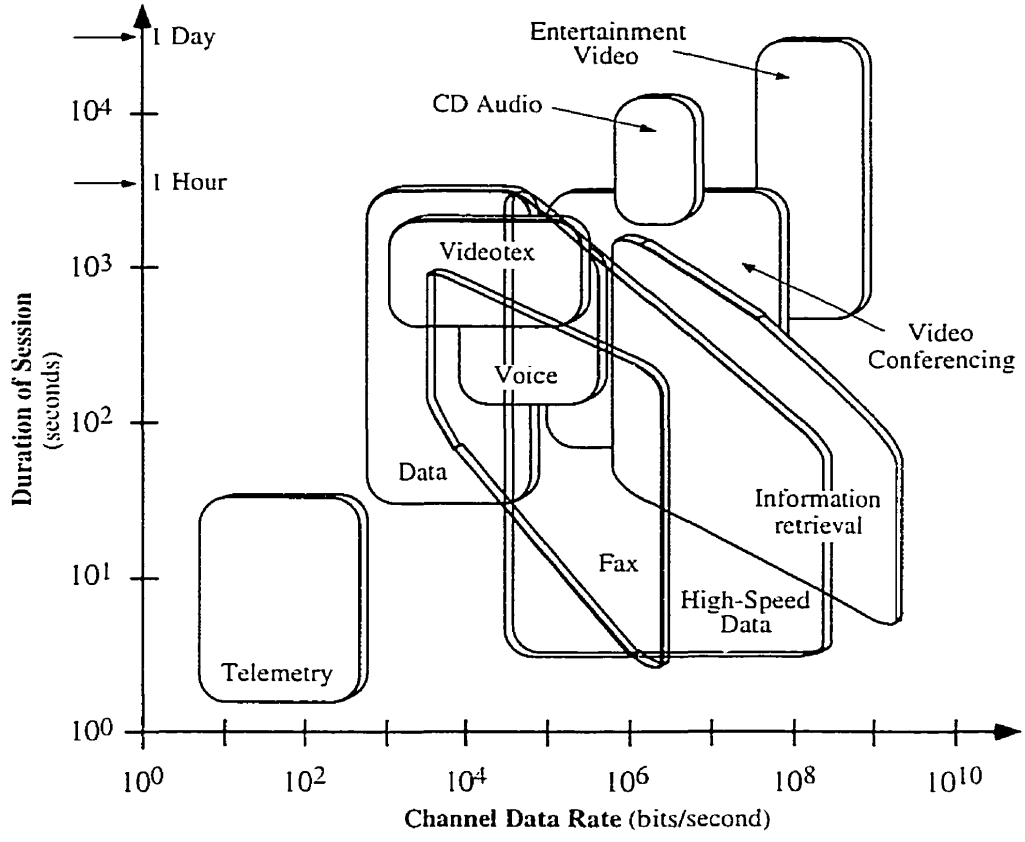


Figure 2.2: Traffic characteristics of broadband services [HIN93].

Previous packet switched networks such as X.25 and frame relay required complex control protocols, such as cyclic redundancy check (CRC) for bit error control and automatic repeat request (ARQ) for error recovery, in order to ensure the successful transmission of packets. This complex control mechanism was needed due to the poor quality of the physical links. Today, the low bit error rates of high quality fiber optics enables the simplification of the control protocols in ATM networks. This simplification leads to a much faster network [DEP93]. The remainder of this chapter will describe

various aspects of ATM including the ATM protocol reference model, ATM traffic management, and ATM switching. It is not the purpose of this chapter to provide a detailed explanation of all aspects of ATM, only an introductory overview. However, since ATM is such a broad topic, even this overview delves deeper into ATM than is needed in order to gain an understanding of subsequent chapters. Further reading on ATM is provided in the references [ATM93], [DEP93]. [HAN94], [MIN89], [ROO94], [SIU95], [STI95], [VET95], [VIC93].

2.3.2 ATM Protocol Reference Model

The seven layer *Open Systems Interconnect* (OSI) model developed by the *International Standards Organization* (ISO) has proven to be a successful representation of communication networks. A similar model has been adopted for ATM networks. The ATM *Protocol Reference Model* (PRM) is composed of three planes (Figure 2.3): a *User Plane* that is concerned with the transmission of user information, a *Control Plane* that handles call control and connection control functions, and a *Management Plane* that is composed of two types of functions known as *Plane Management* and *Layer Management* [HAN94]. *Plane Management* functions provide coordination for all three planes while *Layer Management* functions deal with the *Operation and Maintenance* (OAM) functions for each layer. Each of these planes lie vertically in Figure 2.3. The *Management Plane* spans across all layers in Figure 2.3 while the *Control* and *User Planes* span the *ATM Adaptation Layer* (AAL) and the higher layer protocols.

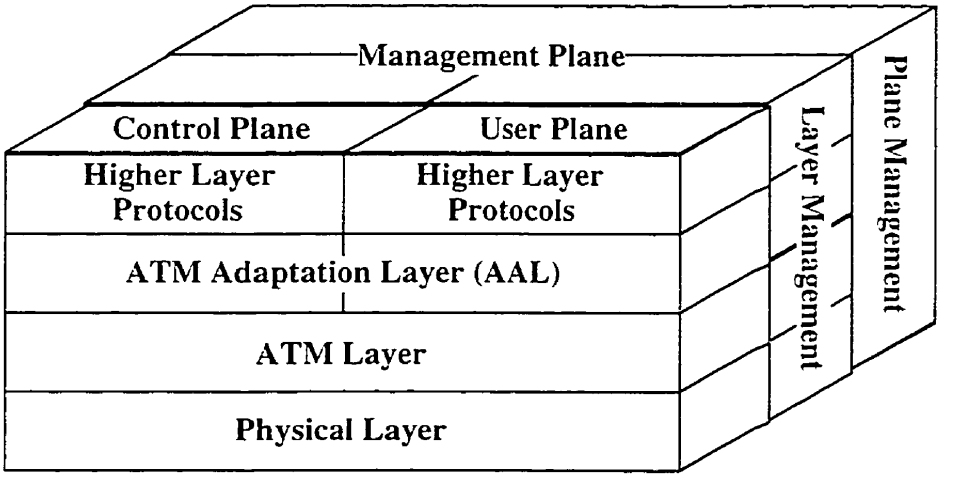


Figure 2.3: The ATM Protocol Reference Model (PRM).

The *User Plane*, the *Control Plane*, and the *Layer Management* portions of the *Management Plane* are further divided into three layers: the *Physical Layer*, *the ATM Layer*, and the *ATM Adaptation Layer* (AAL). This section will focus on these three layers. The *Physical Layer* is responsible for defining the transport method that individual ATM cells will use to travel between ATM entities such as an end user or an ATM switch. The *ATM Layer* is mainly concerned with performing the switching and multiplexing functions for the ATM cells. This layer provides the transparent transfer of fixed-size 53 byte ATM cells. Finally, the *ATM Adaptation Layer* (AAL) is responsible for defining the set of service classes that are required to match the needs of different applications. The AAL adapts user information to the ATM stream and vice versa [VET95].

Layer	Sublayer	Functions
AAL	CS	Convergence
	SAR	Segmentation and reassembly
ATM		Generic flow control Cell VPI/VCI translation Cell multiplex and demultiplex
Physical	TC	Cell rate decoupling HEC header sequence generation/verification Cell delineation Transmission frame adaptation Transmission frame generation/recovery
	PM	Bit timing Physical medium

CS Convergence sublayer
HEC Header error control
PM Physical medium
SAR Segmentation & reassembly
TC Transmission convergence
VCI Virtual channel identifier
VPI Virtual path identifier

Table 2.1: The ATM PRM layers and functions [DEP93].

Finally, the Physical, ATM and AAL layers are themselves composed of sublayers (Table 2.1). The following subsections will present these three layers and their corresponding sublayers (Table 2.1) in greater detail. Again, the detail provided is more than is required to understand subsequent chapters, hence a "light review" is recommended.

2.3.2.1 The Physical Layer

The *Physical Layer* defines the characteristics needed for encoding and decoding data into waveforms that will be transmitted and received over a specific physical media [SIU95]. This layer is subdivided into the *Physical Medium* (PM) and *Transmission Convergence* (TC) sublayers (Table 2.1).

The TC sublayer recognizes individual bits. After bit reconstruction, this sublayer adapts the data to the transmission system in use. A few transmission systems that have been defined for use with ATM are the *Synchronous Optical Network* (SONET), the *Plesiochronous Digital Hierarchy* (PDH), and the *Fiber Distributed Data Interface* (FDDI) [DEP93]. The TC sublayer is responsible for the generation of a *Header Error Check* (HEC) field which provides a means for a receiver to perform proper cell delineation. The HEC field of the ATM header locates and corrects single-bit errors that occur in the header. Due to the high quality of fiber optics and their low bit error rates, error correction functions are simplified. Furthermore, an ATM receiver performs cell delineation on the HEC field of an ATM header and not on the payload. (The structure of an ATM cell will be described shortly.) This feature results in a simpler receiver design. Finally, the TC sublayer is responsible for inserting and suppressing unassigned ATM cells according to the portion of the channel bandwidth that is being used by user applications. (An unassigned cell is identified by a standardized virtual path identifier (VPI) and virtual channel identifier (VCI) value, which has been generated and does not carry information from an application using the ATM Layer service [ATM93].) This is known as cell rate decoupling [DEP93].

The PM sublayer is the lowest sublayer and deals only with physical medium dependent functions such as bit alignment, bit timing, line coding, and electrical/optical data conversion. For the most part, the physical medium will be optical fiber although other media such as coaxial and twisted pair cables may be used. The next subsection will discuss the *Synchronous Optical Network* (SONET) protocol. SONET is a transmission system that may be used to transport ATM cells.

2.3.2.1.1 SONET

This section will provide a detailed description of SONET. Readers may omit this section without any loss of continuity. The reader must only be aware that SONET is an optical transport system that is capable of carrying ATM traffic.

The *Synchronous Optical Network* (SONET) is a synchronous transmission standard for optical networks that has been acknowledged as an international transmission standard used for framing and synchronization at the physical layer. SONET (also known as SDH in Europe) is able to carry several *plesiochronous* signals - signals operating at roughly the same data rates with any variation in rates being confined within specified limits - in a synchronous environment [BAL89], [OMI93].

The fundamental unit of the SONET signal hierarchy is the 51.84 Mbit/s *Synchronous Transport Signal - Level 1* (STS-1). The *Optical Carrier - Level 1* (OC-1) signal is derived from the STS-1 after scrambling and electrical-to-optical conversion. Scrambling is performed to aid in receiver clock recovery by avoiding long strings of ones or zeroes [BAL89]. The STS-1 frame structure can be viewed as a two-dimensional entity with 90 columns and 9 rows of 8-bit bytes (Figure 2.4). The order of transmission is row by row, from left to right, starting at 0 µs and ending at 125 µs. The time interval required to send an entire STS-1 frame - 125 µs - is the same time needed to transmit one sample of 8 bits over an STM voice network. The first three columns of the STS-1 frame comprise the section and line overhead bytes while the remaining 87 bytes are for the *Synchronous Payload Envelope* (SPE). The SPE is used to carry SONET payloads. The SPE also includes 9 bytes of path overhead.

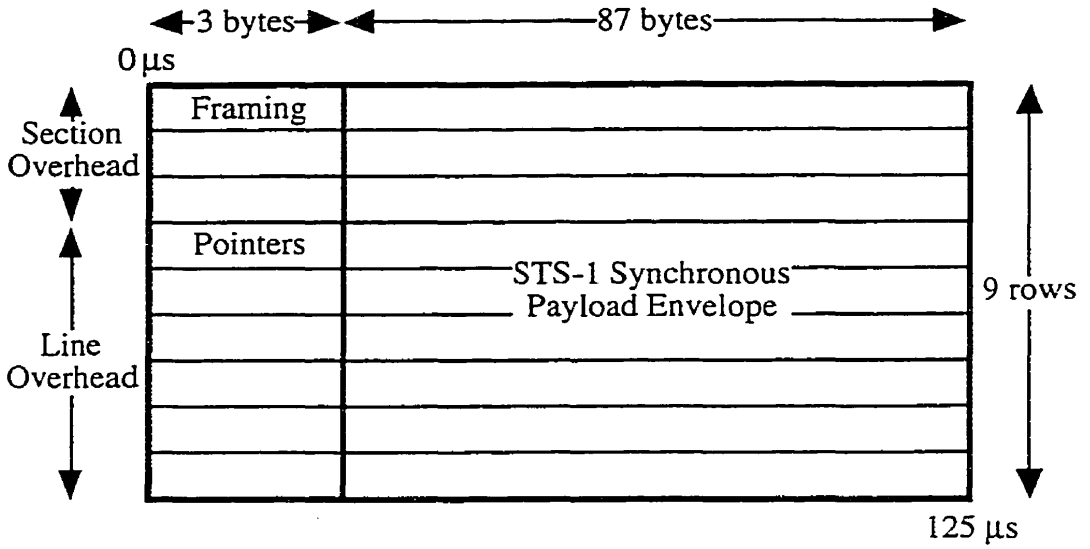


Figure 2.4: The SONET STS-1 frame structure.

The SONET STS-1 frame contains overhead bytes which are divided into section, line, and path layers. Figure 2.4 illustrates that the section and line overheads occupy the first three columns of a SONET frame. The path overhead is embedded within the SPE. The section overhead contains two framing bytes that signal the start of each STS-1 frame, an STS-1 identification byte, an 8-bit Bit-Interleaved Parity (BIP-8) check to monitor section overhead errors, a channel for maintenance purposes, and user channels for future use. When a SONET frame is scrambled, the framing bytes and the identification byte of the section overhead are left unscrambled. The line overhead includes three STS-1 pointer bytes (which will be discussed shortly), a BIP-8 check for monitoring line overhead errors, and additional channels for messaging and future use. Finally, the path overhead of a SONET STS-1 frame includes a path BIP-8 error check for payload error monitoring, a label field to identify the type of payload being carried, and others.

One main advantage of SONET is the ease of forming higher data rate signals by byte-interleaving *N* frame-aligned STS-1 signals to form an STS-*N* signal (Figure 2.5) [BAL89]. All the section and line overhead bytes from the STS-1 #1 signal are used in the STS-*N* frame. However, many of the overhead bytes from the remaining *N* - 1 STS-1 signals are unused. The only bytes that are kept from the remaining signals are the framing, STS-1 ID, and BIP-8 channels from the section overhead and the pointer and BIP-8 channels from the line overhead. The STS-*N* frame is then scrambled to form an Optical Carrier - Level *N* (OC-*N*) signal. In this manner, higher bit-rate frames may easily be derived from the fundamental STS-1 signal. Typical OC-*N* levels are OC-3 (155.52 Mbit/s), OC-12 (622.08 Mbit/s), and OC-48 (2.48832 Gbit/s).

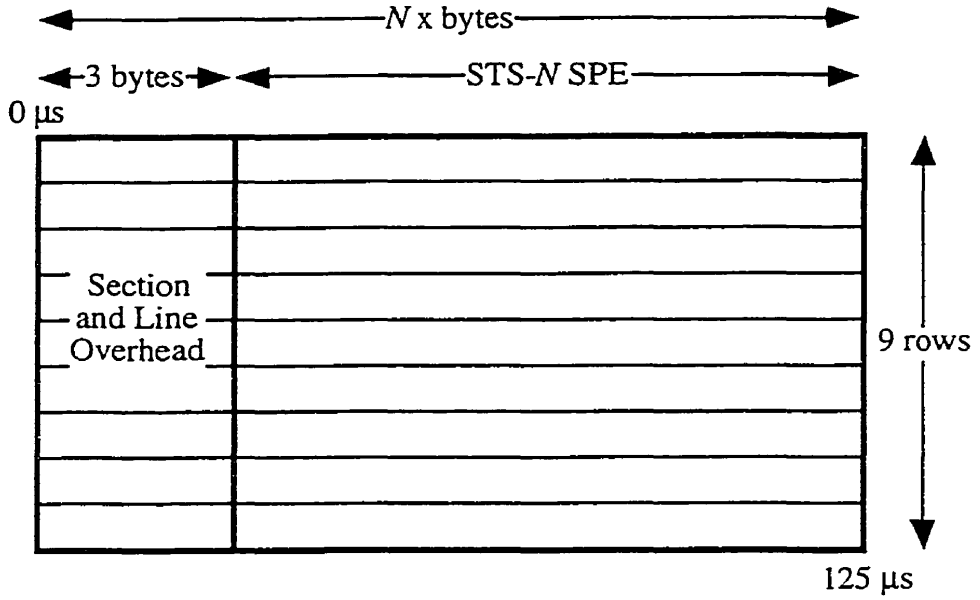


Figure 2.5: The SONET STS-*N* frame structure.

Maintaining synchronization when multiplexing payloads into a higher rate signal is accomplished in SONET via the use of pointers which are located in the line overhead. The payload pointer is a value stored in bytes H1 and H2 (Figure 2.6) of each STS-1 line overhead. The payload pointer indicates the location of the starting byte of the STS-1 SPE payload within the STS-1 frame. Thus, the payload is not set at a fixed location within the STS-1 payload. Small frequency variations that occur in the payload of an STS-1 frame are accounted for by either incrementing or decrementing the pointer value (Figure 2.6). In this manner, the synchronous frame structure of SONET is able to deal with plesiochronous signals. For example, if the payload signal rate is slightly higher than the SONET frame rate then the pointer value is decremented by one and the H3 line overhead byte (next to the H2 pointer byte) is used to carry payload data for one frame (Figure 2.6 (a)). Likewise, the payload pointer value is incremented by one and the data byte immediately following the H3 line overhead byte is nulled for one frame if the payload signal rate is slightly less than the SONET frame rate (Figure 2.6 (b)). In effect, the size of the SONET payload may be incremented or decremented by one byte in a plesiochronous environment. Using pointers avoids data slips and the resulting data loss that occur when using more conventional methods such as bit-stuffing and fixed location mapping to multiplex payloads into higher rate signals [BAL89].

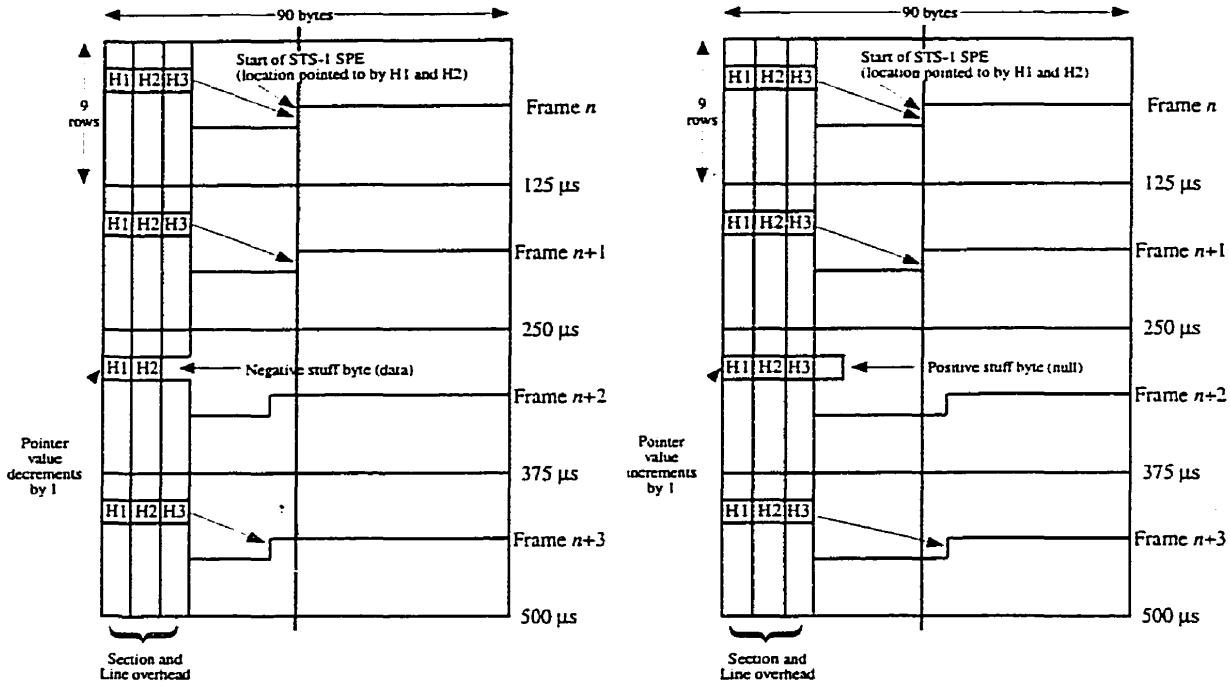


Figure 2.6: (a) Negative STS-1 pointer adjustment operation and (b) positive STS-1 pointer adjustment operation [BAL89].

Finally, if a high data rate service requires *N* STS-1 payloads, then the phase and frequency of *N* STS-1 payloads must be locked together. This is done by grouping together the STS-1 pointers in an STS-*N* signal and aligning all STS-1 signals within the STS-*N* frame [BAL89]. A *concatenation indication* is used in the second and third STS-1 pointers. This *concatenation indication* simply informs any STS-1 pointer processor that the value of this pointer must be the same as the previous STS-1 pointer. Therefore, by using pointer concatenation and frame aligning STS-*N* signals, larger payloads formed from multiple STS-1 payloads may be created. The resulting signal is known as an STS-*N*c signal where the "c" stands for concatenation. European telecommunication groups such as the ITU-T had little use for the STS-1 signal. They preferred a base signal that operated near 150 Mbit/s. For this reason, the STS-3c, which operates at 155.52 Mbit/s, was chosen to be the building block of the signal hierarchy for SDH. In SDH, the STS-3c signal is known as the *Synchronous Transport Module - Level 1* (STM-1).

The reader is referred to [BAL89] for further details on the SONET protocol.

2.3.2.2 The ATM Layer

The ATM layer serves as an interface between the AAL and the physical layers and is independent of the transmission medium used to transport the ATM cells. This layer defines a fixed ATM cell size of 53 bytes. The choice of this cell size resulted from a compromise between North American and European telecommunication organizations. In high-speed fast packet switching, the header of a cell must be processed extremely quickly. North American observers felt that a cell with a 64 byte payload would give switching systems enough time to process the header. Europeans were focusing more on the integration of services, including voice. At 64 kbit/s, a 64 byte cell would take 8 ms to be filled with voice traffic. This increase in delay for voice services would require additional funds to employ echo cancellers, something the Europeans were opposed to do. Thus, the Europeans proposed a cell payload size of 32 bytes. Finally, a compromise was reached where the cell would be comprised of a 5 byte header and a 48 byte payload (the average of 32 and 64). This is one of the reasons why ATM is sometimes referred to as a compromise technology [VIC93].

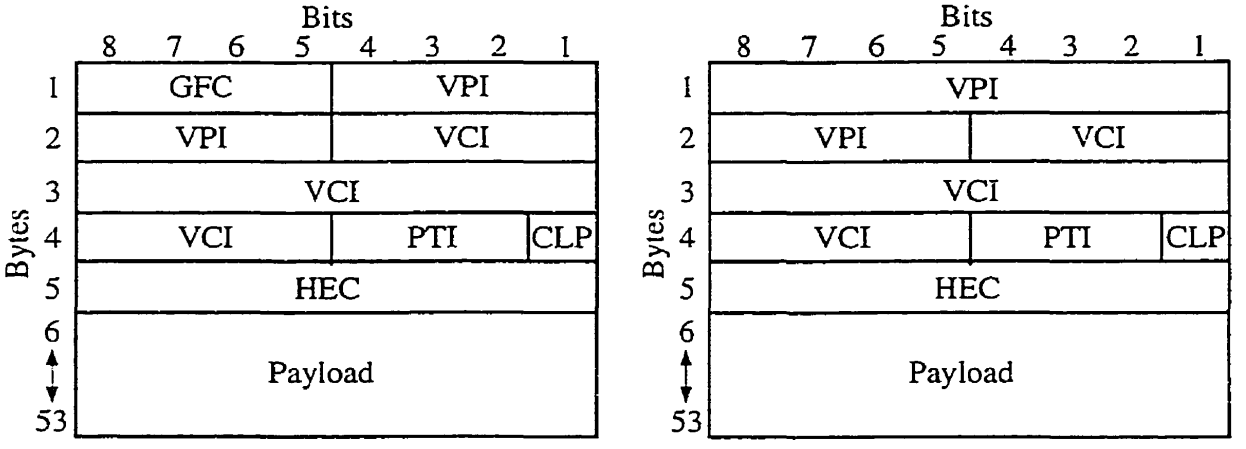


Figure 2.7: ATM cell format for the UNI (a) and NNI (b).

The 5-byte ATM cell header formats are shown in Figure 2.7. The header formats are defined for the *User-to-Network Interface* (UNI) and the *Network-to-Node Interface* (NNI). The UNI defines the interface at the user end of the network while the NNI defines the interface between ATM switches and/or nodes. The only difference between the two is that the four most significant bits of the first byte in the UNI cell format house the *Generic Flow Control* (GFC) fields, whereas the NNI cell uses these bits for the *Virtual Path*

Identifier (VPI). ATM bytes are transmitted in increasing order starting with byte 1. Within a byte, the bits are sent in decreasing order beginning with bit 8 and ending with bit 1 (Figure 2.7).

ATM layer functionality is defined by the cell header format shown in Figure 2.7. The following is a summary of the main functions provided by the ATM layer [DEP93].

1) The multiplexing and demultiplexing of cells of several different connections are performed with the aid of the *Virtual Path Identifier* (VPI) and *Virtual Channel Identifier* (VCI) fields.

2) VPI/VCI cell header translation is accomplished when switching an ATM cell from one physical link to another across an ATM switch.

3) The ATM layer provides a certain *Quality of Service* (QoS) class for a connection through the *Cell Loss Priority* (CLP) field.

4) Management functions such as congestion indication are handled in the ATM layer using the *Payload Type Identifier* (PTI) field.

5) Extraction of the header is performed before the cell is sent to the AAL layer, while insertion of the header occurs after the AAL layer delivers the cell to the ATM layer.

6) Flow control is performed in the ATM layer through the *Generic Flow Control* (GFC) bits of the ATM cell header. (This field is found only in the UNI ATM cell format.)

7) A simple error checking scheme for the ATM cell header is provided by the *Header Error Control* (HEC) field.

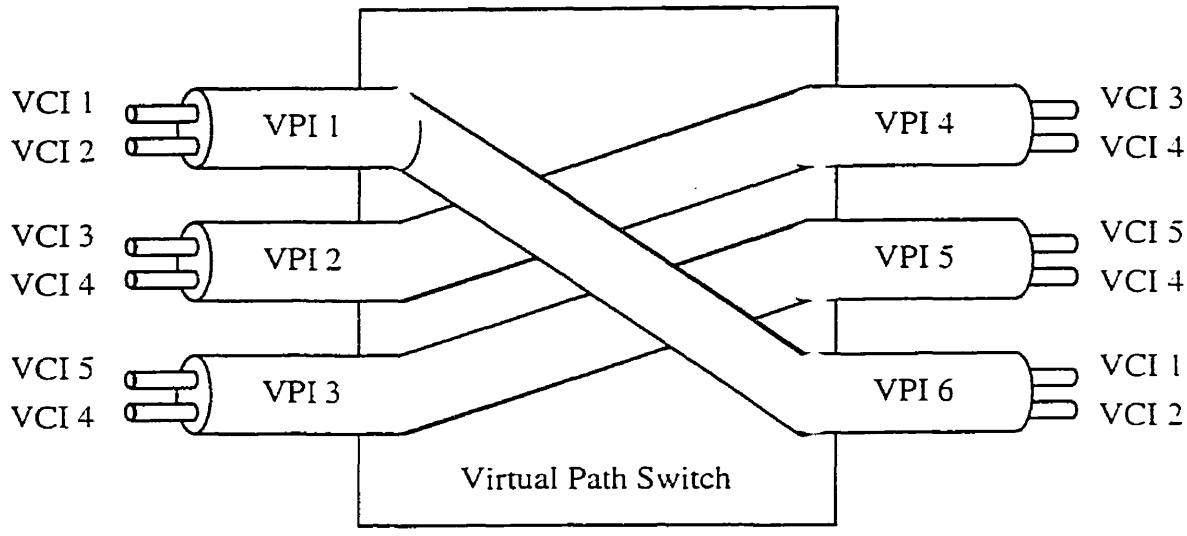


Figure 2.8: Virtual path switching [HAN94].

ATM cells of different connections are identified and distinguished by their VCI and VPI fields. The VPI and VCI fields of the ATM header are used in the routing of ATM cells in a connection. The VCI identifies a single channel while the VPI groups (or multiplexes) individual *Virtual Channels* (VCs) with different VCIs, so that an entire group of VCs may be switched as a single entity. Upon entering an ATM switch on a given link, an ATM cell's VPI field is used to determine the output port that the incoming cell must take. In addition, a new VPI field will be inserted to replace the old VPI field (VPI cell header translation) and the cell is forwarded to the next switch or end-user. This is known as VP switching and is illustrated in Figure 2.8. From this figure, it should be noted that virtual channels within a VP are switched as one entity. However, VPI and VCI fields in a header may also be translated simultaneously in what is known as VP/VC switching. This is shown in Figure 2.9.

It should be stressed that the VPIs and VCIs have significance only on the local physical link. The VPI and VCI values in the ATM cell header will generally change as the cell travels from link to link. The VPI and VCI fields in the UNI cell format comprise 24 bits. This corresponds to over 16 million simultaneous connections (2^{24} = 16,777,216). For the NNI format, there are 28 bits in the VPI and VCI fields which translates to over 260 million connections (2^{28} = 268,435,456) [SIU95].

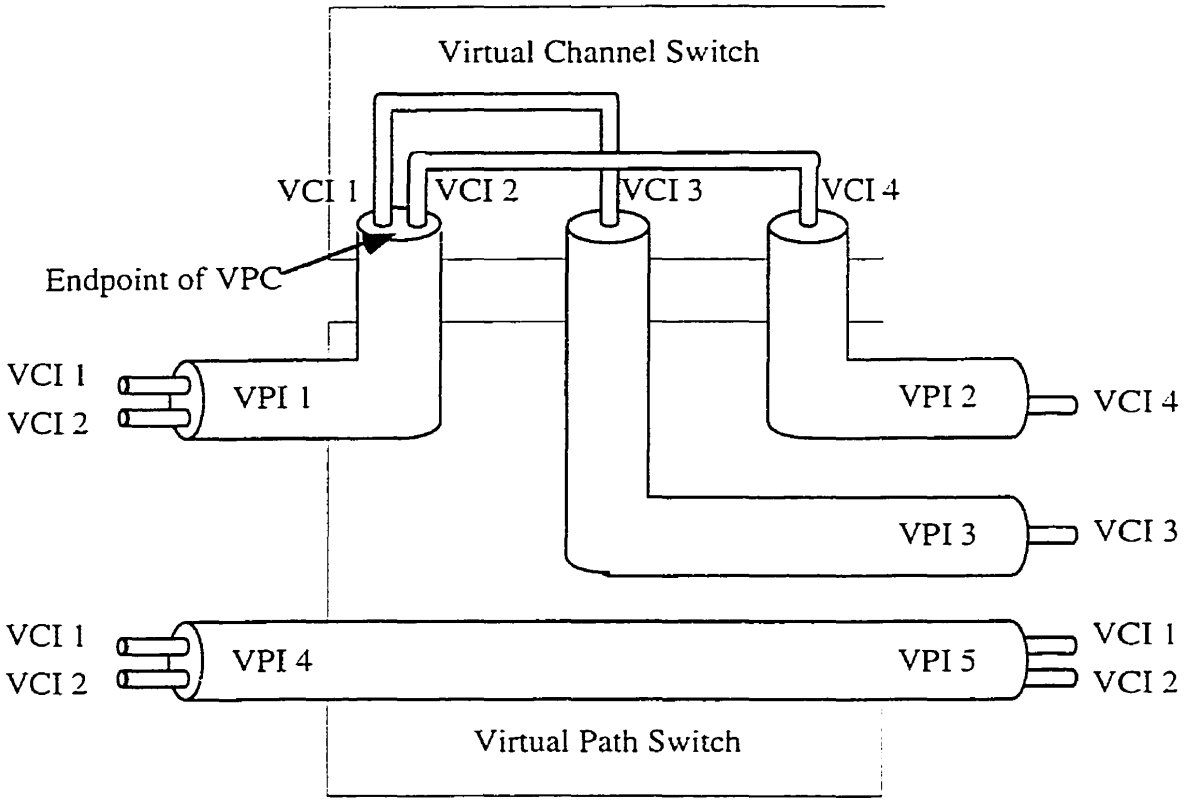


Figure 2.9: Virtual channel/virtual path switching [HAN94].

The GFC field (Figure 2.7) is used by the UNI to control the traffic rate entering the network. This four bit field allows the UNI to limit the amount of data entering the network during periods of congestion. In the NNI, these four bits are used for the VPI.

The PTI field (Figure 2.7) is used to distinguish between ATM cells that are carrying user information and cells that are carrying control information. This field allows control and signaling data to be transmitted on other subchannels which are different from the user channels. In addition, the PTI field may be used to indicate traffic congestion.

The CLP field (Figure 2.7) is used to ensure that a particular connection is offered a certain *Quality of Service* (QoS) class. This field is used to indicate whether an ATM cell may be discarded during periods of network congestion. For example, voice and video applications can usually tolerate a minor amount of data loss whereas data applications cannot. Therefore, a higher cell loss priority will be given to voice and video traffic enabling some of those cells to be dropped during periods of network congestion. The CLP field may also be used by the network to tag cells that exceed the negotiated bandwidth limit allocated to that particular connection.

The HEC field (Figure 2.7) is used to reduce the errors that may occur in the header but not in the payload field. Multiple-bit errors are detected with high probability. In addition, single-bit errors can be corrected. This is usually sufficient since for ATM connections traversing fiber optic links the probability of a bit error is less than 10^{-9}. Therefore, most header errors may be corrected with single bit error correction [SIU95].

2.3.2.3 The ATM Adaptation Layer

The purpose of the *ATM Adaptation Layer* (AAL) is to provide a link between higher layer protocols and the ATM layer. The *Protocol Data Units* (PDUs) of higher layer protocols are mapped by the AAL into the information field of one or more 53-byte ATM cells of a virtual connection. In the other direction, the AAL takes the ATM cells and reassembles the payloads into a format that higher layer protocols can understand. This process is called *Segmentation and Reassembly* (SAR) and is performed by the SAR sublayer of the AAL. The *Convergence Sublayer* (CS) of the AAL (Table 2.1) is service dependent and may include functions such as higher-layer error detection and correction, allocating buffer sizes, source clock recovery at the receiver, and handling of cell delay variation [DEP93].

	Class A	**Class B**	**Class C**	**Class D**
Timing between source and destination	Required		Not required	
Bit rate	Constant	Variable		
Connection mode	Connection oriented			Connectionless

Figure 2.10: ATM B-ISDN service classes.

Services that are to be transported over ATM networks are grouped into 4 classes (Figure 2.10), each of which has its own specific requirements on the AAL. The classification of these services is performed according to three different parameters:

1) The time relation between the transmitter and the receiver in a connection is taken into account. Some services such as voice require a strict time relation

between the source and destination. Services that require a time relation are sometimes referred to as *real time* services.

2) The bit rate that is characteristic of a particular service is considered. Some services have a constant bit rate while others have a variable bit rate.

3) Also taken into consideration is the connection mode that is used. Services are either connection oriented or connectionless.

These 3 parameters result in 8 possible combinations for traffic types although only 4 of these result in valid existing services. These 4 service classes are named A, B, C, and D (Figure 2.10) [DEP93].

Class A: Constant Bit Rate (CBR) service. This service class defines a connection-oriented service whose bit rate is constant and where a timing relation is required between the sender and receiver. AAL1 is responsible for this type of service class. An example of a Class A service is 64 kbit/s voice that current telephone networks carry.

Class B: Variable Bit Rate (VBR) service. This service class defines a connection-oriented service whose bit rate is variable but requires a bounded delay for proper delivery. This service class utilizes the AAL2 protocol. An example of a Class B service is compressed packetized voice or video.

Class C: Connection-oriented data service. This is a connection-oriented service whose bit rate is variable and whose delay for delivery is not bounded. This class includes data communications where a connection is set up prior to transmission. The ITU-T had originally recommended two AAL protocols for this class. These two protocols, however, have been merged into a single type called AAL3/4. The complexity of this protocol has prompted the proposal of a simpler protocol, the AAL5, to support this service class.

Class D: Connectionless data service. This class defines those services whose bit-rate is variable, where no time relation is required between sender and receiver, and whose operation does not require a connection to be set up. An example of a Class D service is datagram traffic. Either AAL3/4 or AAL5 may be used to support Class D services.

Although each traffic class has a specific AAL protocol that is optimized for it, there is no requirement that an AAL designed for one class may not be used for another class. In

fact, many ATM equipment vendors have manufactured products that use AAL5 to support all 4 classes of traffic. In addition, some standards organizations such as the ATM Forum have been focusing their activities on AAL5 [SIU95].

2.3.3 ATM Traffic Management

ATM traffic management is essential for guaranteeing the desired *Quality of Service* (QoS) demanded by integrated broadband services, while maximizing the utilization of available resources. Problems arise during periods of heavy traffic load especially when these traffic demands cannot be predicted in advance. For this reason, the most important aspect of traffic management is congestion control [JAI95], [SIU95]. This section will offer only a brief introduction to ATM traffic management and its main issues. Further information may be obtained from the references [JAI95], [SIU95], [BON95], [KUN95], [JAI90], [NEW94].

During call setup, user applications are able to specify the following parameters related to the desired QoS [JAI95].

- *Peak Cell Rate* (PCR): The maximum instantaneous transmission rate of the user.
- *Sustained Cell Rate* (SCR): The average transmission rate measured over an extended time interval.
- *Cell Loss Ratio* (CLR): $\text{CLR} = \frac{\#\ \text{Lost Cells}}{\#\ \text{Transmitted Cells}}$.
- *Cell Transfer Delay* (CTD): The time it takes for a cell to travel from its source to its destination.
- *Cell Delay Variation* (CDV): The variance of the CTD.
- *Burst Tolerance* (BT): This is the maximum burst size that can be transmitted at the peak rate.
- *Minimum Cell Rate* (MCR): The minimum instantaneous transmission rate of the user.

Using these parameters, a traffic contract is made between the user and the network during call setup in order provide a guaranteed QoS. This contract includes a connection traffic descriptor and a conformance definition. However, not all connections need to have a specified QoS. If the opposite were true then ATM network resources may be wasted since connections may not be utilizing the full capacity that their QoS contracts allocated. ATM networks can support unspecified QoS contracts on a *best-effort* basis.

An ATM traffic contract specifies four classes of traffic. *Constant Bit Rate* (CBR) traffic, which is used to emulate circuit switching, was discussed previously along with *Variable Bit Rate* (VBR) traffic. A third traffic class, *Unspecified Bit Rate* (UBR), is designed for data applications that wish to use any left-over bandwidth and are not sensitive to cell loss or cell delay. Data file transfers that run in the background of a work station with minimal service requirements are an example of a UBR application [BON95]. The fourth traffic class, *Available Bit Rate* (ABR), serves applications with vague requirements for bandwidth and delay. In these cases, the ATM network will attempt to allocate a *best-effort* service so as not to waste network resources. For this reason, ABR is sometimes referred to as a best-effort traffic class. Typically, the user will define a *Minimum Cell Rate* (MCR) which is guaranteed by the network if the traffic contract is made. Most connections will request an MCR of zero and those that do not may be denied service if enough network resources are not available.

As was previously mentioned, congestion control remains one of the most vital aspects of ATM traffic management. Primary functions of a congestion control scheme include ensuring adequate bandwidth resources and delay performance for a particular connection, maintaining a fair allocation of network resources to each user, and providing policing mechanisms to protect connections that obey their traffic contracts from those that do not [BON95]. The problem of congestion control becomes more difficult for unspecified QoS traffic such as ABR whose traffic characteristics are often very bursty and unpredictable.

A rather simple method to avoid network congestion is to accept new connections, whose QoS requirements must be guaranteed, during call setup only when sufficient network resources are available to provide the requested QoS. This method is known as *Connection Admission Control* (CAC) and is used for connections where the QoS must be guaranteed. An example of CAC is the "busy" signal that one hears on a telephone when telephone networks are congested.

The *Generic Cell Rate Algorithm* (GCRA) is another method used for congestion control. The GCRA, also known as the "leaky bucket" algorithm, transforms a highly irregular bursty data stream into a more regular pattern. Essentially, this algorithm places incoming cells into a bucket which is drained at the *Sustained Cell Rate* (SCR) defined in the traffic contract. It is possible for the bucket to overflow if too many cells arrive at once. The network may not accept these overflowing (or non-conforming) cells. Non-conforming ATM cells will have their header *Cell Loss Priority* (CLP) bit set so that these cells will be the first to be discarded in case of network overload.

The leaky bucket algorithm performs *traffic shaping* meaning that traffic originating from the sender is "shaped" so as to fit within the traffic bounds (such as sustained and peak cell rates) that were negotiated prior to connection setup. Since these traffic parameters cannot be dynamically adjusted should network congestion occur, these traffic shaping algorithms are termed open loop. However, in closed loop or feedback congestion control methods, end users are informed dynamically whether to increase or decrease their transmission rate according to the amount of network congestion.

Two major approaches to feedback congestion control have appeared in recent years - credit-based congestion control [KUN95] and rate-based congestion control [BON95]. Without going into detail (the reader is referred to the references for a more in depth account of these two feedback congestion control mechanisms), it can be said that the rate-based scheme was chosen overwhelmingly by the ATM Forum over the credit-based scheme largely because of the per-VC (virtual channel) queueing required by the latter. Per-VC queueing makes any switching node have a complexity that is proportional to the number of VCs. Also making the credit-based option less attractive was the need for link-by-link flow control. The rate-based schemes, on the other hand, used the *Explicit Forward Congestion Indication* (EFCI) located in the PTI field of the ATM cell header to avoid network congestion. This field could be set by switches during periods of congestion. As a result, the rate-based methods required only end-to-end flow control and were thus simpler and more flexible to implement. An approach that combined the best features of the rate-based and credit-based alternatives was proposed [RAM95] but it was rejected largely because of the need for per-VC queueing. Thus, rate-based congestion control was chosen by the ATM Forum in September 1994 to support ABR services [SIU95], [JAI95].

2.4 Broadband ATM Switching

The central component of an ATM network is the ATM switch (Figure 2.11). An *NxN* ATM switch provides a means of connecting *N* inputs to *N* outputs without resorting to the use of a fully connected *N*-node network. As Simcoe and Pei eloquently stated, "Switches allow the sharing of links in a way that provides the illusion that each node is connected to every other node in the network" [SIM95]. An ATM switch consists of a switch fabric which routes ATM cells from input ports to the correct output ports; a management control processor which communicates with the switch fabric and the port control processors in order to facilitate switch operation, administration, and management; and input and output port control processors which perform various functions such as ATM

cell queueing, cell duplication for broadcast purposes, VPI/VCI header translation, and multiplexing and demultiplexing traffic.

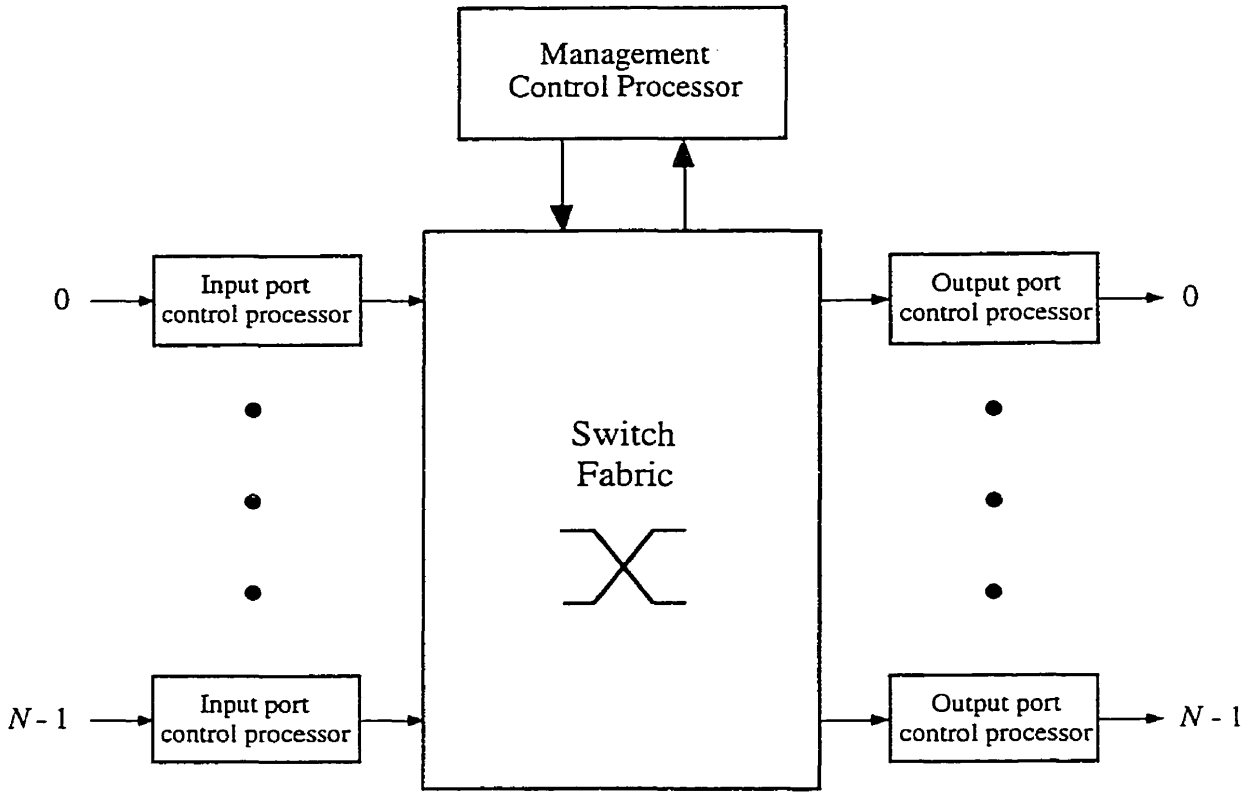


Figure 2.11: Model of an ATM switch.

For a *strictly nonblocking* switch, it is always possible to connect an idle input port to an idle output port without disturbing any of the input/output connections that have already been made [BEN62]. Switches that are *nonblocking in the wide sense* will have blocking states that may be avoided by following a rule to make the input/output connections and thus making the network effectively nonblocking. Finally, there exist switches where a connection between an idle input and an idle output may be made by rearranging existing connections (if necessary). Such switches are *rearrangeably nonblocking*. Switching networks that use multistage arrays of smaller switching elements are usually of the latter two classes of nonblocking switches. *Multistage switching networks* will be discussed shortly.

Even if a switch is strictly nonblocking, output port contention may occur when two or more inputs try to go to the same output port simultaneously. If the contention is severe and persists for an extended period of time the usual way to deal with this sort of traffic congestion is to use feedback traffic control in conjunction with cell queueing. If the contention is less major simple cell queueing may be used. This section will focus on cell

queueing and the switch fabric. For more information on ATM switching architectures, there are several articles that provide a detailed introduction [AHM89], [GAR94], [PAT93], [ROO94], [SIM95], [TOB90], [ZEG93].

2.4.1 ATM Switching Fabrics

There exist three broad classes of ATM switching fabrics. These are the shared medium switches which are similar to existing LANs, the multistage switching fabrics or self-routing switch arrays, and the crossbar [SIM95].

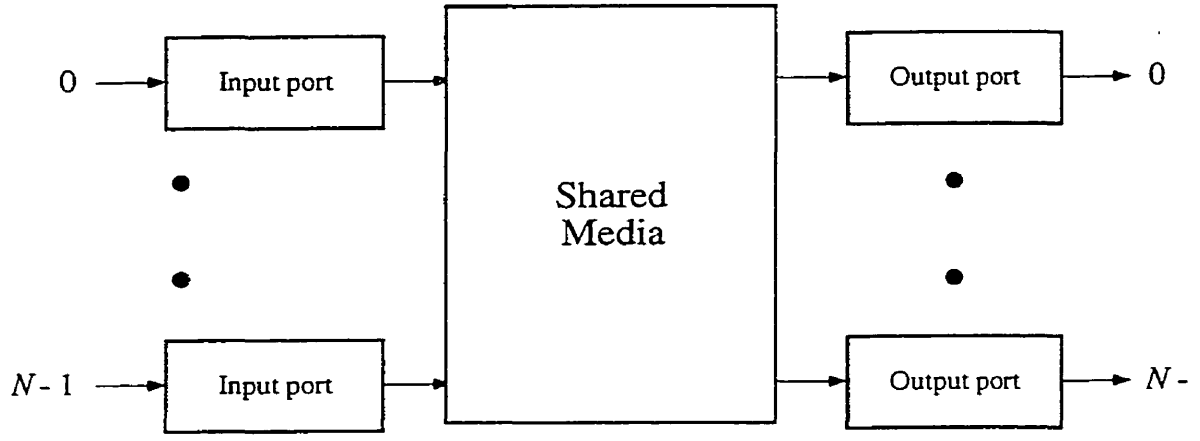


Figure 2.12: Shared media switch.

Shared media switches work by sharing a high speed medium such as a bus, a ring, or a bus with an attached memory system. Examples of this class of switching architectures are *Token Ring*, *Ethernet*, or *Fiber Distributed Digital Interface* (FDDI). Shared media switches have a bandwidth that is limited by the distances traversed by the bus; the type, the number, and the distribution of the loads; and the bus width. Multicasting and broadcasting of data are relatively easy to implement when using shared media-type switches since all output ports have access to the data that has been stored on the shared media. Also, shared media switches are viewed as being the most cost effective when implemented with off-the-shelf technology at lower data rates. However, due to the limitations of bus loading, shared media switches are not scaleable and have a relatively low bandwidth [SIM95].

An example of a crossbar topology is shown in Figure 2.13. Here, the bold lines represent a connection made between input 1 and output 2. For a more general NxN crossbar, there are N^2 paths between N input ports and N output ports. The crossbar contains a single contact pair for each input-output combination. Broadcasting is permitted

in a crossbar switch since an input row may be connected to more than one output column. Not surprisingly, the crossbar is one of the most simplest switching architectures. However, scalability becomes a problem since the cost of a crossbar switch is on the order of N^2. In this case, the cost is described in terms of the number of contact pairs required. However, it is now known that the limiting effect of crossbar size when using VLSI CMOS technologies is not the number of contact pairs but the size of the chip bonding pads and the number of pins on currently available chip packages [SIM95]. In other words, using current VLSI technologies, it is possible to fit a larger array of contact pairs onto a chip but this would prove futile since there would not be enough bonding pads to accommodate all the inputs and outputs required by the number of contact pairs. This is an example of the limitations incurred by current chip packaging technologies that may be avoided when using optics. (This point will be discussed in the following chapter.)

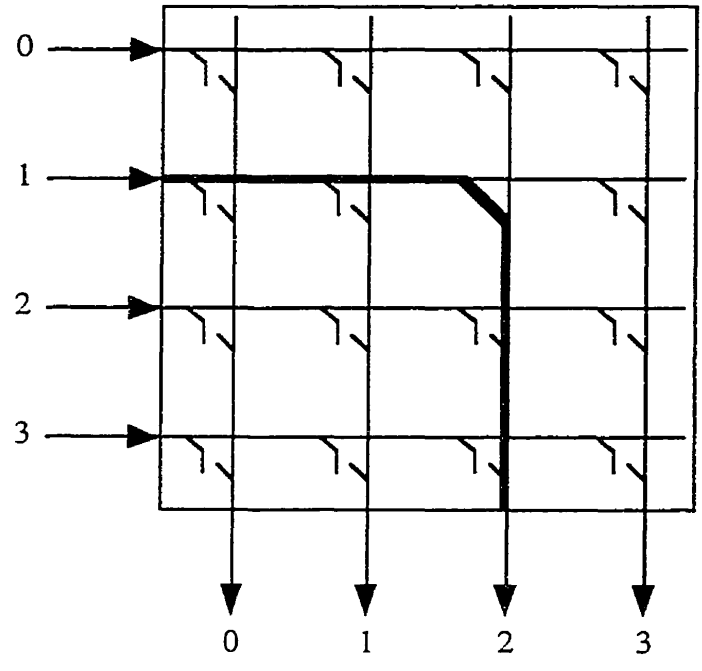


Figure 2.13: An *NxM* crossbar switch ($N = 4$ and $M = 4$).

Multistage switches comprise the third class of switching architectures. Multistage switches are composed of arrays of simpler switching elements. Delta networks are examples of multistage switches (Figure 2.14). Figure 2.14 (a) shows a 3-stage 8x8 delta network composed of 2x2 switching elements (a delta-2 network). Self-routing of ATM cells is accomplished by a technique known as the Perfect-Shuffle [ROO94] whereby the bits of the destination address provide the required routing tags. For example, in Figure 2.14 (a) the most significant bit is used to set the state of the switching element in stage 1, the second most significant bit sets the state of the stage 2 switching element, and the least

significant bit is used for the third stage switching element. If the bit is 1 the input port of a switching element is connected to output port 1. If it is 0 then output port 0 is connected to the input port.

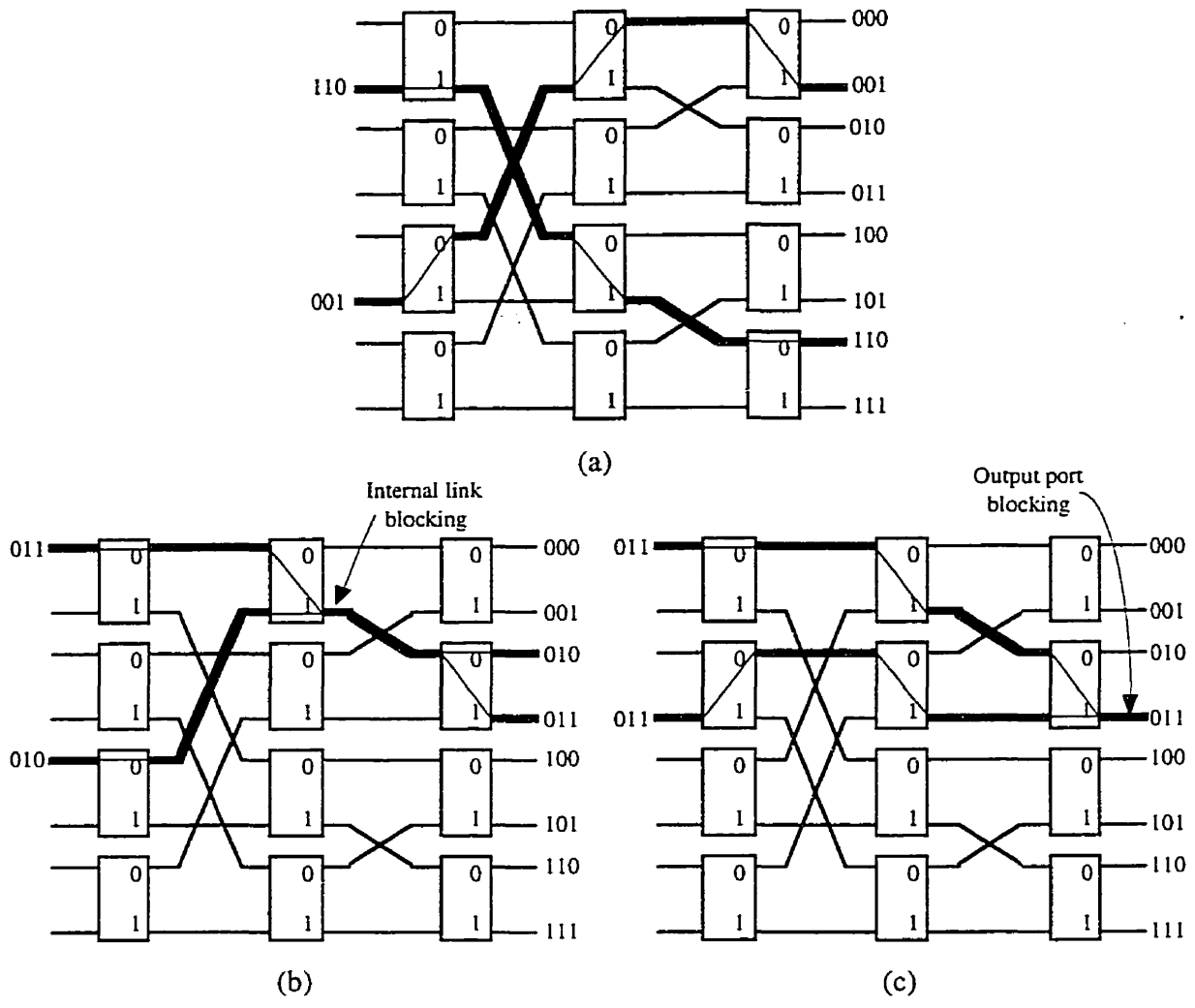


Figure 2.14: (a) An 8x8 delta-2 network with (b) internal link blocking and (c) output port blocking [AHM89].

In general, in an N-input port switch composed of bxb switching elements, there are K stages where $K = \log_b N$ and N/b switching elements in each stage. Therefore, the switch complexity is on the order of $N \log_b N$ making these architectures theoretically more scaleable. For a central office class sized switch, N is preferably on the order of 10^3 or 10^4. However, according to [BAN91] and [ITO91], physical limits like chip pins, bonding pads, printed circuit board (PCB) size, and card connector systems have prevented switches with $N > 128$ from being fabricated. Again, these limitations are ones that optical systems hope to avoid.

Another drawback of multistage switches is in the area of traffic congestion. Multistage switches have somewhat more difficulty dealing with traffic congestion than other switch classes because of the extensive pipelining and queueing that occurs within the switch [SIM95]. Once contention has been detected, an ATM cell may be too far inside the multistage switch requiring the cell to be either stored or redirected back to the inputs.

Figures 2.15 (b) and (c) show two different types of blocking that can occur in a multistage switching network. The first type of blocking (or contention) occurs within an internal link of the network. The second occurs when two or more inputs wish to access the same output port. There are several methods that reduce internal or output blocking and they are briefly summarized below [ROO94].

- Place a distribution network in front of the network to reduce or eliminate the probability of blocking. This is much like rearranging input-output connections to avoid blocking in a rearrangeably nonblocking network.
- Include blocked ATM cells in the next cycle by recirculating them back to the input ports.
- At the input ports, add a contention resolution phase.
- Add a handshaking protocol between stages of the network to control the flow of traffic across the stages.
- Add additional stages and paths to the network making the network strictly nonblocking in the best case or reducing the blocking probability in the worst.
- Insert queues either at the input ports, output ports, or within each switching element (to be discussed in the next subsection).
- Speedup the operation of the switching fabric relative to the speed of the external input and output links.
- Decrease the blocking probability by using several networks in parallel.

2.4.2 ATM Cell Queueing

For less severe instances of traffic congestion, cell queueing may be used to reduce the probability of output or link contention. Queueing strategies may be used for the input ports, the output ports, within the switch fabric, or a combination of the three. However, conventional analysis of queueing networks has used an independence assumption whereby ATM cell traffic is modeled as being uniformly distributed across all N inputs of a switch fabric with cell destinations also being uniformly distributed among the N output ports. These assumptions may not be valid for real ATM traffic such as the Available Bit Rate (ABR) traffic class discussed earlier [SIM95].

When queues are positioned at the input ports of an NxN switch ATM cells may be subject to *Head of Line* (HOL) blocking. HOL blocking occurs in *First-In-First-Out* (FIFO) Queues when cells in a FIFO queue whose destined output ports are not busy are not able to transmit to their respective output ports because these cells follow an HOL cell which is blocked because its destined output port is busy. The throughput of an input queued switch has been shown to be limited to 58.6% of its capacity [KAR87]. Ideally, one would like to have the switch utilization at 100% especially in *Wide Area Networks* (WANs) where the bandwidth required is expected to be large.

Placing queues at the output ports avoids HOL blocking at the expense of increasing the hardware complexity. Each output queue must be able to accept the simultaneous arrival of a maximum of N cells (in an NxN switch). This requires that the switch fabric have a speedup of up to N times the speed that the external input and output links are running [SIM95].

In order to limit the speedup factor needed for the switch fabric, it was originally proposed that some cells be dropped in the rare cases that large numbers of cells would access the same output port. This is the basis of the famous *Knockout* switch [YEH87]. Later, switching fabrics using a combination of input and output queueing were proposed. It has been shown that when using queues at the input and output ports and a marginal speedup of $L = 3$ for the switching fabric, the output link utilization can be increased to 97.6% in the limit corresponding to a cell loss probability on the order of 10^{-6} [OIE89]. Therefore, it does not seem necessary to use the maximum speedup of N for an NxN switch.

Other more complex queueing strategies include the use of shared queues and the placement of input, output and/or shared queues within individual switching elements in a multistage switching network. Input and output queueing allocates individual buffers to each input or output port. In shared queueing, every input and/or output port has access to

the same queue. This alternative is the most complex in terms of hardware and will not be discussed further.

2.5 Summary

This chapter provided a concise overview of ATM. Topics that were discussed included the motivation behind the development of ATM. This chapter also provided a description of the ATM cell, the ATM network layer model, ATM traffic management, and the SONET transmission standard. The chapter concluded with a discussion of ATM switching fabrics and cell queueing.

CHAPTER 3
Free-Space Optical Interconnects

3.1 Introduction - The Need for Optical Interconnects

ATM networks and high performance computing applications are driving the need for faster digital systems. However, as physical limits such as clock skew, interconnection density, and power dissipation of an electrical based system are approached, it becomes clear that new alternatives to interconnection are necessary. One solution that could meet at least some of these challenges is to incorporate optics into digital systems. It is not the purpose of optics to supplant electronics completely. In fact, the most logical avenue for optics to follow is one where optics complements electronics in a way where optics only replaces its electronic counterparts when a sufficient advantage can be gained. One possible application of this approach is in the area of board-to-board interconnects [HAR86], [HIN94], [HIN95], [NOR92], [NOR95], [SZY95].

In high speed digital systems such as an ATM switching fabric, distances between adjacent integrated circuits (ICs) or adjacent printed circuits boards (PCBs) become comparable to the wavelength of the propagating signal [NOR95]. In these situations, transmission lines are required as a propagating medium for signals. An important design rule is to minimize the interconnecting lengths so that these lengths are small relative to the wavelength of the propagating signal (i.e., $l < \lambda/10$ where l is the length of the interconnection and λ is the wavelength). Another guide based on the same principle would be to keep the maximum bit rate (BR) below a certain value as in equation (3.1)

$$BR \leq \frac{c}{50l\sqrt{\varepsilon_r}} \tag{3.1}$$

where ε_r is the relative dielectric constant of the packaging medium[1] , c is the speed of light, and where the rise time of the propagating signal is assumed to be one-fifth of the signal's period [NOR95]. As long as these rules are followed, one may avoid resorting to complex transmission line techniques in order to maintain signal integrity [NOR95]. If transmission lines are required it should be noted that transmission lines are best suited for relatively short distances. As distances are increased, the propagating signals become attenuated by the "skin effect" and parasitic capacitances and inductances further reduce the available bandwidth [JOH88].

Telecommunication switches from AT&T, NEC, and Fujitsu with bandwidth capacities exceeding 100 Gbit/s [HUI96] have already been announced. A free-space optical interconnect/backplane solution may be required in order to obtain higher bandwidth digital systems - such as the aforementioned switches - approaching 1 Tbit/s capacities. This approach may be necessary as parasitic effects mentioned above become substantial in these high-bandwidth systems. In particular, as the technologies required in a free-space optical backplane mature, the performance of an optical backplane may be increased to meet or exceed the bandwidth required in a Tbit/s telecommunication switch or a Teraflop supercomputer.

Figure 3.1 shows an example of what a free-space optical backplane might look like. Board-to-board communications are accomplished through the use of the *Smart Pixel Arrays* (SPAs) and the *Optical Communication Channels* (OCCs). SPAs are optoelectronic components composed of optical input and output windows and associated electronic circuitry. A PCB wishing to transmit information to another PCB across the backplane would use a SPA to convert the electrical signal into an optical form for transmission over an OCC. Similarly, a receiving PCB will accept the transmitted optical signal from an OCC via a SPA which converts the signal back into electrical form [HIN95], [SZY95]. Obtainable interconnection densities may exceed 10,000 high bandwidth connections per PCB for an aggregate bandwidth exceeding 1 Tbit/s. Furthermore, the associated electronics in each SPA enables signal processing to occur giving the SPA and the OCCs some intelligence.

[1] The dielectric constant of free space, ε_0, is equal to 8.854×10^{-12} C^2/Nm^2. For PCBs, the typical value for ε_r is 2.8 to 4.5 times ε_0. [JOH93].

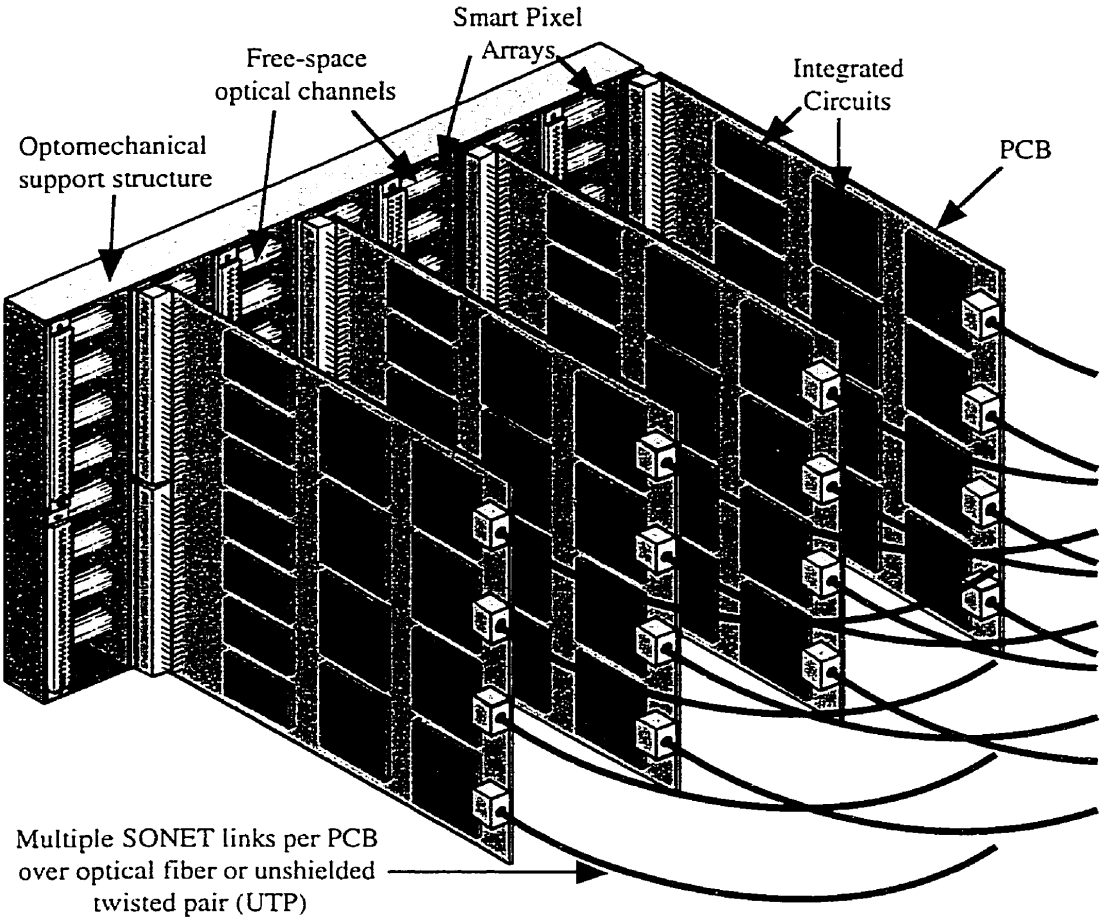


Figure 3.1: Free-space optical backplane.

Several groups are currently investigating high speed free-space optical interconnects. The former AT&T (now Lucent Technologies) in Naperville, Illinois has produced a number of demonstrators ranging from a simple switching node to a 32 x 16 switching fabric with a system bit rate of 50 Mb/s [HIN94]. This thesis is part of a multi-year project at McGill University to demonstrate representative portions of an optical backplane for use in high performance systems such as an ATM switching fabric [CIT95]. A system demonstrator has been built showing both high channel data rates (100 Mb/s) and large interconnection densities (2222 channels/cm^2) [ROL96]. One of the purposes of this thesis was to build an electronic backplane capable of working with the CITR 1998 optical backplane demonstrator with little or no adjustments. In particular, the goal is to build and demonstrate a PCB which receives ATM cells from a SONET payload,

performs VCI/VPI header translation, transmits these ATM cells to the OCCs via the SPAs for switching, and receives ATM cells from the OCCs to be sent over fiber or unshielded twisted pair (UTP) cable within a SONET payload.

3.2 HyperPlane Smart Pixels

A smart pixel is composed of optical inputs and outputs along with electronic circuitry for signal processing to give it its intelligence [HIN95]. A generic *Smart Pixel Array* (SPA) is shown in Figure 3.2. In this example, data is injected to and extracted from the *Optical Communication Channels* (OCCs) by the electrical inputs and outputs of each smart pixel. Each SPA which is packaged and mounted on a PCB is able to communicate optically with other SPAs that are mounted on other PCBs. Standard VLSI technologies enable the packaging of a few thousand smart pixels within a square centimeter [MIL95], leading to very high interconnection densities. In addition, the SPA is able to communicate with ICs on the PCB through electrical channels. Data injection or extraction between a PCB and the OCCs is possible in this manner.

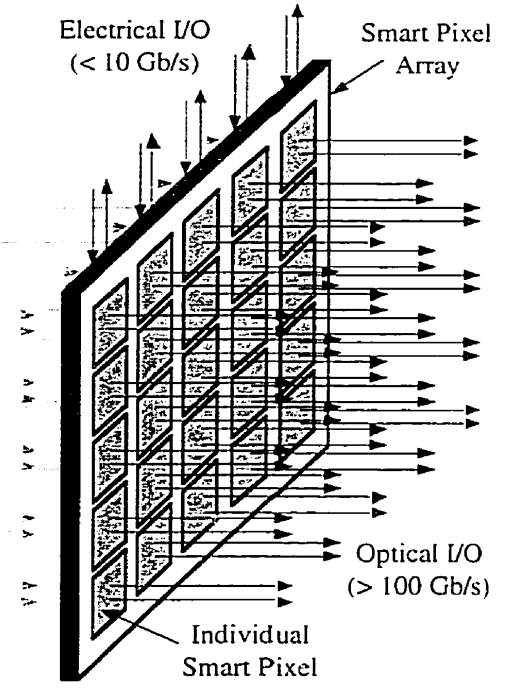


Figure 3.2: Smart pixel arrays [HIN95].

One typical SPA is illustrated in Figure 3.3. Here, one can see the optical inputs and outputs along with their associated electrical circuitry. PCB-to-PCB communication is accomplished through the use of the *Optical Communication Channels* (OCCs) and the extractor and injector channels of a SPA. Each smart pixel has four basic states:

1) *Transparent state*: Data is not exchanged between the PCB and the smart pixel. Optical data is allowed to pass uninterrupted from the optical input to the optical output.

2) *Transmitting state*: Data is injected into the smart pixel from the PCB on an injector channel and is transmitted to the optical output.

3) *Receiving state*: Data received from the optical input is sent to the PCB on an extractor channel.

4) *Transmitting and Receiving state*: In this case, actions for the transmitting state and the receiving state occur simultaneously.

Due to current chip fabrication limitations, smart pixel electronics have a modest complexity of approximately 100 gates. As the technology matures, complex circuit designs of several hundred gates or more may be incorporated into the electrical portion of each smart pixel.

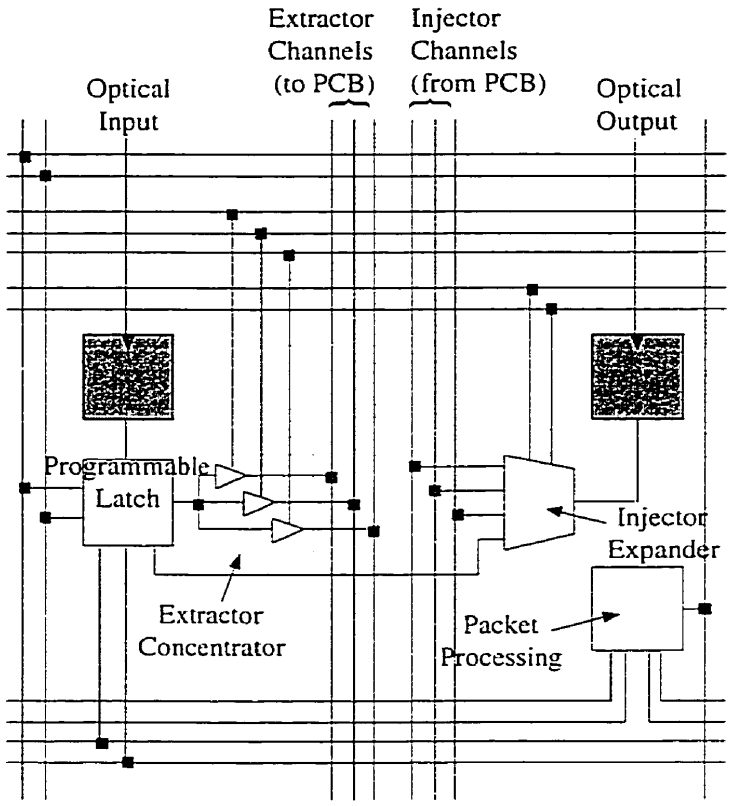


Figure 3.3: Typical *HyperPlane* smart pixel [SZY96].

Several rapidly evolving technologies are available for fabricating SPAs. Each of these technologies can produce SPAs using fairly standard VLSI processes. Packaging is accomplished in the same manner as in current VLSI IC chips. Among these technologies are FET-SEED smart pixels. These monolithic optoelectronic devices are produced by combining MESFETs and self-electro-optic effect devices (SEEDs). These MESFETs are processed on a GaAs substrate which contains the SEED multiple quantum well structure [HIN93a], [GOO89], [MIL90], [MIL84]. Multiple quantum wells are composed of thin alternating layers of narrow and wide bandgap semiconductor materials. Since carriers are confined within the quantum wells, an absorption spectrum will show distinct peaks known as exciton peaks. These peaks can be moved by applying an electric field perpendicular to the plane of the quantum wells. This phenomena is known as the quantum-confined Stark effect (QCSE). Thus, SEEDs can be used as optical modulators in smart pixel transmitters by having the SEED modulate the optical power from an off-chip laser. They may also be used as optical detectors in smart pixel receivers. One advantage of FET-SEED smart pixels is that they operate extremely quickly. However, it

is currently quite difficult to obtain dense, low power designs with the FET-SEED smart pixel electronic circuitry making this approach unattractive.

At McGill, research has been focused on using Hybrid-SEED (or CMOS-SEED) and VCSEL/MSM smart pixels in its optoelectronic systems [CIT95]. Hybrid-SEED technology involves creating arrays of quantum well diodes from a SEED substrate. These arrays are then turned and solder-bump bonded onto a silicon CMOS circuit. Another name for this bonding process is flip-chip bonding. Once the bonding is complete, the SEED substrate is removed leaving isolated quantum wells bonded to a CMOS circuit [MIL95], [RED87]. Smart pixels fabricated using the hybrid-SEED technology are fast and are able to support relatively dense electronics.

A VCSEL/MSM smart pixel utilizes vertical cavity surface emitting lasers (VCSELs) and metal-semiconductor-metal (MSM) detectors along with electronics [HIN95], [IGA88]. Since VCSELs (which serve as optical transmitters) produce light, there is no need for external lasers and associated optical components as in the modulator-based technologies. As this technology improves, VCSELs drive power will be reduced making this a very attractive option.

3.3 The McGill *HyperPlane* Architecture

Novel optoelectronic technologies alone cannot produce an effective high-performance optical backplane system. Architectural innovations must also be realized in order to maximize the performance. A novel high performance backplane architecture called the *HyperPlane* has been defined. The *HyperPlane* is a free-space optical backplane architecture capable of interconnecting communicating nodes through a number of reconfigurable network topologies. Optical embeddings for linear arrays, meshes, toroids, hypercubes, crossbars, dilated crossbars, hypermeshes, shuffle-based networks, and multi-stage networks have been mapped into the *HyperPlane* architecture, demonstrating its versatility as a digital system for use in both telecommunication and parallel computing applications [SZY96].

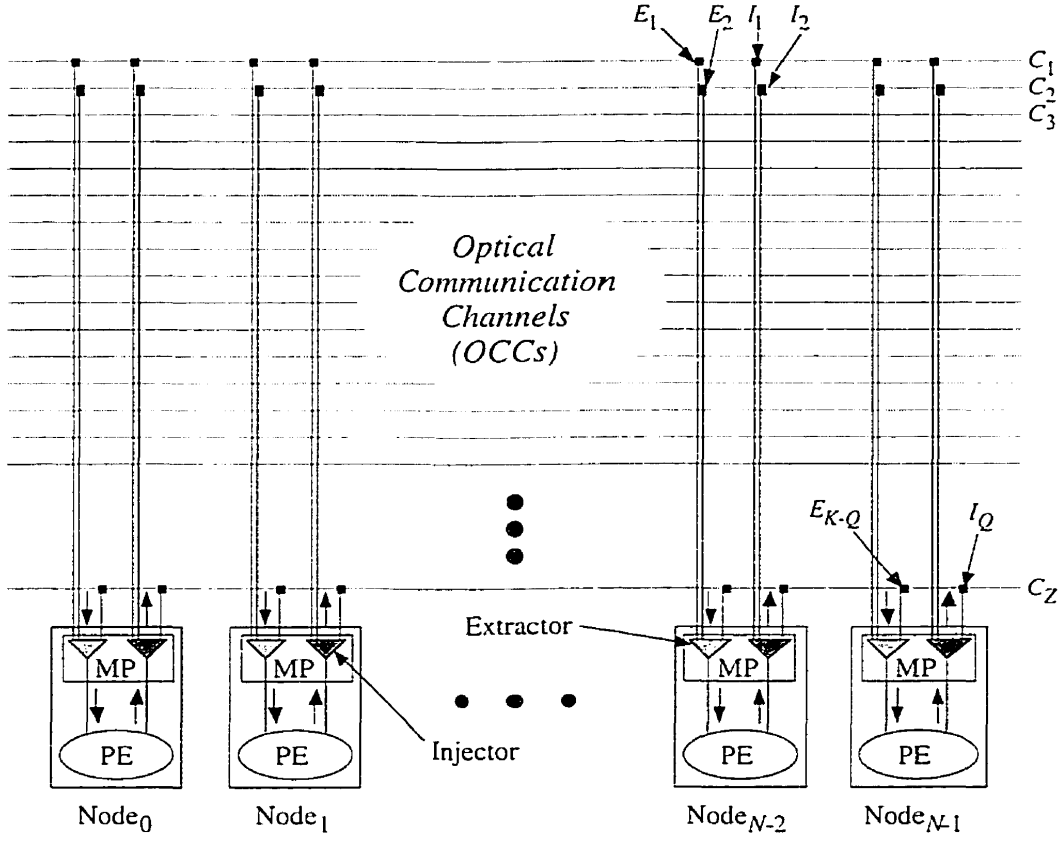


Figure 3.4: *HyperPlane* node connectivity [SZY94].

The architecture of the *HyperPlane* is composed of M printed circuit boards (PCBs) or multi-chip modules (MCMs) which are interconnected by the *Optical Communication Channels* (OCCs) of the free-space optical backplane. (This thesis involves the design and implementation of the printed circuit boards for ATM switching.) Each PCB contains multiple nodes where each node includes a *Processing Element* (PE) and a *Message Processor* (MP). General purpose microprocessors and ATM switching nodes are examples of PEs. The message processors enable point-to-point or broadcast connections between nodes through the control of node access to Z OCCs $\{C_1, C_2, \ldots, C_Z\}$.

Through the use of injectors and extractors, each message processor has access to a selected subset of the OCC channels. More specifically, there are X injector and Y extractor access channels for each message processor labeled $\{I_1, I_2, \ldots, I_X; X \le Z\}$ and $\{E_1, E_2, \ldots, E_Y; Y \le Z\}$, respectively. By allowing access to a selected subset of OCCs through the injectors and a subset through the extractors, a multi-dimensional interconnection design space known as the *HyperPlane* is formed [SZY96]. The *HyperPlane* allows the embedding of all N node interconnection networks of degree K. In addition, by allowing the message processors to dynamically alter node access configurations to the OCCs, the notion of a *dynamically programmable interconnection network* is introduced. As a result, the *HyperPlane* is able to implement crossbars, meshes, hypercubes, and other networks as required for each application.

Figure 3.5 illustrates how an actual *HyperPlane* optical backplane system may look. The *Processing Elements* (PEs) and the *Message Processors* (MPs) are found on each PCB. The PCBs shown in Figure 3.5 are called *ATMCards*. The purpose of this thesis is the design, implementation, and description of the *ATMCard*. Each *ATMCard* discussed in this thesis contains one SONET I/O as opposed to the multiple SONET I/Os shown in Figure 3.5. Future versions of the *ATMCard* or other second generation PCBs will support multiple SONET I/Os. The *ATMCard* will be described in detail in subsequent chapters.

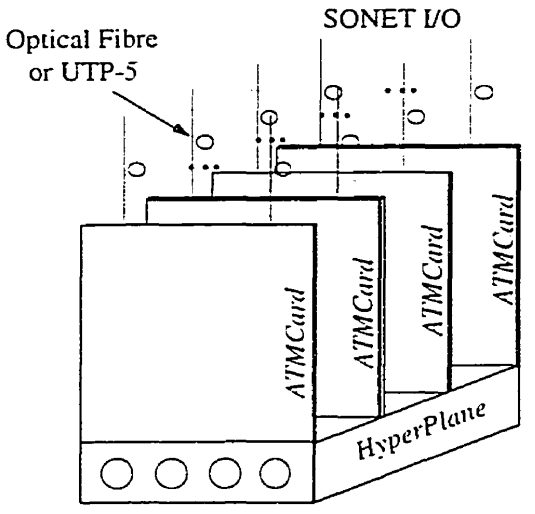


Figure 3.5: *ATMCard* and the *HyperPlane* system.

McGill/CITR is in the midst of an ongoing project to develop free-space optical backplanes. Several generations of *Smart Pixel Arrays* (SPAs) have been developed for this project. Figure 3.3 illustrated one version of a SPA. The June 1994 SPA uses *Time-Division Multiplexing* (TDM) and assumes an ATM switching function. All subsequent references to the *HyperPlane* assume that the June 1994 SPA is used. It should be noted that the 1998 optical backplane demonstrator funded by the CITR will use essentially the same SPA. Hence, the *ATMCard* may be used in the 1998 demonstrator with only minor modifications.

3.4 Summary

In this chapter, the motivation behind the use of free-space optical interconnects was explained. A description of smart pixels - one of the main components of a free-space optical interconnect system - followed. As an example, a typical smart pixel developed at McGill University was outlined. Finally, the chapter concluded with a description of a free-space optical backplane - known as the *HyperPlane* - being developed at McGill University.

CHAPTER 4
System Specification

4.1 Introduction

One of the goals of this thesis was to design the *ATMCard* for the 1998 CITR demonstrator with minimal adjustments. In order to facilitate the incorporation of the *ATMCard* within the 1998 demonstrator, a specification of the *ATMCard* and its environment is required. Chapter 4 provides a description of the environment the *ATMCard* will function in - a smart-pixel based optical backplane switching system. It will also describe *ATMCard* functions within this environment, such as insertion and extraction of ATM cells to and from SONET frames and ATM switching.

4.2 Smart Pixel Array Specifications

The block diagram of the June 1994 *HyperPlane Smart Pixel Array* (SPA) is shown in Figure 4.1 [SZY94a]. Each *HyperPlane* SPA has four 8-bit wide optical channels labeled C_1, C_2, C_3, and C_4. The *ATMCard* has access to each of these 4 channels (Figure 4.1). Data is transmitted into the *HyperPlane* on a frame-by-frame basis. Therefore, the *ATMCard* must wait for the start of a frame before it can start sending data, which in this case is an ATM cell. Since the 1998 CITR optical *HyperPlane* is currently under development, it has been emulated on a Xilinx FPGA for the purposes of this thesis.

Typical parameters for an optical backplane system that are foreseeable in the near future include a system that is able to support a range of 4 to 16 PCBs which can accept external ATM-carrying SONET streams and switch them over the backplane. The number of optical channels envisioned in this future system vary from 4 to 32 in the backplane, with each channel being 8 to 32 bits wide. Each optical channel would be clocked at frequencies from 50 MHz to 200 MHz, enabling aggregate bandwidths on the order of a few hundred Gbit/s. This SPA forms the basis for the 1998 CITR optical backplane demonstrator.

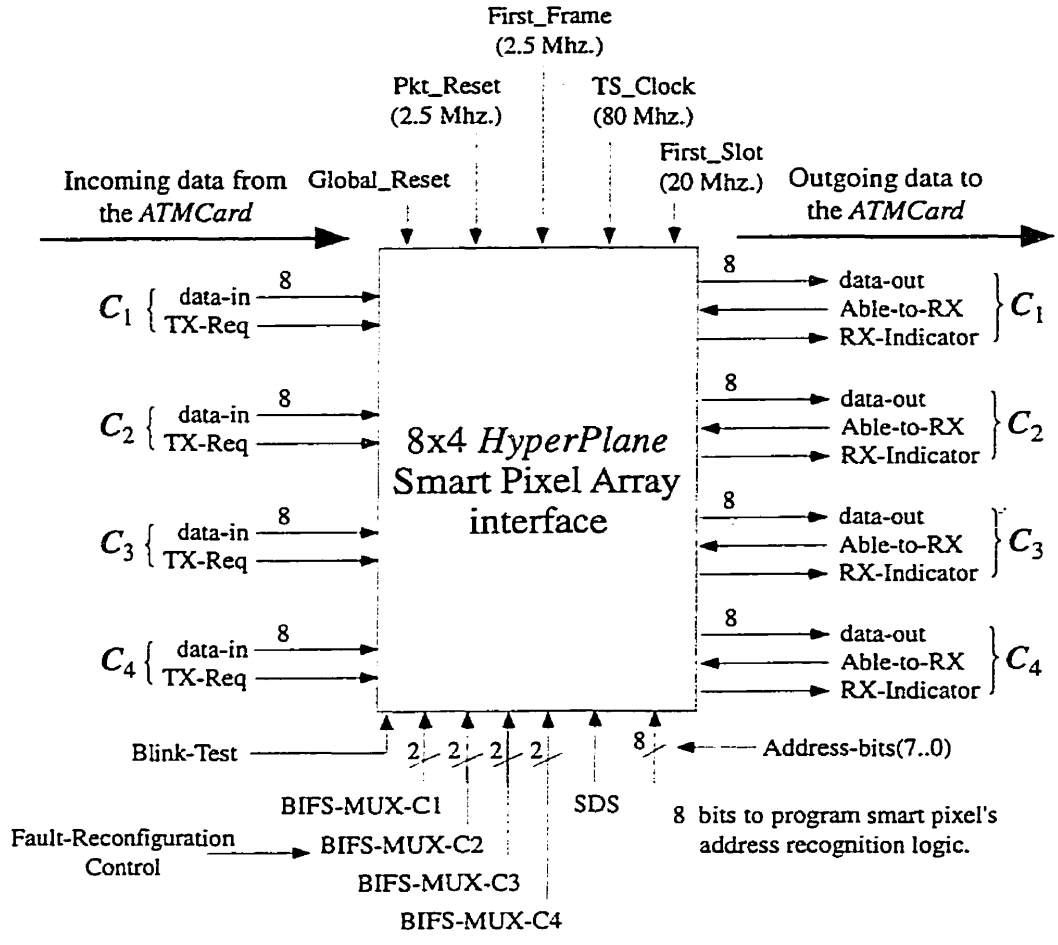


Figure 4.1: Block diagram of an 8x4 *HyperPlane* smart pixel array used by the *ATMCard*.

The 1998 CITR demonstrator will support 4 PCBs with 8 16-bit wide optical channels clocked at frequencies between 50 and 300 MHz. Again, the *ATMCard* is designed to fit within the 1998 demonstrator with minimal adjustments.

The specification of the smart pixel array presented in this chapter is only an overview. A separate document contains a detailed specification of the smart pixel array used with the *ATMCard* [SZY94a]. Included in this document are detailed timing diagrams, block diagrams, and a listing of the SPA functionality. For further information regarding this document please contact Prof. Szymanski or the CITR whose mailing address may be found in the References.

4.3 *ATMCard* Specifications

The *ATMCard* is a transmit/receive card designed for a CITR free-space optical backplane demonstrator. Figure 4.2 illustrates the main input/output ports of the *ATMCard*.

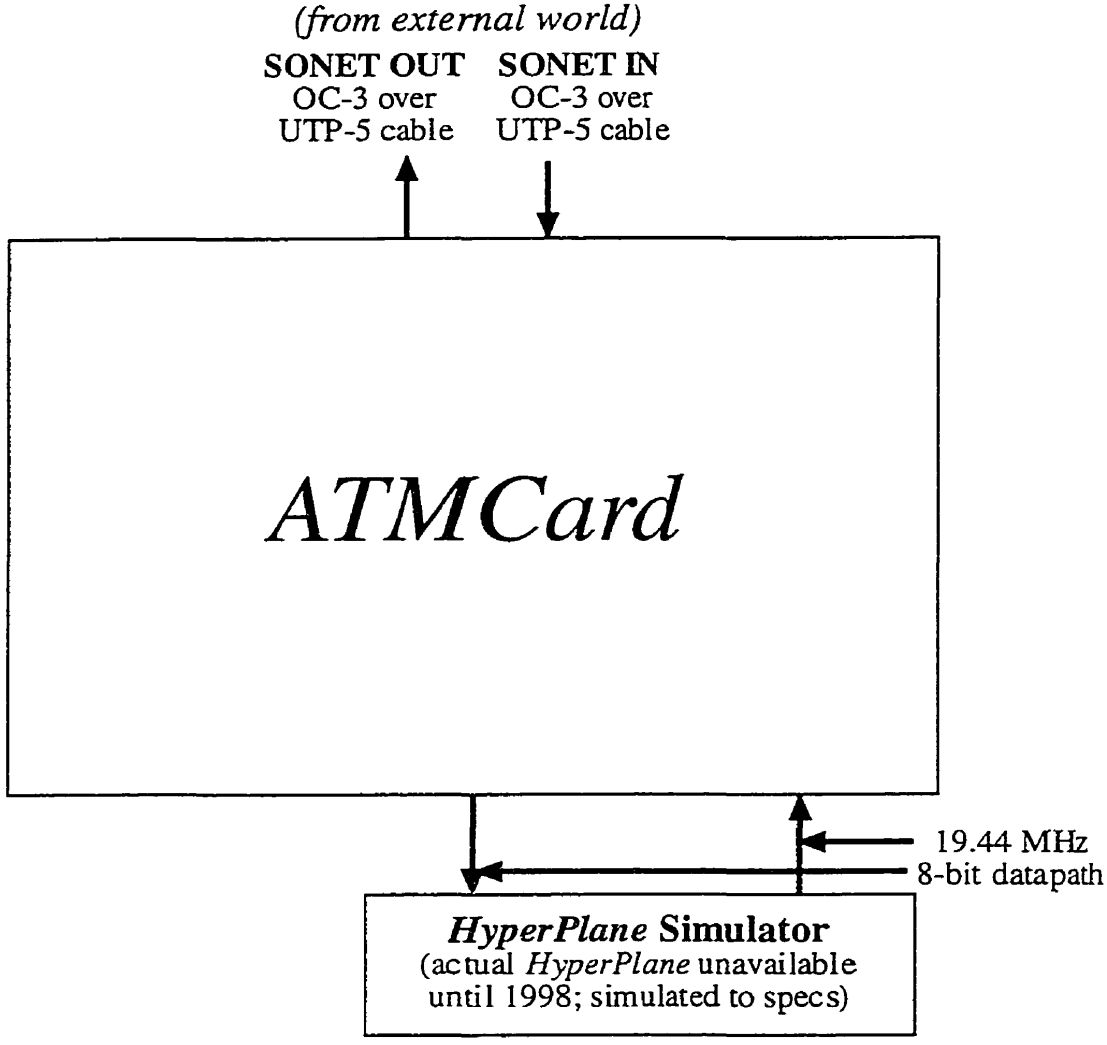


Figure 4.2: High-level block diagram of the *ATMCard*.

Referring to Figure 4.2, each *ATMCard* will connect to two SONET UTP-5 cables. One cable will carry incoming SONET traffic to the *ATMCard* and the other will carry outgoing traffic away from the *ATMCard*. In the future, the UTP-5 cables may be replaced with optical fibre. Adding optical fibre would also entail the addition of optical transceiver components to the *ATMCard*. The data format of the incoming and outgoing serial

SONET links is OC-3 (please see section 2.3.2.1.1 for an explanation of SONET data formats) with a data rate of 155.52 Mbit/s in each direction.

ATM cell buffering is performed within the TNETA1500A transceiver of the *ATMCard*. This transceiver is capable of buffering 3 ATM cells in the incoming and outgoing directions.

The mechanical format of the *ATMCard* is a standard 6U VME board. The 1998 CITR demonstrator also uses 6U VME boards to connect to the backplane making the *ATMCard* mechanically compatible with the 1998 demonstrator. The pin configuration for the edge connectors of the *ATMCard* have been defined to match the pin configuration of the system used to test the *ATMCard*. As such, the as-yet-to-be-defined pin configuration of the 1998 demonstrator may differ from the current configuration found on the *ATMCard*. This issue is not serious since the pin configuration on the *ATMCard* can easily be re-wired to match the configuration of the 1998 demonstrator.

4.4 *ATMCard* Functions

The *ATMCard* exhibits several key functions which are listed here.

1) The *ATMCard* performs the reception of external ATM-carrying SONET frames from UTP-5 cable or optical cable from the external world.
2) The *ATMCard* performs the transmission of ATM cells onto SONET frames traveling over UTP-5 cable or fibre optic cable from the external world.
3) ATM cell buffering is handled on the *ATMCard*.
4) ATM VPI header translation is supported by the *ATMCard*.
5) The *ATMCard* provides a transmit interface to the optical backplane. The *ATMCard* synchronizes the ATM traffic to the optical backplane timing signals and inserts ATM cells into empty time slots of the backplane.
6) The *ATMCard* provides a receive interface to the optical backplane. It synchronizes ATM cells received from the backplane to the *ATMCard* timing signals and transmits the cells over a SONET frame.

4.5 UTP-5 Cables

UTP-5 cables are used to connect end-users in the external world to PCBs (in this case, *ATMCards*) in the optical backplane switching system. Traditional copper-based wires are not capable of providing the bandwidth needed to support 155.52 Mbit/s SONET OC-3 traffic. Instead, fiber optic cables or unshielded twisted pair (UTP) copper transmission lines are required to effectively carry such traffic. SONET OC-3 signals used by the *ATMCard* travel over UTP Category 5 (UTP-5) cables. It has been shown that for relatively short distances (≈ 100 m) UTP-5 cables may be used to transport 155.52 Mbit/s SONET signals [BAN92], [IM93]. Therefore, UTP-5 wiring is ideal for high-bandwidth local area networks (LANs) where distances are relatively short. Testing of the *ATMCard* will be performed under very short distances (< 10 m) allowing the use of UTP-5 cables. Higher data rate signals such as SONET OC-12 and OC-48 will require optical fiber cables as a transmission medium.

4.6 Summary

This chapter has presented a brief specification of the optical backplane system in which the *ATMCard* is to operate, and a list of functions the *ATMCard* is to perform. For a more detailed specification of the smart pixel array or the 1998 CITR Optical Backplane Demonstrator, the reader is referred to [SZY94a] or the CITR.

CHAPTER 5
ATMCard Overview

5.1 Introduction

This chapter will present an overview of the ATM/SONET transmit/receive card implemented on a prototyping PCB. The prototyping PCB utilizes *Field-Programmable Gate Arrays* (FPGAs) and Speedwire™ [1] prototyping terminals and will be incorporated within the *HyperPlane* system. This transmit/receive card has been dubbed the *ATMCard*. Chapter 5 begins with an explanation of the *ATMCard* system design goals. It then continues with a description of the hardware components used on the *ATMCard*.

5.2 ATMCard System Design Goals

When designing and choosing the components of the *ATMCard*, care was taken to meet certain design goals. With competition in research and industrial sectors reaching a feverish pace, it becomes necessary to design and test prototype digital systems as quickly and as efficiently as possible. Therefore, a reusable single hardware platform enabling the rapid and efficient testing of digital system designs becomes attractive. With these philosophies in mind, the *ATMCard* was designed within a prototyping framework which enables rapid testing and debugging of the *ATMCard*'s various transmit and receive functions. Specifically, the *ATMCard* makes use of FPGAs and Speedwire™ prototyping terminals. Furthermore, large arrays of Speedwire™ terminals enable the inclusion of a wide variety of integrated circuits with varying footprints. This feature makes the choice of components used in the ATMCard less restrictive.

Both Speedwire™ and wirewrap technologies avoid the use of permanent connections between integrated circuits (ICs) and instead use conventional copper wire. During system testing, a re-design that requires certain connections to be altered may be

[1] Speedwire is a trademark of VERO Electronics, Inc.

performed by simply removing the wires connecting terminals in the previous design and making the new connections. Further advantages over wire-wrap that are demonstrated by Speedwire™ include [BIC94]:

1) Speedwire™ connections require less time to make than wirewrap connections since wire does not have to be stripped and wrapping of wire around terminals is not required.

2) PCBs using Speedwire™ may be packed closer together in a PCB card cage since Speedwire™ terminals are shorter than two-level or three-level wirewrap terminals.

3) Speedwire's™ low profile minimizes the "antenna" effect at high frequencies that is generated by radiation from the top of the wirewrap terminals.

4) Sockets for dual in-line package (DIP) ICs are not required.

One drawback of using Speedwire™ as opposed to wirewrap is cost. However, the time that is saved from making connections in Speedwire™ more than justifies this additional cost.

The *ATMCard* system design goals and many design features on the current version of the *ATMCard* were developed after much discussion and help from Ted Szymanski and John Walker of McGill University. The remainder of this chapter will discuss the components that are used in the *ATMCard*. The *ATMCard* is a transmitter/receiver card that transports ATM cells which are extracted from a SONET frame and then transports (or switches) these cells across the *HyperPlane*. In addition to its function as a transmit/receive card, the *ATMCard* also performs ATM VPI/VCI header translation on the ATM cells.

5.3 Components of the ATMCard

This section will briefly describe each of the components that will be used on the *ATMCard*. A simplified block diagram of the *ATMCard* is shown in Figure 5.1. In this Figure, the flow of ATM cell data is shown. Note that *incoming cells* are defined to be those cells that are transmitted from the TNETA1500A transceiver to the *HyperPlane*. Similarly, *outgoing cells* are those cells that are transmitted from the *HyperPlane* to the TNETA1500A transceiver (Figure 5.1).

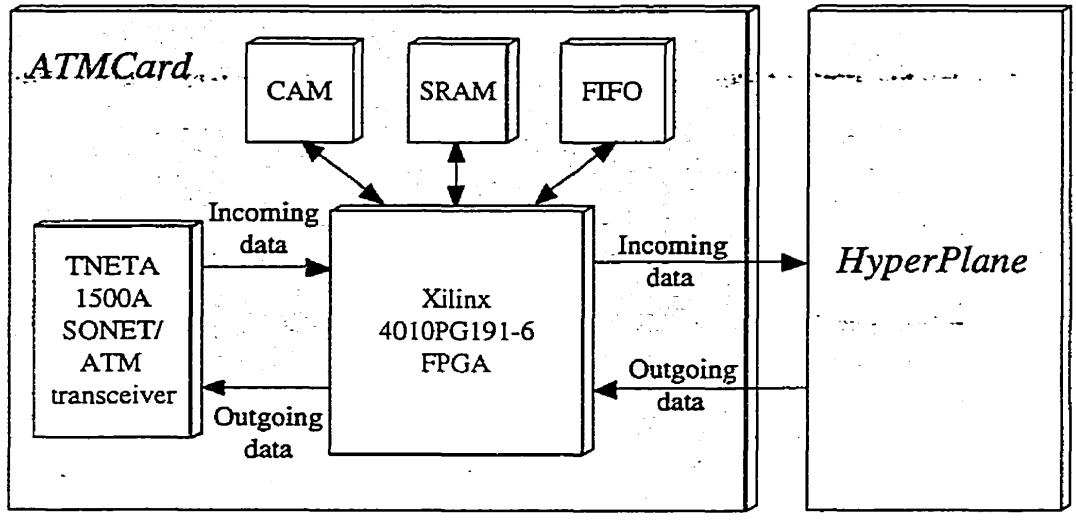


Figure 5.1: Simplified block diagram of the *ATMCard* and the *HyperPlane* showing the major components and *incoming* ATM cell data flow and *outgoing* ATM cell data flow.

The main components of the *ATMCard* are also shown in Figure 5.1. The main control unit of the *ATMCard* resides within the FPGA. Other components on the *ATMCard* include the TNETA1500A SONET/ATM transceiver, a *Content-Addressable Memory* (CAM), an SRAM, and a *First-In First-Out* (FIFO) queue. It should be noted that since the optical *HyperPlane* is still under development and not scheduled for completion until 1998, a functionally equivalent version of the *HyperPlane* has been emulated within the FPGA for the purposes of this thesis. Subsequent sections of this chapter will describe each of the *ATMCard* components.

5.3.1 Xilinx FPGAs

The reconfigurability of the *ATMCard* is realized through the use of *Field-Programmable Gate Arrays* (FPGAs). FPGAs are "general-purpose, multi-level programmable logic devices that are customized in the package by the end users" [TRI94]. A major difference between FPGAs and their cousins, the *Mask Programmed Gate Arrays* (MPGAs), is that in the latter custom masks are required during the manufacturing process. Custom masks make low-volume production of MPGAs expensive [TRI94]. Therefore, low-volume designs such as digital system prototypes favour the use of FPGAs from a cost perspective. One drawback, however, is that designs implemented using FPGAs generally require more gates than when implemented on an MPGA. In addition, interconnections are slower in an FPGA due to the programming circuitry. Therefore, once a design has been thoroughly tested and is ready for mass-volume production, one can expect that a design using MPGAs will not only be faster but will also require less gate area.

FPGAs are composed of logic blocks that are connected by a programmable interconnect and are surrounded by input/output (I/O) blocks (Figure 5.2). There are many commercially available FPGAs, each with varying logic block complexities and programmable interconnect strategies. Currently, the most popular are the SRAM-based FPGAs such as those produced by Xilinx, AT&T, and Altera. The *ATMCard* uses the XC4000 family of FPGAs from Xilinx.

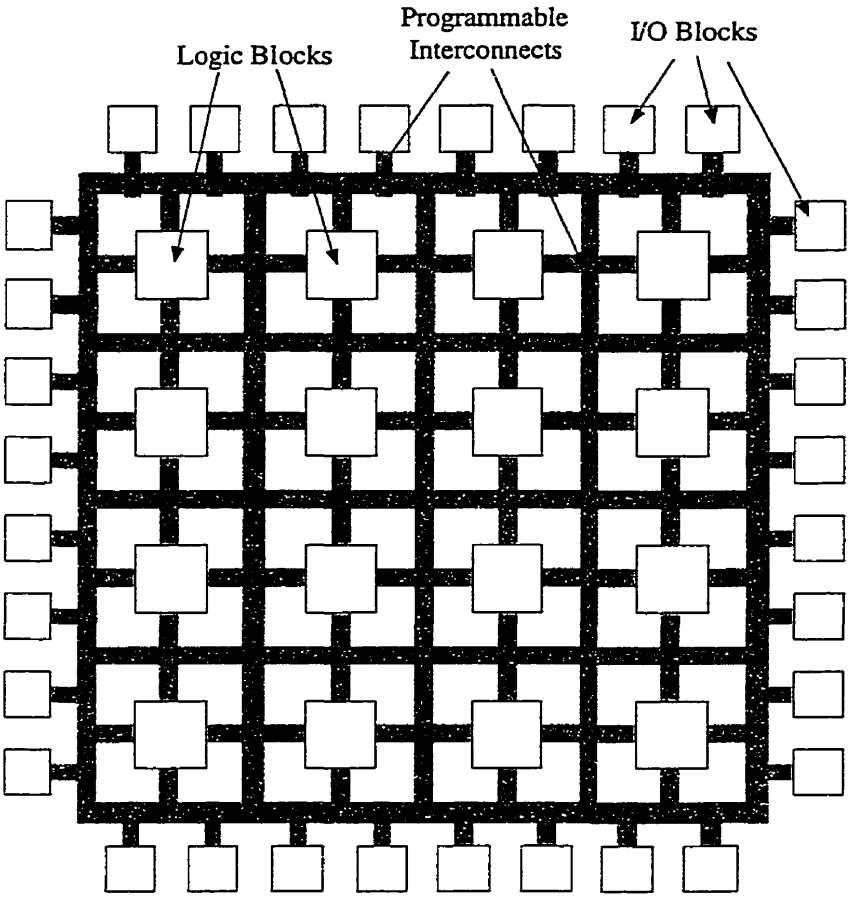


Figure 5.2: Generalized FPGA architecture [ROS93].

The *ATMCard* utilizes the Xilinx XC4010pg191 FPGA. This FPGA implements combinatorial logic in small look-up tables [XIL94]. The XC4010pg191 can hold an approximate gate count of 10,000 gates where a gate is defined as a 2 input NAND [BRO92]. Furthermore, the XC4010pg191 contains 160 user-definable I/O ports. The gate density and number of I/O ports are sufficient for the *ATMCard*. If larger or faster designs are to be implemented on the *ATMCard* requiring higher gate densities and additional I/O ports, then FPGAs such as the Xilinx XC4013 or 4025, the Altera *FLEX 10K* family, or the AT&T ORCA families of FPGAs may be utilized. These devices boast gate densities exceeding 100,000 gates with user I/O ports numbering greater than 400 and clock speeds up to two or three times the clock speeds found on the XC4010pg191-6.

5.3.1.1 FPGA Design Methodology

Portions of the *ATMCard* design were implemented using Xilinx XC4010pg191-6 FPGAs. The design implemented within the FPGA was performed according to a standard design methodology. The hardware description language, VHDL (*V*ery high speed integrated circuit *H*ardware *D*escription *L*anguage), was chosen as the design level entry for the various hardware modules that are implemented within the FPGA. Its ease of use, debugging and modification features were reasons for choosing VHDL. Synthesis of the VHDL code into a Xilinx netlist format (XNF) was performed by Synopsys version 3.4b synthesis tool. Finally, partitioning, placement, and routing (PPR) of the design was performed on the XNF files by the Xilinx Automated CAD Tools (XACT) version 5.2.1.

5.3.2 Texas Instruments' SONET-ATM Transceiver: The TNETA1500

The TNETA1500 is a single chip BiCMOS SONET/ATM transceiver [TEX94]. Hereafter, we will refer to this chip as the SONET/ATM transceiver or just transceiver. The SONET/ATM transceiver along with the Xilinx FPGA form the cornerstones of the *ATMCard*. ATM insertion and extraction to and from SONET frames is performed by the transceiver, while ATM data flow control and the *HyperPlane* simulator are implemented in the Xilinx FPGA. The transceiver implements the physical layer of the ATM Protocol Reference Model by transporting ATM cells over a SONET frame running at the STS-3c rate of 155.52 Mbit/s. Incoming SONET frames are accepted serially by the transceiver (Figure 5.1). The device performs clock recovery, frame alignment, serial-to-parallel data conversion, SONET payload identification, and ATM cell boundary establishment on those frames. In the *ATMCard*, ATM cells are extracted from the SONET payload, descrambled, and then sent to an incoming first-in first-out (FIFO) queue. From there, the *ATMCard* processes the ATM cells before transmitting them to the McGill *HyperPlane* (Figure 5.1). Outgoing ATM cells received from the *HyperPlane* are placed into a transmit input FIFO queue located within the transceiver. From here, these ATM cells are then scrambled and inserted into the synchronous payload envelope (SPE) of an STS-3c frame. The 155.52 Mbit/s output clock is generated by an analog phase lock loop (APLL) from an external 19.44 MHz oscillator. Figure 5.3 shows a simplified block diagram of the major components of the SONET/ATM transceiver.

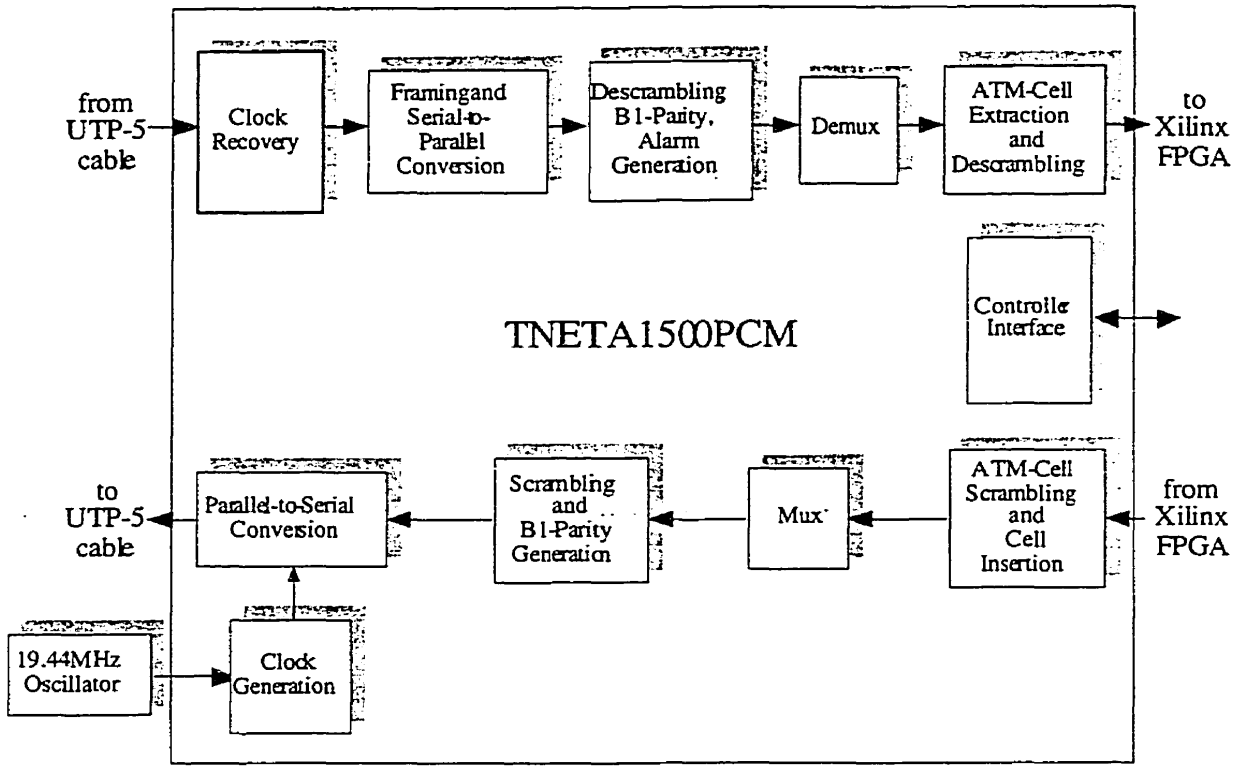


Figure 5.3: Simplified functional block diagram of the TNETA1500 [TEX94].

5.3.3 MUSIC Semiconductor's Content-Addressable Memory

Content-addressable memories (CAMs) operate in a fashion opposite to their Random Access Memory (RAM) counterparts [MUS94]. In a RAM, an address is presented to the device and the data that is pointed to by that address is accessed. However, in a CAM data is presented at the inputs and the address where that data is located is output. A flag is also generated to signal a hit if the data has an associated address within the CAM; a miss is generated if there is no associated address for the data. Alternatively, a CAM may output associated data instead of the matching address. The speed at which CAMs operate make them ideal for data search processes. In the *ATMCard*, ATM header translation is accomplished through the use of CAM and SRAM memories.

MUSIC Semiconductors' MU9C1480 LANCAM® is a 1K x 64 bit fixed-width CMOS CAM used in Local Area Network (LAN) address filtering applications and in ATM VPI/VCI header translation and tagging (Figure 5.4). External communications with the LANCAM is done on a 16-bit wide data bus. The internal data path, however, is 64-bits wide thus requiring 4-way multiplexing in order to connect with the external bus. Control flexibility of the LANCAM is obtained through four control signals: *Write Enable*, *Chip*

Enable, *Command Enable*, and *Comparison Enable*. Furthermore, an extensive instruction set minimizes software overhead in systems. The internal memory array of the LANCAM may be configured as a mixture of CAM and RAM on 16-bit boundaries. Data stored in the memory array may be accessed randomly as in RAMs or associatively as in CAMs. Finally, when data is input into the Comparand register for comparison with the CAM array, individual bits may be masked by setting the appropriate Mask register bits during both read and write comparison operations. All these features allow easy use of the LANCAM in ATM header translation applications.

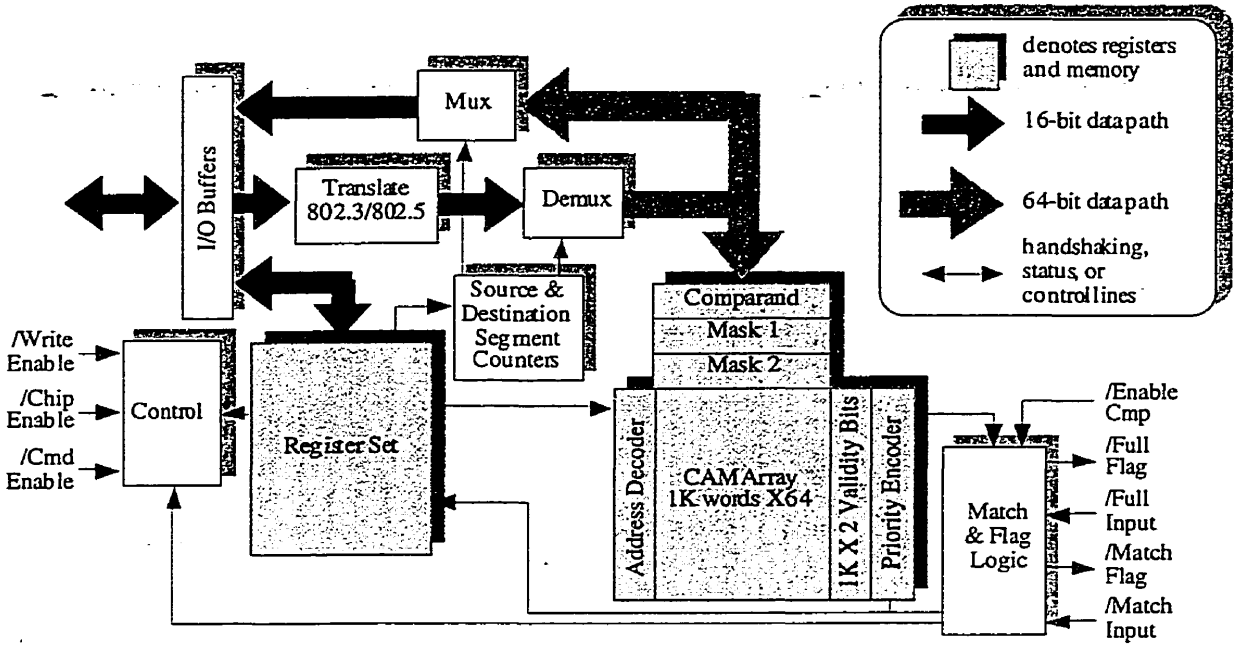


Figure 5.4: Simplified functional block diagram of the LANCAM [TEX94].

5.3.4 Memory

Memory devices used in the *ATMCard* include 256-Kbytes Static RAMs (SRAMs), 64x4 bit and 64x5 bit parallel FIFOs, and 512-Kbytes UV-erasable EPROMs. After a CAM comparison has been made, the associated data that is pointed to by the CAM output is located in the SRAMs. The *ATMCard* utilizes high-speed SRAMs[1] manufactured by Integrated Device Technology, Inc. which have a read-write access time of 35 ns [INT94]. FIFOs are located on the *ATMCard* to pipeline the design [INT94a]. A 4-bit wide and a 5-bit wide FIFO are configured together as a 9-bit wide FIFO[2] . The extra bit is used as a tag field. 512-Kbyte EPROMs[3] are used to store the bit-stream used to program the Xilinx FPGAs [TEX89]. Further details on the use of the CAM and SRAM memory components are discussed in the following chapter.

5.4 Summary

This chapter introduced the *ATMCard* and presented a brief description of the major components used on the *ATMCard.* The next chapter will provide a detailed functional and structural description of the *ATMCard* and how the *ATMCard* utilizes these components.

[1] SRAM part number is IDT71256L35P.
[2] FIFO part numbers are IDT72401L35P (4-bit wide FIFO) and IDT72402L25P (5-bit wide FIFO).
[3] EPROM part number is 27C512.

CHAPTER 6
ATMCard Design Implementation

6.1 Introduction

Chapter 6 will provide a detailed description of the *ATMCard*. The *ATMCard* is a transmit/receive card that transports ATM cells between fiber or UTP-5 cable carrying SONET frames and the McGill *HyperPlane*. After the reception of SONET frames from optical fiber or UTP-5 cables, the *ATMCard* extracts ATM cells from these SONET frames. ATM cell header translation is then performed on these ATM cells, after which the cells are transmitted to the *HyperPlane*. The *HyperPlane* serves as the switching medium; based on the value of the VPI and VCI fields of the ATM cell header, ATM cells are switched to other *ATMCards* via the *HyperPlane* (Figure 6.1). ATM cells received by an *ATMCard* from the *HyperPlane* are then inserted into SONET frames for transport over fiber or UTP-5 cables. It should be noted that the optical *HyperPlane* demonstrator is not scheduled for operation until 1998. Hence, a VHDL implementation of the *HyperPlane* emulated on a Xilinx FPGA was used in this thesis. This *HyperPlane* Simulator was created by Stephane Gagnon and Palash Desai when they were with McGill University [SZY94a], [DES95a].

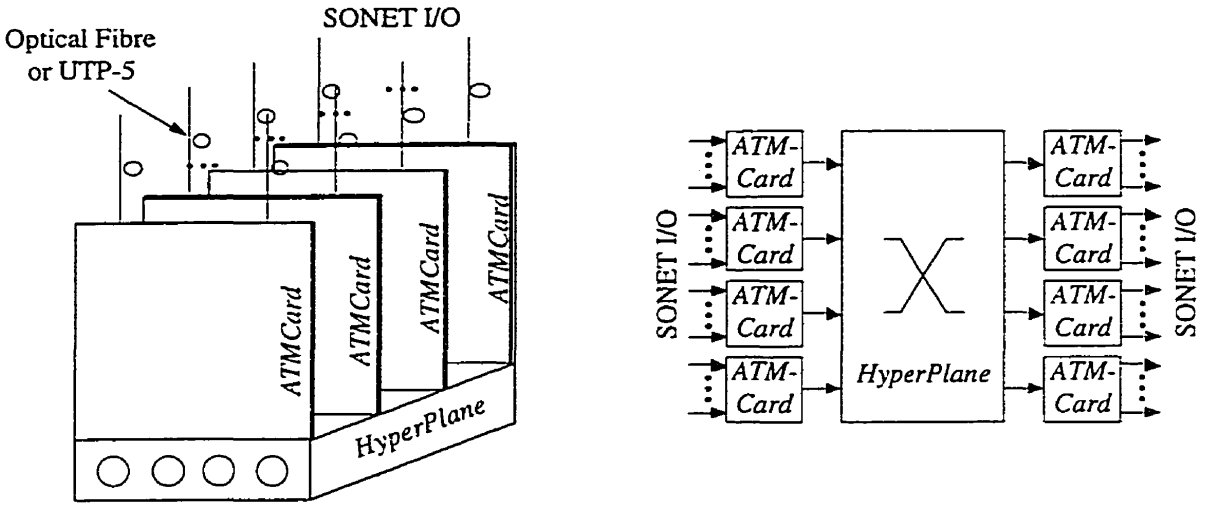


Figure 6.1: Equivalent representations of the *ATMCard* and the McGill *HyperPlane* system. Next generation *ATMCards* will support multiple SONET I/Os per *ATMCard*.

Portions of the *ATMCard* design are based upon the work of students in McGill University's 304-494 Project Laboratory, supervised by Dr. Ted Szymanski. Nabbus and Telfer's [NAB95] project involved the VHDL design of a unidirectional interface that extracted ATM cells from the TNETA1500A transceiver and transmitted them to the *HyperPlane*. This design stored an entire 53-byte ATM cell in a FIFO queue that was designed using VHDL and targeted for a Xilinx 4010PG191 FPGA. Since space is at a premium on a Xilinx 4010 FPGA, it was decided to use an external FIFO for the *ATMCard*. The second 304-494 project that the *ATMCard* is based on was by Li and Wai [LI95]. In this project, an ATM header translation component was designed that makes use of a CAM and a replacement arbitration unit. Both the CAM and the replacement arbitration unit were coded in VHDL for implementation on a Xilinx FPGA. In order to save space, the *ATMCard* uses an external CAM. The *ATMCard* design incorporates portions of these two projects. The *ATMCard* is also loosely based on a design described in an application note by MUSIC Semiconductors [BRI94].

This chapter will provide a description of the *ATMCard* design. The functionality of the various VHDL entities that make up the *ATMCard* will be explained. Also, through the aid of figures, a structural description will be provided. The design will be presented in a top-down fashion.

6.2 *ATMCard* Design Overview

This section will present an overview of the *ATMCard* design. Figure 6.2 shows a block diagram of the major components that comprise the *ATMCard*. The *ATMCard* is comprised of commercially available components - the TNETA1500A transceiver, the CAM, the FIFO, and the SRAM - and components designed in VHDL that are implemented in an FPGA - the *TNET_Interface*, the *HyperPlane_Interface*, and the *TNET_TX_Interface*. For practical design purposes, the *HyperPlane* Simulator will be implemented on the same FPGA as the other VHDL components. Since the previous chapter dealt with the TNETA1500A, the CAM, the FIFO, and the SRAM memories, this chapter will concern itself primarily with the VHDL components.

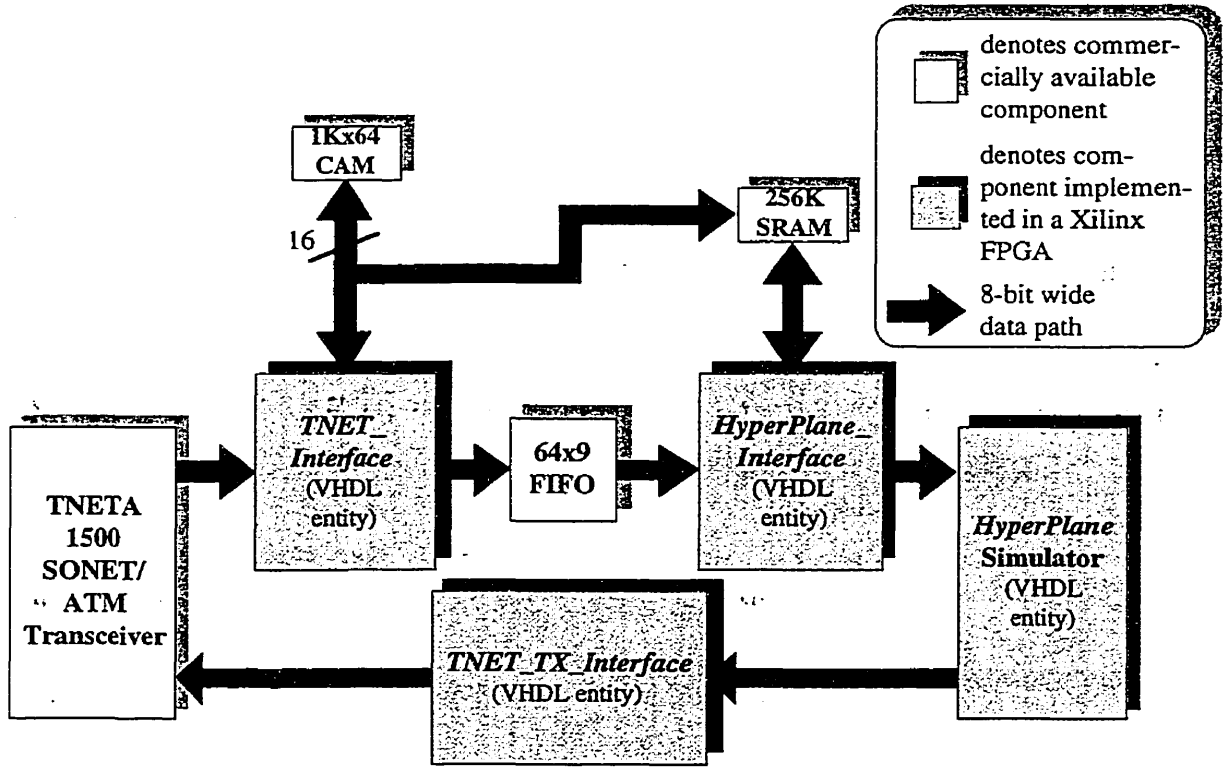


Figure 6.2: *ATMCard* block diagram (all data path widths are 8-bits except data path connecting *TNET_Interface* and CAM which is 16 bits).

The *ATMCard* serves three main functions. First, the *ATMCard* controls the transport of ATM cells from SONET frames received from the SONET/ATM transceiver to the *HyperPlane*. Second, it performs rudimentary ATM header translation before transmitting the cells across the *HyperPlane*. Finally, the *ATMCard* controls the transport

of ATM cells that are received from the *HyperPlane* and destined for the SONET/ATM transceiver. The switching core is implemented within the *HyperPlane* as illustrated in Figure 6.1.

ATM cell data flow within the *ATMCard* is shown in Figure 6.3. With the aid of this figure, the reader should get a better understanding of how the *ATMCard* functions. In Figure 6.3 (a), the first byte of an ATM cell is received by the *TNET_Interface* component of the *ATMCard*. (This point is labeled (i).) After reading in the 5-byte ATM header from the SONET/ATM transceiver, the *TNET_Interface* loads the VCI and VPI values of the header into the CAM (Figure 6.3 (b) point (ii)). The CAM, together with the SRAM, serves as a look-up table; according to the value of the VCI and VPI fields received from the *TNET_Interface*, the CAM will perform a search and - if it finds a match - will provide the SRAM address where the new VCI and VPI fields are located. In Figure 6.3 (c), a CAM match is assumed to have occurred. Thus, the *TNET_Interface* reads out the matching SRAM address field from the CAM look-up table and this data is supplied to the SRAM (Figure 6.3 (c) point (iii)). Concurrently, the *TNET_Interface* reads out the payload of the ATM cell and transmits this payload to the external 64x9 FIFO (Figure 6.3 (c) point (iv)).

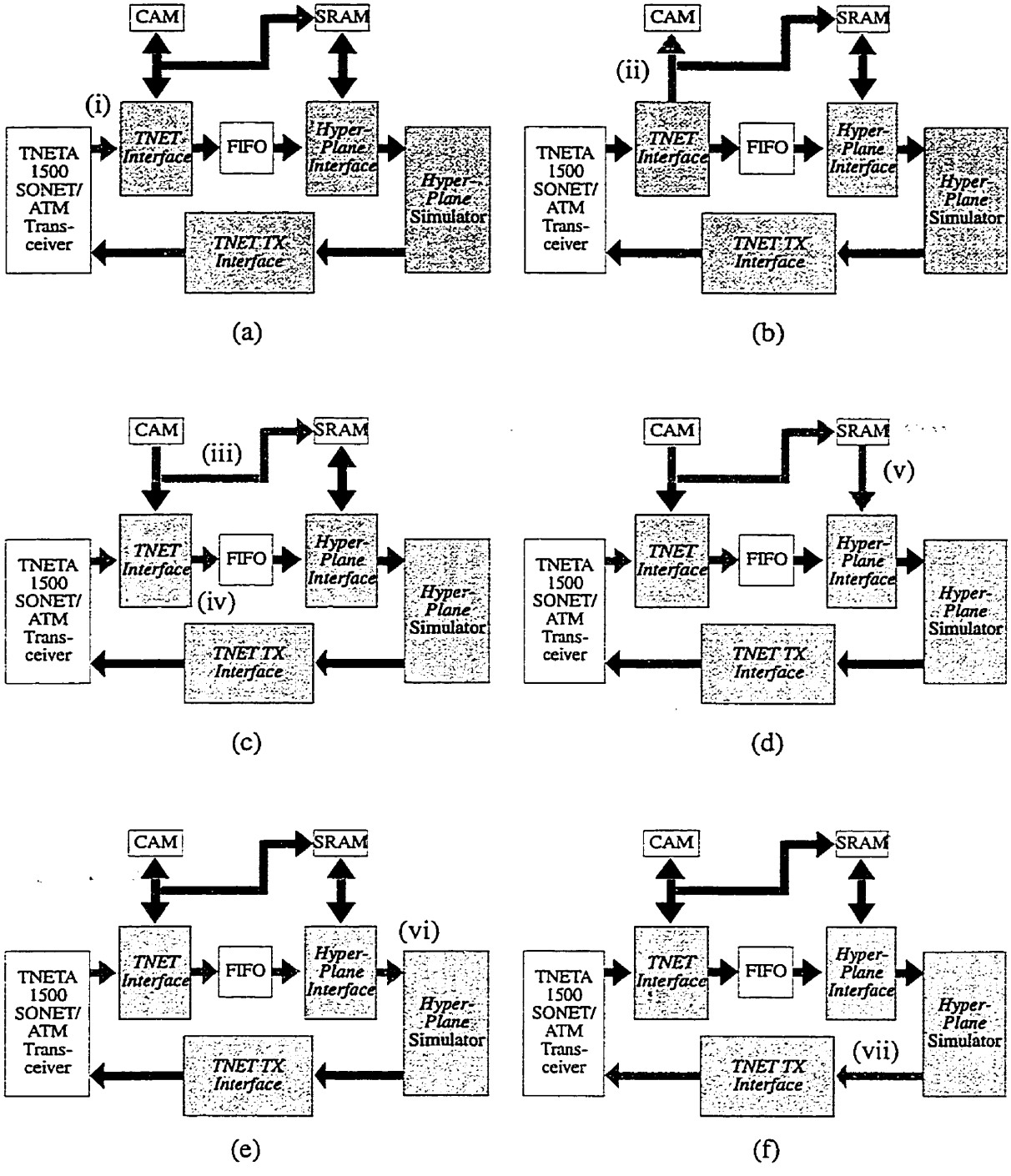


Figure 6.3: Operation and ATM cell data flow in the *ATMCard*.

While the SRAM address from the CAM is valid, the *HyperPlane_Interface* reads and stores the new 6-byte ATM cell header from the SRAM (Figure 6.3 (d) point (v)). The header is now 6-bytes because an additional byte has been prepended. This byte is for internal *HyperPlane* use and is needed to identify the destination *ATMCard* this ATM cell is intended for. After the new header has been stored in the *HyperPlane_Interface*, the

HyperPlane_Interface then waits for a "start-of-frame" signal from the *HyperPlane*. It then transmits the 6-byte header to the *HyperPlane* followed by the 48-byte payload that it receives from the external 64x9 FIFO (Figure 6.3 (e) point (vi)). For an *ATMCard* receiving an ATM cell from the *HyperPlane*, the *TNET_TX_Interface* is first informed by the *HyperPlane* that an ATM cell is arriving (Figure 6.3 (f) point (vii)). After removing the prepended byte from the header, the *TNET_TX_Interface* then transmits this 53-byte ATM cell to the TNETA1500A SONET/ATM transceiver.

The purpose of the external 64x9 FIFO memory in the data flow cycle described above is to pipeline the design. During header translation, the *ATMCard* can simultaneously read in the rest of the ATM cell and store it within the 64x9 FIFO. In a sense, the *ATMCard* is multitasking the two functions of cell transmission to the *HyperPlane* and cell header translation. While header translation within the CAM and SRAM is being processed, the *TNET_Interface* continues to read in the 48-byte ATM payload from the SONET/ATM transceiver. In order to avoid losing any clock cycles, the time required for the *ATMCard* to perform header translation should be equal to or less than the time it takes to read in the 48-byte header from the SONET/ATM transceiver. Otherwise, if ATM cells with valid data were to arrive in every time slot the ATM buffers in the SONET/ATM transceiver would overflow causing ATM cell loss to occur.

The 64x9 external FIFO also contains a "fall-through" mode. In this mode, data arriving at the input of an empty FIFO is propagated to the output after a certain fall-through delay [INT94a]. This fall-through delay is much less than the time required for data to reach the FIFO output after visiting each element of the FIFO queue. Because of this feature, the *HyperPlane_Interface* does not have to wait for data to propagate through all 64 memory elements of an empty 64x9 FIFO; data arriving at an empty FIFO input will appear at the output after a very short fall-through delay. In addition, it should be noted that transmission of ATM cells to the *HyperPlane* and reception of cells from the *HyperPlane* may occur simultaneously.

Figures 6.4 (a) and (b) present a more detailed illustration of the *ATMCard*. Figures 6.4 (a) and (b) are magnified versions of Figure 6.2 with signals being identified. The main VHDL components responsible for header translation and the transmission of ATM cells from the SONET/ATM transceiver to the *HyperPlane* are the *TNET_Interface* and the *HyperPlane_Interface* (Figure 6.4 (a)). In the other direction, the VHDL component which handles ATM cell transport from the *HyperPlane* to the SONET/ATM transceiver is the *TNET_TX_Interface* (Figure 6.4 (b)).

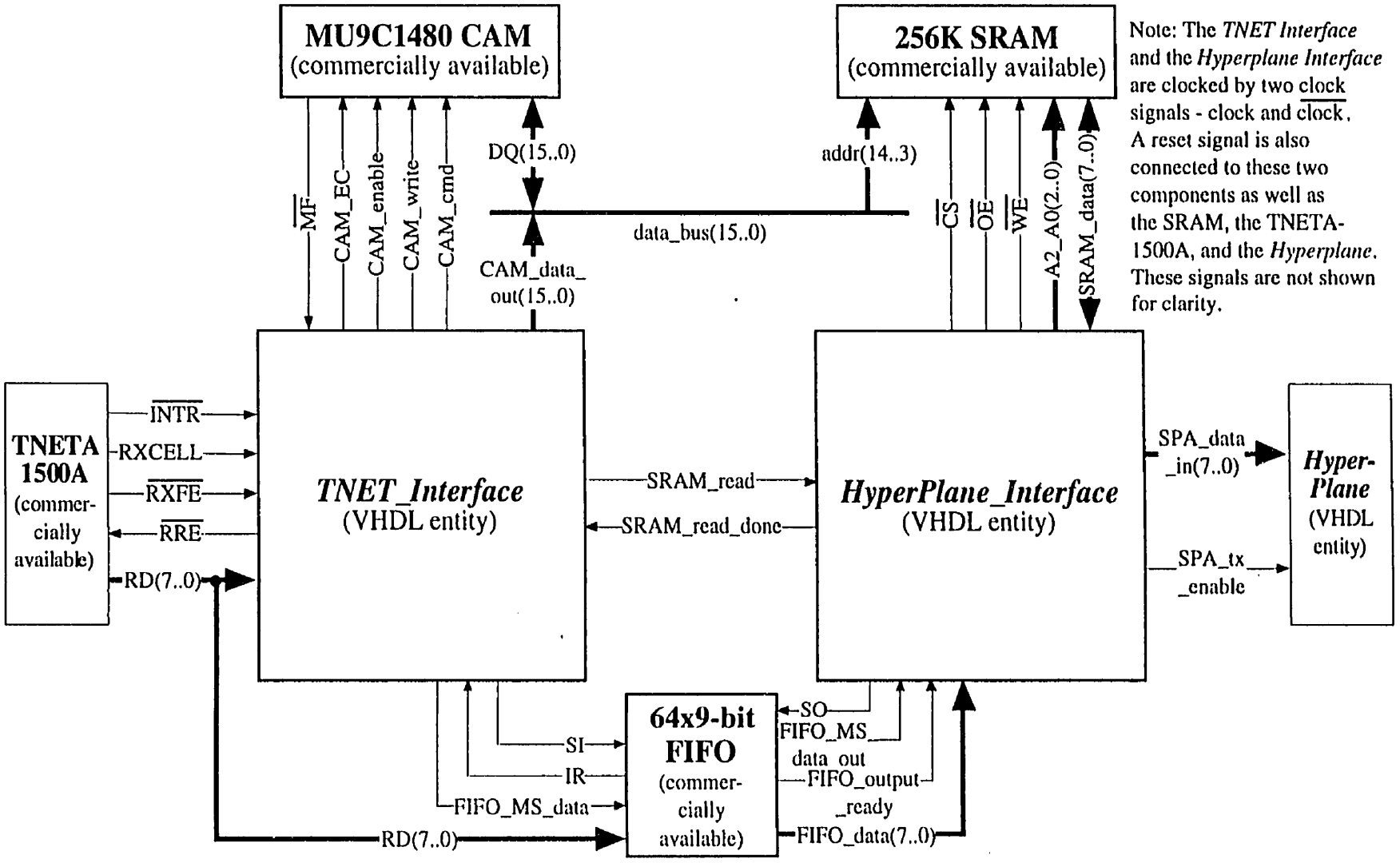


Figure 5.4 (a): Block diagram of the *ATMCard* components involved in the transmission of ATM cells from the SONET/ATM transceiver (TNETA1500A) to the *HyperPlane*.

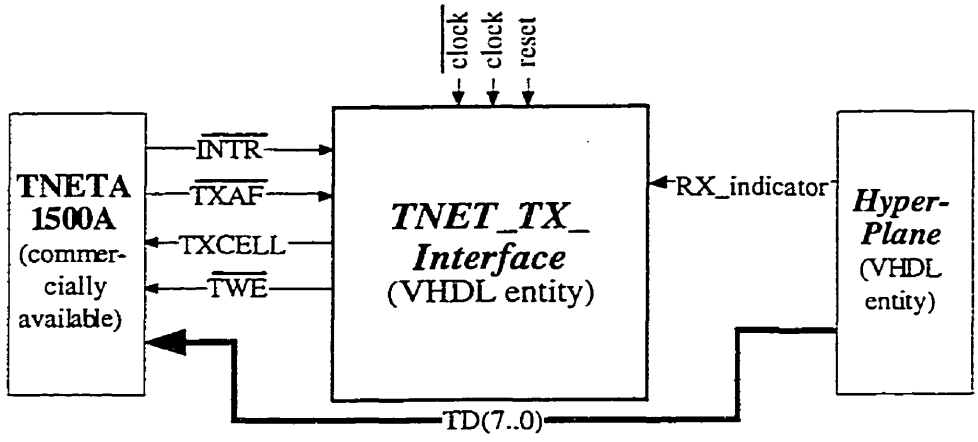


Figure 6.4 (b): The *HyperPlane*-to-SONET Interface block diagram.

This section has provided a high-level description of the function and structure of the *ATMCard*. Figure 6.5 illustrates the hierarchical structure of the VHDL entities/components of the *ATMCard*. In continuing with a top-down approach to describing the *ATMCard*, the remainder of this chapter will concentrate on providing a detailed description of each of these three main components and all their subcomponents shown in Figure 6.5.

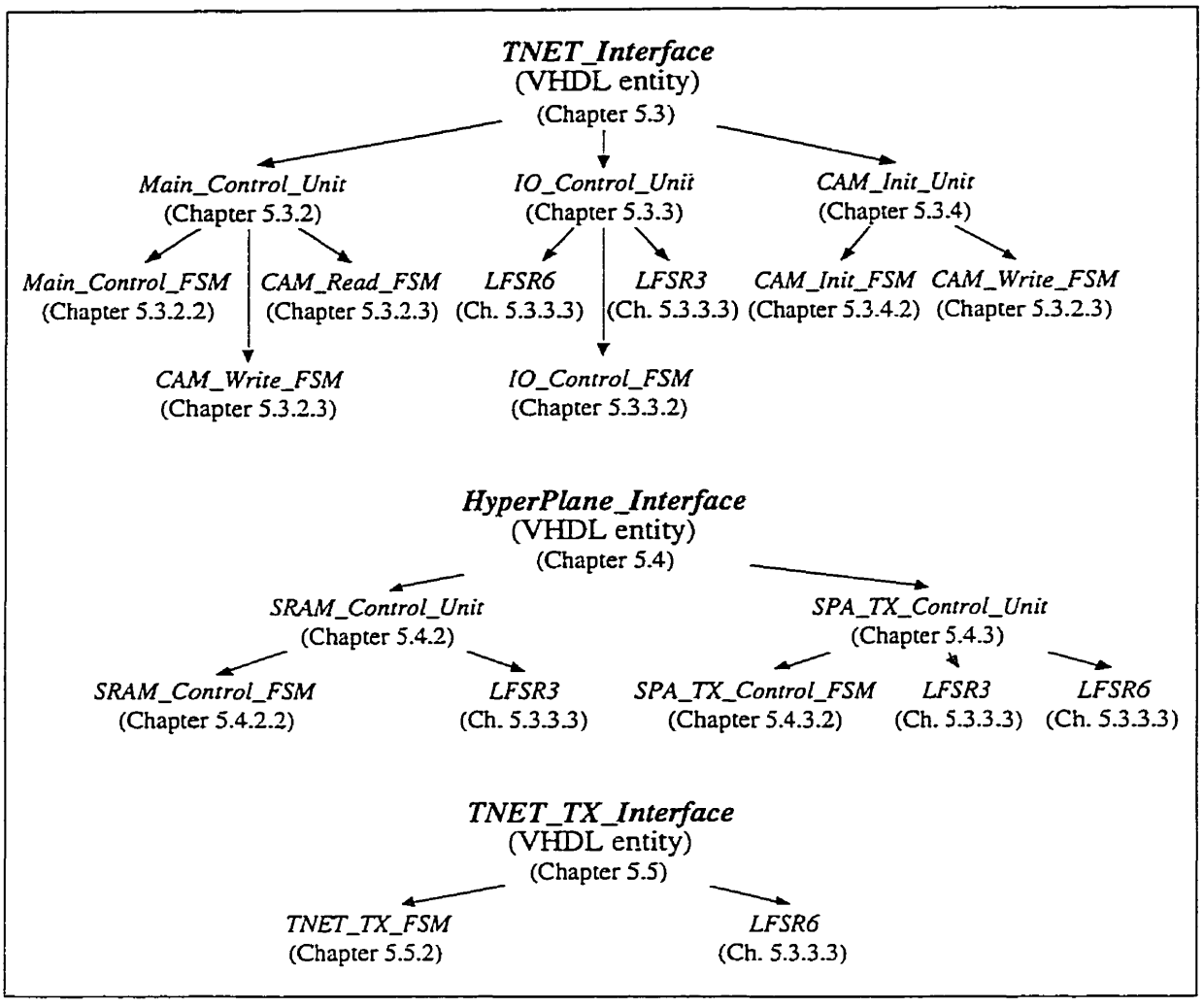


Figure 6.5: VHDL hierarchical organization of the three main *ATMCard* VHDL components. The three top level entities - *TNET_Interface*, *HyperPlane_Interface*, and *TNET_TX_Interface* - are shown in **bold**.

6.3 The *TNET_Interface*

6.3.1 Introduction

The *TNET_Interface* and the *HyperPlane_Interface* components of the *ATMCard* are responsible for controlling ATM cell header translation and the transmission of ATM cells from the SONET/ATM transceiver to the *HyperPlane*. This section will focus on the *TNET_Interface*. A detailed block diagram of the *TNET_Interface* is shown in Figure 6.6. The *TNET_Interface* is comprised of three main VHDL entities - the *Main_Control_Unit*, the *IO_Control_Unit*, and the *CAM_Init_Unit* - in addition to a 5x8-bit internal FIFO,

multiplexers, and logic gates. The overall VHDL hierarchical organization of the *TNET_Interface* was illustrated in Figure 6.5.

The *TNET_Interface* serves three main functions. First, it is responsible, through the *CAM_Init_Unit*, for the initialization and configuration of the CAM. Second, it controls the reading of the ATM header from the SONET/ATM transceiver and the subsequent writing of this header to the CAM during the header translation process. Finally, the *TNET_Interface* transmits the ATM payload from the SONET/ATM transceiver to the external 64x9 FIFO queue.

The central control component of the *TNET_Interface* is the *Main_Control_Unit* (Figure 6.6). It becomes operational after the *CAM_Init_Unit* has completed the initialization and configuration of the CAM. The *Main_Control_Unit* is specifically responsible for directing the *IO_Control_Unit* to read bytes of an ATM cell from the SONET/ATM transceiver. The *Main_Control_Unit* is also responsible for initiating the header translation process by writing the ATM header to the CAM and waiting to see if a match has occurred. If a matching field has been found within the CAM the *Main_Control_Unit* will read out the SRAM address value that is stored inside the CAM.

The *IO_Control_Unit* controls the byte-by-byte transfer of ATM cells from the SONET/ATM transceiver to the internal 5x8 FIFO (*FIFO5* in Figure 6.6) and also to the external 64x9 FIFO. The *IO_Control_Unit* performs these actions under the direction of the *Main_Control_Unit*.

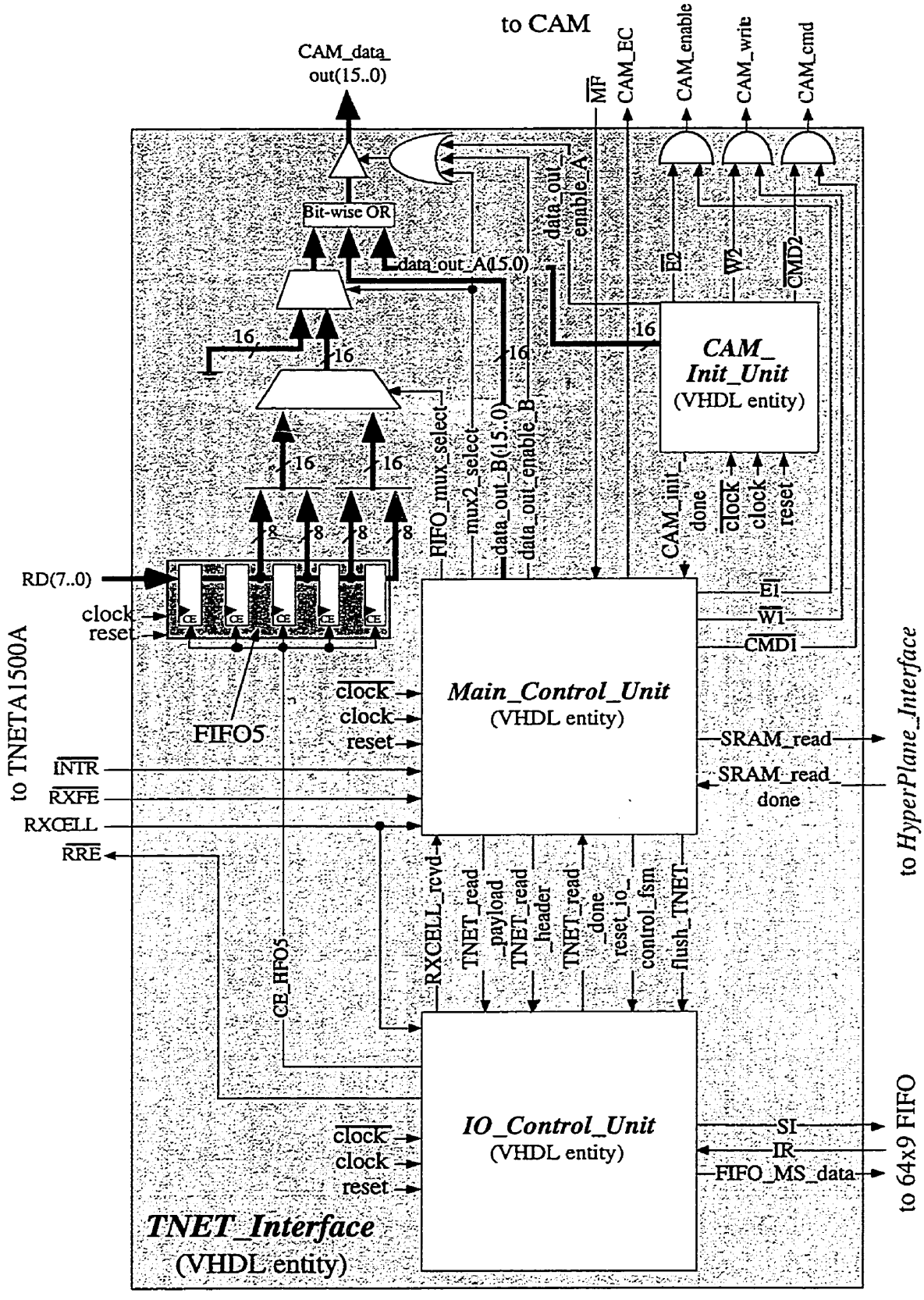


Figure 6.6: Block diagram of the *TNET_Interface*.

Following Figure 6.3 for an overview and Figure 6.6 for the block diagram, a description of the actions that the components of the *TNET_Interface* perform during header translation and ATM payload transmission will be given. In the point labeled (i) in Figure 6.3 (a), the start of an ATM cell is received by the *TNET_Interface*. The SONET/ATM transceiver sends a signal to the *Main_Control_Unit* that the first byte of an ATM cell is being output on *RD(7..0)* by setting the *RXCELL* signal to a logical value of '1'. Once the *Main_Control_Unit* receives this signal, it instructs the *IO_Control_Unit* - by setting the *TNET_read_header* signal to '1' - to read the ATM header (the first 5 bytes of the ATM cell) and store it into *FIFO5*.

Once the ATM header has been stored into *FIFO5*, the *TNET_Interface* writes this header to the CAM comparand register for comparison (Figure 6.3 (b), label (ii)). In order to perform this action, the *Main_Control_Unit* must first output the header on *CAM_data_out(15..0)*. The *Main_Control_Unit* accomplishes this by properly setting the two 2-to-1 multiplexers and the tri-state gate using the signals *FIFO_mux_select*, *mux2_select*, and *data_out_enable_B* (Figure 6.6). Since the *CAM_data_out(15..0)* output is 16 bits wide, the header can be written to the CAM comparand register 2-bytes at a time using two CAM write cycles. (The fifth byte of the header, the HEC field, is not used in header translation and, therefore, is not written to the CAM.) Once the CAM receives the 4-bytes of the header, it proceeds to search its memory contents for a matching field. In the LANCAM, each address location is partitioned into a 4-byte CAM field that is used in comparisons and a 2-byte RAM field that acts as the associated data. This associated data field stores the SRAM address pointer that points to the SRAM location where the new ATM header is located.

A match from the CAM is signaled by the assertion of a logical '0' value on the $\overline{MF}$ (Match Flag) signal. At this point, the *Main_Control_Unit* reads the associated SRAM address value from the CAM and supplies this value to the SRAM (Figure 6.3 (c), label (iii)). It then instructs the *HyperPlane_Interface*, via the *SRAM_read* signal, to read the new header from SRAM at the address pointed to by the CAM (Figure 6.6). The *Main_Control_Unit* will terminate reading the SRAM address from the CAM only when the *HyperPlane_Interface* has finished reading in the new header. The *HyperPlane_Interface* uses the *SRAM_read_done* signal to inform the *Main_Control_Unit* that it has finished loading the new header from SRAM (Figure 6.6). While header translation processes are being executed, the *Main_Control_Unit* uses the *TNET_read_payload* signal (Figure 6.6) to instruct the *IO_Control_Unit* to read the ATM

payload from the SONET/ATM transceiver and write this payload to the external 64x9 FIFO queue (Figure 6.3 (c), label (iv)).

Conversely, if the ATM header does not produce a match within the CAM the ATM cell is discarded. The *Main_Control_Unit* instructs the *IO_Control_Unit* to discard the rest of the ATM cell through the *flush_TNET* signal (Figure 6.6). The *IO_Control_Unit* will then flush the remaining bytes of the ATM cell from the TNETA1500A by setting $\overline{RRE}$ (Receive Read Enable) to '0' until the start of a new ATM cell is received.

This section has provided a description of the function and structure of the *TNET_Interface*. The following sections will discuss the three main components of the *TNET_Interface*: the *Main_Control_Unit*, the *IO_Control_Unit*, and the *CAM_Init_Unit*.

6.3.2 The *Main_Control_Unit*

6.3.2.1 Introduction

The block diagram for the *Main_Control_Unit* is shown in Figure 6.7. The *Main_Control_FSM*, the *CAM_Write_FSM*, and the *CAM_Read_FSM* are the three main VHDL components that comprise the *Main_Control_Unit* (Figures 6.5 and 6.7). CAM read and write operations are handled by the *CAM_Read_FSM* and the *CAM_Write_FSM* components, respectively. The CAM read and write functions were placed in separate components to simplify the *Main_Control_FSM*. It was also advantageous to separate these functions since they can be used again in other components; the *CAM_Write_FSM* is used by the *CAM_Init_Unit* during CAM initialization. This partitioning of finite state machines for the purposes of design component simplification is done repeatedly in the VHDL components of the *ATMCard*. The next two sections will discuss these three components of the *Main_Control_Unit*.

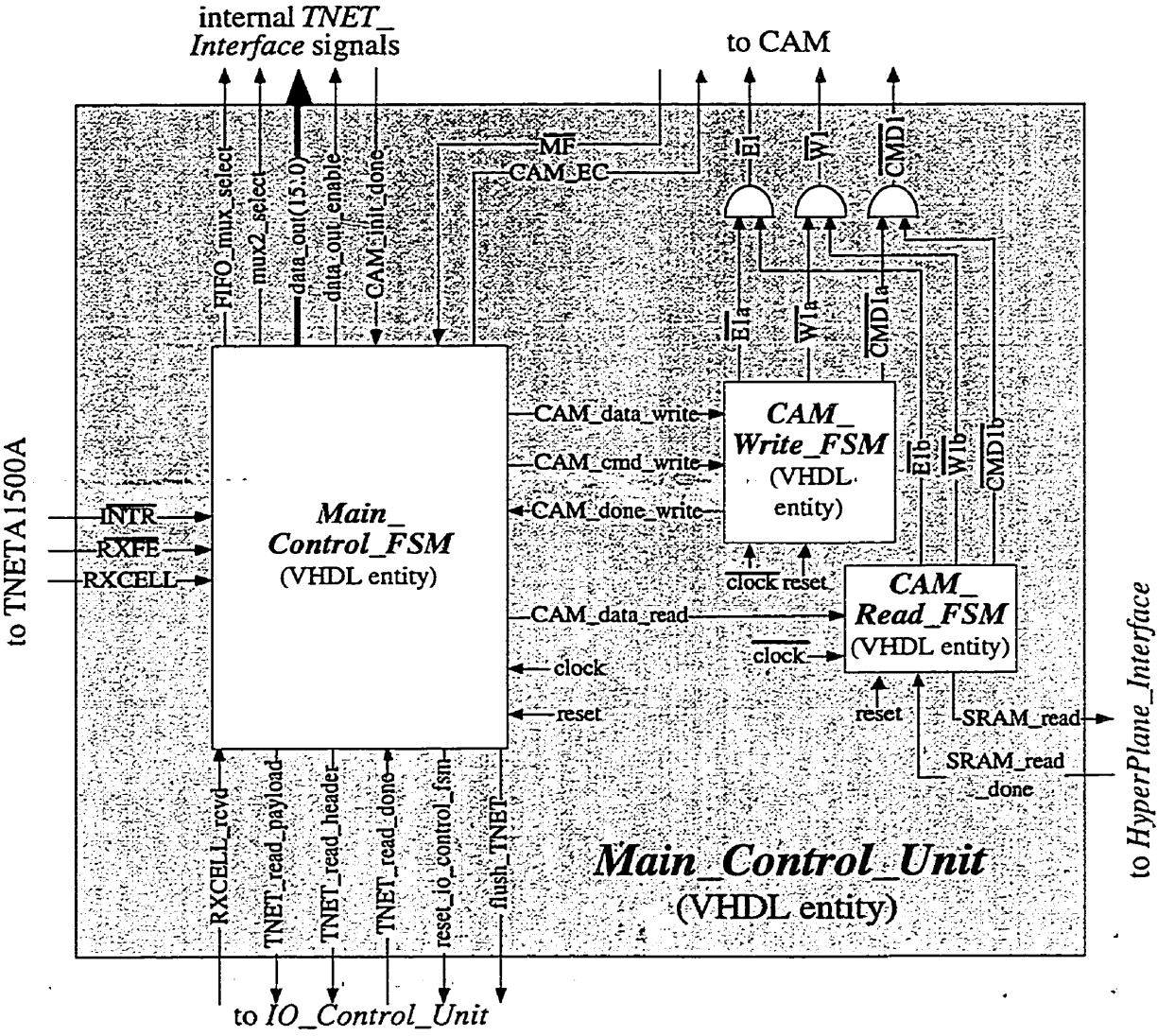


Figure 6.7: Block diagram of the *Main_Control_Unit*.

6.3.2.2 The *Main_Control_FSM*

When the *ATMCard* is first powered on, the *CAM_Init_Unit* of the *TNET_Interface* configures and initializes the CAM. Once this process is complete, control is transferred to the *Main_Control_FSM*. The *Main_Control_FSM* generates several control signals which control components in the *TNET_Interface*. The *Main_Control_FSM* is responsible for writing ATM headers it receives from the SONET/ATM transceiver to the CAM comparand register for comparison. In case of a CAM match condition, the *Main_Control_FSM* is also responsible for instructing the *IO_Control_Unit* of the *TNET_Interface* to transmit the ATM payload from the transceiver to the external 64x9 FIFO. In addition, if an ATM cell

does not appear on the output data lines (*RD(7..0)*) of the SONET/ATM transceiver the *Main_Control_FSM* instructs the *IO_Control_Unit* of the *TNET_Interface* to flush data from the SONET/ATM transceiver output queues until the start of an ATM cell appears on *RD(7..0)*. Finally, during ATM header writes to the CAM, the *Main_Control_FSM* handles the proper set up of the two 2-to-1 multiplexers and the tri-state gate that are located within the *TNET_Interface*.

The *Main_Control_FSM* is a 13-state finite state machine with 29 transitions and with 10 inputs and 27 outputs. A flow chart outlining the operation of the *Main_Control_FSM* is shown in Figure 6.8.

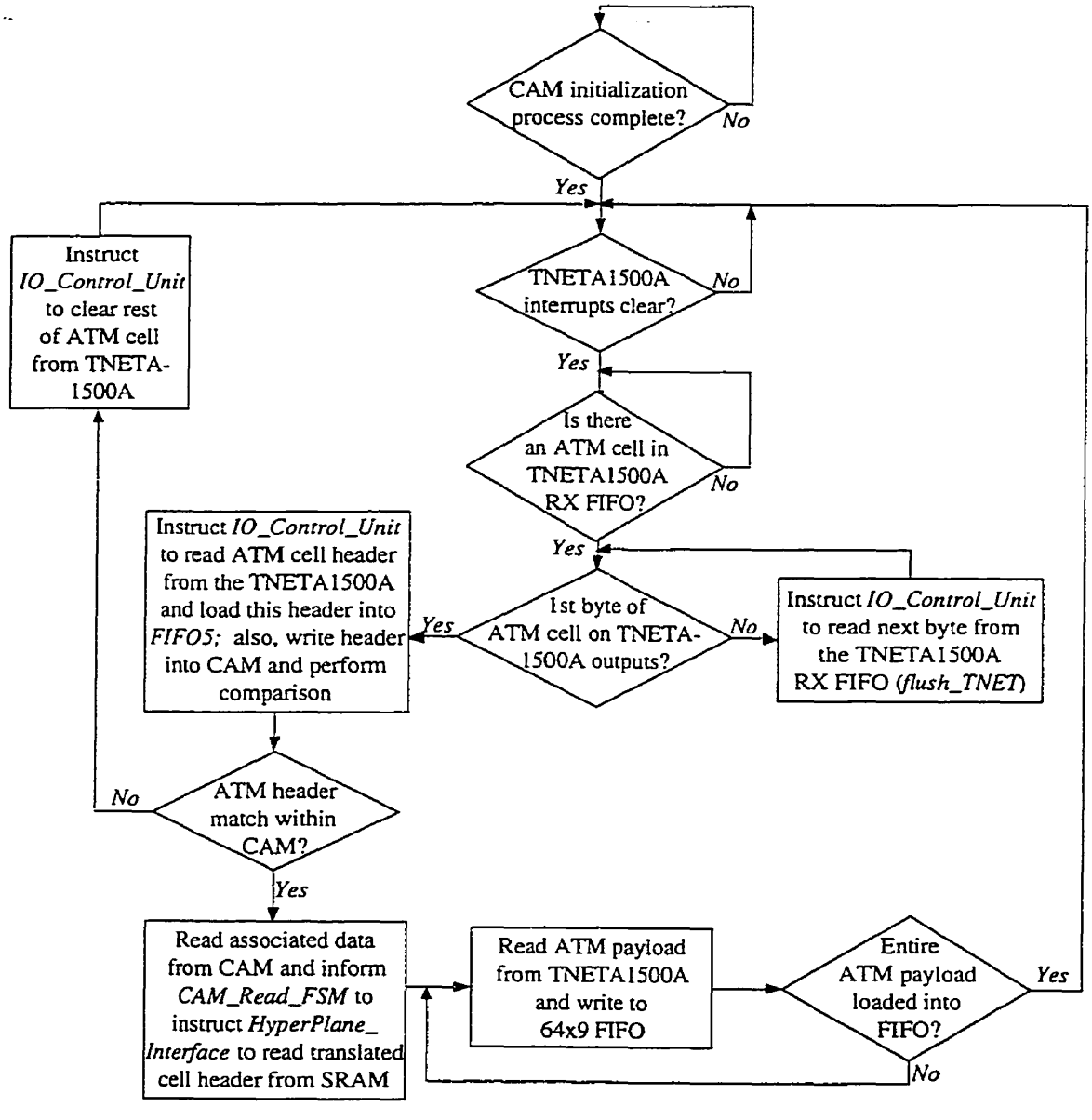


Figure 6.8: Functional flow chart for the *Main_Control_FSM*.

Starting at the top of Figure 6.8, the *Main_Control_FSM* first waits for the *CAM_Init_Unit* to complete the initialization and configuration of the CAM. Once this process is complete, the *Main_Control_FSM* will wait for all SONET/ATM transceiver interrupts to clear. This event is signaled when $\overline{INTR}$ = '1' (Figure 6.7). Following down the flow chart in Figure 6.8, the *Main_Control_FSM* waits for $\overline{RXFE}$ = '1' which means that there is an ATM cell in the internal receive queue of the SONET/ATM transceiver. Next, the *Main_Control_FSM* must wait for the first byte of the ATM cell to appear on *RD(7..0)* (Figure 6.6).

The appearance of the first ATM byte on *RD(7..0)* occurs when *RXCELL* = '1'. If the first byte does not appear on the output the *Main_Control_FSM* instructs the *IO_Control_Unit* to flush data from the internal receive queue of the SONET/ATM transceiver until *RXCELL* = '1'. Otherwise, the *Main_Control_FSM* instructs the *IO_Control_Unit* to read the 5-byte ATM header and to load this header into *FIFO5* (Figure 6.6). After setting up the multiplexers and the tri-state gate to allow data from *FIFO5* to appear on *CAM_data_out(15..0)* in Figure 6.6, the *Main_Control_FSM* then instructs the *CAM_Write_FSM* in Figure 6.7 to write the header from *FIFO5* to the CAM for comparison. If a match does not occur ($\overline{MF}$ = '1'), then the *Main_Control_FSM* instructs the *IO_Control_Unit* to flush the rest of the ATM cell from the SONET/ATM transceiver and proceeds back to the top of the flow chart (Figure 6.8) to begin the process again. If a match does occur, the *Main_Control_FSM* instructs the *CAM_Read_FSM* to read the associated data from the CAM. The *CAM_Read_FSM*, in turn, instructs the *HyperPlane_Interface* to read the new ATM header from SRAM. Meanwhile, the *Main_Control_FSM* instructs the *IO_Control_Unit*, through the *TNET_read_payload* signal, to read the 48-byte ATM payload from the SONET/ATM transceiver and write this into the external 64x9 FIFO, after which the *Main_Control_FSM* proceeds to the top of the flow chart in Figure 6.8 to begin the process again.

This section has attempted to provide a brief functional description of the *Main_Control_FSM*. The following section will discuss the *CAM_Write_FSM* and the *CAM_Read_FSM*.

6.3.2.3 The *CAM_Write_FSM* and the *CAM_Read_FSM*

The *Main_Control_Unit* utilizes the *CAM_Write_FSM* and the *CAM_Read_FSM* during CAM read and write cycles (Figure 6.7). With the LANCAM, there are two

different types of read and write cycles. There are read and write cycles that access the control registers of the CAM. These cycles are known as command reads or writes. Similarly, read or write cycles that access the memory portion of the CAM are known as data reads or writes. The *TNET_Interface* performs both data and command writes to the CAM but only executes data reads from the CAM.

Read and write cycle timing diagrams are illustrated in Figures 6.9 and 6.10. In both figures, the *CAM_Write_FSM* and the *CAM_Read_FSM* are responsible for controlling the CAM chip enable signals ($\overline{E1a}$ and $\overline{E1b}$), the CAM read/write signals ($\overline{W1a}$ and $\overline{W1b}$), and the CAM command/data signals ($\overline{CMD1a}$ and $\overline{CMD1b}$). The latter are used to notify the CAM whether a command read/write cycle is occurring or a data read/write cycle. Setting up of data on *CAM_data_out(15..0)* is controlled by the *Main_Control_FSM*.

Looking at the write cycle timing diagram in Figure 6.9, $\overline{W1a}$ is pulsed low at (i) to signify a write. The CAM does not care about the value of $\overline{W1a}$ before this point. A CAM command or data write is chosen by selecting the appropriate value for $\overline{CMD1a}$ where $\overline{CMD1a}$ = '1' denotes a data write and $\overline{CMD1a}$ = '0' denotes a command write. Once these signals have been set, the data has settled on *CAM_data_out(15..0)*, and after a certain setup time defined in the CAM data sheets [MUS94], $\overline{E1a}$ is pulsed low (point (ii) in Figure 6.9). Data appearing on *CAM_data_out(15..0)* is written into the CAM on the falling edge of $\overline{E1a}$.

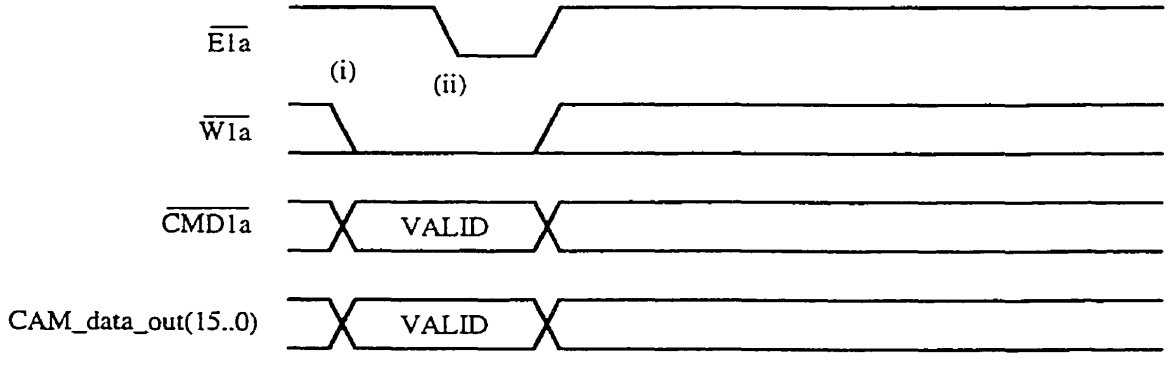


Figure 6.9: *ATMCard* CAM write cycle timing diagram
(signals named according to Figure 6.7).

Next, the CAM read cycle timing diagram shown in Figure 6.10 will be discussed. The *CAM_Read_FSM* is slightly more complex then the *CAM_Write_FSM* because it must

inform the *HyperPlane_Interface* when it can read a new ATM header from the SRAM. The CAM read cycle begins similarly to the write cycle in that the $\overline{W1b}$ and the $\overline{CMD1b}$ signals must be set before the CAM is enabled by pulsing $\overline{E1b}$ low (Figure 6.10, points (i) and (ii)). For command reads, $\overline{CMD1b}$ = '0' and for data reads $\overline{CMD1b}$ = '1'. Only data read cycles are used by the *TNET_Interface*. Also, the duration that $\overline{E1b}$ is pulsed low controls the length of time the data on *CAM_data_out(15..0)* is valid.

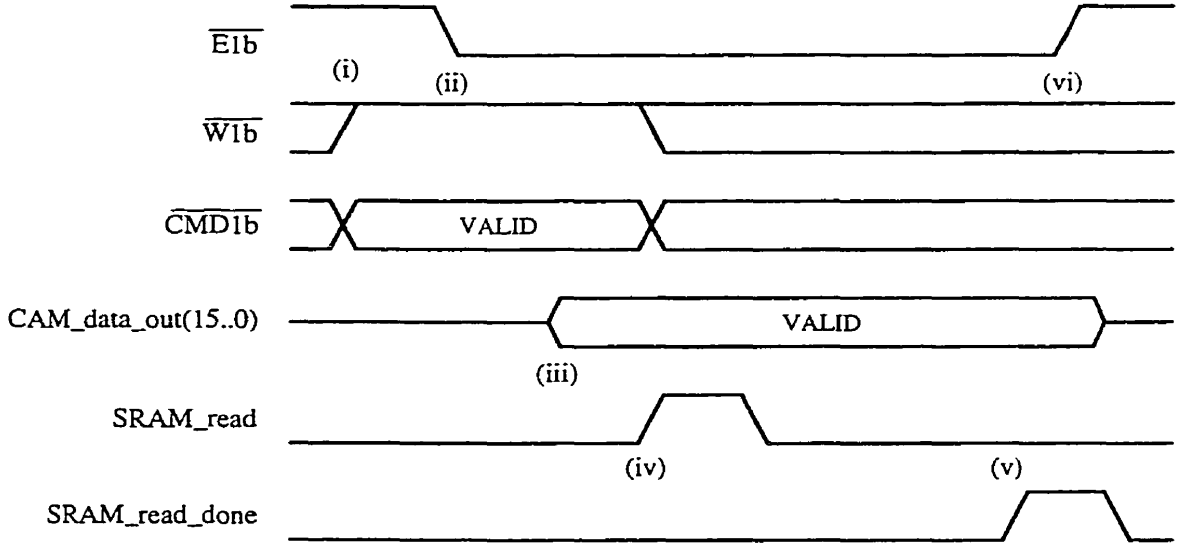


Figure 6.10: *ATMCard* CAM read cycle timing diagram (signals named according to Figure 6.7).

After a certain delay specified in the CAM data sheets [MUS94], data read from the CAM becomes valid on *CAM_data_out(15..0)* (Figure 6.10, point (iii)). In the *ATMCard*, this data will also be supplied to the address signals of the 256K SRAM. Recall that the associated data read from the CAM after a CAM match has occurred is the address location in the SRAM where the new ATM header can be found. In order to inform the *HyperPlane_Interface* that it can now read the new header from SRAM, the *CAM_Read_FSM* sets the *SRAM_read* signal to '1' (Figure 6.10, point (iv)). The *CAM_Read_FSM* then waits for the *HyperPlane_Interface* to notify it that it has completed reading in the new header. When the header read is done, the *HyperPlane_Interface* sets the *SRAM_read_done* signal to '1' (Figure 6.10, point (v)) and the *CAM_Read_FSM* can now release the SRAM address data that was appearing on *CAM_data_out(15..0)* (Figure 6.10, point (vi)).

Finally, for completeness, functional flow charts for the *CAM_Write_FSM* and the *CAM_Read_FSM* are shown in Figures 6.11 and 6.12.

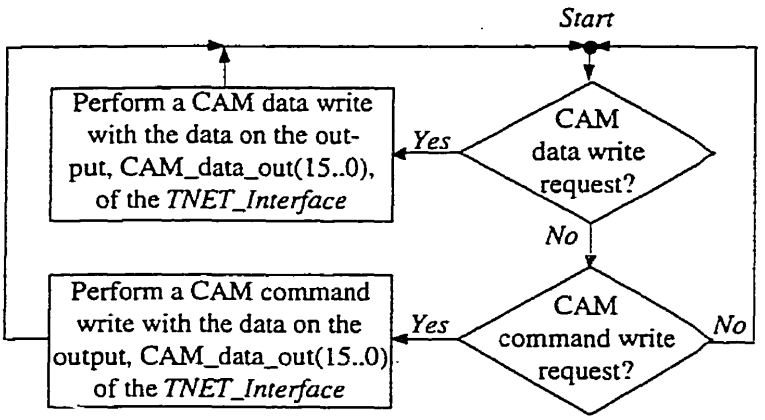


Figure 6.11: Functional flow chart for the *CAM_Write_FSM*.

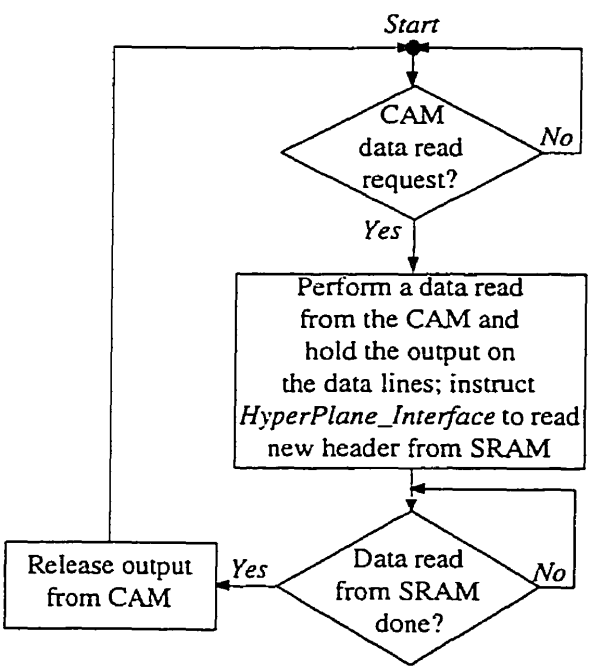


Figure 6.12: Functional flow chart for the *CAM_Read_FSM*.

6.3.3 The *IO_Control_Unit*

6.3.3.1 Introduction

The second main component of the *TNET_Interface* is the *IO_Control_Unit*. The floorplan of the *TNET_Interface* was shown in Figure 6.6. A block diagram of the *IO_Control_Unit* is shown in Figure 6.13. Through commands received from the *Main_Control_Unit*, the *IO_Control_Unit* is responsible for controlling the byte-by-byte transfer of ATM cells from the SONET/ATM transceiver to the internal *FIFO5* queue of the *TNET_Interface* (Figure 6.6) and also to the external 64x9 FIFO. The three main VHDL components of the *IO_Control_Unit* are the *IO_Control_FSM*, the *LFSR6* component, and the *LFSR3* component. The *LFSR6* and the *LFSR3* components are 6-bit and 3-bit Linear Feedback Shift Registers (LFSR), respectively. These two components will be discussed shortly. The purpose of the two LFSRs is to act as counters. The following two sections will discuss the *IO_Control_FSM* and the two LFSRs.

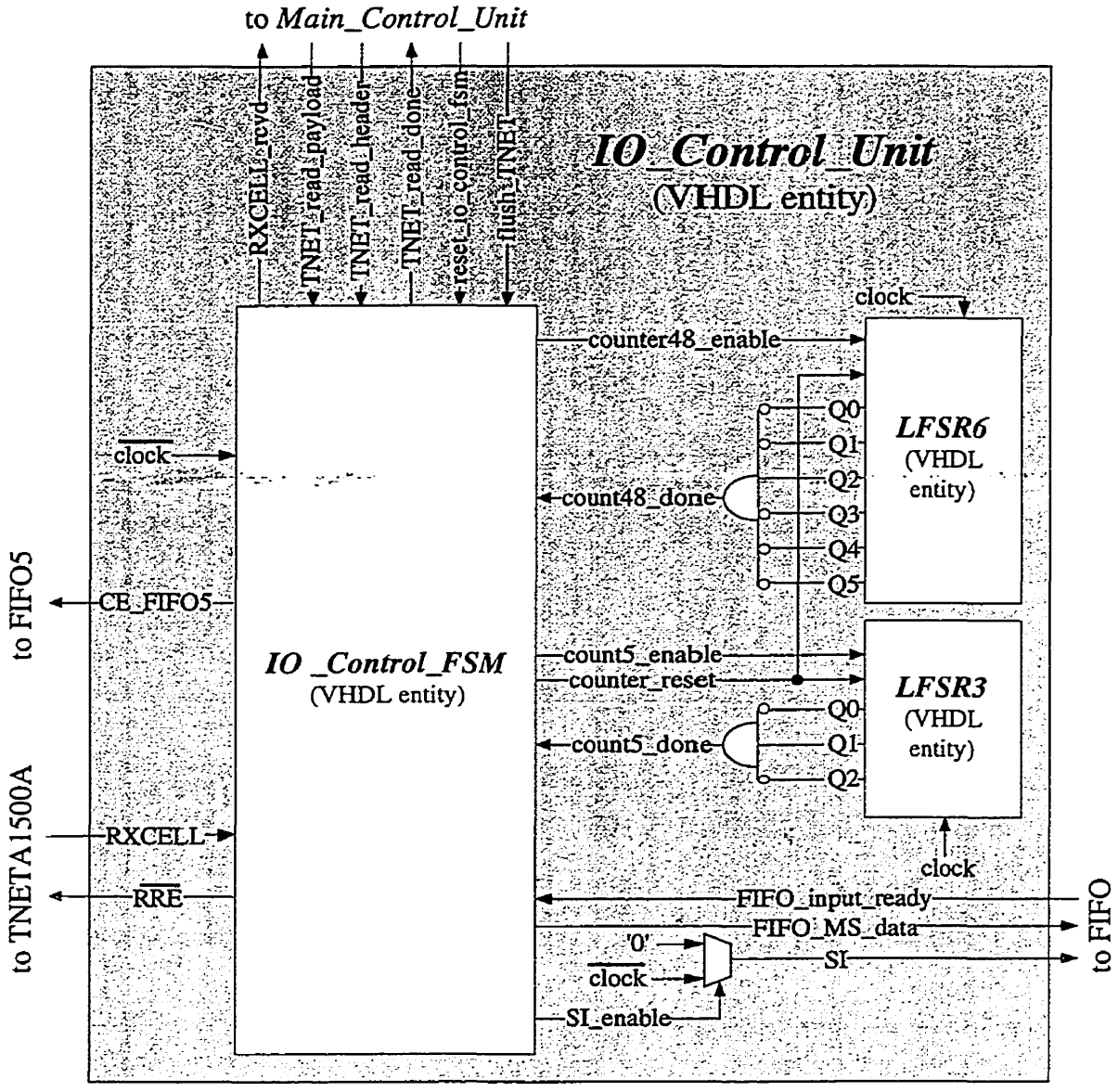


Figure 6.13: Block diagram of the *IO_Control_Unit*.

6.3.3.2 The *IO_Control_FSM*

The *IO_Control_FSM* is responsible for controlling the transfer of ATM cells from the SONET/ATM transceiver to the *FIFO5* queue of the *TNET_Interface* (Figure 6.6) or to the external 64x9 FIFO queue. If an ATM cell does not appear on the output port *RD(7..0)* of the SONET/ATM transceiver, the *IO_Control_FSM* is then responsible for flushing the internal receive queue of the SONET/ATM transceiver until the start of an ATM cell appears on *RD(7..0)*.

The *IO_Control_FSM* is a 9-state finite state machine with 21 transitions and has 9 inputs and 9 outputs. A flow chart outlining the operation of the *IO_Control_FSM* is shown in Figure 6.14.

In Figure 6.14, the *IO_Control_FSM* first checks to see if *RXCELL* = '1'. If this is not true then the *IO_Control_FSM* will wait for the *Main_Control_Unit* to instruct it to flush the SONET/ATM transceiver internal receive queue until the start of an ATM cell appears on the data lines, *RD(7..0)*. The *Main_Control_Unit* instructs the *IO_Control_FSM* to begin flushing by setting the *flush_TNET* signal to '1' (point (i) in Figure 6.14). Since the *IO_Control_FSM* is always waiting for instructions from the *Main_Control_Unit*, the *Main_Control_Unit* may be seen as the master component while the *IO_Control_Unit* may be seen as the slave component.

Once the start of an ATM cell appears on *RD(7..0)*, the *IO_Control_FSM* informs the *Main_Control_Unit* of this event and then proceeds to wait for the *Main_Control_Unit* to instruct it to read the header (point (ii) in Figure 6.14). When the *Main_Control_Unit* sets the *TNET_read_header* signal to '1', the *IO_Control_FSM* begins to read the 5-byte header from the TNETA1500A transceiver and loads this header into the internal *FIFO5* queue (point (iii) in Figure 6.14). The *IO_Control_FSM* uses the *LFSR3* counter to notify itself when all 5-bytes have been written to *FIFO5*. After the header has been loaded, the *IO_Control_FSM* informs the *Main_Control_Unit* that it is done and waits again for the *Main_Control_Unit* to instruct it to read the 48-byte ATM payload (Figure 6.14). The *Main_Control_Unit* instructs the *IO_Control_FSM* to read the payload by setting the *TNET_read_payload* signal to '1' (point (iv) in Figure 6.14). When this event occurs, the *IO_Control_FSM* first verifies that the external 64x9 FIFO is ready to accept data. Once the external 64x9 FIFO is ready, the *IO_Control_FSM* commences the transmission of the ATM payload from the SONET/ATM transceiver to the external FIFO (Figure 6.14). The *IO_Control_FSM* uses the *LFSR6* counter to notify itself when all 48-bytes have been written to the external FIFO. It should be noted that the external FIFO is 9-bits wide so that the *HyperPlane_Interface* can identify the start of a 48-byte payload. When the first byte of an ATM payload is written to the external 64x9 FIFO, the *IO_Control_FSM* sets the most significant bit (the ninth bit) to one. This bit is set to zero for all other bytes in the ATM payload. When the entire payload has been written to the external FIFO, the *IO_Control_FSM* returns to the top of Figure 6.14 and the process begins anew.

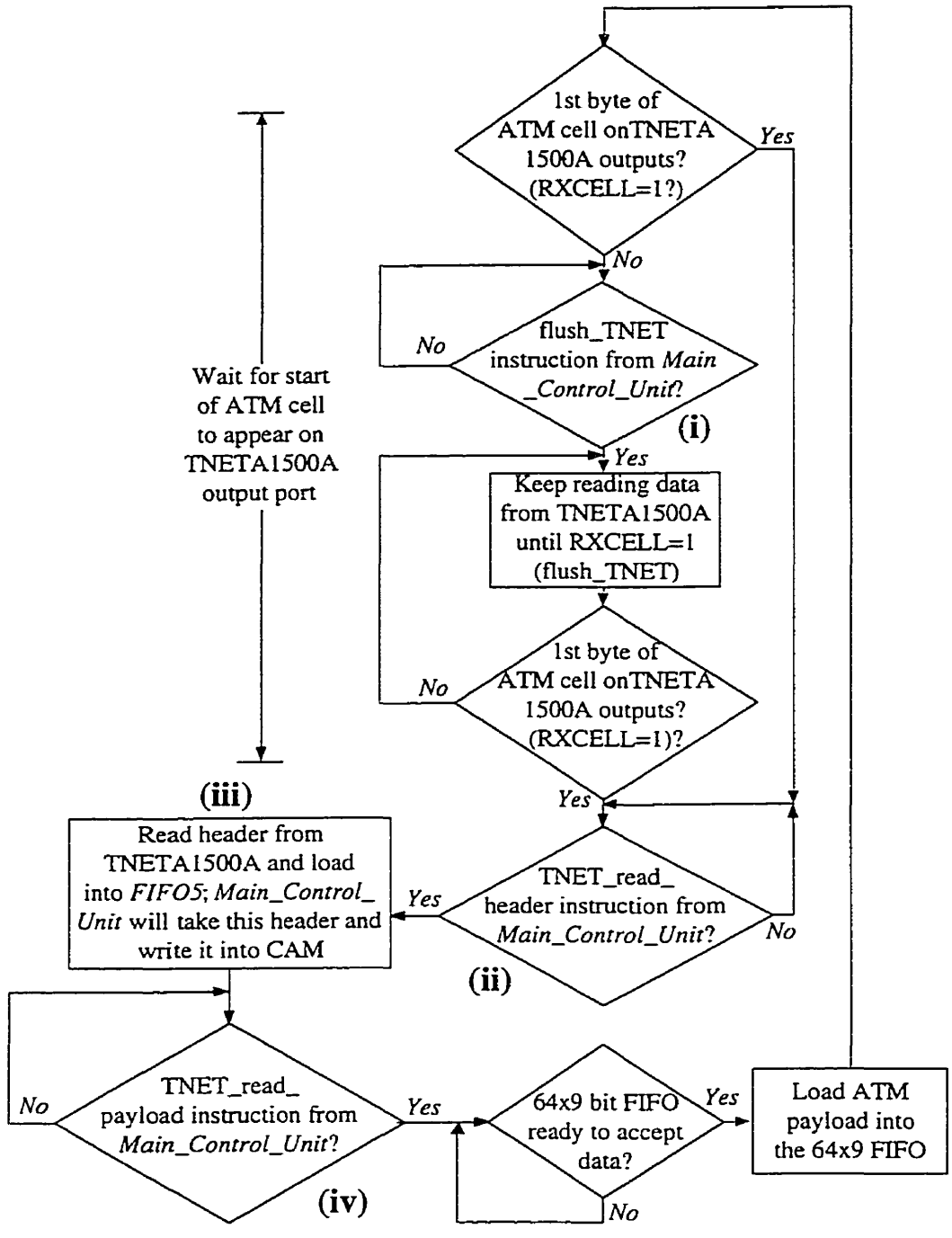


Figure 6.14: Functional flow chart for the *IO_Control_FSM*.

6.3.3.3 Linear Feedback Shift Register Counters

All counters that are used on the *ATMCard* are implemented as *Linear Feedback Shift Registers* (LFSR). An LFSR counter differs from conventional binary counters in their high speed and simple structure, which is gained at the expense of sacrificing the

binary count sequence [ALF94], [ALF95]. The size and speed advantages that are inherent in LFSR counters make them ideal for high speed telecommunication applications such as the *ATMCard*. An *n*-bit LFSR counter is capable of having a maximum sequence length of $(2^n - 1)$. An *n*-bit LFSR is simply an *n*-bit shift register with an XNOR gate in the feedback path from selected outputs of the shift register to the first input, D_0 (Figure 6.15). The *ATMCard* uses a 3-bit LFSR (*LFSR3*) and a 6-bit LFSR (*LFSR6*) enabling counting operations up to the number 63. In order for the two LFSRs to function, outputs Q_1 and Q_2 must be fed to the inputs of the XNOR gate of *LFSR3*, while outputs Q_4 and Q_5 must be fed to the XNOR inputs of LFSR6 (Figure 6.15). These configurations guarantee a pseudo-random count having the maximum sequence length of $(2^n - 1)$.

The outputs Q_x, $x = 0,1,...,n-1$, of an *n*-bit LFSR are used by components in the *ATMCard* to determine when a count to a certain number has been reached. For example, in the counting sequence followed by LFSR3 the number five is represented by Q_0='0', Q_1='1', and Q_2='0'. Therefore, a signal that alerts a component that a count to five has been reached (such as the *count5_done* signal of Figure 6.13) will be the output of a 3-input AND gate that has as its inputs the signals Q_0='0', Q_1='1', and Q_2='0'.

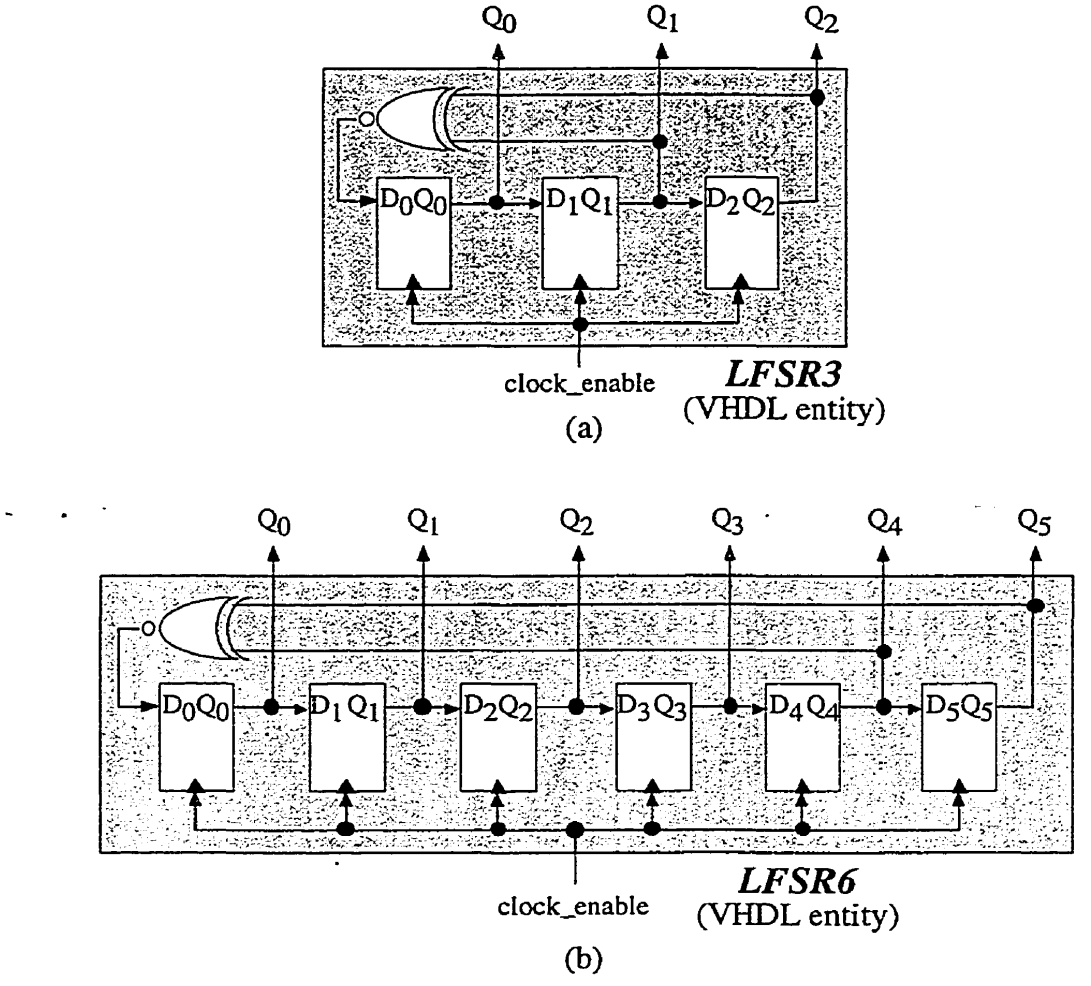


Figure 6.15: Linear feedback shift register (LFSR) counters with maximum sequence length of $(2^n - 1)$. (a) A 3-bit LFSR counter (*LFSR3*) and (b) a 6-bit LFSR counter (*LFSR6*) are shown (clock and reset signals are not shown) [ALF94].

6.3.4 The *CAM_Init_Unit*

6.3.4.1 Introduction

The third and final VHDL component of the *TNET_Interface* which was shown in Figure 6.6 is the *CAM_Init_Unit* (Figure 6.16). The *CAM_Init_Unit* is composed of the *CAM_Init_FSM* and the *CAM_Write_FSM*. When powering up the *ATMCard*, the *CAM_Init_Unit* is responsible for the initialization and configuration of the LANCAM. Once its task is complete, the *CAM_Init_Unit* passes control to the *Main_Control_Unit* so

that normal *TNET_Interface* operations may proceed. At this point, the *CAM_Init_Unit* remains dormant. The following section will discuss the *CAM_Init_FSM* component. Since the *CAM_Write_FSM* has already been described in section 6.3.2.3, it will not be discussed here.

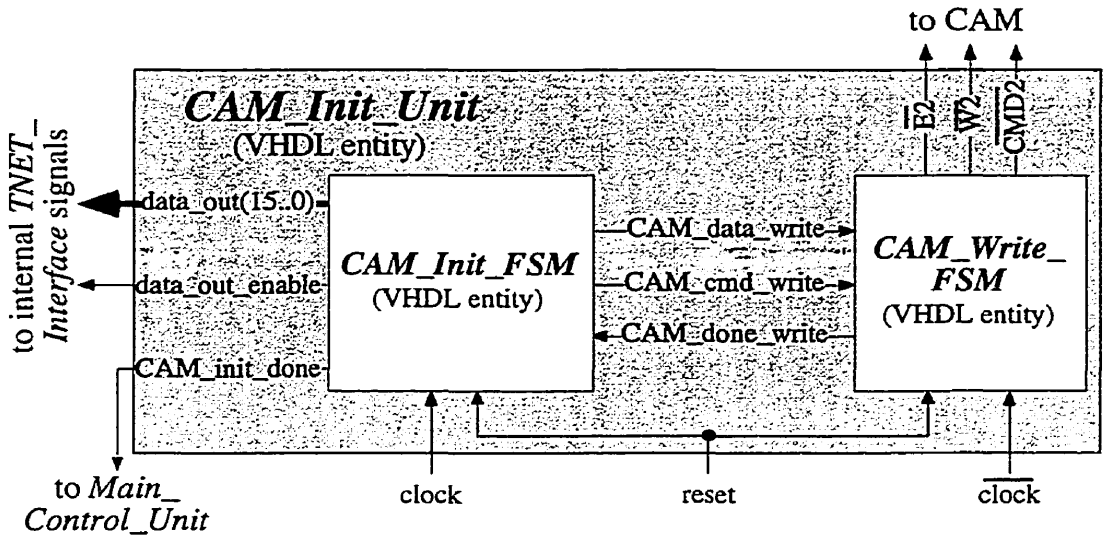


Figure 6.16: Block diagram of the *CAM_Init_Unit*.

6.3.4.2 The *CAM_Init_FSM*

The CAM write control signals are managed by the *CAM_Write_FSM*. During CAM initialization and configuration, the *CAM_Write_FSM* is, in turn, controlled by the *CAM_Init_FSM*. The *CAM_Init_FSM* also controls what is to be written to the CAM and whether the write actions are to be CAM data writes or CAM command writes.

The CAM is configured by the *CAM_Init_FSM* to enable the 5-byte ATM header comparison cycles that were described previously. To recapitulate, four bytes of the 5-byte header (the HEC field is not used) are written to the CAM comparand register by the *Main_Control_Unit*. The CAM then proceeds to search its memory contents for a field matching these four bytes. In the case of a match, the *Main_Control_Unit* reads from the CAM the 2-byte data that is associated with this matching field. These two bytes comprise the address where the new ATM header can be found in the 256K SRAM.

Since the LANCAM manipulates data as 64-bit words (8 bytes), the CAM memory is partitioned so that the most significant 6 bytes act as CAM fields and the least significant 2 bytes serve as the associated RAM fields. Therefore, during CAM comparison cycles

only the most significant 6 bytes of the CAM memory are searched. This partitioning, however, is not enough. The LANCAM allows a user to mask CAM memory bits during comparisons. The *CAM_Init_FSM* masks the most significant 16 bits (2 bytes) of the 6-byte CAM memory field leaving only 4-bytes which are used during CAM comparison cycles. These 4-bytes correspond to the 4-bytes of the ATM header that are used during comparisons. Similarly, the 2-byte SRAM address pointer is stored in each 2-byte RAM field of the CAM memory, allowing the address pointer to be output during a CAM comparison match.

The instruction sequence that the *CAM_Init_FSM* uses to configure and initialize the MU9C1480 CAM also partitions and masks the CAM memory so that all the above actions may occur. Table 6.1 illustrates the command sequence that is followed by the *CAM_Init_FSM* in order to properly initialize and configure the CAM. It should be noted that the CAM *Instruction Register* (IR) is the default destination of all command writes (CW) to the CAM. In order to write to other CAM control registers, a *Temporary Command Overwrite* (TCO) instruction must be written to the *Instruction Register* (IR) telling the CAM which control register is to be written. The next command write after the TCO instruction will be destined for the new control register. Similarly, the CAM Comparand Register is the default destination of all CAM data writes (DW). In order to perform a data write to the CAM memory or the Mask Registers, a Select Persistent Destination (SPD) instruction must be written to the Instruction Register. Once the new destination has been chosen with the SPD instruction, it will become the default destination for all subsequent data writes until another SPD instruction has been made [MUS94].

Command Abbreviation	*Description*
1) CW TCO CT	Command write (CW) the Temporary Command Overwrite (TCO) instruction. This command makes the next CW instruction write to the Control Register (CT). The default destination for command writes is the Instruction Register.
2) CW 0000h	Command write the value 0000h to the CT register. This resets the device and accounts for power up anomalies.
3) CW TCO CT	Same command as 1.
4) CW FE5Ch	Command write FE5Ch to the CT register. Writing this value to the CT register configures the lowest 16 bits of a CAM memory location as a RAM field and the remaining 48 bits as CAM fields. Therefore, during comparisons only the most significant 48 bits are used. Also, this command masks all comparisons with the contents of Mask Register 1 (MR1).
5) CW SPD MR1	Command write the Select Persistent Destination (SPD) instruction. This command makes all subsequent data writes go to the selected destination. In this case, the selected destination is Mask Register 1 (MR1).
6) DW 0000h	Data write the value 0000h to the lowest significant 16 bits of the 64-bit MR1.
7) DW 0000h	Data write the value 0000h to the second lowest significant 16 bits of MR1.
8) DW 0000h	Data write the value 0000h to the third lowest significant 16 bits of MR1.
9) DW FFFFh	Data write the value FFFFh to the most significant 16 bits of MR1. Writing all ones to the most significant 16 bits means that these bits are masked. All CAM comparisons will only use the least significant 4 bytes of an 6 byte CAM field.
10) CW SPD CR	Select the Comparand Register (CR) as the persistent destination for data writes. After the CAM is initialized and configured, all data writes from the *Main_Control_Unit* will go to CR for comparison operations.
11) CW TCO SC	Select the Segment Control (SC) Register as the destination for the next command write instruction.
12) CW 1024h	Write the value 1024h to the SC Register. Writing this value configures the CAM so that only the least significant 48 bits (4 bytes) of a 64-bit word are written to the comparand register during comparisons (the most significant 16 bits are masked). Also, during all data reads only the least significant 16 bits (2 bytes) are read and the remaining 48 bits are ignored. The 16 bit field that is read is the SRAM address pointer.
13) CW SPS M@HPM	Command write the Select Persistent Source (SPS) instruction. This command makes all subsequent data read cycles read from the memory location where the highest priority match (HPM) from a comparison occurred.

Table 6.1: *CAM_Init_FSM* configuration and initialization command sequence. (CW = command write; DW = data write)

CAM command abbreviations are shown in Table 6.1. For abbreviations involving CAM data writes (DW), the 16-bit hexadecimal field represents the 16-bit value appearing on the CAM data ports that will be written into the CAM. For example, "DW 0000h" signifies that the 16-bit hexadecimal value 0000h will be written into the CAM during a data

write cycle. The command write (CW) TCO, SPD, and SPS instructions shown in the first column of Table 6.1 are 16-bit instructions appearing on the CAM data ports that are written into the CAM during command write cycles [MUS94].

This section completes the description of the *TNET_Interface* component and its three main subcomponents - the *Main_Control_Unit*, the *IO_Control_Unit*, and the *CAM_Init_Unit*. The next section will discuss the *HyperPlane_Interface* and its subcomponents shown in Figure 6.5.

6.4 The *HyperPlane_Interface*

6.4.1 Introduction

The *HyperPlane_Interface* works in conjunction with the *TNET_Interface* in order to perform ATM cell header translation and in order to transmit ATM cells from the SONET/ATM transceiver to the *HyperPlane*. An overview of the *HyperPlane* was presented in Chapter 3. This section will discuss the *HyperPlane_Interface* VHDL component. A detailed block diagram of the *HyperPlane_Interface* is illustrated in Figure 6.17. The *HyperPlane_Interface* is comprised of two main components - the *SRAM_Control_Unit* and the *SPA_TX_Control_Unit* - in addition to a 6x8-bit FIFO (*FIFO6*), multiplexers, and an OR gate. The overall VHDL hierarchical organization of the *HyperPlane_Interface* was shown in Figure 6.5.

There are two main functions that the *HyperPlane_Interface* serves. Through its *SRAM_Control_Unit* subcomponent, the *HyperPlane_Interface* is responsible for reading the new 6-byte ATM header from the external 256K SRAM and writing this header into the internal *FIFO6* queue (Figure 6.17). During ATM header translation, the new header is 6-bytes long because an extra byte has been prepended. This prepended byte is used internally by the *HyperPlane* to direct the ATM cell to the destined *ATMCard*. The other subcomponent - the *SPA_TX_Control_Unit* - is responsible for the second main function of the *HyperPlane_Interface*. This function is the transmission of the ATM header from the *FIFO6* queue to the *HyperPlane* and also the transmission of the ATM payload from the external 64x9 FIFO to the *HyperPlane*.

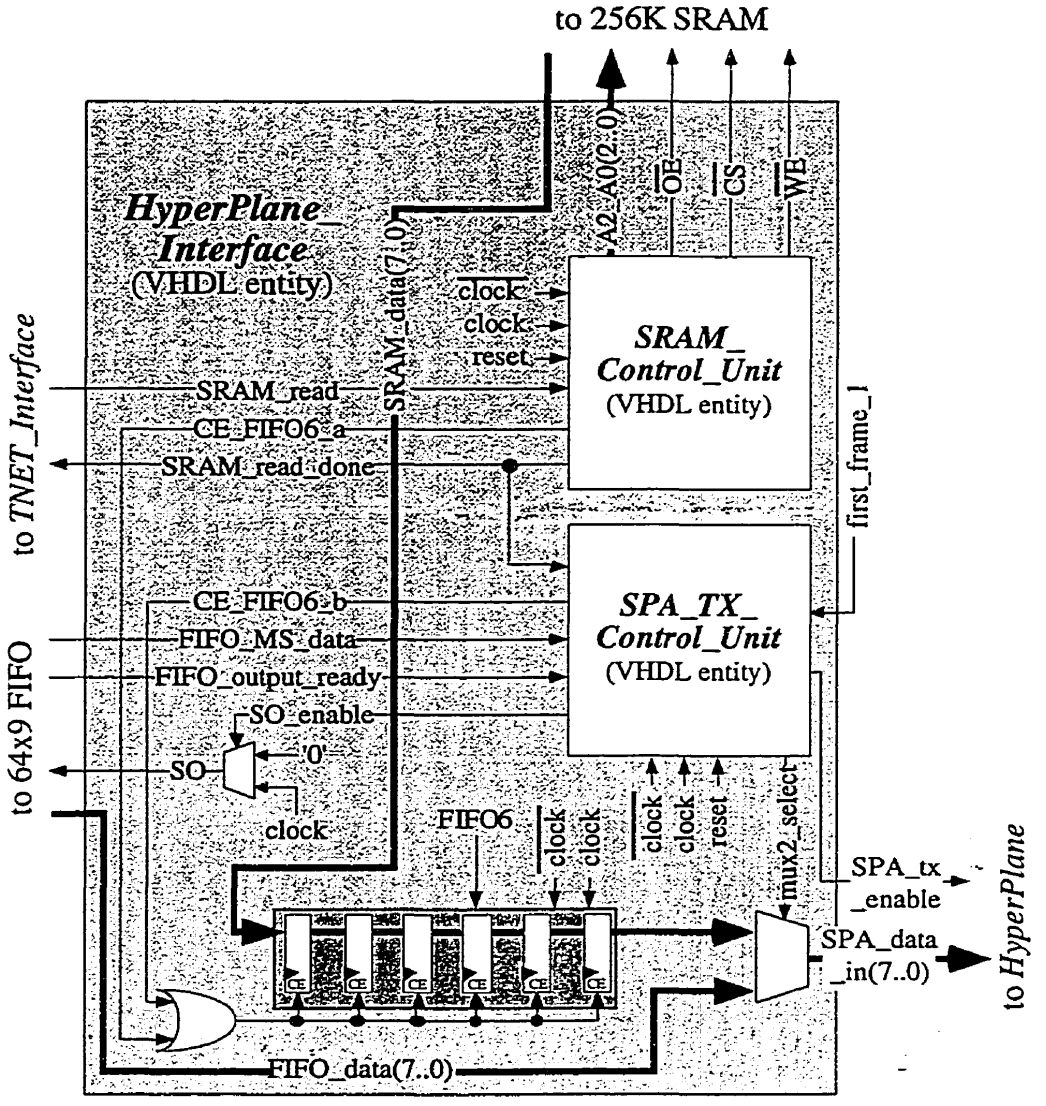


Figure 6.17: Block diagram of the *HyperPlane_Interface*.

A description of the procedures used by the *HyperPlane_Interface* during ATM header translation and ATM cell transmission to the *HyperPlane* will be given. The reader is asked to refer to Figure 6.3 during this discussion. Once a matching SRAM address field has been found within the CAM memory (Figure 6.3, label (iii)), the *TNET_Interface* instructs the *SRAM_Control_Unit*, via the *SRAM_read* signal (Figure 6.17), to read the new 6-bit ATM header from the SRAM. Upon receiving an instruction to read from the SRAM (*SRAM_read* = '1'), the *SRAM_Control_Unit* proceeds to read the 6-byte header from the SRAM and loads this header into the internal *FIFO6* queue of the *HyperPlane_Interface* (Figure 6.3, label (v)). The *SPA_TX_Control_Unit* may then transmit this 6-byte header together with the ATM payload received from the external 64x9 FIFO only when a special

clock signal, *first_frame_1* (Figure 6.17), is equal to a logical value of '1'. Since the *HyperPlane* functions on a time-division multiplexed (TDM) scheme, it is imperative that the *SPA_TX_Control_Unit* transmit the ATM cell only at the beginning of a *HyperPlane* frame (signaled by *first_frame_1* = '1'). When *first_frame_1* = '1' the *SPA_TX_Control_Unit* first transmits the 6-byte header from *FIFO6* to the *HyperPlane*, and then the ATM payload from the external 64x9 payload (Figure 6.3, label (vi)).

This section has provided a brief description of the *HyperPlane_Interface*. The following sections will discuss the two main components of the *HyperPlane_Interface*: the *SRAM_Control_Unit* and the *SPA_TX_Control_Unit*.

6.4.2 The *SRAM_Control_Unit*

6.4.2.1 Introduction

The first main VHDL component of the *HyperPlane_Interface* is the *SRAM_Control_Unit*. The block diagram for the *SRAM_Control_Unit* is illustrated in Figure 6.18. The *SRAM_Control_Unit* is responsible for reading the new 6-byte ATM header from SRAM and loading it into the internal 6x8-bit FIFO (*FIFO6*) of the *HyperPlane_Interface*. Upon completion of this task, the *SRAM_Control_Unit* then transfers control to the *SPA_TX_Control_Unit* of the *HyperPlane_Interface* (Figure 6.17), which transmits the 6-byte translated ATM header along with the 48-byte ATM payload into the *HyperPlane*. The main VHDL components of the *SRAM_Control_Unit* are the *SRAM_Control_FSM* and the 3-bit LFSR counter (*LFSR3*). The following section will discuss the *SRAM_Control_FSM*. The 3-bit LFSR counter will not be described in this section since it was previously discussed.

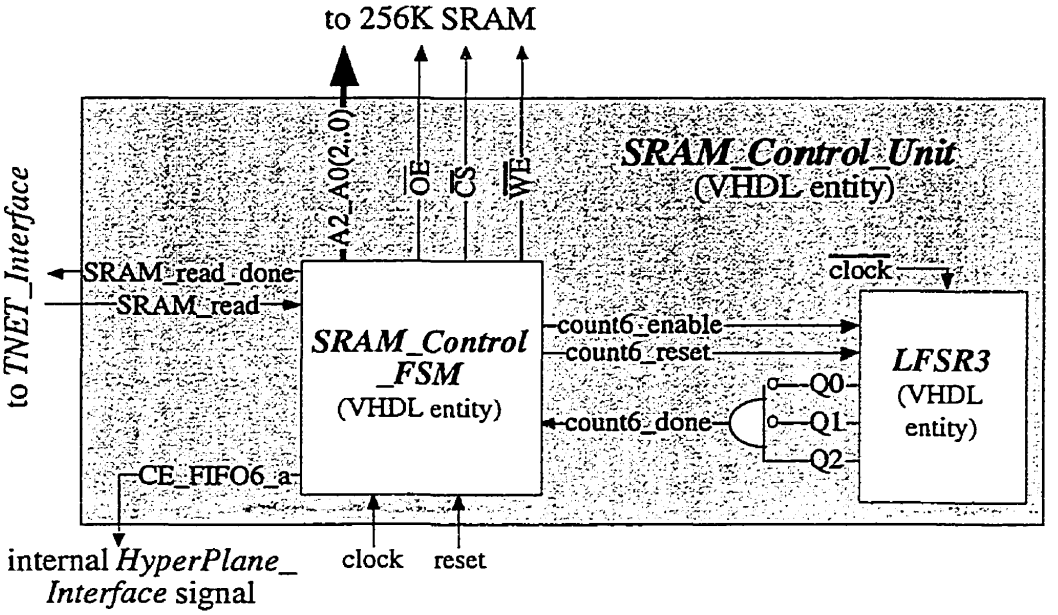


Figure 6.18: Block diagram of the *SRAM_Control_Unit*.

6.4.2.2 The *SRAM_Control_FSM*

The *SRAM_Control_FSM* is responsible for reading the 6-byte translated ATM header and loading it into the internal 6x8 FIFO (*FIFO6*) of the *HyperPlane_Interface*. The SRAM address that the *SRAM_Control_FSM* should read from is obtained from the CAM after a match has occurred. Once the 6-byte header has been written into *FIFO6*, the *SRAM_Control_FSM* passes control over to the *SPA_TX_Control_FSM*.

The *SRAM_Control_FSM* is a 6-state finite state machine (FSM) with 8 transitions between states and it has 4 input ports and 10 output ports. A flow chart outlining the operation of the *SRAM_Control_FSM* is illustrated in Figure 6.19.

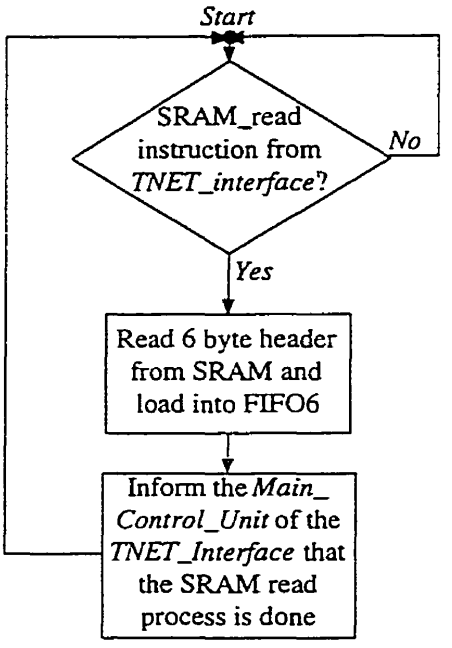


Figure 6.19: Functional flow chart for the *SRAM_Control_Unit*.

The operation of the *SRAM_Control_Unit* is very simple. Beginning at the top of Figure 6.19, the *SRAM_Control_Unit* waits for an SRAM read instruction from the *TNET_Interface* (*SRAM_read* ='1'). When the instruction arrives the *TNET_Interface* has ensured that the proper address has been sent to the SRAM. This address originates from the associated data field of the CAM memory. It points to the SRAM address where the start of the new ATM header is located. Therefore, the *SRAM_Control_Unit* need only concern itself with sequentially reading the new 6-byte ATM header from the SRAM and loading this header into the *FIFO6* queue of the *HyperPlane_Interface*. Once it has finished loading *FIFO6*, the *SRAM_Control_Unit* must notify the *TNET_Interface* that it is done so that the *Main_Control_Unit* of the *TNET_Interface* may terminate the CAM read cycle. At this point, the *SRAM_Control_Unit* proceeds back to the top of Figure 6.19 and waits for the next SRAM read instruction.

6.4.3 The *SPA_TX_Control_Unit*

6.4.3.1 Introduction

The second main VHDL component of the *HyperPlane_Interface* is the *SPA_TX_Control_Unit*. Figure 6.20 illustrates the block diagram of the *SPA_TX_Control_Unit*. After ATM header translation has been performed, the *SPA_TX_Control_Unit* is responsible for the transmission of these translated ATM cells to the *HyperPlane*. To accomplish this transfer, the *SPA_TX_Control_Unit* first transmits the 6-byte ATM header stored in *FIFO6* to the *HyperPlane* (Figure 6.17). It then transmits the ATM payload from the external 64x9 FIFO queue to the *HyperPlane*. These transmissions must be performed in a synchronous manner so that the ATM cell is inserted into a valid frame of the *HyperPlane*.

The main VHDL components of the *SPA_TX_Control_Unit* are the *SPA_TX_Control_FSM* and the *LFSR3* and *LFSR6* counters (Figure 6.20). The following section will discuss the *SPA_TX_Control_FSM*. The LFSR counters will not be described in this section since they were described previously.

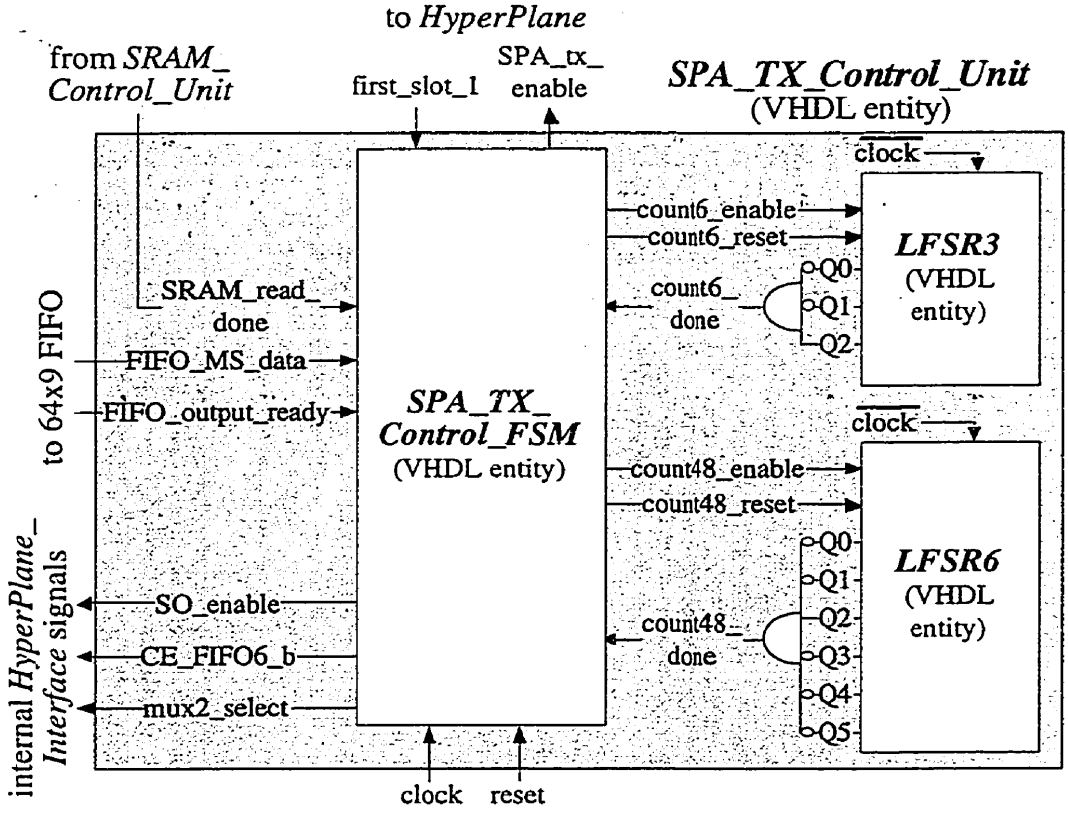


Figure 6.20: Block diagram of the *SPA_TX_Control_Unit*.

6.4.3.2 The *SPA_TX_Control_FSM*

The *SPA_TX_Control_FSM* is responsible for transmitting an ATM cell to the *HyperPlane*. An additional byte has been prepended to the ATM header. This prepended byte is used internally by the *HyperPlane* in order to identify the correct destination *ATMCard*. In order to transmit the ATM cell, the *SPA_TX_Control_FSM* first reads the 6-byte header from *FIFO6* (Figure 6.17) and sends the header to the *HyperPlane*. Immediately after the header has been sent, the *SPA_TX_Control_FSM* transmits the 48-byte ATM payload from the external 64x9 FIFO queue to the *HyperPlane*.

The *SPA_TX_Control_FSM* is a 6-state finite state machine with 13 transitions between states and it has 8 input ports and 8 output ports. A flow chart outlining the operation of the *SPA_TX_Control_FSM* is illustrated in Figure 6.21.

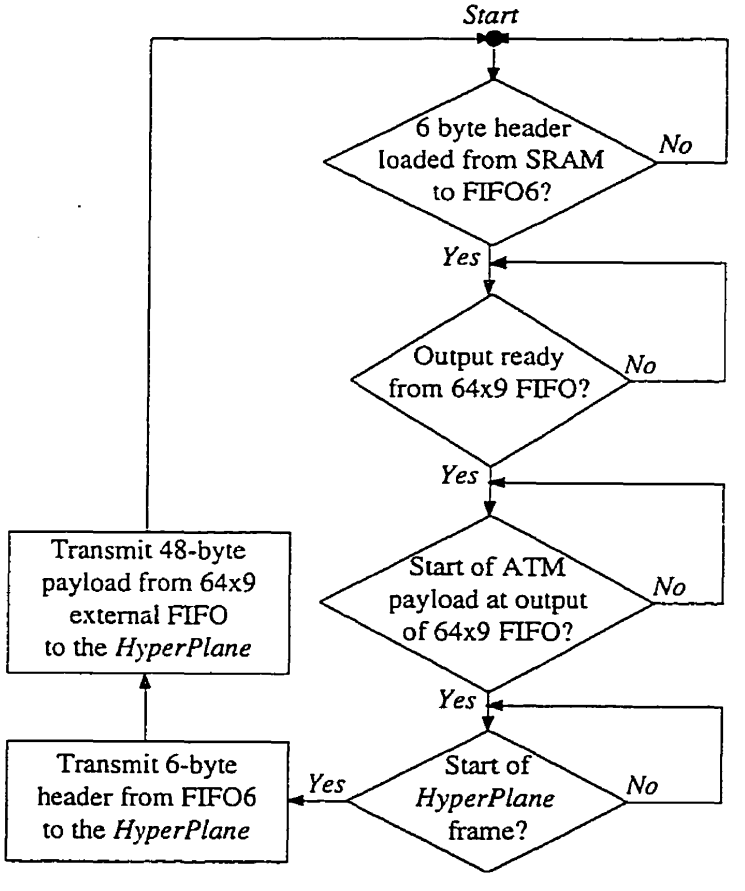


Figure 6.21: Functional flow chart for the *SPA_TX_Control_Unit.*

Before the *SPA_TX_Control_Unit* is able to send an ATM cell to the *HyperPlane*, it must wait for the *SRAM_Control_Unit* to complete its loading of *FIFO6* with the new 6-byte ATM header (top of Figure 6.21). When the loading is complete, the *SRAM_Control_Unit* informs the *SPA_TX_Control_FSM* of this event by setting the *SRAM_read_done* signal to a logical value of '1' (Figure 6.20). Once this signal is set to true, the *SPA_TX_Control_FSM* waits for the first byte of the 48-byte ATM payload to appear on the output port of the external 64x9 FIFO. When *FIFO_MS_data* is asserted, the first byte of the ATM payload is on the output port (Figure 6.20).

Finally, before transmission of an ATM cell to the *HyperPlane* can begin, the *SPA_TX_Control_FSM* must wait for the start of a frame. Recall that data sent across the *HyperPlane* is time-division multiplexed. Therefore, it is important for the *SPA_TX_Control_FSM* to begin transmission of an ATM cell at the start of a frame. In this case, the *SPA_TX_Control_FSM* may begin ATM cell transmission only at the start of a *HyperPlane* time frame (Figure 6.22). At this point, the *SPA_TX_Control_FSM* begins transmitting the 6-byte header from *FIFO6* to the *HyperPlane*. The *SPA_TX_Control_FSM* uses the LFSR counter, *LFSR3*, to notify it when the entire 6-byte header has been sent and when it should begin transmitting the 48-byte payload from the external 64x9 FIFO to the *HyperPlane*. Similarly, the *SPA_TX_Control_FSM* uses the *LFSR6* counter to signal the end of the 48-byte payload. When the *LFSR6* counter has reached the value of 48, the *SPA_TX_Control_FSM* stops and proceeds to the top of Figure 6.21 to await the transmission of the next ATM cell.

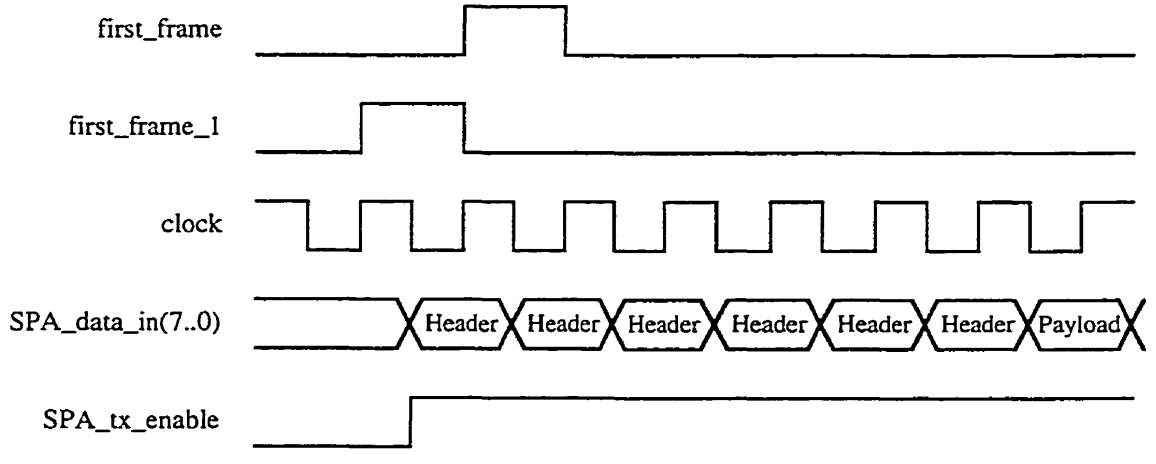


Figure 6.22: Timing diagram showing the transmission of an ATM cell to the *HyperPlane* [SZY94a].

This section completes the description of the *HyperPlane_Interface* VHDL component and its two main subcomponents - the *SRAM_Control_Unit* and the *SPA_TX_Control_Unit*. The next section will describe the final component which is the *TNET_TX_Interface*.

6.5 The *TNET_TX_Interface*

6.5.1 Introduction

While the *TNET_Interface* and the *HyperPlane_Interface* together are responsible for the transmission of ATM cells from the SONET/ATM transceiver to the *HyperPlane*, the *TNET_TX_Interface* is solely responsible for the transmission in the reverse direction, that is, from the *HyperPlane* to the SONET/ATM transceiver. This final section of this chapter will describe the *TNET_TX_Interface*. A detailed block diagram of the *TNET_TX_Interface* is shown in Figure 6.23. The *TNET_TX_Interface* is very simple compared with its counterparts the *TNET_Interface* and the *HyperPlane_Interface*, since its sole task of transmitting ATM cells from the *HyperPlane* to the SONET/ATM transceiver is much simpler. Also, unlike the *TNET_Interface* or the *HyperPlane_Interface*, the *TNET_TX_Interface* does not utilize any external memory components.

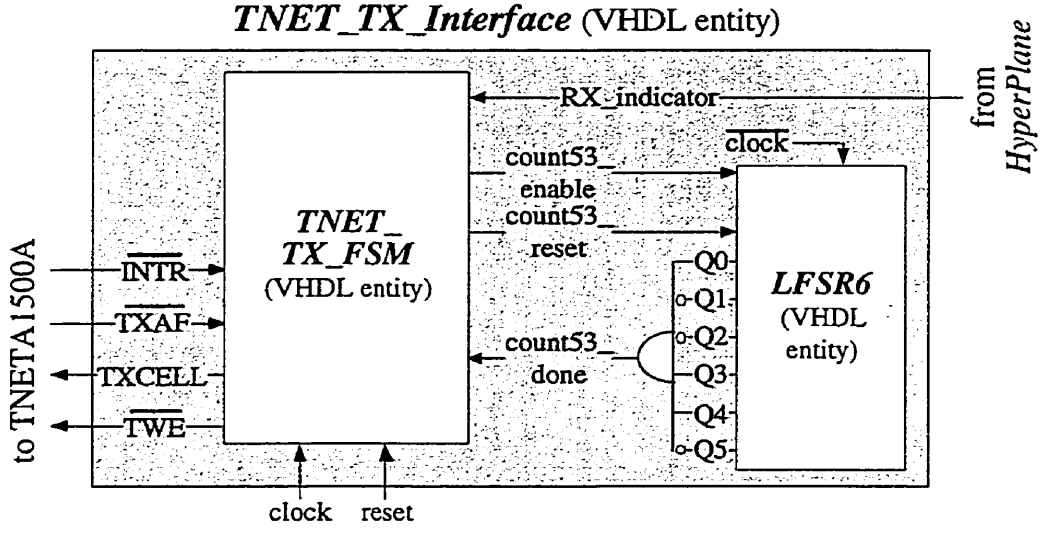


Figure 6.23: Block diagram of the *TNET_TX_Interface*.

The only task that the *TNET_TX_Interface* must perform is the transmission of ATM cells it receives from the *HyperPlane* to the SONET/ATM transceiver (Figure 6.3 (f), label (vii)). Before transmitting a cell, however, the *TNET_TX_Interface* must remove the prepended header byte that was used internally by the *HyperPlane*. Once this byte has been removed, the *TNET_TX_Interface* can send the entire 53-byte ATM cell to the SONET/ATM transceiver. Figure 6.23 shows that the two main components of the

TNET_TX_Interface are the *TNET_TX_FSM* and the *LFSR6* counter. Since LFSR counters have been discussed previously, the next section will only describe the *TNET_TX_FSM*.

6.5.2 The *TNET_TX_FSM*

The *TNET_TX_FSM* is responsible for transmitting ATM cells received from the *HyperPlane* to the SONET/ATM transceiver. Before the transmission, however, the *TNET_TX_FSM* must remove the prepended header byte that was used by the *HyperPlane*.

The *TNET_TX_FSM* is a 6-state finite state machine with 13 transitions between states and it has 6 input ports and 4 output ports. A flow chart outlining the operation of the *TNET_TX_FSM* is illustrated in Figure 6.24.

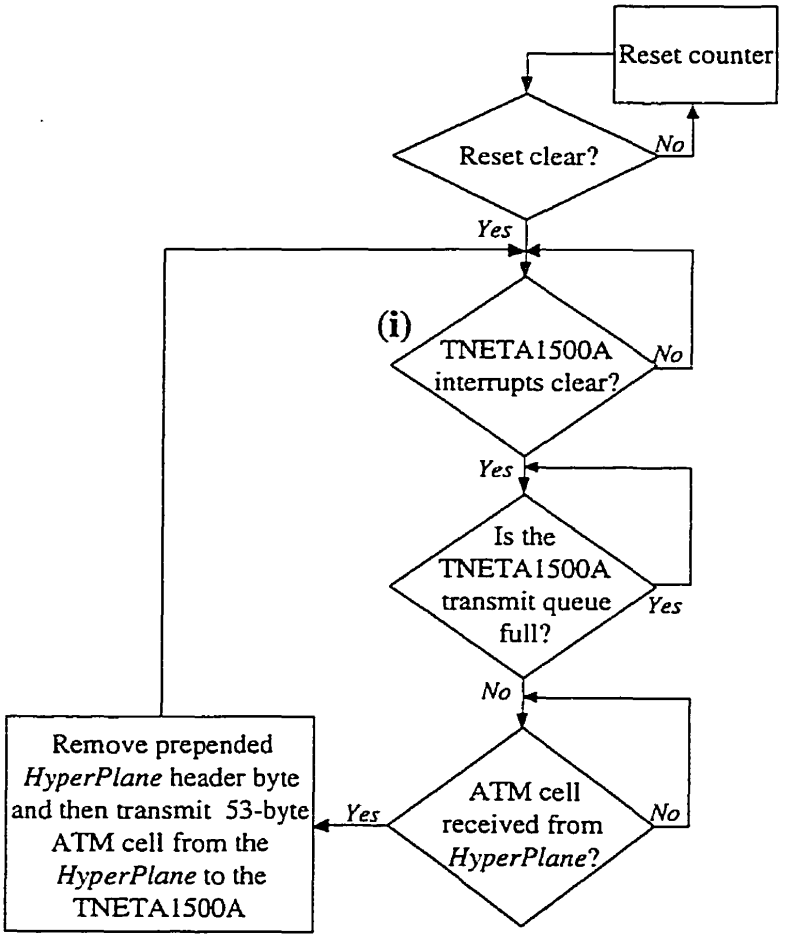


Figure 6.24: Functional flow chart for the *TNET_TX_FSM*.

Before any ATM cell transmission can occur, the *TNET_TX_FSM* must first verify that the SONET/ATM transceiver is operational. First, it waits until all SONET/ATM transceiver interrupts are clear (point (I) in Figure 6.24). Next, if the SONET/ATM transceiver internal transmit queue is full then it cannot accept any additional ATM cells. The *TNET_TX_FSM* waits until there is room within the SONET/ATM transceiver internal transmit queue for an additional 53-byte ATM cell (Figure 6.24). Once it has been verified that the SONET/ATM transceiver can receive ATM cells, the *TNET_TX_FSM* waits for the arrival of an ATM cell from the *HyperPlane*. When the *RX_indicator* signal is equal to '1' (Figure 6.23), it means that an ATM cell has arrived from the *HyperPlane*. At this point, the *TNET_TX_FSM* discards the first prepended header byte. It then proceeds to transmit the rest of the 53-bytes of the ATM cell to the SONET/ATM transceiver. The *LFSR6* counter (Figure 6.23) is used to notify the *TNET_TX_FSM* when all 53-bytes have been transmitted.

6.6 Summary

This chapter provided a functional and structural description of the *ATMCard*. In particular, the three main components of the *ATMCard* - the *TNET_Interface*, the *HyperPlane_Interface*, and the *TNET_TX_Interface* - and their subcomponents were described in a top-down fashion (Figure 6.5). The following chapter will discuss the actual implementation of the *ATMCard*.

CHAPTER 7
ATMCard Evaluation

7.1 Introduction

This chapter will discuss the experimental evaluation and the VHDL synthesis results of the *ATMCard*. Figure 7.1 illustrates how the functioning of a two-board system was implemented on a single *ATMCard*. Since the *HyperPlane* is currently under development, FPGA emulation of the *HyperPlane* was required (Figure 7.1). In Figure 7.1, the components in the path from point A to point B (the ATM Cell Generator, the *TNET_Interface*, the *HyperPlane_Interface*, the *HyperPlane* simulator, and the *TNET_TX_Interface*) were implemented in the Xilinx FPGA of one *ATMCard*. The components labeled ATM Cell Generator', *TNET_Interface'*, *HyperPlane_Interface'*, and *TNET_TX_Interface'* in the path from A' to B' were not implemented since only one *ATMCard* was available for testing. However, Figure 7.1 illustrates how it was possible to mimic the behaviour of two *ATMCards* using only one *ATMCard*. The only difference was the fact that only a unidirectional datapath from point A to point B could be implemented and tested.

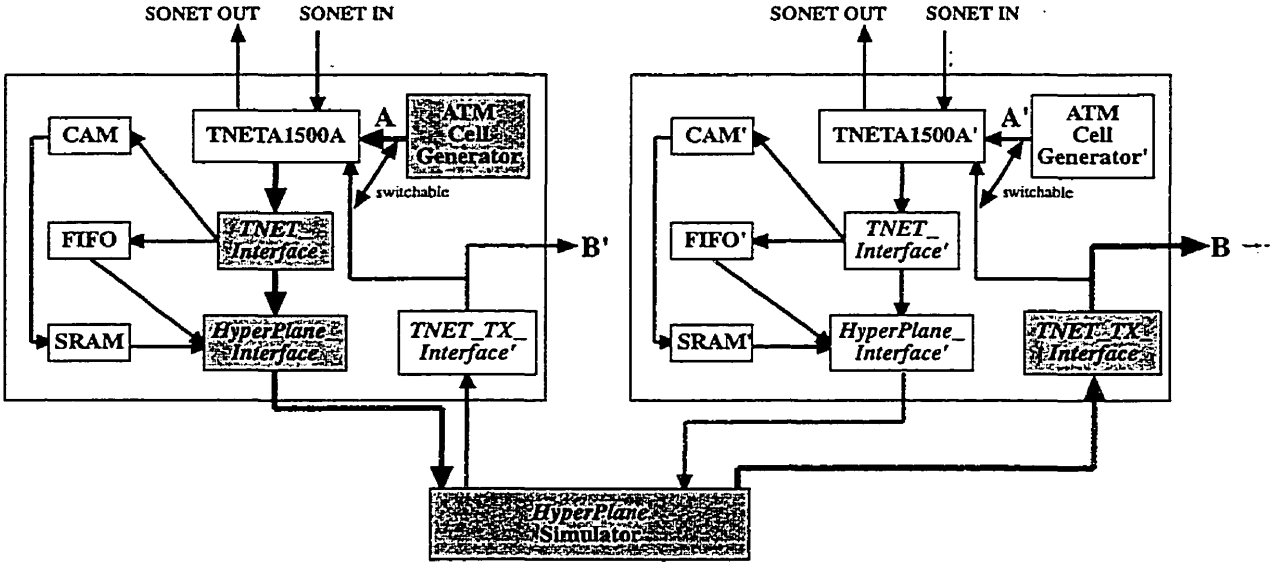


Figure 7.1: *ATMCard* evaluation setup showing two *ATMCards* connected together. Shaded blocks were implemented in a single Xilinx FPGA.

Of particular note in Figure 7.1, is the *ATM Cell Generator*. When the *ATMCard* was implemented, a source for ATM cells was required. The *ATM Cell Generator* is a VHDL entity implemented as an 8-bit *Linear Feedback Shift Register* (LFSR) that generates a pseudo-random sequence of 8-bit words (section 6.3.3.3). LFSRs also generate pseudo-random bit sequences but in this case, the LFSR is used to output a pseudo-random *byte* sequence. Upon each clock cycle, the bits in the LFSR shift one position and the 8-bit word is read out. Since the *ATM Cell Generator* is an 8-bit LFSR, a total of $(2^8 - 1)$ or 255 unique 8-bit words are generated. Recall from Chapter 6 that in order to transmit an ATM cell into the SONET/ATM transceiver, the assertion of a *TXCELL* signal was required to allow the SONET/ATM transceiver to identify the first header byte of an ATM cell. Combinatorial logic within the *ATM Cell Generator* ensures that only one of the possible 255 unique bytes that are generated from the 8-bit LFSR result in the *TXCELL* signal being asserted. Since an ATM cell is only 53-bytes long, the remaining 202 bytes that are generated by the 8-bit LFSR are not transmitted into the SONET/ATM transceiver. The 255-byte sequence is generated repeatedly.

The SONET/ATM transceiver is only able to receive and transmit a single stream of ATM cells. Since the *ATM Cell Generator* is already transmitting ATM cells into the SONET/ATM transceiver, the *TNET_TX_Interface* (Figure 7.1) described in Chapter 6 will not be able to transmit ATM cells received from the *HyperPlane* into the SONET/ATM transceiver at the same time. Therefore, the ATM cells transmitted by the *TNET_TX_Interface* are output to point B in Figure 7.1. These ATM cells are routed to a logic analyzer for observation. A second observation point (point A in Figure 7.1) appears in between the SONET/ATM transceiver and the *TNET_Interface*. A logic analyzer is also able to view the ATM cells at this point. In conclusion, ATM cells are generated by the *ATM Cell Generator* and transmitted into the SONET/ATM transceiver. The transceiver is put in loopback mode so that any ATM cells received from the *ATM Cell Generator* are looped back and transmitted to the *TNET_Interface*. From the *TNET_Interface*, the ATM cells are sent to the *HyperPlane_Interface* for transmission into the *HyperPlane*. The *TNET_TX_Interface* receives these ATM cells from the *HyperPlane* and outputs these cells to point B in Figure 7.1.

A successful demonstration of the *ATMCard* setup shown in Figure 7.1 was accomplished. ATM traffic was shown to travel from the ATM Cell Generator (point A) to the *TNET_TX_Interface* (point B). This demonstration showed that the FPGA components functioned properly. In addition, the *HyperPlane* simulator functionality was verified. Finally, correct header translation was observed at point B demonstrating the proper

functioning of the header translation components which included the CAM and SRAM memories.

A photograph of the *ATM Card* is shown in Figure 7.2. In addition to previously mentioned components such as the Xilinx FPGA and various memory ICs, several other components are shown. The SONET/ATM transceiver is housed in a *Plastic Quad Flat-Pack* (PQFP) chip package which has I/O pins that are incompatible with the Speedwire™ terminals used on the *ATMCard*. Therefore, the SONET/ATM transceiver was placed in a *Zero-Insertion Force* (ZIF) adapter that enabled it to be mounted on the Speedwire™ terminals of the *ATMCard*. The 19.44 MHz oscillator (Figure 7.2) is used by the transceiver to generate a 155 MHz clock used in the serial transmission of SONET OC-3 data. The 512K *Erasable Programmable Read-Only Memory* (EPROM) is used to store the bit-stream that configures the Xilinx FPGA. Finally, DIP switches and LEDs are used to reset the *ATMCard* and monitor key signals, respectively.

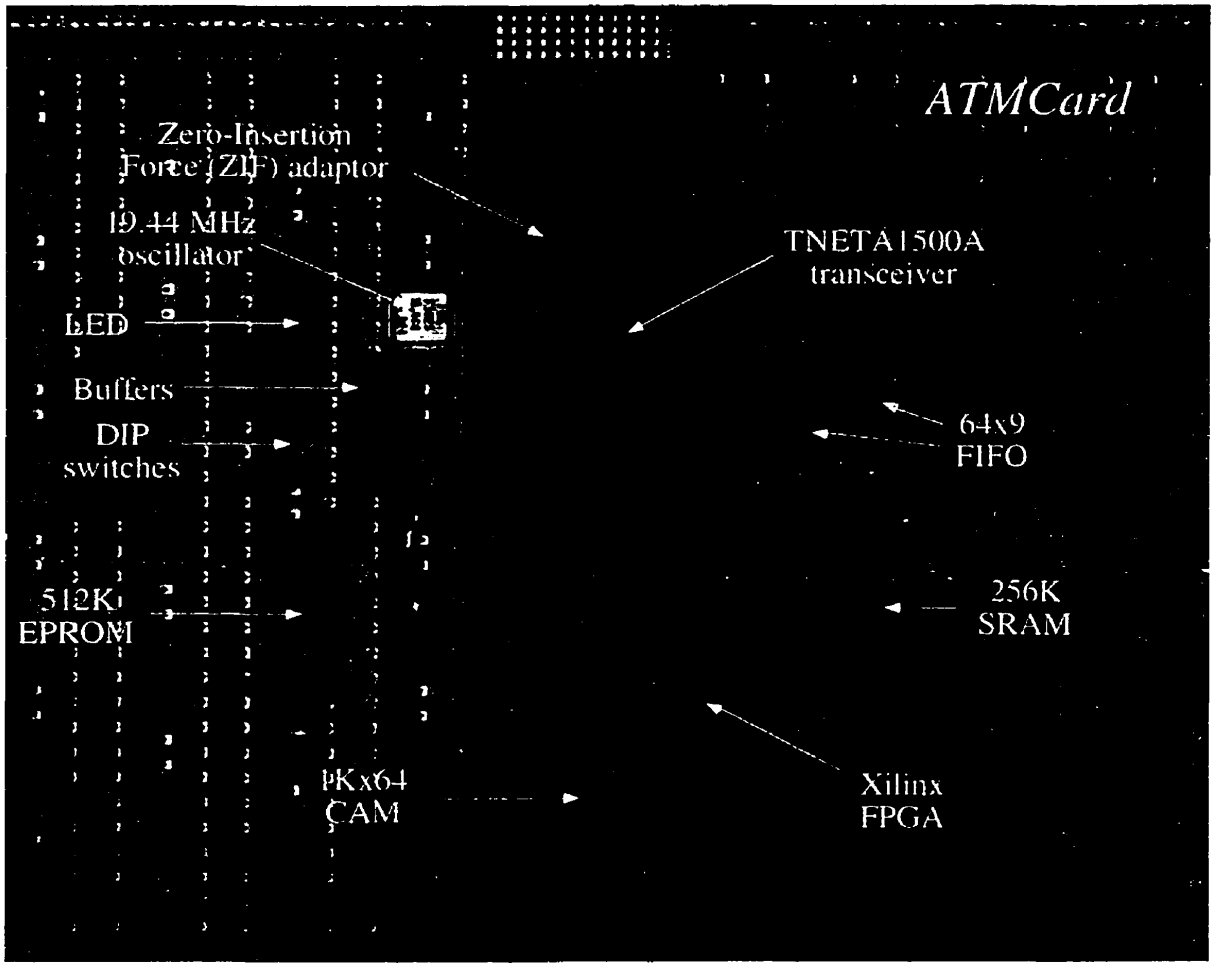


Figure 7.2: Photograph of the *ATMCard*.

7.2 VHDL Synthesis Results

Synthesis of the *ATMCard* VHDL entities was accomplished using the Synopsys VHDL compiler version 3.4b. Synopsys produces a netlist file known as the *Xilinx Netlist Format* (XNF). The XNF file is then used by the Xilinx XACTStep version 5.2.1 CAD tool. Specifically, XACTStep performs placement, partitioning, and routing (PPR) of the design and maps it to the selected Xilinx FPGA. For the case of the *ATMCard*, the 4010PG191-6 FPGA was used. A final bit-stream file is produced by XACTStep that will be stored on an EPROM to program the FPGA.

The use of VHDL in the design of the *ATMCard* was highly beneficial in terms of rapid design prototyping. If a design change was required in one of the VHDL entities the designer simply edited the VHDL code, resynthesized, and downloaded the newly synthesized design onto the FPGA.

XACTStep also provides area and timing statistics. Recall from Chapter 4 that a Xilinx FPGA is composed of a matrix of interconnected logic blocks known as *Configurable Logic Blocks* (CLB). These CLBs implement combinatorial logic in small look-up tables whose outputs are fed to flip-flops, other CLBs, or I/O ports. Each CLB implements these look-up tables in three function generators labeled *F*, *G*, and *H*. A 4010PG191-6 Xilinx FPGA contains a 20x20 matrix of CLBs. The area statistics are expressed in terms of the CLBs used, the flip-flops used, and the function generators used. Table 7.1 shows the area statistics for the VHDL entities of the *ATMCard* synthesized with the *HyperPlane* simulator.

FPGA Resource	Number used	Max. Available in 4010PG191-6	Percentage Used
CLBs	352	400	88%
F and *G* function generators	385	800	48%
H function generators	88	400	22%
CLB flip-flops	281	800	35%

Table 7.1: FPGA area statistics for the *ATMCard* with the *HyperPlane simulator* (area optimization used).

It should be noted that the FPGA area resources required by the *HyperPlane* simulator amount to less than 15% of the total resources available on the FPGA. This result is much lower than those found in [DES95a]. Also, the *HyperPlane* simulator used by the *ATMCard* uses 8-bit wide channels whereas the SPA described in [DES95a] uses 4-bit wide channels.

This seemingly contradictory finding is due to the fact that many of the SPA ports shown in Figure 5.1 in Chapter 5 were not used in the *HyperPlane* simulator. To conserve FPGA resources, the unused ports were removed from the VHDL code of the *HyperPlane* emulator. For example, referring to Figure 5.1, the *ATMCard* demonstrator used only one of the four available channels (i.e., only channel C_1 was used). Hence, when the simplified SPA was placed on the FPGA, the number of I/O pins was reduced by a factor of four. It is interesting to note that the 1998 demonstrator faced the same constraints. To conserve I/O pins, each SPA utilized only one channel rather than the four channels as shown in Figure 5.1.

During VHDL synthesis and optimization, Synopsys would prune out logic that was associated with these unused ports. The resulting area-optimized design would still have its original design functionality preserved. To demonstrate this point, the 8x4 SPA *HyperPlane* simulator was first synthesized with ports removed in the VHDL code - leaving one electrical channel per SPA - and then again with all ports left intact. The results are shown in Tables 7.2 and 7.3.

FPGA Resource	Number used	Max. Available in 4010PG191-6	Percentage Used
CLBs	58	400	14%
F and *G* function generators	30	800	3%
H function generators	2	400	0%
CLB flip-flops	54	800	6%

Table 7.2: FPGA area statistics of the *HyperPlane* simulator with one electrical channel per SPA (with three of the four ports removed in VHDL code).

FPGA Resource	Number used	Max. Available in 4010PG191-6	Percentage Used
CLBs	343	400	85%
F and *G* function generators	445	800	55%
H function generators	135	400	33%
CLB flip-flops	266	800	33%

Table 7.3: FPGA area statistics of the *HyperPlane* simulator with all ports intact in the VHDL code (area optimization used).

In addition to area statistics, XACTStep provides timing statistics. The FPGA device used, the 4010PG191-speed grade 6, is the slowest available FPGA in the Xilinx 4010 family. The speed grade defines the worst case toggle rate of the FPGA flip-flops in nanoseconds. The lower this number is, the faster the device. A new family of FPGAs known as the 4000E family has been introduced. These devices are faster than the 4000 series and have speed grades as low as 3. Timing statistics for an *ATMCard* targeted for the 4010EPG191-speed grade 3 FPGA are also presented in Tables 7.4 and 7.5 for comparison.

	4010PG191 Speed Grade 6	4010EPG191 Speed Grade 3
Pad to Setup	75.1 ns	38.8 ns
Clock to Setup	36.2 ns	19.4 ns
Clock to Pad	27.6 ns	15.2 ns
Maximum Frequency of *HyperPlane* Time Slot Clock	13.3 MHz	25.8 MHz
Maximum Frequency of *ATMCard* clock	3.3 MHz	6.4 MHz

Table 7.4: FPGA timing statistics for the *ATMCard* with the *HyperPlane* Simulator embedded in same FPGA.

In order for the *ATMCard* to utilize the full SONET OC-3 bandwidth of 155 Mbit/s available on the SONET/ATM transceiver, the FPGA on the *ATMCard* must be clocked at 19.44 MHz. The timing statistics for the FPGA are shown in Table 7.4. When the FPGA includes the *HyperPlane* simulator, the maximum *ATMCard* clock frequency is 6.4 MHz using the speed grade 3 device (Table 7.4). This clock rate is below the objective of 19.44

MHz. However, in a real system, the FPGA will not include the *HyperPlane* simulator. Hence, faster clock frequencies are attainable.

	4010PG191 Speed Grade 6	**4010EPG191 Speed Grade 3**
Pad to Setup	104.1 ns	59.7 ns
Clock to Setup	65.7 ns	33.3 ns
Clock to Pad	56.8 ns	32.1 ns
Maximum Frequency of *ATMCard* with no *HyperPlane* simulator	9.6 MHz	16.8 MHz

Table 7.5: FPGA timing statistics for the *ATMCard* without the *HyperPlane* simulator embedded in the same FPGA.

Table 7.5 shows the expected maximum clock frequencies for an *ATMCard* implemented without the VHDL *HyperPlane* emulation. The clock frequencies are higher in this case since the *ATMCard* is no longer restricted to implementing the *HyperPlane* simulator. Note that with the use of the 4010EPG191-3 FPGA, the maximum operating of 16.8 MHz almost reaches the operating frequency of 19.44 MHz required in order to utilize the full SONET OC-3 bandwidth. The use of faster FPGAs from Xilinx or from other vendors will easily permit an operating frequency of 19.44 MHz or more.

7.3 Summary

Chapter 7 described the successful evaluation of the *ATMCard*. VHDL synthesis issues were discussed and the CAD tools used were outlined. Area and timing statistics for the *ATMCard* and for the *HyperPlane* simulator were presented. Most importantly, since minimal adjustments are required, Chapter 7 has illustrated the successful operation of a PCB supporting ATM switching capable of working with the 1998 CITR optical backplane demonstrator.

CHAPTER 8
Conclusion

This thesis has described the design and implementation of an ATM transmit/receive card called the *ATMCard*. The *ATMCard* is targeted for a free-space electro-optical backplane, known as the *HyperPlane*, which is under development at McGill University. The main purpose of the *ATMCard* is to facilitate the control of ATM traffic flow between incoming ATM-carrying SONET streams and the *HyperPlane*. In addition, the *ATMCard* performs ATM cell header translation before these cells are transmitted to the *HyperPlane*. Portions of the *ATMCard* were implemented in a programmable logic device known as a *Field-Programmable Gate Array* (FPGA). Furthermore, cell header translation functions necessitated the use of separate memory ICs. Finally, since the *HyperPlane* is currently under development and is scheduled to be operational in 1998, a hardware emulation of the *HyperPlane* was also implemented on the FPGA.

Chapter 2 began this thesis with a broad overview of ATM. The motivations behind the introduction of ATM were outlined. Chapter 2 also included a description of the ATM network layer model, ATM traffic management, and broadband ATM switching and queueing.

Free-space optical interconnects were discussed in Chapter 3. In particular, Chapter 3 provided a description of the McGill *HyperPlane* architecture. It also described smart pixel technology including an overview of the *HyperPlane* smart pixels.

In Chapter 4, a high-level introduction to the *ATMCard* was presented and the individual hardware components used on the *ATMCard* were described. These components included an FPGA, and SONET/ATM transceiver, a *Content-Addressable Memory* (CAM), and SRAM and FIFO memory ICs.

A high-level system specification of the *ATMCard* was provided in Chapter 5. Included in this specification was a brief description of the smart pixel array that was used with the *ATMCard*.

Detailed functional and structural descriptions of the *ATMCard* VHDL components were given in Chapter 6. These descriptions were presented in a top-down manner. The

interaction between the VHDL components and external hardware components such as the CAM and the SONET/ATM transceiver was also described.

The evaluation of the *ATMCard* was described in Chapter 7. The hardware test apparatus and the FPGA area and timing results were discussed.

With minimal adjustments, the *ATMCard* may be interfaced with the 1998 CITR optical backplane demonstrator. This project may be performed with more confidence since a proof-of-concept system was successfully demonstrated using an emulation of the optical backplane. With the knowledge obtained from the design and implementation of the *ATMCard*, it is anticipated that the subsequent design of a more advanced *ATMCard* or other application-oriented PCB will be much more tractable.

There are a number of recommendations for the design of the next generation SONET/ATM PCBs. First, in order to approach the bandwidth required for a terabit photonic backplane, each PCB must include multiple OC-3 or OC-12 SONET transceivers. Furthermore, faster and more powerful FPGAs will be required. Second, the speedwire concept is very useful for prototyping, but in order to achieve a larger bandwidth capacity a custom PCB would be necessary. A final recommendation is to include more complex data link protocols such as error and flow control into the *ATMCard.* With the current foundation of the *ATMCard* firmly established, the addition of more complex protocols into the *HyperPlane* interface is feasible for a second generation *ATMCard.*

References

[AHM89] H. Ahmadi and W. E. Denzel, "A survey of modern high-performance switching techniques", *IEEE JSAC*, vol. 7, no. 7, pp. 1091-1193, September 1989.

[ALF94] P. Alfke, "Linear feedback shift register counters", *The Programmable Logic Data Book 1994*, Xilinx, pp. 9-24, 1994.

[ALF95] P. Alfke, "Efficient shift registers, LFSR counters, and long pseudo-random sequence generators", Xilinx application note, August 1995.

[ATM93] The ATM Forum, *ATM User-Network Interface Specification*, version 3.0, PTR Prentice Hall: Englewood Cliffs, New Jersey, 1993.

[BAL89] R. Ballart and Y.-C. Ching, "SONET: Now it's the standard optical network", *IEEE Communications Magazine*, vol. 27, no. 3, pp. 8-15, March 1989.

[BAN91] T. C. Banwell, R. C. Estes, S. F. Habiby, G. A. Hayward, T. K. Helstern, G. R. Lalk, D. D. Mahoney, D. K. Wilson, and K. C. Young, Jr., "Physical design issues for very large ATM switching systems", *IEEE JSAC*, vol. 9, no. 8, pp. 1227-1238, October 1991.

[BAN92] T. C. Banwell, W. E. Stephens, and G. R. Lalk, "Transmission of 155 Mbit/s (SONET STS-3) signals over unshielded and shielded twisted pair copper wire", *Electronics Letters*, vol. 28, no. 12, pp. 1102-1104, 4th June 1992.

[BEN62] V. E. Benes, "On rearrangeable three-stage connecting networks", *The Bell System Technical Journal*, vol. XLI, no. 5, pp. 1481-1492, September 1962.

[BIC94] BICC-VERO Electronics, *Product Handbook '94*.

[BON95] F. Bonomi and K. W. Fendick, "The rate-based flow control framework for the available bit rate ATM service", *IEEE Network*, vol. 9, no. 2, pp. 25-39, March/April 1995.

[BRI94] R. van den Brink, "VPI/VCI translation and cell tagging in ATM with the MU9C1480 LANCAM®" MUSIC Semiconductors application note AN-N9, Rev. 0, August 5, 1994.

[BRO92] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers: Boston, 1992.

[CIT95] Canadian Institute for Telecommunications Research (CITR), *Annual Report 1995*. For more information please contact the offices of CITR at 3480 University St., Suite 7543, Montreal, Quebec, Canada H3A 2A7.

[DEP93] M. De Prycker, *Asynchronous Transfer Mode: Solution for Broadband ISDN*, Second Edition, Ellis Horwood: New York, 1993.

[DES95] P. Desai, *Embeddings of a Cray T3D Supercomputer onto a Free-Space Optical Backplane*, Master's Thesis, McGill University, October 1995.

[DES95a] P. Desai and T. H. Szymanski, "Simulation and modelling of a smart pixel array optical backplane using FPGAs", *Proceedings of the Third Canadian Workshop on Field Programmable Devices*, Montreal, Canada, pp. 38-43, May/June 1995.

[GAR94] J. Garcia-Haro and A. Jajszczyk, "ATM shared-memory switching architectures", *IEEE Network*, vol. 8, no. 4, pp. 18-26, July/August 1994.

[GOO89] K. W. Goosen, G. D. Boyd, J. E. Cunningham, W. Y. Jan, D. A. B. Miller, D. S. Chemla, and R. M. Lum, "GaAs-AlGaAs multiple quantum well reflection modulators grown on GaAs and silicon substrates", *IEEE Photonic Technology Letters*, vol. 1, no. 10, October 1989.

[HAN94] R. Händel, M. N. Huber, and S. Schröder, *ATM Networks: Concepts, Protocols, Applications*, Second Edition, Addison-Wesley:Wokingham, England, 1994.

[HAR86] D. H. Hartman, "Digital high speed interconnects: a study of the optical alternative", *Optical Engineering*, vol. 25, no. 10, pp. 1086-1102, October 1986.

[HIN93] H. S. Hinton, *An Introduction to Photonic Switching Fabrics*, Plenum Press: New York, 1993.

[HIN93a] H. S. Hinton and A. L. Lentine, "Multiple quantum-well technology", *IEEE Circuits and Devices*, pp. 12-18, March 1993.

[HIN94] H. S. Hinton, T. J. Cloonan, F. B. McCormick, Jr., A. L. Lentine, and F. A. P. Tooley, "Free-space digital optical systems", *Proceedings of the IEEE*, vol. 82, no. 11, pp. 1632-1649, November 1994.

[HIN95] H. S. Hinton and T. H. Szymanski, "Intelligent optical backplanes", *Proceedings of the Second International Conference on Massively Parallel Processing using Optical Interconnections (MPPOI '95)*, pp. 133-143, October 23-24, 1995.

[HUI96] J. Y. Hui and K.-W. Cheung, "Optical versus electronic switching for broadband networks", *IEEE Network*, vol.10, no. 6, pp. 21-25, November/December 1996.

[IGA88] K. Iga, F. Koyama, and S. Kinoshita, "Surface emitting semiconductor lasers", *IEEE Journal of Quantum Electronics*, vol. 24, no. 9, pp. 1845-1855, September 1988.

[IM93] G.-H. Im and J. J. Werner, "Bandwidth-efficient digital transmission up to 155 Mb/s over unshielded twisted pair wiring", *Proceedings of the IEEE*

International Conference on Communications '93 (ICC '93), pp. 1797-1803, May 23-26, 1993.

[INT94] Integrated Device Technology, Inc., "CMOS STATIC RAM 256K (32K x 8-bit): IDT71256S/L data sheets", May 1994.

[INT94a] Integrated Device Technology, Inc., "CMOS Parallel FIFO 64 x 4-bit and 64 x 5-bit: IDT72401/2/3/4 data sheets", July 1994.

[ITO91] A. Itoh, W. Taakahashi, H. Nagano, M. Kurisaka, and S. Iwasaki, "Practical implementation and packaging technologies for a large-scale ATM switching system", *IEEE JSAC*, vol. 9, no. 8, pp. 1280-1288, October 1991.

[JAI90] R. Jain, "Congestion control in computer networks: Issues and trends", *IEEE Network*, vol. 4, no. 3, pp. 24-30, May 1990.

[JAI95] R. Jain, "Congestion control and traffic management in ATM networks: Recent advances and a survey", Draft Version: January 26, 1995, invited submission to *Computer Networks and ISDN Systems*.

[JOH88] Carl T. A. Johnk, *Engineering Electromagnetic Fields and Waves*, Second Edition, John Wiley & Sons: New York, 1988.

[JOH93] Howard W. Johnson and Martin Graham, *High-Speed Digital Design: A Handbook of Black Magic*, Prentice Hall PTR: Englewood Cliffs, New Jersey, 1993.

[KAR87] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queueing on a space-division packet switch", *IEEE Trans. Comm.*, vol. COM-35, no. 12, pp. 1347-1356, December 1987.

[KES93] G. C. Kessler, *ISDN*, Second Edition, McGraw-Hill, Inc.: New York, 1993..

[KUN95] H. T. Kung and R. Morris, "Credit-based flow control for ATM networks", *IEEE Network*, vol. 9, no. 2, pp. 40-48, March/April 1995.

[LI95] C.-Y. P. Li and H. B. S. Wai, "ATM header translator", from the Project Laboratory, course 304-494B with T. H. Szymanski, McGill University, April 13, 1995.

[MIL84] D. A. B. Miller, D. S. Chomla, T. C. Damen, A. C. Gossard, W. Wiegmann, T. H. Wood, and C. A. Burrus, "Novel hybrid optically bistable switch: the quantum well self electro-optic effect device", *Applied Physics Letters*, vol. 45, no. 1, pp. 13-15, 1 July 1984.

[MIL90] D. A. B. Miller, "Quantum well self electro-optic effect devices", *Optical and Quantum Electronics*, vol. 22, pp. S61-S98, 1990.

[MIL95] D. A. B. Miller, "Hybrid SEED - massively parallel optical interconnections for silicon ICs", *Proceedings of the Second International Conference on Massively Parallel Processing using Optical Interconnections (MPPOI '95)*, pp. 2-7, October 23-24, 1995.

[MIN89] S. E. Minzer, "Broadband ISDN and asynchronous transfer mode (ATM)", *IEEE Communications Magazine*, vol. 27, no. 9, pp. 17-24, 57, September 1989.

[MUS94] MUSIC Semiconductors, *The MU9C1480 LANCAM® Handbook*, November 1994.

[NAB95] A. Nabbus and R. Telfer, "SONET to photonic backplane interface", from the Project Laboratory, course 304-494B with T. H. Szymanski, McGill University, April 10, 1995.

[NEW94] P. Newman, "Traffic management for ATM local area networks", *IEEE Communications Magazine*, vol. 32. no. 8, pp. 44-50, August 1994.

[NOR92] R. A. Nordin, A. F. J. Levi, R. N. Nottenburg, J. O'Gorman, T. Tanbun-Ek, and R. A. Logan, "A systems perspective on digital interconnection technology", *Journal of Lightwave Technology*, vol. 10, no. 6, pp. 811-827, June 1992.

[NOR95] R. A. Nordin, W. R. Holland, and M. A. Shahid, "Advanced optical interconnection technology in switching equipment", *Journal of Lightwave Technology*, vol. 13, no. 6, pp. 987-994, June 1995.

[OIE89] Y. Oie, M. Murata, K. Kubota, and H. Miyahara, "Effect of speedup in nonblocking packet switch", *Proc. IEEE International Conference on Communications '89*, pp. 410-414, June 1989.

[OMI93] C. G. Omidyar and A. Aldridge, "Introduction to SDH/SONET", *IEEE Communications Magazine*, vol. 31, no. 9, pp. 30-33, September 1993.

[PAT93] A. Pattavina, "Nonblocking architectures for ATM switching", *IEEE Communications* Magazine, vol. 31, no. 2, pp. 38-48, February 1993.

[RAM95] K. K. Ramakrishnan and P. Newman, "Intergration of rate and credit schemes for ATM flow control", *IEEE Network*, vol. 9, no. 2, pp. 49-56, March/April 1995.

[RED87] U. K. Reddy, R. Houdre, G. Munns, G. Li, H. Morkoc, M. Longerbone, L. Davis, B. P. Gu, and N. Otsuka, "Investigation of GaAs/AlGaAs multiple quantum wells grown on Ge and Si substrates by molecular beam epitaxy", *Journal of Applied Physics*, vol. 62, no. 12, pp. 4858-4862, 15 December 1987.

[ROL96] D. R. Rolston, D. V. Plant, T. H. Szymanski, H. S. Hinton, W. S. Hsiao, M. H. Ayliffe, D. Kabal, M. B. Venditti, P. Desai, A. V. Krishnamoorthy, K. W. Goosen, J. A. Walker, B. Tseng, S. P. Hui, J. C. Cunningham, and W. Y. Jan, "A hybrid-SEED smart pixel array for a four-stage intelligent optical backplane demonstrator", *IEEE Journal of Selected Topics in Quantum Electronics*, Special Issue on Smart Pixels, pp. 97-105, April 1996.

[ROO94] R. Rooholamini and V. Cherkassky, "Finding the right ATM switch for the market", *IEEE Computer*, vol. 27, no. 4, pp. 16-28, April 1994.

[ROS93] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "Architecture of field-programmable gate arrays", *Proceedings of the IEEE*, vol. 81, no. 7, pp. 1013-1029, July 1993.

[SIU95] K.-Y. Siu and R. Jain, "A brief overview of ATM: Protocol layers, LAN emulation, and traffic management", *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 2, pp. 6-20, April 1995.

[SIM95] R. J. Simcoe and T.-B. Pei, "Perspectives on ATM switch architecture and the influence of traffic pattern assumptions on switch design", *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 2, pp. 93-105, April 1995.

[STI95] B. Stiller, "A survey of UNI signaling systems and protocols for ATM networks", *ACM SIGCOMM Computer Communication Review*, vol. 25, no. 2, pp. 21-33, April 1995.

[SZY94] T. Szymanski and H. S. Hinton, "Architecture of a terabit free-space photonic backplane", *Proceedings of the International Conference on Optical Computing (OC '94)*, August 22-25, 1994.

[SZY94a] T. H. Szymanski, P. Desai, M. C. Kim, and S. Gagnon, *CITR Project 94-3-4: Large ATM Architectures - Milestones and Deliverables-94/95*. For more information please contact Prof. T. Szymanski at 3480 University St., Suite 633, Montreal, Quebec, Canada H3A 2A7.

[SZY95] T. Szymanski and H. S. Hinton, "Design of a terabit free-space photonic backplane for parallel computing", *Proceedings of the Second International Conference on Massively Parallel Processing using Optical Interconnections (MPPOI '95)*, pp. 16-27, October 23-24, 1995.

[SZY96] T. H. Szymanski and H. S. Hinton, "A reconfigurable intelligent optical backplane for parallel computing and communications", *Applied Optics*, Special Issue on Optical Computing, pp. 1253-1268, March 1996.

[TEX89] Texas Instruments, *MOS Memory: Commercial and Military Specifications Data Book*, pp. 6.105-6.117, 1989.

[TEX94] Texas Instruments, "TDC1500A: 155.52-Mbits/s SONET/SDH ATM receiver/transmitter", product preview SDNS021A, March 1994, revised November 1994.

[TOB90] F. A. Tobagi, "Fast packet switch architectures for broadband integrated services digital networks", *Proceedings of the IEEE*, vol. 78, no. 1, pp. 133-167, January 1990.

[TRI94] S. M. Trimberger, ed., *Field-Programmable Gate Array Technology*, Kluwer Academic Publishers: Boston, 1994.

[VET95] R. J. Vetter, "ATM concepts, architectures, and protocols", *Communications of the ACM*, vol. 38, no. 2, pp. 30-38, 109, February 1995.

[VIC93] R. Vickers, "The development of ATM standards and technology: A retrospective", *IEEE Micro*, vol. 13, no. 6, pp. 62-73, December 1993.

[XIL94] Xilinx, Inc., *The Programmable Logic Data Book*, 1994

[YEH87] Y.-S. Yeh, M. G. Hluchyj, and A. S. Acampora, "The knockout switch: A simple, modular architecture for high-performance packet switching", *IEEE JSAC*, vol. SAC-5, no. 8, pp. 1274-1283, October 1987.

[ZEG93] E. W. Zegura, "Architectures for ATM switching systems", *IEEE Communications Magazine*, vol. 31, no. 2, pp. 28-37, February 1993.

Related Architectural Publications at McGill University[1]

P. Desai, *Embeddings of a Cray T3D Supercomputer onto a Free-Space Optical Backplane*, Master's Thesis, McGill University, October 1995.

S. Gagnon, *Virtual Hardware Based ATM Switching Node Test System*, Master's Thesis, McGill University, July 1995.

W. Ho, *Performance Modelling of the Terabit Free Space Optical Backplane*, Master's Thesis, McGill University, January 1997.

T. Obenaus, *Topology of a High Speed Free-Space Photonic Network*, Master's Thesis, McGill University, October 1995.

M. Verghese, *A Software Based Design Space Exploration of a Free-Space Photonic Backplane*, Master's Thesis, McGill University, June 1995.

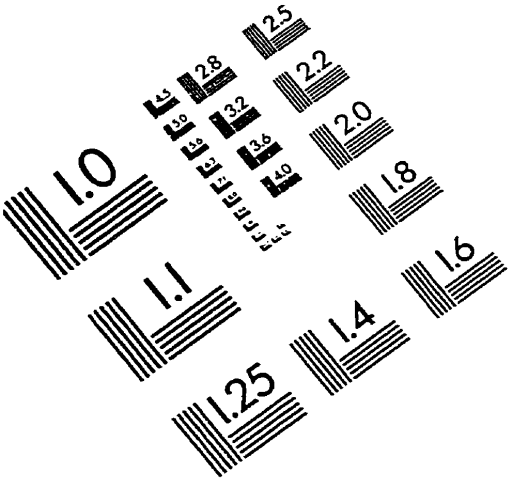
A. Nabbus and R. Telfer, "SONET to photonic backplane interface", from the Project Laboratory, course 304-494B with T. H. Szymanski, McGill University, April 10, 1995.

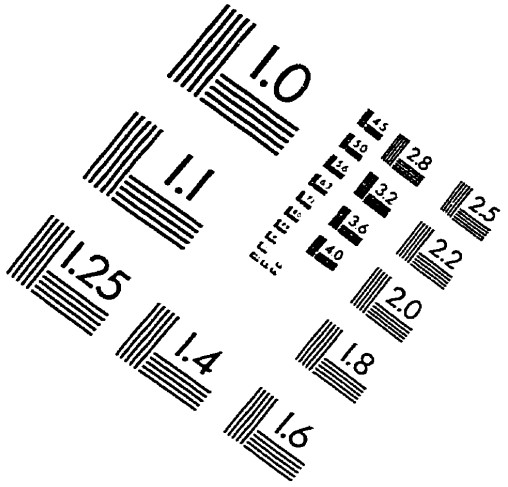
C.-Y. P. Li and H. B. S. Wai, "ATM header translator", from the Project Laboratory, course 304-494B with T. H. Szymanski, McGill University, April 13, 1995.

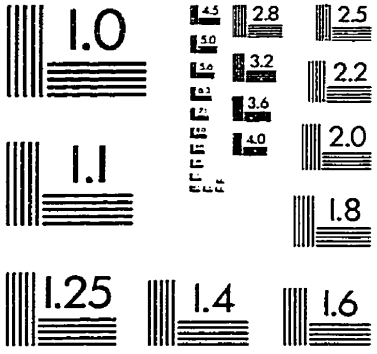
T. H. Szymanski, P. Desai, M. C. Kim, and S. Gagnon, *CITR Project 94-3-4: Large ATM Architectures - Milestones and Deliverables-94/95*. For more information please contact the offices of CITR at 3480 University St., Suite 7543, Montreal, Quebec, Canada H3A 2A7.

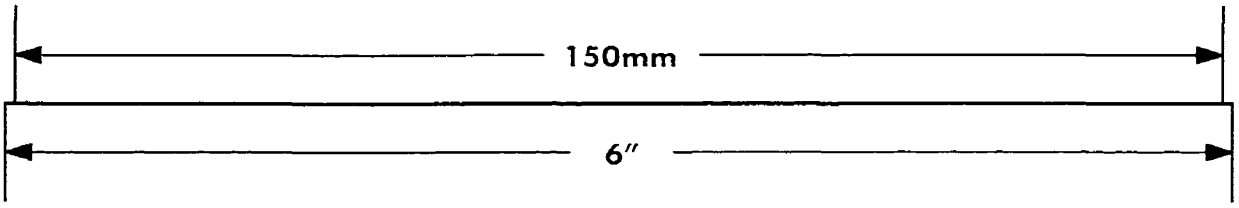
[1] Information on these publications may be obtained from Dr. T. H. Szymanski.

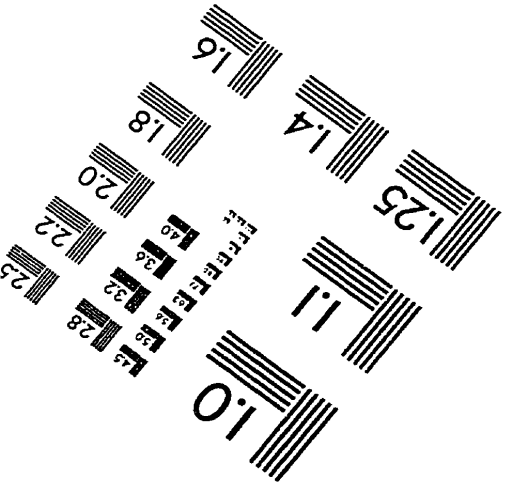
IMAGE EVALUATION
TEST TARGET (QA-3)

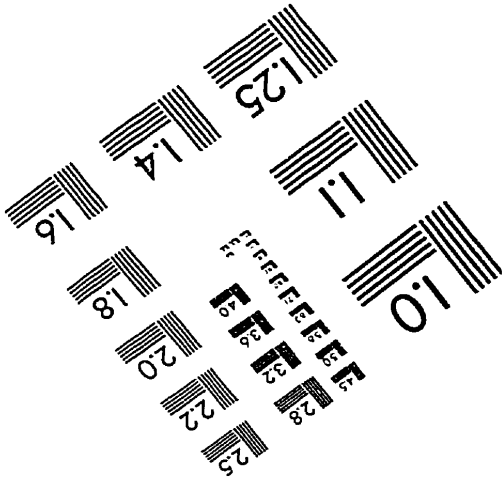












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989