

**DESIGN AND IMPLEMENTATION  
OF AN ATM  
TRAFFIC GENERATION AND CAPTURE CARD**

**BY  
JEFF GIESBRECHT**

**A Thesis  
Submitted to the Faculty of Graduate Studies  
in Partial Fulfillment of the Requirements  
for the Degree of**

**MASTER OF SCIENCE**

**Department of Electrical and Computer Engineering  
University of Manitoba  
Winnipeg, Manitoba**

**(c) January, 2000**



National Library  
of Canada

Acquisitions and  
Bibliographic Services

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

Bibliothèque nationale  
du Canada

Acquisitions et  
services bibliographiques

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file Votre référence*

*Our file Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-51712-8

**Canada**

**THE UNIVERSITY OF MANITOBA**  
**FACULTY OF GRADUATE STUDIES**  
\*\*\*\*\*  
**COPYRIGHT PERMISSION PAGE**

**Design and Implementation of an ATM Traffic Generation and Capture Card**

**BY**

**Jeff Giesbrecht**

**A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University  
of Manitoba in partial fulfillment of the requirements of the degree**

**of**

**Master of Science**

**JEFF GIESBRECHT © 2000**

**Permission has been granted to the Library of The University of Manitoba to lend or sell copies of this thesis/practicum, to the National Library of Canada to microfilm this thesis/practicum and to lend or sell copies of the film, and to Dissertations Abstracts International to publish an abstract of this thesis/practicum.**

**The author reserves other publication rights, and neither this thesis/practicum nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.**

---

# Abstract

As a result of recent increases in demand for network bandwidth, it is becoming more important that high-speed telecommunication networks be installed, managed, and used as efficiently as possible. In order for these objectives to be realized, it is essential that network operation be studied and accurate models of its behavior developed. Unfortunately, the data acquired from testing with network equipment is often less than ideal due to the cost and capabilities of test equipment currently available commercially. Testing suffers as a result of the equipment's high cost, inability to generate network traffic with certain types of properties, and the limited amount of traffic which can be recorded for analysis. A network interface card designed specifically to overcome the limitations of commercial test equipment is one solution for improving the results of current network testing. This card would not duplicate all of the capabilities of existing test equipment. Instead, it would only strive to improve the traffic generation capabilities to allow all possible types of network traffic sources to be simulated, extend network traffic capture periods to provide a sufficient amount of data for analysis, and be built at a low-enough cost to be a viable alternative to commercial equipment. This thesis will describe the development of a card designed for generating and capturing ATM network traffic at OC-3 rates.

---

# Acknowledgements

Several people deserve to be acknowledged for their role in helping to realize a working prototype of the ATM traffic generation and capture card discussed in this thesis. I would like to begin by thanking my advisor, Professor Bob McLeod, and Yair Bourlas for developing the concept of the card. Thanks also go to Hung Nguyen, who designed the card's architecture, as well as to Kent Felske and Dan Reader for their part in getting prototype boards fabricated.

Thanks should also go to the staff at *TRLabs*. In particular, Dan Erickson and Andy Parker for their help in overcoming the many technical hurdles that were encountered during the card's development and Dr. Jose Rueda for managing the project at *TRLabs*.

I would also like to acknowledge the assistance provided by *TRLabs*, Nortel Networks, and the Canadian Microelectronics Corporation in terms of personnel, equipment, and other resources. Finally, I would like to thank my family for their encouragement and support.

---

# Table of Contents

ABSTRACT .....	ii
ACKNOWLEDGEMENTS .....	iii
LIST OF FIGURES .....	vi
LIST OF TABLES .....	vii
LIST OF ABBREVIATIONS .....	ix
1 INTRODUCTION .....	1
1.1 Motivation .....	1
1.2 Report Structure .....	7
2 ASYNCHRONOUS TRANSFER MODE .....	8
2.1 Supported Traffic Types .....	8
2.2 Cell Structure .....	10
2.3 Connections .....	13
2.4 Payload Formatting .....	15
2.5 Cell Transmission .....	16
SONET .....	17
2.6 ATM Reference Model .....	22
Physical Layer .....	22
ATM Layer .....	24
ATM Adaptation Layer .....	25
Higher Layers .....	27
Planes in the ATM reference model .....	28
3 DESIGN SPECIFICATION .....	29
3.1 Required Features and Functionality .....	29
3.2 Host Interface .....	31
3.3 ATM Traffic Stream Representation .....	32
4 CARD DESIGN AND IMPLEMENTATION .....	34
4.1 Basic Architecture .....	34
ATM Network Interface .....	35
Host PC Interface .....	35
Traffic Stream Translator .....	35
Data Flow .....	36
4.2 Component Selection and Complete Architecture .....	37
ATM Network Interface .....	37
Host PC Interface .....	39

---

---

Traffic Stream Translator . . . . .	42
FPGA Configuration . . . . .	46
Complete Architecture . . . . .	47
4.3 System Clock . . . . .	48
4.4 PCB Design. . . . .	50
5 SUPPORT SOFTWARE FOR THE ATM TRAFFIC GENERATION AND CAPTURE CARD . . . . .	52
5.1 FPGA Configuration . . . . .	52
5.2 Host-Card Communication . . . . .	56
Transfer Mechanism. . . . .	57
Host-Card Relationship . . . . .	58
Signalling. . . . .	59
5.3 Low-Level PCI Communication Functions . . . . .	62
5.4 Card Initialization. . . . .	63
5.5 Traffic Stream Transfer. . . . .	64
6 TESTING AND VERIFICATION . . . . .	66
6.1 Component Testing. . . . .	66
FPGA Configuration . . . . .	67
PCI Interface. . . . .	69
S/UNI. . . . .	69
RAM. . . . .	71
6.2 System testing . . . . .	71
Traffic Capture. . . . .	71
Traffic Generation . . . . .	77
7 DISCUSSION. . . . .	85
8 RECOMMENDATIONS AND FUTURE WORK . . . . .	95
9 CONCLUSION . . . . .	100
10 REFERENCES. . . . .	103

---

---

# List of Figures

Figure 1.1	Switch load test . . . . .	3
Figure 1.2	Capturing network traffic for analysis and characterization . . . . .	5
Figure 1.3	Applications of the ATM traffic generation and capture card. . . . .	6
Figure 2.1	ATM cell structure . . . . .	10
Figure 2.2	ATM cell header formats . . . . .	11
Figure 2.3	Relationship between VCs, VPs, and transmission paths . . . . .	12
Figure 2.4	SONET OC-1 frame structure . . . . .	19
Figure 2.5	Mapping of ATM cells to an OC-1 frame . . . . .	20
Figure 2.6	SONET OC-3 frame structure . . . . .	21
Figure 2.7	ATM reference model ([13]). . . . .	23
Figure 2.8	Sublayers within the physical layer ([13]) . . . . .	24
Figure 2.9	Sublayers within the ATM adaptation layer ([13]) . . . . .	27
Figure 3.1	Operating environment of the ATM traffic generation . . . . . and capture card	32
Figure 4.1	Basic architecture of the ATM traffic generation and capture . . . . . card	36
Figure 4.2	Internal architecture of the S/UNI-LITE. . . . .	39
Figure 4.3	Internal architecture of the AMCC S5933. . . . .	41
Figure 4.4	Internal architecture of the traffic stream translator . . . . .	44
Figure 4.5	Traffic stream translator development . . . . .	45
Figure 4.6	Complete architecture of the ATM traffic generation. . . . . and capture card	48
Figure 4.7	Photographs of prototype ATM traffic generation . . . . . and capture card	51
Figure 5.1	Operations performed by microcontroller used for configuring . . . . . the FPGA for ATM traffic generation and capture	56





---

# List of Tables

Table 2.1	Common SONET/SDH transports . . . . .	22
Table 5.1	Commands defined for host-microcontroller communication . . . . .	53
Table 5.2	Responses defined for host-microcontroller communication. . . . .	54
Table 5.3	Commands defined for host-card communication . . . . .	60
Table 5.4	Responses defined for host-card communication . . . . .	60

---

# List of Abbreviations

AAL	ATM adaptation layer
ABR	available bit rate
ANSI	American National Standards Institute
API	application programming interface
ASIC	application-specific integrated circuit
ATM	asynchronous transfer mode
B-ISDN	broadband-ISDN
CBR	constant bit rate
CLB	configurable logic block
CLP	cell loss priority
CS	convergence sublayer
FIFO	first-in, first-out
FPGA	field-programmable gate array
GFC	generic flow control
HEC	header error control
I/O	input/output
ISDN	integrated services digital network
ITU-T	International Telecommunications Union, Telecommunications Standard- ization Sector
LAN	local area network
LED	light-emitting diode
LOH	line overhead
Mbps	mega-bits per second
NNI	network-node interface
NRT	non-real-time
OC	optical carrier
OS	operating system
PC	personal computer
PCB	printed circuit board
PCI	peripheral component interconnect
PHY	physical layer
PMD	physical medium dependent
POH	path overhead
PROM	programmable read-only memory
PT	payload type
PVC	permanent virtual connection
QoS	quality of service
RAM	random access memory
RT	real-time
RTL	register transfer level
SAR	segmentation and reassembly
SDH	synchronous digital hierarchy
SOH	section overhead
SONET	synchronous optical network

---

SPE	synchronous payload envelope
STM	synchronous transport module
STS	synchronous transport signal
SVC	switched virtual connection
TC	transmission convergence
TOH	transport overhead
UBR	unspecified bit rate
UNI	user-network interface
VBR	variable bit rate
VC	virtual channel
VCi	virtual channel identifier
VHDL	VHSIC hardware description language
VHSIC	very high speed integrated circuit
VP	virtual path
VPI	virtual path identifier

---

# **Chapter 1**

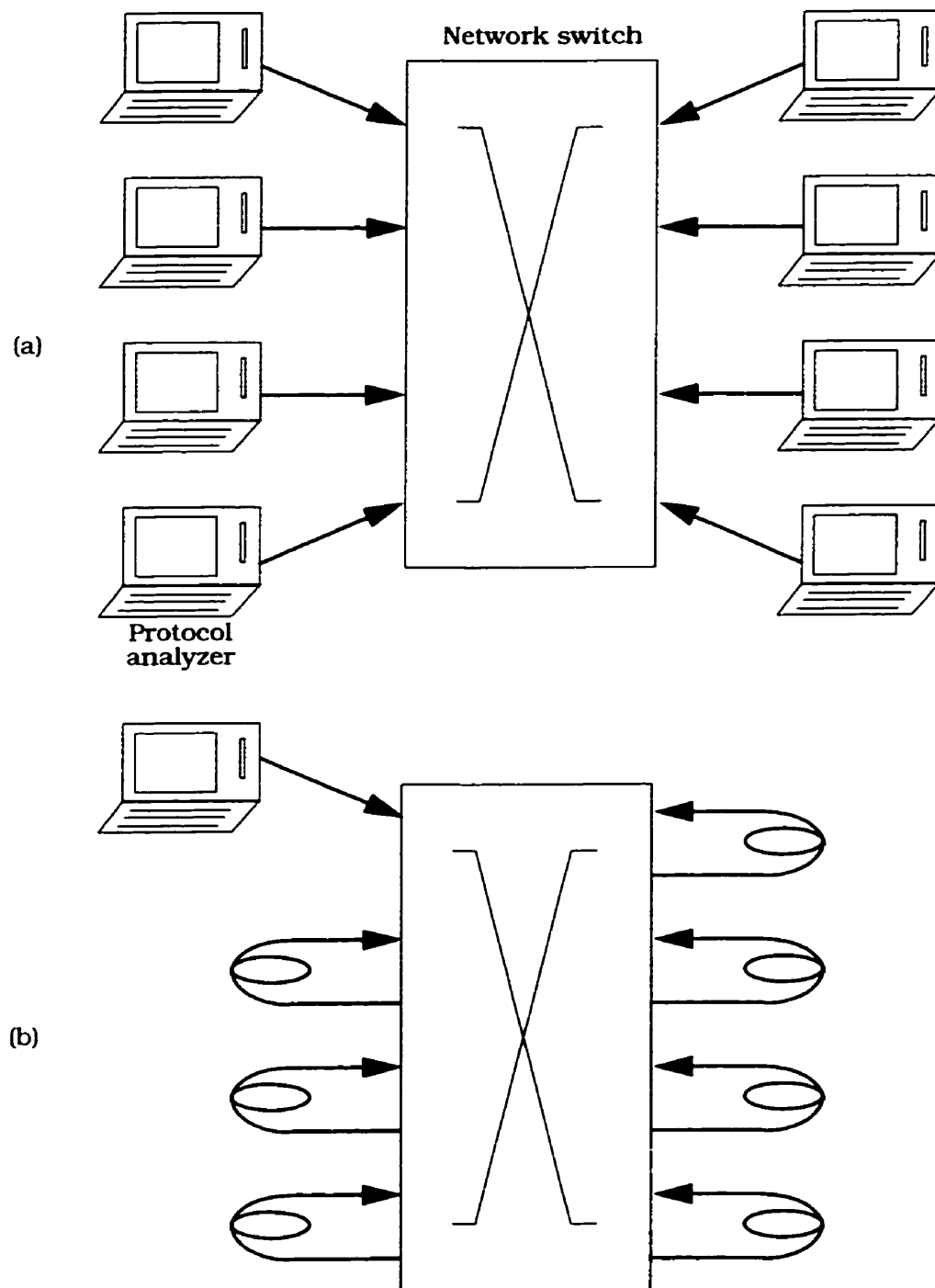
## **Introduction**

### **1.1 Motivation**

In recent months, there has been increased installation and use of high-speed digital telecommunication networks based on technologies such as asynchronous transfer mode (ATM). This additional demand for network resources can be attributed partly to the popularity of applications such as the Internet, video conferencing, distance education, and interactive multimedia as well as to the availability of computer and networking equipment able to support such applications. As a result of this increased need for bandwidth, the importance of efficient installation, use, and management of network resources has never been greater. In order to realize these objectives, a thorough understanding and accurate models of network operation and behavior must be obtained. This can be accomplished through software simulations or by testing with actual network equipment. Testing with physical hardware is the preferred approach for studying network operation as it is inherently more accurate than software models which may not capture all details of the network's behavior.

Unfortunately, there are several problems with the current methods of performing network testing. Rather than attempting to acquire, configure, and operate a variety of common network equipment (e.g. computer terminals, video cameras, telephones,

etc.) to provide traffic for testing, what is typically done instead is to configure test equipment such as network protocol analyzers to act as traffic sources for loading the equipment or network being studied. Figure 1.1(a) illustrates such a set-up, where protocol analyzers are being used to drive the inputs of a network switch for the purpose of performing a switch load test. In this test, the behavior of the switch is studied under a variety of realistic as well as some not-so-realistic traffic loading conditions. With protocol analyzers costing hundreds of thousands of dollars each and with switches or networks having many ports to populate, it is often not possible to have each of the inputs of a switch or network receive traffic from its own network analyzer. What is done instead is to have a single protocol analyzer drive one of the inputs to a switch and to supply the other input ports with traffic looped-back from the outputs of the same switch using an arrangement like that shown in Figure 1.1(b). While this approach to providing data traffic to each of the inputs of a switch is certainly less expensive in terms of the amount of test equipment required, it creates an unrealistic test environment as there is no independence in the traffic streams arriving at each of the switch inputs; the properties of the data traffic arriving at one of the switch inputs may be similar to that appearing at other ports. The timing of the traffic stream may also become distorted as it is looped through the switch. Protocol analyzers are also limited in the types of network traffic that they are capable of generating; it is quite difficult to configure a network analyzer to provide a stream of traffic with the long-range dependence and self-similarity properties that have been found to occur in some types of network traffic ([5], [17]).



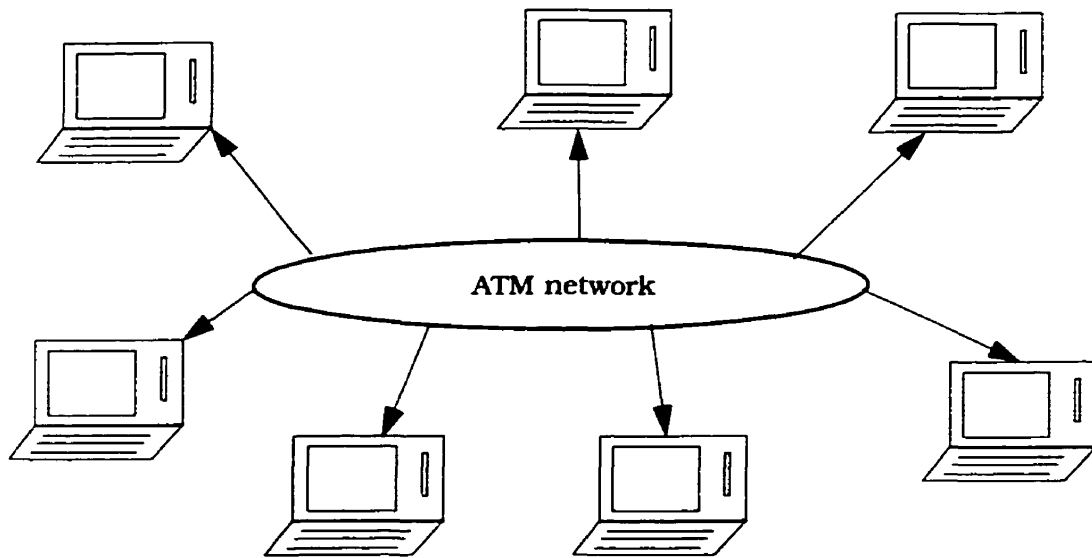
**Figure 1.1** Switch load test: (a) with protocol analyzers driving individual input ports, (b) with input ports connected in loop-back

Characterization of network behavior and that of individual traffic sources suffers as a result of the capabilities of currently-available commercial network test equipment. In an environment like that shown in Figure 1.2, protocol analyzers are used to capture traffic leaving the output ports of a network. Traffic recorded at each of the outputs is analyzed in an attempt to characterize properties of the traffic sources and examine how data transfers across a network are affected by various network loading conditions. In order to develop models that can accurately describe and predict network behavior, it is essential that these capture intervals last for durations as long as possible. This is necessary for observing the long-range dependence tendencies which have been found to occur with various types of data traffic in analysis performed over periods of hours, days, and weeks. Commercial test equipment typically<sup>1</sup> permits only a few seconds of OC-3 ATM network traffic to be captured. This is not a sufficient amount of data from which accurate models of network behavior can be developed. Also, the expense of the network analyzers limits the number of ports from which traffic can be captured.

---

1. GN Nettest has recently introduced an option for its ATM network analyzers which would allow approximately 81 million cells, or up to 4 minutes of OC-3 traffic, to be captured. Unfortunately, only a fraction of the bandwidth in an OC-3 stream can be captured.

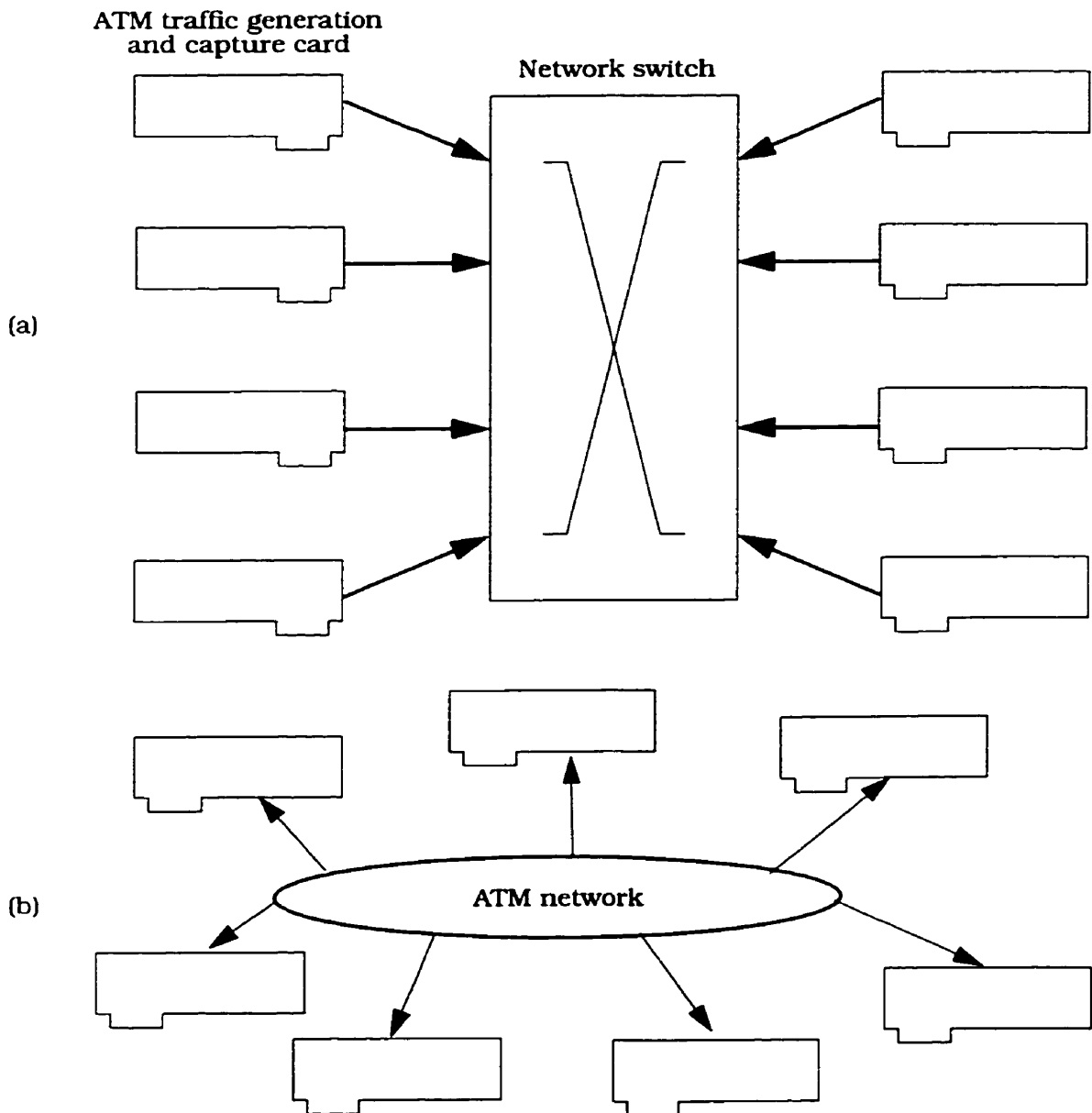




**Figure 1.2** Capturing network traffic for analysis and characterization

A network interface card designed specifically for the purpose of generating and capturing network traffic and available at a low cost is one possible approach to overcoming the limitations of performing network testing and analysis with current commercial test equipment. The intent of this traffic generation and capture card would not be to duplicate the protocol checking and error injection and analysis capabilities of network protocol analyzers. Rather, the card would extend the traffic generation and traffic capture capabilities of these devices. The traffic generation capability would be enhanced to allow a larger variety of network traffic sources to be simulated including those that exhibit properties of self-similarity. Extended traffic capture capabilities would permit network traffic to be logged for periods of time long enough for accurate characterization and modelling of traffic sources and network behavior to be realized. The objective of maintaining a low cost is crucial for making this traffic generation and capture card an economically-viable alternative to using protocol analyzers for generating and capturing network traffic. An inexpensive implementation of the card would

permit more input and output ports to be populated when performing switch load tests or capturing network traffic. As Figure 1.3 shows, this traffic generation and capture card could find application in testing typically performed with network protocol analyzers.



**Figure 1.3** Applications of the ATM traffic generation and capture card: (a) in a switch load test, (b) for network traffic capture

## **1.2 Report Structure**

The purpose of this thesis is to describe the design and implementation of a traffic generation and capture card developed for testing ATM networks. The next chapter will provide a brief introduction to ATM and background for the remainder of the discussion on the card's development. A discussion of the features and functionality required of the ATM traffic generation and capture card will be presented in Chapter 3. Chapter 4 of this thesis will describe the design and implementation of the generation and capture card while Chapter 5 will discuss the software developed to support the card's operation. Testing of these hardware and software modules will be described in Chapter 6. Chapter 7 will discuss the card's design and Chapter 8 will provide suggestions for future work and development on the card in terms of extensions and improvements. Chapter 9 will conclude the thesis and summarize the results achieved.

The ATM traffic generation and capture card discussed in this thesis was developed in partnership with *TRLabs*. As a result, certain aspects of the card's design and operation are proprietary and cannot be discussed in detail.

---

# **Chapter 2**

## **Asynchronous Transfer Mode**

Asynchronous transfer mode (ATM) is a high-speed, digital telecommunications technology which was designed for the purpose of integrating a variety of services and applications on a common network. Applications for which ATM was designed to support range from low bandwidth, constant bit rate telephony to bursty, high-bandwidth video and data transfers. ATM is the transport used in the implementation of B-ISDN (broadband-integrated services digital network).

This chapter will describe the basic features and operation of ATM networks. Specifically, the chapter will discuss asynchronous transfer mode as it relates to the ATM traffic generation and capture card implemented in this thesis project.

### **2.1 Supported Traffic Types**

ATM is designed to be capable of supporting a variety of network traffic types including:

- traffic generated at constant bit rates and at variable bit rates in large or small bursts
- traffic originating from source applications that require a connection to be established with the destination application prior to the transmission of data and traffic

from applications that transfer data without a connection present

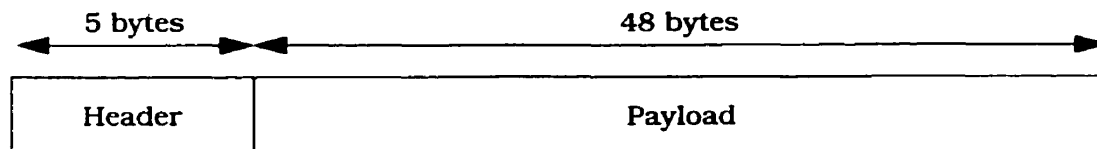
- traffic from applications which are sensitive to any loss of data caused by mis-routing or corruption during the transfer and traffic from applications which are more tolerant to the occurrence of data loss
- traffic from applications that require data to be transferred in real-time with minimal delay and jitter as well as traffic from applications having less-stringent requirements in the delays encountered during the data transfer

Each of these ATM network traffic types can be classified as belonging to one of several service categories defined by the Telecommunications Standardization Sector of the International Telecommunications Union (ITU-T). A constant bit rate (CBR) service class is defined for traffic sources which generate data at fixed bit rates and require data to be transported in real-time but without any error checking, flow control, or other processing applied to the traffic stream. Uncompressed audio and video streams are typical applications which would be classified as belonging to the CBR service category. The variable bit rate (VBR) service class is one which serves applications which generate data traffic at a variable rate. This service class is divided into a real time (RT) sub-class, where stringent requirements are placed on the transfer delay and the amount of jitter experienced, and a non-real time (NRT) sub-class, where the timing of the data transfer is less critical. Compressed video traffic in a video-conferencing environment would fall into the RT-VBR service class while e-mail with a video attachment would belong to the NRT-VBR class. The available bit rate (ABR) service class is defined for bursty traffic sources which have known minimum levels of bandwidth requirements. For the ABR service class, the network guarantees to support this minimum data transfer rate and may provide additional bandwidth, if resources become available. Web browsers are one example of an application belonging to the ABR class. The unspecified bit rate (UBR) service class is for applications which place no minimum bandwidth requirements on

the network, but will accept and utilize whatever level of bandwidth that the network is able to provide at any given time. File transfers and e-mail messages fall into this service class.

## **2.2 Cell Structure**

All data is transferred across an ATM network in the form of a fixed-length block called a cell. As Figure 2.1 illustrates, ATM cells are 53 bytes in size and are comprised of a 5-byte header and a 48-byte payload. The header portion of a cell contains the information needed to ensure that the cell is correctly transferred across a network from a source node to one or more destinations. The payload portion of a cell contains the data that is to be transferred. The cell payload can contain user data, control signalling, or operation and administration information.

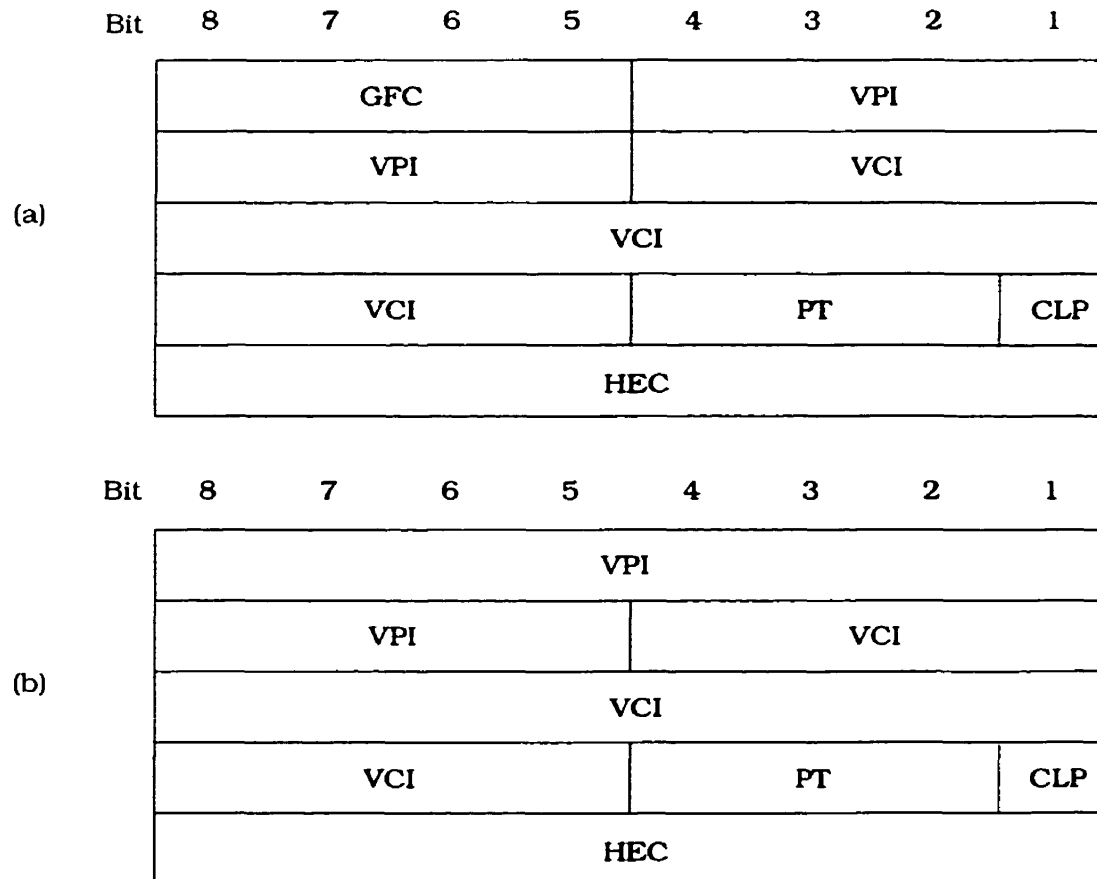


**Figure 2.1** ATM cell structure

There are two header formats defined for ATM cells. The user-network interface (UNI) header structure is the format used for cells transferred between a user application and an ATM network while the network-node interface (NNI) header format is used for cells transferred between nodes within the network. The UNI and NNI header formats, shown in Figure 2.2, differ only in the generic flow control and virtual path identifier fields.

The 4-bit generic flow control (GFC) field is present only in the UNI cell header format. This field was intended to implement flow control between a user and a network.

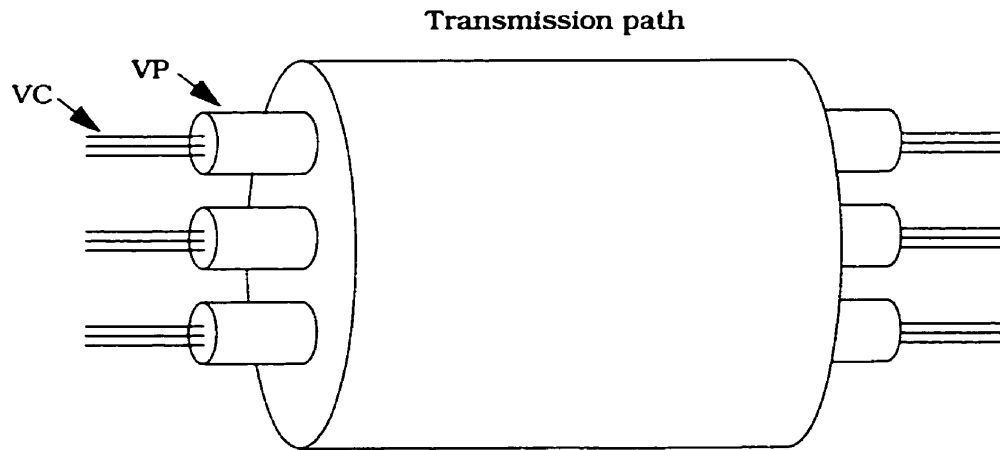
but its use is presently not defined in ATM standards ([3]). As a result, the GFC field in a UNI cell header is typically set to zero and ignored by ATM switches in a network.



**Figure 2.2** ATM cell header formats: (a) user-network interface style, (b) network-node interface style

The virtual path identifier (VPI) and virtual channel identifier (VCI) fields contain the address information necessary for switching a cell during its transfer across an ATM network. The VPI field is 8 bits in length in the UNI header format and 12 bits in the NNI header format. The VCI field is 16 bits long in both the UNI and NNI header formats. The VCI field allows for differentiation of cells from a number of virtual channels (VCs) sent over a common virtual path (VP). The VPI field identifies a particular VP in a trans-

mission path. Figure 2.3 illustrates the relationship between virtual channels, virtual paths, and transmission paths.



**Figure 2.3** Relationship between VCs, VPs, and transmission paths

The 3-bit payload type (PT) field is used to describe the type of data being carried in the cell payload. This field indicates whether the payload is carrying user data or network management information. When the payload contains user data, the PT field can also indicate whether a cell has experienced congestion in its transfer across a network.

The cell loss priority (CLP) field is a single bit used to indicate the priority of an ATM cell. When traffic congestion occurs in a network, ATM switches in the network can drop cells in an attempt to reduce the level of congestion. The CLP bit allows a network switch to differentiate between the two priority levels assigned to cells: low priority cells are discarded by a switch before those with a high priority are dropped.

The last field in the cell header is an 8-bit header error control (HEC). The HEC field is a checksum applied to the first four bytes of the cell header in order to prevent the mis-routing of cells with errored headers. This field makes it possible to correct all single-bit errors and detect most multi-bit errors that can occur in the cell header. The HEC is not applied to the cell payload.



Cell transmission begins with the most-significant bit of the first byte of the cell header and ends with the least-significant bit of the last (or forty-eighth) byte in the cell payload.

## **2.3 Connections**

ATM networks are connection-oriented. Before data can be transferred between a source and a destination, a connection with the network must first be established. A connection is a set of links which define the path across the network that all cells transferred between these two entities will follow. Connection establishment reserves network resources for data transfer between a particular source and destination or destinations, in the event that data is to be broadcast. As a result of the connection-oriented nature of ATM networks, cells will be received at the destination node in the same order that they were sent by the source, provided that no cells are discarded or mis-routed while in transit across the network.

A connection with an ATM network can be established as one of two types: as a permanent virtual connection or as a switched virtual connection. A permanent virtual connection (PVC) is a connection that is created and released manually by the network provider's personnel. A switched virtual connection (SVC), on the other hand, is a connection which is created and released through meta-signalling exchanged between network switches and equipment at the source and destination nodes. SVCs are established without human intervention by hardware and/or software within the network switches and user equipment. SVCs are dynamically established and released, typically existing only for the duration that data needs to be transferred, while PVCs tend to be maintained for longer periods before and after a data transfer is completed. SVCs have the advantage of making more-efficient usage of network resources than

PVCs.

Regardless of whether a connection is established as an SVC or as a PVC, the connection set up procedure involves negotiation between a data source and the ATM network to which it is attached. During connection set up, the data source indicates to the network its quality of service (QoS) requirements for the connection using parameters which include its tolerance for cell loss, delay in the cell transfer, and variation in the cell transfer delay. The data source also specifies the peak, average (sustained), and minimum rates that it requires for cells being transferred across the network. The network evaluates these requirements and determines whether or not it has sufficient resources to be able to meet the source's QoS requirements. The connection request is accepted only if the network is able to support the source's QoS requirements for the connection and if the QoS requirements of existing connections are not compromised as a result of accepting the connection. The request is refused if either of these two conditions is not satisfied. In the event that a network refuses a source's connection request, the source can resubmit its request if it is willing to accept a lower QoS for the connection. The values of the QoS parameters that finally get agreed upon by the two entities become what is known as the traffic contract.

Once a connection has been established, the data source is informed of the values that it must assign to the VPI and VCI fields of all data cells that it sends to the network. The specific values of the VPI and VCI fields allow cells associated with one connection to be distinguished from cells sent on other connections by other data sources.

When a data source begins transmitting cells on an established connection, it is the responsibility of the switches in the network to police the traffic on the connection through a mechanism known as usage parameter control. This policing of a source is

necessary to ensure that it does not violate the terms agreed upon in the traffic contract, ultimately causing network congestion or compromising the QoS of other connections. Cells which are found to be in violation of the traffic contract can be marked to be discarded immediately or at some later point in time.

## **2.4 Payload Formatting**

As discussed earlier, data is transferred across an ATM network in the payload of cells. The sending application must therefore format the data that it wishes to transfer to fit within the cell payload. The application at the destination node must extract the data from the payload of the cells it receives and reformat the data to meet its usage requirements. Formatting of the data being transferred to the payload of an ATM cell involves more than just simply chopping the data into blocks of 48 bytes. In order to meet the QoS requirements of each of the service classes, additional control information is inserted into the cell payload along with the data. The control information that is added can include:

- a payload sequence number
- an indication of the relative position of the payload within the frame from which it originated
- the length of the frame from which the cell payload was extracted
- an identifier for differentiating between multiple sessions multiplexed over a common connection
- error correction/detection applied to the entire cell payload or only the control information included in the payload

The specific type, format, and amount of control information inserted in the payload is dependent upon the service class of the connection

## **2.5 Cell Transmission**

ATM transfers cells across a network asynchronously, as cells become available from source applications. Rather than assigning a dedicated time slot during which cells from a particular connection are to be transferred, cells are transmitted to an ATM network as they are created by a source. Cells on different connections are multiplexed to form a single stream of cells which are transferred across a network. The order that cells from different connections are multiplexed into a single stream is dependent upon their order of arrival and, possibly, the QoS requirements of the individual connections involved. This asynchronous approach to cell transfer makes most-optimal use of network resources by allowing bandwidth which is freed-up by a source which temporarily does not have any data to send to be shared by other sources which have an immediate need for data to be transferred. In the absence of cells carrying user or control information (also referred to as assigned cells), idle/unassigned cells are inserted into the stream transferred to the network as a means of cell rate decoupling.

At switches within an ATM network, cells arriving on an incoming link are queued to allow the VPI and VCI fields in the header to be examined. These fields are decoded to select the switch's output port to which the cell should be routed. ATM switches may also examine the HEC field to detect and correct errors in the cell headers, translate the VPI and VCI fields for transfer further across the network, or drop cells in an attempt to control network congestion. Cells leaving an output port of a switch are multiplexed to form a single cell stream to be transferred to other nodes in the network.

There are a number of physical transport mechanisms which can be used to transfer ATM cells across the transmission paths between nodes in an ATM network. Cells can be transferred using framed transports such as the synchronous optical network (SONET) and synchronous digital hierarchy (SDH) or even in their raw form with-

out any additional transport overhead. Some of the physical transports for transferring ATM cells, as defined by organizations such as the ATM Forum, the ITU-T, and the American National Standards Institute (ANSI), include:

- 44.736 Mbps DS-3/T-3 transport over coaxial cable
- 51.84 Mbps SONET STS-1 over category 3 unshielded twister pair
- a 100 Mbps data rate over a fiber optic medium using the FDDI/TAXI (fiber distribute data interface/transparent asynchronous exchange interface) architecture
- 155.52 Mbps SONET STS-3c/SDH STM-1 over fiber optic cable or coaxial cable
- a 155.52 Mbps data rate over fiber or category 5 unshielded twisted pair using the Fiber Channel architecture
- 622.08 Mbps SONET STS-12c/SDH STM-4 over a fiber optic medium
- and a number of low-speed transports which provide data transfer rates of a few mega-bits per second over shielded and unshielded twisted pair and coaxial cable

At the output port of a source node, ATM cells are mapped to one of these transports for transfer across the physical links between nodes in a network. At switches in the network, cells are extracted from the physical transport, are switched and multiplexed with cells from other sources, and are mapped again to a physical transport for transfer further through the network. When the destination node is reached, the cells are extracted from the transport and sent to the receiving application.

One physical transport that is commonly used, and the one that will be important in this thesis, is the SONET hierarchy. The remainder of this section will describe the basics of this transport mechanism.

## **SONET**

The SONET hierarchy is an optical communications interface standard maintained by ANSI. It is a framed transport which allows direct, synchronous multiplexing

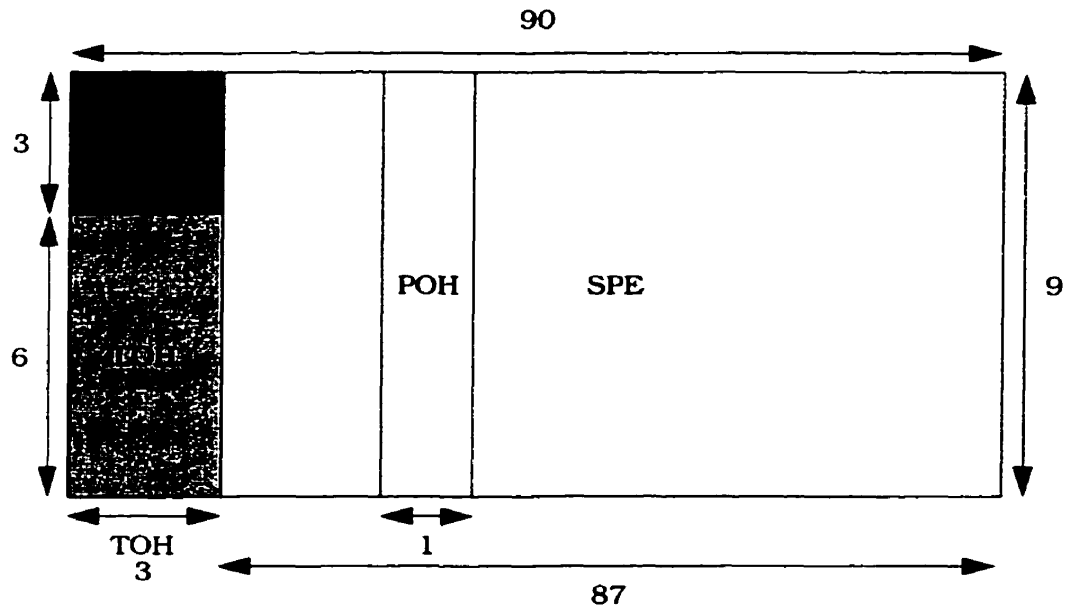
of traffic streams to be used to realize high data transfer rates. In addition to being a transport for data, SONET provides network management and maintenance capabilities.

The basic structure used for data transfer in the SONET hierarchy is the level 1 optical carrier (OC-1) frame as shown in Figure 2.4. The OC-1 frame is 9 rows by 90 columns in size, with the intersection of a row and a column representing one byte of data. Three columns in the frame contain transport overhead (TOH) consisting of 9 bytes of section overhead (SOH) and 18 bytes of line overhead (LOH). The remaining 87 columns in the frame form the synchronous payload envelope (SPE). The SPE contains one column of path overhead (POH) data and 86 columns of user information in which services such as ATM are mapped. The section, line, and path overhead together provide such transport management and diagnostic information as frame alignment detection, performance monitoring, fault isolation, a pointer to the start of the SPE in the frame, and a voice channel for network maintenance personnel. The equivalent electrical version of an OC-1 frame is referred to as a level 1 synchronous transport signal (STS-1).

SONET frames are transferred beginning with the byte at the left-most column and top-most row of the frame, proceeding rightward across the row towards the last column, and continuing in a similar manner with the second and subsequent rows until all the bytes in the frame have been transmitted; the cycle begins again with the next frame. Frames at an OC-1 rate, as well as at all other optical carrier levels, are generated and transmitted across a link at a rate of 8000 frames per second. This frame size and transmission rate allows a data transfer rate of 51.84 Mbps to be realized. Of this 51.84 Mbps, 49.536 Mbps is available for transferring user information.

As mentioned previously, the user information that is to be transferred is mapped into the SPE portion of a SONET frame. In the case where SONET is used as the transport for ATM, cells are mapped head to tail into the SPE in a manner like that

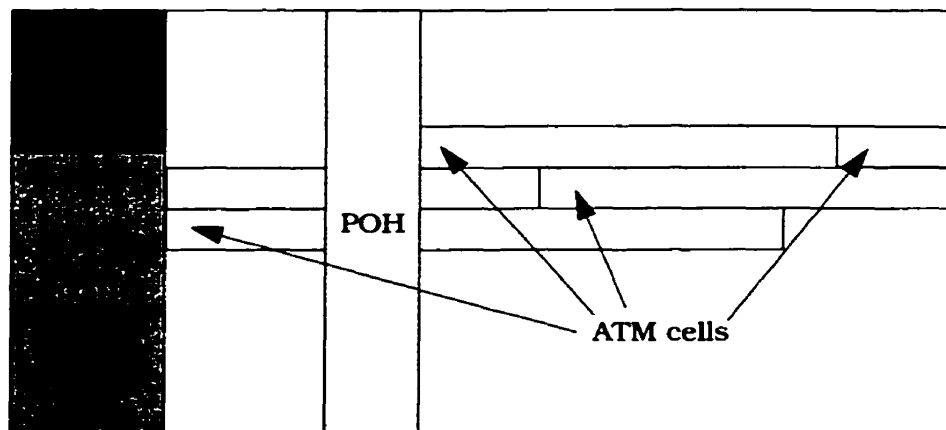
shown in Figure 2.5. The SPE can begin at any location in the payload of a frame and can, consequently, extend across the boundary of one frame and into a second frame. A pointer in the line overhead part of the transport overhead indicates where the SPE begins and facilitates the demapping of the payload from the frame when it reaches the destination node in a network.



**Figure 2.4** SONET OC-1 frame structure

Data transfer rates higher than 50 Mbps are achieved by multiplexing a number of OC-1 signals together to form a new signal in the SONET transport hierarchy. An OC- $n$  signal is created when  $n$  OC-1 signals are multiplexed, with the OC- $n$  frame being  $n$  times larger and having  $n$  times the data transfer capacity of an OC-1 signal. In the OC- $n$  frame, the SOH, LOH, and POH sections of the frame also increase  $n$  times in size from the OC-1 frame. As an example, Figure 2.6 illustrates the OC-3 frame that is formed when three OC-1 frames are multiplexed; an OC-3 frame is 9 rows by 270 columns in size and contains 9 columns of TOH and 3 columns of POH. In general, an OC- $m$  signal

is formed when  $k$  OC- $l$  signals are multiplexed, where  $k$ ,  $l$ , and  $m$  are integers and  $m$  is the product of  $k$  and  $l$ . An OC- $m$  signal is  $k$  times larger and has  $k$  times the data transfer rate of an OC- $l$  signal. The SONET hierarchy allows signals which have already been multiplexed to be multiplexed further to achieve yet higher data transfer rates. Four OC-3 signals, for example, can be multiplexed to create an OC-12 signal; this is also equivalent to multiplexing 12 OC-1 signals. Lower-level signals in a multiplexed stream can be recovered through demultiplexing of the higher-level signal. Table 2.1 lists some of the transport signals defined in the SONET hierarchy, the resulting data transfer rates, and the equivalent level of the synchronous transport module (STM) signal in the synchronous digital hierarchy (SDH).

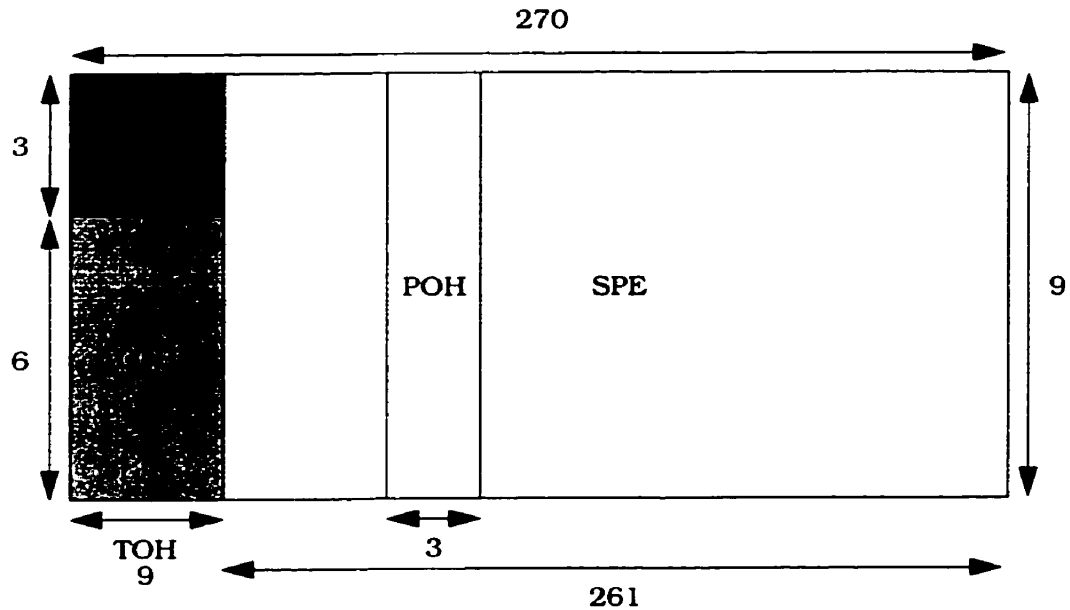


**Figure 2.5** Mapping of ATM cells to an OC-1 frame

Higher-level signals in the SONET hierarchy are formed through synchronous, byte-interleaved multiplexing of lower-level signals. Using the notation from the previous paragraph, an OC- $m$  signal is created by taking a single byte sequentially from each of the  $k$  OC- $l$  frames. This creates an arrangement where the first group of  $k$  bytes in the OC- $m$  frame is the first byte of each of the  $k$  OC- $l$  frames, the second group of  $k$  bytes in the OC- $m$  frame is the second byte of each of the  $k$  OC- $l$  frames, etc. As the multiplexing



is synchronous, bytes which were adjacent to each other in the lower-level OC-*l* signal are always only *k* bytes apart in the OC-*m* signal.



**Figure 2.6** SONET OC-3 frame structure

The SONET hierarchy also defines transport signals which allow higher data transfer rates to be realized without using the byte-interleaved scheme of multiplexing lower-level signals described previously. These signals are differentiated from the optical carrier signals described earlier by appending a letter 'c' designator to the optical carrier level value (OC-3c, for example). OC-*nc* signals have a frame size and data transfer rate which is identical to that of an OC-*n* signal. The frame structure of an OC-*nc* signal is identical to that of an OC-*n* with the exception of the POH portion of the frame; the POH in an OC-*n* signal is *n* columns wide, but is only a single column wide in an OC-*nc* frame. Data is mapped to the SPE of OC-*nc* signal in a manner similar to what was described previously for an OC-1 signal. Because an OC-*nc* signal is not created from multiplexing of lower-level signals, a byte-interleaved multiplexing scheme is not used to

map data into the SPE of these frames. Consequently, an OC-*nc* signal cannot be demultiplexed.

**Table 2.1** Common SONET/SDH transports

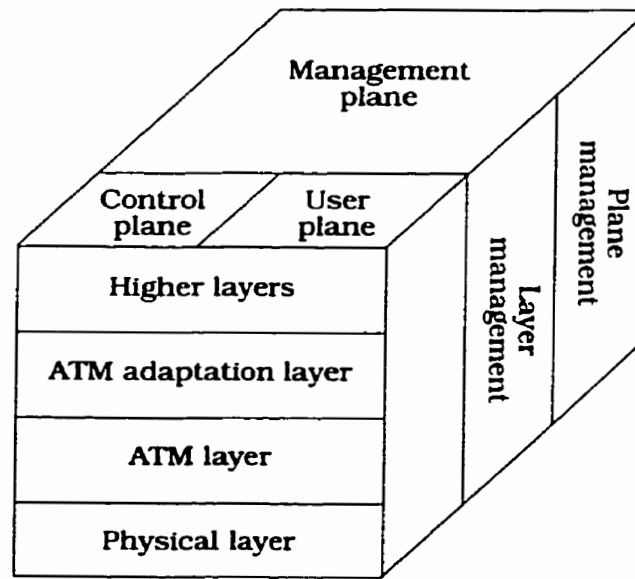
Data transfer rate (Mbps)	SONET optical carrier	SONET electrical carrier	SDH carrier
51.84	OC-1	STS-1	-
155.52	OC-3	STS-3	STM-1
622.08	OC-12	STS-12	STM-4
2488.32	OC-48	STS-48	STM-16
9953.28	OC-192	STS-192	STM-64

## **2.6 ATM Reference Model**

Figure 2.7 shows the protocol reference model defined for ATM. This section will briefly describe the function of each of the layers and planes in this reference model and relate them to the discussion earlier in the chapter.

### **Physical Layer**

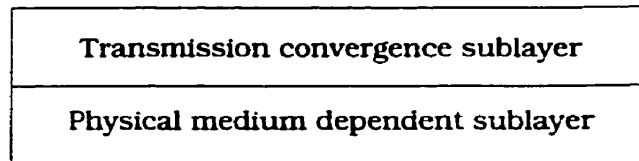
The lowest layer in the ATM protocol reference model is the physical layer (PHY). The purpose of this layer is to ensure accurate transmission and reception of ATM cells across the links that connect nodes in an ATM network. It serves as an interface between the physical transmission medium used for data transfer between network nodes and the next-higher layer in the reference model, the ATM layer. The physical layer is further sub-divided into two sublayers, the transmission convergence (TC) and physical medium dependent (PMD) sublayers, as illustrated in Figure 2.8.



**Figure 2.7** ATM reference model ([13])

The transmission convergence sublayer looks after the mapping and demapping of ATM cells to and from one of the physical transport mechanisms listed earlier. The specific functions performed by the TC sublayer include transmission frame generation and recovery, transmission frame adaptation, cell delineation, cell rate decoupling, and HEC field generation and verification. At the source node in an ATM network, the TC sublayer calculates the value inserted in a cell's HEC field, creates the frame used by the physical transport mechanism, and maps cells to the data payload portion of the frame. The TC sublayer also inserts idle cells into the frame's payload as a means of cell rate decoupling in the event that the total data transfer requirements are less than the transport mechanism's data transfer capacity. The TC sublayer at the destination node processes the received data stream to locate the start of the transport mechanism frame structure as well as the beginning of ATM cells so that they can be demapped from the frame's payload. The TC sublayer at the destination node discards idle cells that were

inserted at the source for cell rate decoupling. Also at the destination, the TC sublayer recalculates a cell's HEC and compares it to the HEC field that was received in the header for the purposing of detecting, and possibly correcting, errors that have occurred in the cell header. The TC sublayer typically does not pass cells with errored headers (that cannot be corrected) up to the ATM layer.



**Figure 2.8** Sublayers within the physical layer ([13])

The physical medium dependent sublayer of the physical layer handles the coding of transmission frames for the purpose of transmission across links in a network. At a source node, the PMD sublayer converts individual bits in the transmission frame received from the TC sublayer into electrical or optical signals (depending on the transmission medium used) with the appropriate voltage or light intensity levels and bit timing. At a destination node, the PMD sublayer forms the transmission frame from the signal levels received on the transmission media.

## **ATM Layer**

The next-higher layer in the reference model is the ATM layer. This layer interfaces with the physical layer and the ATM adaptation layer. The ATM layer exchanges ATM cells with the physical layer, passing down ATM cells that it wishes to transmit to the physical layer and receiving cells that the physical layer has demapped from the physical transport mechanism. The payload portion of a cell is passed between the ATM layer and the ATM adaptation layer.

Some of the responsibilities of the ATM layer in the protocol reference model

include:

- attaching a cell header to payload information received from the ATM adaptation layer at the source node where the cell begins its transfer through network
- removal of the header from the cell and passing of the payload up to the ATM adaptation layer when the cell reaches its destination
- multiplexing and demultiplexing of cells from different traffic streams
- switching and address (VPI/VCI) translation of cells at intermediate nodes in a network
- cell rate decoupling through insertion and removal of idle/unassigned cells in the traffic stream
- connection establishment and release using connection admission control procedures
- processing of a header's CLP bit for the purpose of congestion control in a network
- traffic policing (usage parameter control), shaping, and accounting (for billing purposes) of data originating from individual sources
- processing of cell headers to identify reserved meta-signalling or network management cells
- all cell processing and management tasks necessary for proper network operation and for meeting a variety of QoS requirements

### **ATM Adaptation Layer**

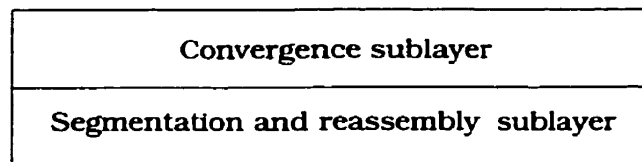
The layer above the ATM layer in the reference model is the ATM adaptation layer (AAL). The AAL acts as an interface between the ATM layer and higher layers in the model. This layer operates to map the services required by user applications to the services provided by the ATM layer. In addition to translating between the data format used by sending and receiving applications and the ATM cell structure which is used for transferring data across a network, the AAL is also responsible for the overhead informa-

tion included in the payload for the purpose of meeting applications' QoS requirements. This overhead serves as a mechanism for handling variations in cell transfer delays, the occurrence of transfer errors, the mis-insertion or loss of cells during transport, and the recovery of source timing information at destination nodes. The AAL deals with ATM cell payloads only and does not have access to the cell headers.

A number of types of ATM adaptation layers are defined to meet the differing requirements of each of the available service classes. ATM adaptation layer type 1 (AAL-1) is defined for the CBR service class which requires data generated at a constant bit rate to be transferred in real-time and with a connection to be established. Included with the user data in the payload of AAL-1 cells are fields for a sequence number and a checksum for detecting errors in the sequence number. AAL-2 is for the NRT-VBR service class which transfers variable bit rate data in real-time with a connection present. This AAL includes sequence number, information type identifier, and length indication fields with the user data as well as a field which serves as a checksum over the entire cell payload. AAL-3/4 is for connection-oriented or connectionless applications which require data, generated a variable bit rate, to be transferred without loss, but not necessarily in real-time. This AAL includes a length field, frame sequence numbers, a location identifier (for the position of the cell in the frame from which it originated), a multiplexing identifier field, and an indication of the type of data in the payload of the cell. Like AAL-3/4, AAL-5 is for connection-oriented or connectionless applications. It attaches a field for the length of frame from which the cell was extracted and a checksum. ITU-T specifications ([14]) describe the operation and function of each of the ATM adaptation layers in more detail.

Each of the AAL types share a common sublayer structure: the AAL layer is subdivided into the convergence sublayer (CS) and the segmentation and reassembly (SAR)

sublayer as shown in Figure 2.9. The function of the convergence sublayer is to interface with the sending and receiving applications and to provide the mechanism for supporting each of the defined service classes. This sublayer attaches the AAL-type-specific overhead information to the application data at a source node and removes and processes this overhead at a destination node. The purpose of the SAR sublayer is to do the transformation between the sending and receiving applications' data format and the 48-byte ATM cell payload. At a source node, the SAR sublayer segments the data (received from the convergence sublayer above) into 48-byte blocks to fit within the cell payload. The SAR sublayer at a destination node reassembles the original data structure by joining together each of the cell payloads received from the network, ultimately passing the recovered data up to the CS.



**Figure 2.9** Sublayers within the ATM adaptation layer ([13])

### **Higher Layers**

The *higher layers* in the ATM reference model represent the sending and receiving applications which use the network to transfer data. Among the applications that can be included in this layer are file transfer software, audio and video teleconferencing tools, and world wide web servers and browsers. At a source node, the sending application passes the data that it wishes to transfer across the network to the AAL. When the data has reached the destination node, the receiving application gets the data from the network via the AAL.

## **Planes in the ATM reference model**

The user plane in the ATM reference model handles the transfer of application data from source to destination across a network. The tasks performed by the user plane include data transport, flow and congestion control, and error detection and correction functions.

The purpose of the control plane in the ATM reference model is to support the services that the user plane provides. The control plane looks after connection set up through call admission control, connection tear-down, addressing and switching of data during transfer, traffic policing, and other connection management tasks.

The management plane is concerned with the management of resources and the performance of the network. This plane is subdivided into layer and plane management functions. Layer management is concerned with the operation, administration, and performance of each of the layers in the reference model individually. Plane management looks after the operation and management of the entire system including intercommunication between individual layers and planes in the ATM reference model.



---

# **Chapter 3**

## **Design Specification**

As discussed in the introduction to this thesis, the purpose of the ATM traffic generation and capture card was to provide an economical means of generating and capturing ATM traffic in order to improve the approach currently used for studying network operation and behavior as well as the characterization of network traffic. This chapter will describe the features and functionality that were required for the implementation of the ATM traffic generation and capture card.

### **3.1 Required Features and Functionality**

The design specification for the ATM traffic generation and capture card required that the card be capable of performing the complementary traffic processing operations of ATM traffic generation and ATM traffic capture. Each of these two traffic processing operations was to be performed by a single network interface card. Under traffic generation, the card was to act as a source of network traffic by providing a stream of ATM cells. This stream was to be used to drive the input of a network (or a switch) so that the behavior of the network could be studied under a range of traffic loading conditions. Among the types of network traffic that the card had to be capable of generating were those streams that exhibited characteristics of self-similarity and long range depen-

dence. When capturing traffic, the card was required to record the timing information of an ATM traffic stream that it received from an output port of a switch or a node in a network. The network traffic information that the card captured was to be used for developing models of network operation and network traffic behavior. Although the design requirements specified that the card be capable of both ATM traffic generation and capture, the card was not required to perform both traffic processing operations simultaneously. The card was also not required to perform any connection establishment tasks; rather, it was to generate and capture ATM traffic on connections which had been created ahead of time as PVCs.

The design requirement suggested the duration of the traffic processing interval for which the card had to be capable of operating. In order to improve upon the accuracy of the models that are developed to describe network behavior and to characterize network traffic, the ATM traffic generation and capture card implemented in this project had to extend the traffic processing intervals of conventional network test equipment. This was necessary so that the rare events which occur infrequently during network operation, but which are essential for proper network modelling, could be specified in the generated ATM traffic streams and could be recorded in the captured traffic streams. A target traffic processing interval of a few hours in length was recommended by the design specification.

The ATM traffic generation and capture card's physical layer interface to an ATM network was also determined by the design specification. The card had to process cells using the UNI cell header format and was required to interface to an ATM network through a 155.52 Mbps OC-3c physical transport.

Cost was another aspect of the ATM traffic generation and capture card's implementation that was defined in the design specification. In order for the card to be a via-

ble alternative to testing with commercial network test equipment, the design specification required that the card be built at a low cost. A target cost of a few thousand dollars was placed on the hardware needed for the card's fabrication.

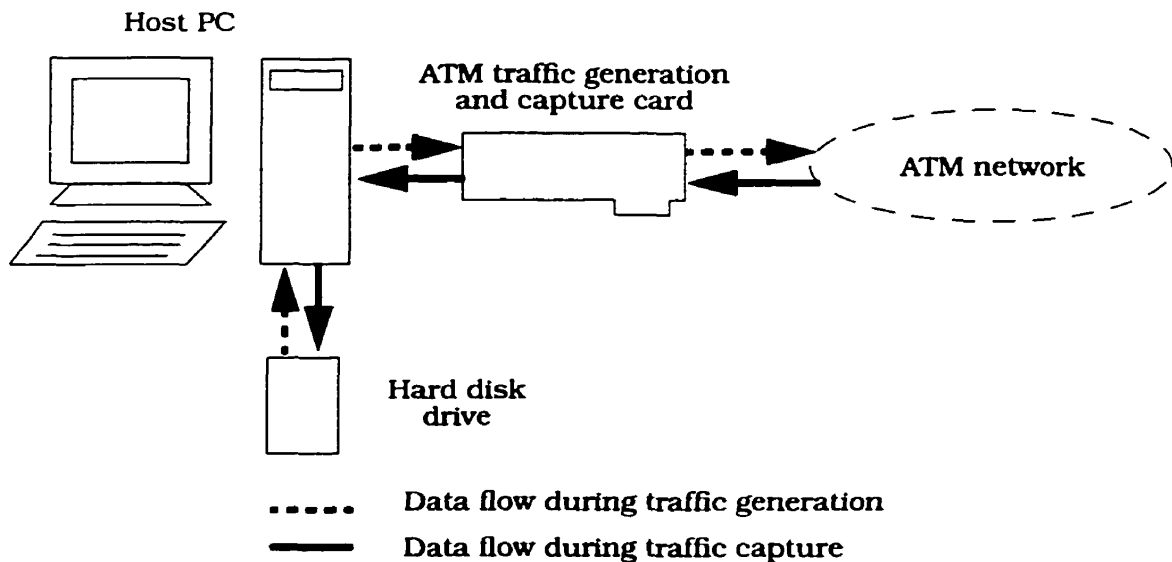
In order to achieve a low hardware cost, the design specification recommended that commercially-available components be used whenever possible in the design of the ATM traffic generation and capture card. This suggestion was made in an attempt to avoid the high fabrication costs associated with the development of an application-specific integrated circuit (ASIC). In the event that no commercial part was available to realize a particular operation, the specification recommended that a field-programmable gate array (FPGA) be used to implement the required functionality.

## **3.2 Host Interface**

The design specification required that the ATM traffic generation and capture card have an interface to a host computer. This interface was required for several purposes. First, it was necessary to allow a host to configure and control the operation of the card. The host had to be able to select whether the card was to perform traffic generation or traffic capture. The host also had to be able to perform any initialization or run-time management tasks as necessary for either traffic processing operation. The interface was required to transfer the ATM traffic stream between the card and the host. During traffic generation, the interface was to be used by the host to supply the card with the ATM traffic stream to "generate" to the network. This traffic stream could be one which the host creates in real time or one which it has read from a pre-computed traffic log file stored on disk. When capturing traffic, the interface was needed by the host to read the ATM traffic stream which the card has received from a network. This allows the host to analyze the traffic stream as it received from the network or to store to

disk for processing off-line. Figure 3.1 illustrates the operating environment of the ATM traffic generation and capture card and the orientation of the ATM traffic stream flow during each of its two modes of operation.

In keeping with the requirement that the ATM traffic generation and capture card be a low-cost alternative to conventional network test equipment, the design specification stipulated that an IBM-compatible personal computer (PC) be employed as a host for supervising the operation of the card. A PC was selected as the host because of its popularity and relatively low cost.



**Figure 3.1** Operating environment of the ATM traffic generation and capture card

### **3.3 ATM Traffic Stream Representation**

As a result of the combined requirements for long traffic generation and capture periods and the high bandwidth of the traffic generation and capture card's ATM network interface, large amounts of data storage space would have been needed for storing raw ATM cells in the traffic log files. For instance, capturing one hour of ATM network traffic at an OC-3c rate requires approximately 67 giga-bytes of disk space on a host PC,

counting storage of the ATM cells only and neglecting any storage overhead that may be needed. A similar amount of storage space would be required for the ATM traffic stream that the card is to supply to the network during traffic generation.

In order to realize the required network transfer rates and traffic processing intervals without any specialized and expensive hardware, it was necessary to apply some of type of compression to the ATM traffic streams. The design specification defined a traffic encoding scheme<sup>2</sup> that made substantial reductions to the amount of storage space required for representing the ATM traffic streams. This encoding scheme recorded the timing data of an ATM traffic stream while maintaining sufficient information such that an encoded traffic stream could be expanded into a stream of ATM cells. This can then be sent to a network during traffic generation and a captured stream of ATM cells could be characterized once encoded. The specific variation of the encoding implemented by the card reduced the storage requirements for the ATM traffic streams to less than one giga-byte per hour.

The design specification required that the ATM traffic stream translation tasks be performed by the ATM traffic generation and capture card rather than by the host PC supervising the card's operation. This requirement was made in order to reduce the amount of data transferred between the host and the card and the amount of ATM traffic stream processing performed by a host during traffic generation or capture. By localizing the traffic stream translation to the card, a host might then be capable of supporting the concurrent operation of a number of cards; a host may be able to supply encoded ATM traffic streams to cards configured for traffic generation and/or to receive encoded traffic streams from cards capturing ATM traffic.

---

2. This encoding scheme is currently being patented by *TRLabs* and will not be defined in this thesis.

---

# **Chapter 4**

## **Card Design and Implementation**

Design of the ATM traffic generation and capture card began with identification of the individual tasks necessary for performing generation and capture of ATM traffic. This served to define the major functional elements and establish the card's basic architecture. Completion of this initial design step led to the selection of components to implement these functions and the design and fabrication of prototype cards. Discussion of the card's development in this chapter will follow this structure and will also include a description of other details of its implementation.

### **4.1 Basic Architecture**

Three modules form the basic architecture of the ATM traffic generation and capture card. These modules include an ATM network interface, an interface to a host PC, and a traffic stream translator. This section will define the purpose of each of these modules and will describe how they interact to realize ATM traffic generation and capture.

## **ATM Network Interface**

In order for the card to be able to generate traffic to an ATM network as well as for it to capture traffic from a network, an interface to the ATM network is required. The network interface has to perform the functions of the ATM reference model's physical layer by completing the mapping between ATM cells and the signalling used to transfer cells across a network. The design requirements for the card specified that it be able to interface to a network using an OC-3c physical transport. During traffic generation, the network interface is required to insert cells generated by the card into the SPE of outgoing SONET frames transmitted to a network. When performing traffic capture, the network interface has to extract ATM cells from the SPE of incoming SONET frames received from a network.

## **Host PC Interface**

The design specification also required that the ATM traffic generation and capture card have an interface to a host PC. In addition to providing a host with the means of controlling and configuring the operation of the card, this interface is necessary for transferring the encoded ATM traffic stream between the card and the host. When generating traffic, the host supplies the card with the ATM traffic stream to send to the network. During traffic capture, the interface is used by the host to receive the ATM traffic stream captured by the card.

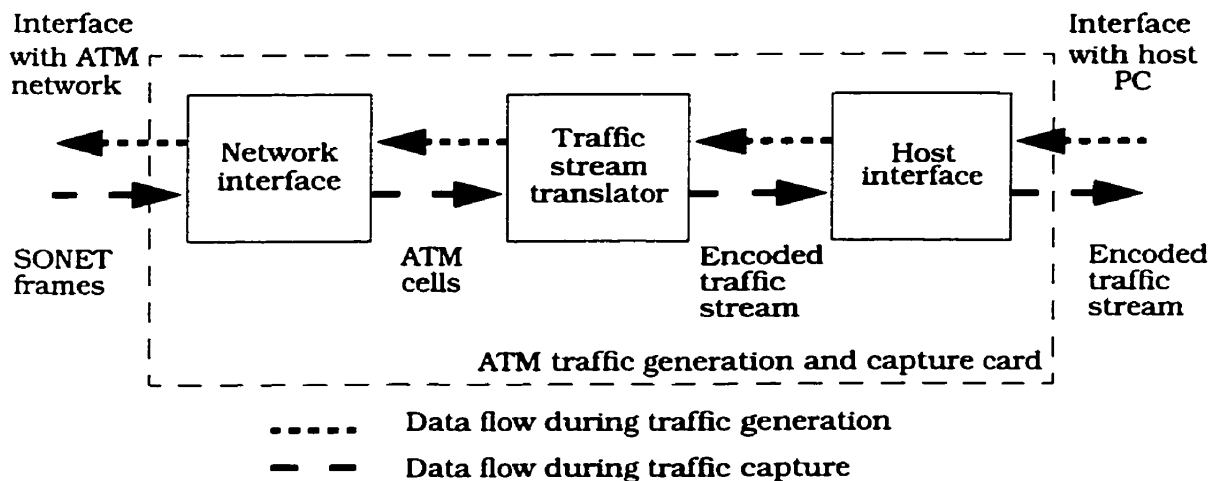
## **Traffic Stream Translator**

The traffic stream translator module is needed for performing the translation between a stream of ATM cells transported by a network and an encoded traffic stream processed by a host PC. During traffic generation, the traffic stream translator expands an encoded traffic stream into a stream of ATM cells which can be transmitted to the

network. During traffic capture, the translator encodes the ATM traffic stream received from the network.

## Data Flow

Figure 4.1 illustrates how the network interface, host interface, and traffic stream translator modules are interconnected to form the basic architecture of the ATM traffic generation and capture card. This figure also shows the orientation and type of data flow during each of the card's two traffic processing operations.



**Figure 4.1** Basic architecture of the ATM traffic generation and capture card

During traffic generation, a host supplies, via the host interface module, an encoded traffic stream to the traffic stream translator. The translator converts the encoded traffic stream into a stream of ATM cells which it transfers to the network interface. The network interface module inserts these cells into the SPEs of SONET frames sent to the network.

The direction of data flow during ATM traffic capture is opposite that during traffic generation. The network interface extracts ATM cells from the SONET SPEs received from a network and transfers these cells to the traffic stream translator. The translator



converts the stream of ATM cells into an encoded traffic stream which it then sends to a host PC via the host interface.

## **4.2 Component Selection and Complete Architecture**

Having defined the function of each of the modules in the card's basic architecture, the next step in the design of the ATM traffic generation and capture card was selection of components to perform these functions. This section will describe the components chosen and the complete architecture developed for the card's implementation.

### **ATM Network Interface**

A physical layer chip from PMC-Sierra called the S/UNI-LITE was selected to implement the card's ATM network interface. The S/UNI performs the ATM cell mapping and demapping and SONET/SDH processing functions at line rates of 51.84 Mbit/s (STS-1) and 155.52 Mbit/s (STS-3c/STM-1) and at ATM Forum mid-range PHY sub-rates of 12.96 Mbit/s and 25.92 Mbit/s<sup>3</sup>. The S/UNI conforms to the ATM Forum UNI Specification ([3]) and the ATM physical layer specification for B-ISDN as described in ITU-T Recommendation I.432 ([15]).

Like the ATM reference model's physical layer, the S/UNI is characterized as having two sides (or interfaces). The line side of the S/UNI interfaces to an ATM network through two independent bit-serial signal lines; one line is used to transmit data to the network and the other receives data from the network. The unit of transfer on the line side of the S/UNI is the SONET/SDH frame.

The side of the S/UNI opposite the line side is the drop side. The drop side of the S/UNI interfaces with ATM layer devices and uses the ATM cell as the unit of transfer.

---

3. Only the 155.52 Mbit/s STS-3c line rate is required for the current implementation of the ATM traffic generation and capture card.

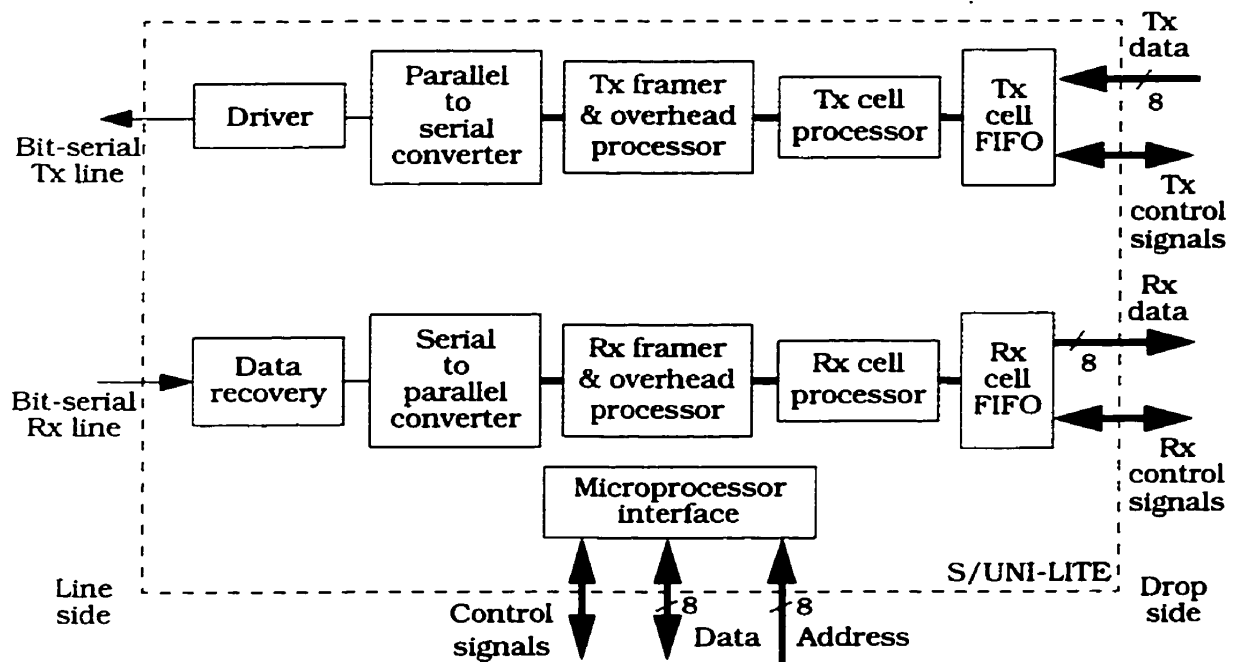
Similar to the line side, the drop side features two independent interfaces; the transmit interface is used to load the S/UNI with the stream of ATM cells that it will send to the network while the receive interface is used to read the ATM cells that the S/UNI has received from the network. Both the transmit and receive drop side interfaces are implemented as synchronous, byte-wide buses which conform to the ATM Forum's UTOPIA Level-1 standard for interfacing ATM layer and physical layer devices ([4]). A total of 53 clock periods are required for a complete cell to be written to or read from the S/UNI; these transfers can be done at a maximum clock frequency of 33 MHz.

The transmit and receive interfaces access separate FIFOs within the S/UNI. Each of the FIFOs can be configured to a maximum depth of four ATM cells. The S/UNI takes cells which have been loaded into the transmit FIFO and maps them into the SPE of SONET/SDH frames sent to the network. When the transmit FIFO reaches an empty condition or less than a complete cell is available in the FIFO, the S/UNI inserts idle/unassigned cells into the SPE. As ATM traffic is received from the network, the S/UNI loads the receive FIFO with the cells that are extracted from the SPE of incoming SONET/SDH frames. A simplified view of the internal architecture of the S/UNI is shown in Figure 4.2.

Also shown in Figure 4.2 is the S/UNI's microprocessor interface. The purpose of this interface is to allow devices external to the S/UNI to monitor and control its operation during ATM cell transmission and reception. The microprocessor interface consists of byte-wide address and data buses which access status and configuration registers within the S/UNI. This allows the S/UNI to be configured to perform such operations as cell filtering on received traffic and statistics such as the number of errored and uncorrectable cell headers received and the total number of cells transmitted to be monitored.

An optical transceiver on the line side of the S/UNI is necessary to complete the

implementation of the card's ATM network interface. This transceiver performs the conversion between the electrical signals processed by the S/UNI and the optical signals used to interface with the network. The particular optical transceiver selected enables the card to connect to a network through multi-mode fiber and optical signals with a 1300 nanometer wavelength.



**Figure 4.2** Internal architecture of the S/UNI-LITE (after [23])

## Host PC Interface

The ATM traffic generation and capture card implements two interfaces to a host PC. One of these interfaces - the PCI Local Bus - was intended to be used to initialize the card for the traffic generation and capture operations and for transferring the encoded ATM traffic stream between a host PC and the card. The card's second interface (an RS-232 interface) to a host will be discussed later.

**PCI Interface.** The PCI (peripheral component interconnect) Local Bus interface

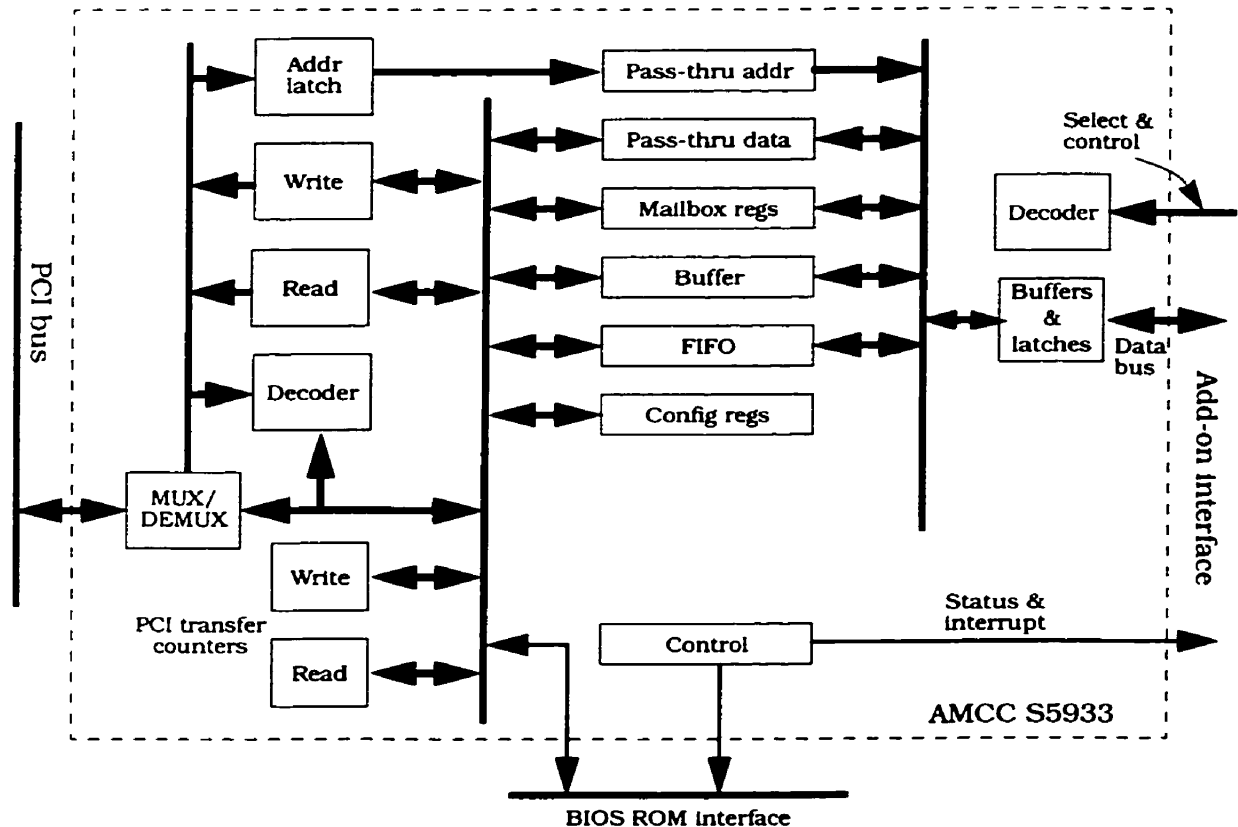
was selected as an interface to the host PC because of its capability for high data transfer rates. Although the current implementation of the card only requires a fraction (a few hundred kilo-bytes per second) of the 132 mega-bytes per second peak data transfer rate that the PCI bus is capable of providing, use of this interface was expected to ensure that the correct operation of the card is not sacrificed as a result of insufficient transfer rates between the host and the card. During traffic capture, for instance, this extra bandwidth prevents the card's captured traffic buffer from overflowing and traffic loss in the captured traffic log files from ultimately occurring.

The AMCC S5933 PCI controller chip was chosen to implement the card's interface to the PCI Local Bus. The S5933 is a general purpose PCI interface chip capable of functioning as a PCI bus master or as a target device. The PCI bus side of the S5933 implements the 32 bit, 33 MHz, 5 volt version of the bus as defined in Revision 2.1 of PCI Local Bus Specification ([20]). The add-on, or expansion, side of the S5933 features a 32 bit data path and is capable of asynchronous operation or synchronous operation at clock rates up to 33 MHz.

The S5933 provides three mechanisms by which devices on the PCI bus are able to exchange data or control and status information with devices on its add-on side. These data transfer mechanisms include FIFOs, mailbox registers, and pass-thru registers. The S5933 also contains status and configuration registers by which devices on either side of the chip can monitor and control the operation of each of the three data transfer mechanisms. The architecture of the S5933 is shown in Figure 4.3.

There are two FIFOs which can be used in transfers between the PCI and add-on sides of the S5933. Each of the FIFOs has a capacity of eight double-words; one FIFO is used for transfers from the PCI bus to the add-on side of the S5933 while the second FIFO is for transfers from the add-on side to the PCI bus. The S5933's FIFOs can be

accessed when the device is operating as a PCI bus master or as a target. Burst transfers with the FIFOs are possible only when the S5933 is the bus master. Only single data phase transactions are possible when the S5933 is functioning as a target device.



**Figure 4.3** Internal architecture of the AMCC S5933  
(after [2])

The S5933 provides eight mailbox registers, each capable of storing a single double-word. Four mailboxes are used for data transfers from the PCI side of the S5933 to its add-on side and four are used for transfers from the add-on side to the PCI side. These mailbox registers can be accessed from the PCI side only through single data phase transactions with the S5933 functioning as a target device; burst reads or writes to the mailbox registers are not possible. The S5933 can be configured to generate inter-

rupts on its PCI and/or add-on sides upon a read or write event to one of the mailbox registers.

Two pass-thru registers, one for each direction of data transfer, allow direct (registered) access between the PCI bus and devices on the add-on side of the S5933 using a handshaking scheme. Accessible only when the S533 is functioning as a target device, the pass-thru registers permit single and multiple (burst) data phase transfers.

### **Traffic Stream Translator**

The last module in the card's basic architecture remaining to be implemented was the traffic stream translator. As a result of the custom nature of the encoding scheme used to represent the ATM traffic stream, there was no commercially-available component that performed the required translation. For this reason, a field-programmable gate array was chosen to implement this module.

A Xilinx XC3195A-1 was selected as the FPGA that implemented the card's traffic stream translator. This part had the largest logic capacity in the XC3000 family and was used in the fastest speed grade available. It was chosen because of experience with this family in the past as well as the part's relatively low cost. Its ability to be reprogrammed infinitely many times (without deterioration in performance) was another important feature. Since simultaneous implementations of both the traffic generation and traffic capture functionalities in the FPGA would have sacrificed its overall performance, a reprogrammable FPGA allowed the translator to be configured for only the traffic processing operation required at a particular time. Reprogrammability also allows changes to be made to the card's functionality, should it be required in the future. The XC3195A-1 features 176 user-programmable input/output (I/O) pins and 484 configurable logic blocks (CLBs) with an equivalent logic capacity of between 6,500 and 7,500 gates. At the time of component selection, it was anticipated that this FPGA would have the speed and

gate capacity necessary to implement the processing required of it for generation and capture of ATM traffic.

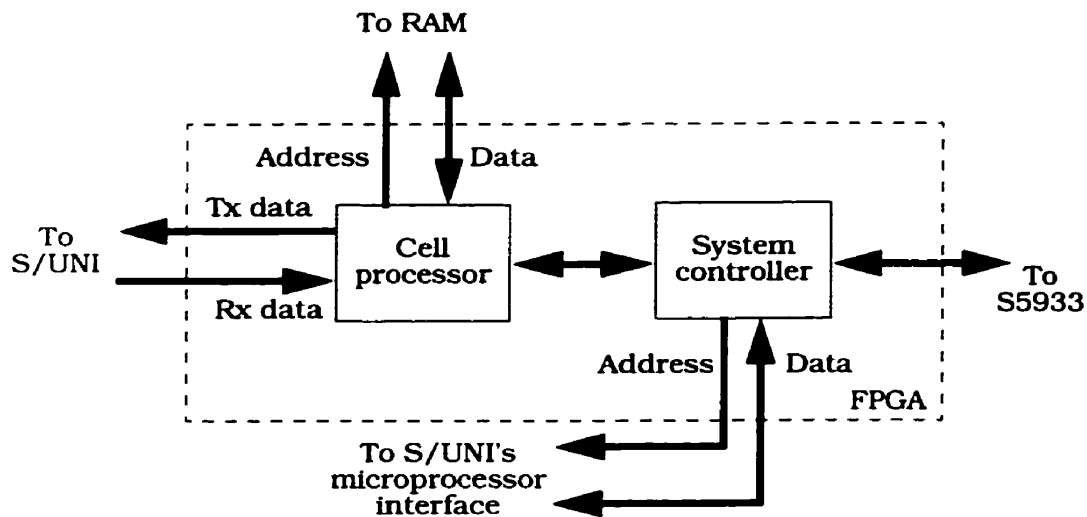
In order to support the operation of the traffic stream translator, RAM was included in the card's architecture. During the traffic generation and traffic capture operations, the translator uses this memory space to facilitate decoding and encoding of the ATM traffic streams. The current version of the card implements a high-speed, 25 nano-second SRAM memory space 2.5 mega-bytes in size.

**Architecture.** The internal architecture of the traffic stream translator contains two modules: a system controller and a cell processor. Although the specific functions performed and the direction of ATM traffic flow differs, this architecture is identical for both traffic generation and traffic capture. Figure 4.4 illustrates the interconnection of these two modules and the other components on the card.

The system controller is responsible for controlling the operation of the traffic stream translator as well as the overall operation of the card. With a direct connection to the S5933, the controller responds to commands received from a host and asserts the appropriate control signals to the other devices in the system. The system controller also handles the transfer of the encoded traffic stream between the host and the cell processor. Configuration and control of the S/UNI's operation is managed by the system controller through its connection with the S/UNI's microprocessor interface.

The cell processor module is responsible for performing the translation between the encoded and raw ATM traffic streams. This module interfaces with the RAM, the transmit and receive interfaces on the S/UNI's drop side, and, internally, with the system controller module. During traffic generation, the cell processor expands an encoded traffic stream received by the system controller into a stream of ATM cells. These cells are then transferred to the S/UNI for transmission to the network. When performing

traffic capture, the cell processor reads a stream of ATM cells from the S/UNI, encodes them, and sends the encoded traffic stream to the system controller for transfer to the host.



**Figure 4.4** Internal architecture of the traffic stream translator

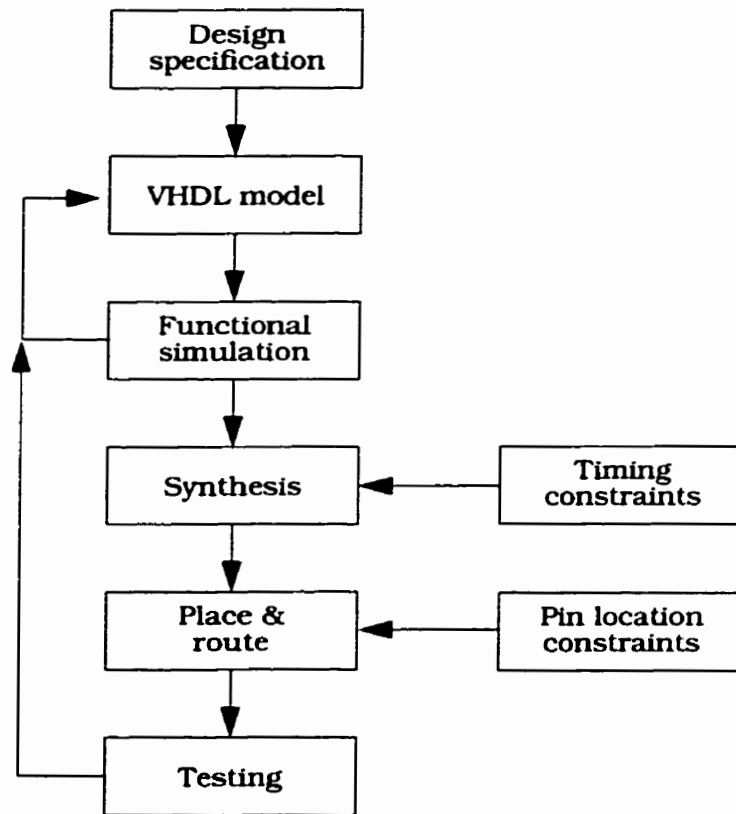
**Implementation.** A high-level design methodology was used in the development of the traffic stream translator. The translator's functionality for ATM traffic generation and traffic capture was specified through individual VHDL descriptions written at the register transfer level (RTL).

Correct functionality of the behavioral descriptions for traffic generation and traffic capture was verified using Synopsys' VHDL simulator. Synopsys' FPGA Compiler was then used to synthesize the VHDL descriptions for implementation in the Xilinx XC3195A FPGA. Timing constraints supplied during the synthesis step helped to ensure that the implemented design would meet the necessary timing requirements. The synthesis step created a netlist of Xilinx primitives (CLBS and I/O blocks).

The netlist was then imported into the Xilinx Alliance software. The automatic



place and route tools were used to complete the implementation and to create the configuration bitstream which is loaded into the FPGA to program its functionality. During this step, a pin constraint file was used to force the place and route software to make pin assignments that conformed to the FPGA's physical connections to other devices on the card. Figure 4.5 summarizes the design flow that was used in the development of the traffic stream translator.



**Figure 4.5** Traffic stream translator development

Of the 176 user I/O pins available on the XC3195A FPGA, 170 of them were used for connections to other components on the card. All 170 of these pins were not necessarily used during both of the card's two traffic processing operations. During ATM traffic generation, some of the signal lines associated with the S/UNI's drop side receive

interface, for example, were idle while some of pins connecting to the S/UNI's drop side transmit interface were unused during traffic capture.

Individual implementations of the ATM traffic generation and traffic capture functionalities in the traffic stream translator made use of 279 and 188 CLBs (respectively) of the 484 CLBs available in the XC3195A. This corresponded to FPGA logic utilizations of 57% and 38%, or approximately 4000 and 2700 gates, for each of the traffic processing operations.

The Alliance software was also able to provide an estimate of the maximum frequency at which the completed designs would function. The implemented traffic generation functionality was rated for a maximum system clock frequency of 24.4 MHz while the traffic capture design was rated a little higher at 29.2 MHz. In both cases, the maximum clock rate estimations were above the system clock rate that had been chosen earlier in the card's development and which will be discussed shortly.

### **FPGA Configuration**

Since the FPGA selected to implement the card's traffic stream translator is configured through a bitstream stored in its internal SRAM, it was necessary to include a mechanism for programming the FPGA's functionality upon power-up. With the need to switch the card's functionality between traffic generation and traffic capture, the programming mechanism had to allow FPGA configuration to be made as simply as possible and with little (if any) assistance from a user.

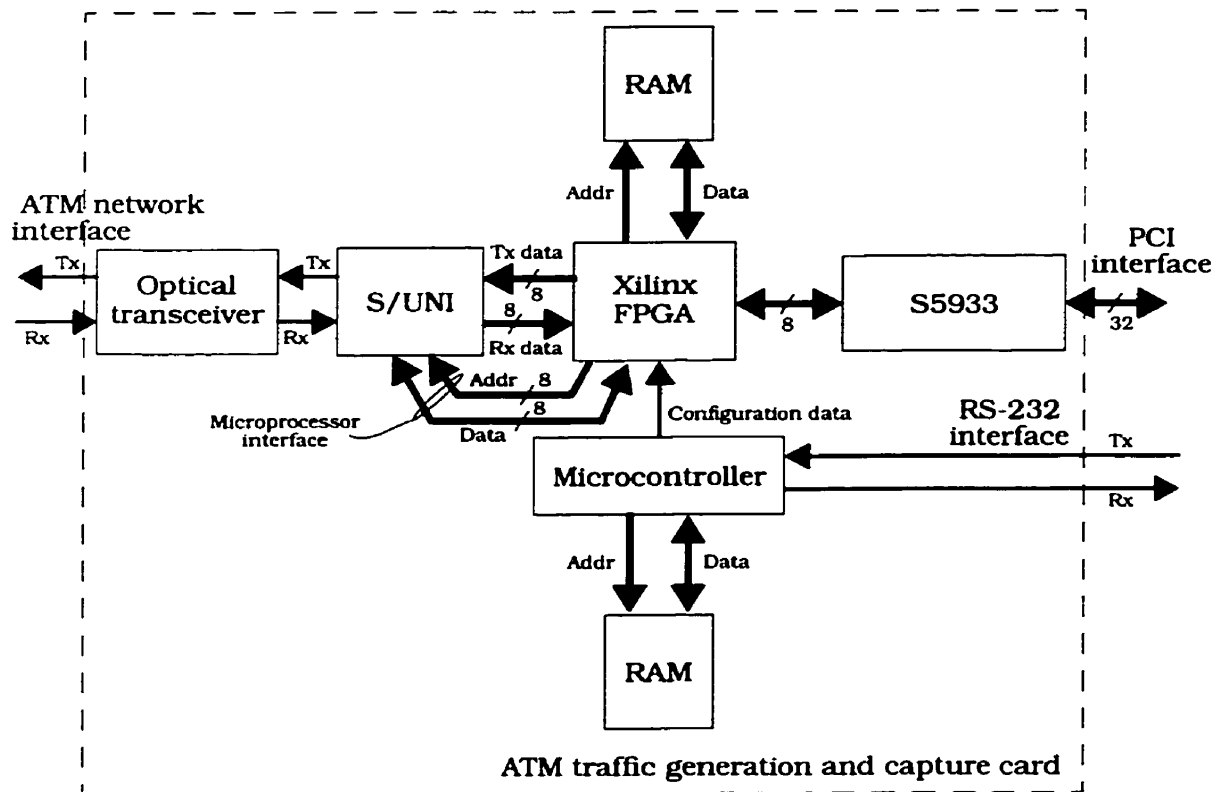
A Dallas Semiconductor DS5001 microcontroller was dedicated to the task of FPGA configuration. This Intel 8051-compatible microcontroller features a built-in RS-232 interface, the card's third interface to a host PC. The microcontroller receives the configuration bitstream downloaded by the host and loads the bitstream into the FPGA to program its functionality. The microcontroller's external RAM stores its program code

and is used to buffer the FPGA's configuration bitstream as it is received from the host. With this microcontroller-based approach for programming the FPGA, no physical modifications need to be made to the card to change its functionality as would have been required had a PROM been used for configuration.

### **Complete Architecture**

Figure 4.6 illustrates the complete architecture of the ATM traffic generation and capture card. For purposes of clarity, this figure only shows the primary data paths in the card's architecture and ignores the associated control signals. Also not shown are the test headers which are connected to each of the I/O pins of the FPGA to facilitate testing and debugging of the card during its development.

One compromise that had to be made in the completed architecture was in the width of the data bus between the FPGA and the S5933 PCI Controller. As a result of a shortage of I/O pins on the FPGA, this bus could not be implemented in the full 32-bit width. It was instead implemented as an 8-bit bus, requiring four individual read or write operations in order for a double-word to be transferred between the FPGA and the S5933.



**Figure 4.6** Complete architecture of the ATM traffic generation and capture card

### 4.3 System Clock

Except for the Dallas microcontroller, which uses its own clock, a single system clock is used to drive all of the components on the board. This system clock controls the rate at which the traffic stream translator loads cells into and reads cells from the S/UNI, transfers data to and from the S5933, accesses RAM, and processes the ATM traffic stream.

An important consideration in the design of the ATM traffic generation and capture card was the selection of the frequency of the system clock. The choice of the clock rate was crucial to ensuring that the card is able to accurately generate and capture ATM traffic. Loading ATM cells into the S/UNI's transmit FIFO during traffic generation

at a clock rate which is too low results in the FIFO frequently reaching an empty state. This causes the S/UNI to insert idle/unassigned cells into the traffic stream sent to the network and ultimately distorts the properties of the traffic source which is to be simulated. Failure to read the S/UNI's receive FIFO quickly enough causes the FIFO to overrun and cells to be dropped, destroying the accuracy of the captured traffic log. In contrast, choice of a clock rate that is too high violates the timing specifications of components on the card.

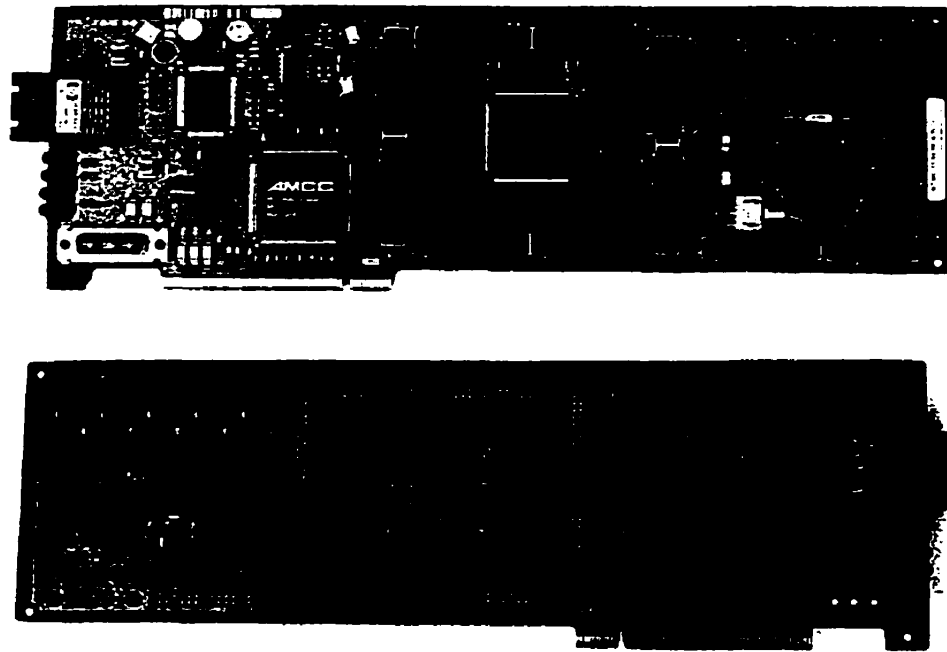
The minimum system clock frequency was determined by calculating the rate at which an OC-3c transport carries ATM cells. With the SPE of each SONET OC-3c frame packed with ATM cells, no more than 353,208 cells per second could be transmitted from or received by the S/UNI's line side. Since the S/UNI's drop-side transfers cells on a byte-wise basis, this cell rate corresponds to a clock rate of 18.72 MHz, where each and every rising edge of the clock is used to transfer one byte to or one byte from the S/UNI's transmit or receive FIFOs respectively.

To provide a margin of safety and to allow time for the traffic stream translator to perform additional processing between cells, the system clock frequency that was chosen for the card's implementation was increased slightly from the minimum rate to 20 MHz. This clock frequency ensures that the S/UNI's transmit FIFO never becomes completely empty and that the receive FIFO never becomes completely full. In fact, through continuous transfers between the S/UNI and the translator, this clock rate establishes an equilibrium condition where the transmit FIFO never contains any less than three cells and the receive FIFO never contains any more than one cell. The 20 MHz clock frequency does not violate the timing specifications of the S/UNI, S5933, or RAM; it was anticipated that the FPGA would also be able to support this clock rate.

## **4.4 PCB Design**

Upon finalizing details of the ATM traffic generation and capture card's architecture, the next step in the card's development was the design and fabrication of printed circuit boards (PCBs). The architecture shown earlier was developed to create electrical schematics which were designed using Viewlogic's schematic-entry software. The schematic's electrical netlist was then transferred to UniCAD for manual placement of components and routing of the associated signal traces on the board. The schematic entry and board layout steps made use of circuit and component placement recommendations from PMC-Sierra, AMCC, and the PCI Special Interest Group ([22], [2], [20]).

Completion of the PCB layout allowed the prototype ATM traffic generation and capture card shown in Figure 4.7 to be fabricated. The card has six layers - a power plane, a ground plane, and four signal planes. The majority of the signal routing is done on the two internal planes, with the external planes used primarily for making the connections to components on the card. As the figure shows, only resistors, capacitors and inductors were installed on the back side of the card while integrated circuits, LEDs, test headers and other components were installed on the front side.



**Figure 4.7** Photograph of prototype ATM traffic generation and capture card

---

# **Chapter 5**

## **Support Software for the ATM Traffic Generation and Capture Card**

Based on the requirements in the design specification, a host PC was assigned the responsibility of configuring and controlling the operation of the ATM traffic generation and capture card. Specific tasks performed by the host include programming the functionality of the FPGA, configuring the card for the desired traffic processing operation, and transferring the encoded ATM traffic stream between itself and the card. This chapter will describe the software developed to perform these operations on an IBM-compatible PC running Microsoft Windows 95 and the firmware developed for the card's (FPGA-configuration) microcontroller.

### **5.1 FPGA Configuration**

Before the ATM traffic generation and capture card can be used for either traffic processing operation, the card's FPGA must be programmed with the required function-



ality. To complete this first step in the card's initialization, the host is required to select the appropriate FPGA configuration bitstream and download it to the card's microcontroller via the RS-232 interface. The microcontroller must receive the configuration bitstream and load it into the FPGA.

A simple communication protocol was developed to facilitate exchanges between the host and the microcontroller. A number of commands were defined for the host to instruct the microcontroller to perform a particular operation or to provide some status information. The microcontroller responds to confirm the execution of a command or to return data. Exchanges between the two devices begin with the byte-long command or response. Depending on the particular command or response being issued, the number of data parameter bytes transferred with the command or response byte is either zero or 16. The commands and responses defined for host-microcontroller communication via the RS-232 interface are listed in Tables 5.1 and 5.2 respectively.

**Table 5.1** Commands defined for host-microcontroller communication

Command	Data parameters
Reset FPGA	none
Start configuration data download	none
Store configuration data	16 - FPGA configuration data
End configuration data download	none
Program FPGA	none
Checking FPGA's configuration status	none
Reset system	none

The first command issued by the host is one which resets the internal configuration SRAM of the FPGA in preparation for programming the functionality of the device. The microcontroller responds with a confirmation that it has asserted the FPGA's reset

signal and returns an idle prompt. The idle prompt is used by the microcontroller to indicate that it is ready to perform the next command. No data bytes are transferred with this command or response; no data is transferred with the idle prompt either.

The next command issued by the host is one instructing the microcontroller to begin receiving FPGA configuration data. The microcontroller responds to this instruction to confirm its ability to receive the data.

**Table 5.2** Responses defined for host-microcontroller communication

Response	Data parameters
Idle prompt	none
FPGA reset confirmation	none
Configuration data download started	none
Configuration data received	16 - FPGA configuration data
Configuration data download stopped	none
FPGA programming successful	none
FPGA programming failed	none
System reset	none

With the microcontroller ready to receive configuration data, the host transfers the configuration bitstream in blocks of 16 bytes, each preceded by a one-byte header. The microcontroller stores each block of configuration data it receives from the host temporarily in its external RAM. It acknowledges the transfer from the host and echoes back the 16 configuration bytes to provide the host with a means of detecting transmission errors.

When the entire configuration bitstream has been transferred, the host sends a command to the microcontroller to inform it that the download is complete. The microcontroller acknowledges this command and returns the idle prompt.

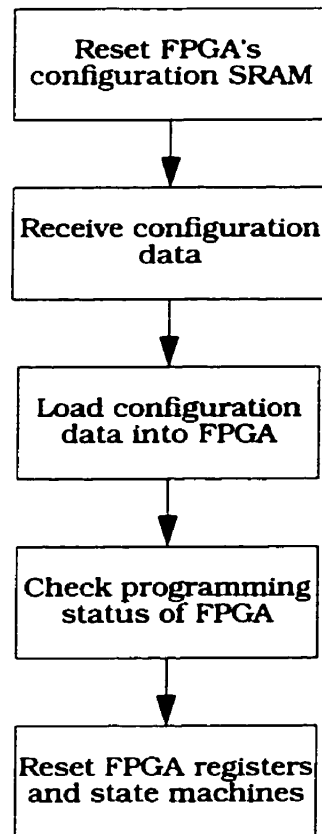
The host then issues a command to the microcontroller to program the FPGA. The microcontroller retrieves the configuration bitstream which it stored in RAM and loads it into the FPGA. Upon completion of programming, the microcontroller responds to the host with the FPGA's configuration status, which either reports successful programming or a failure.

A command is also defined to allow the FPGA-based traffic stream translator to be reset (system reset). This reset serves to clear some of the registers and resets the finite state machines within the FPGA; it does not reset the FPGA's configuration SRAM to reprogram its functionality.

Figure 5.1 lists the tasks performed by the microcontroller to program the functionality of the FPGA.

The host software for communicating with the card's microcontroller and controlling the configuration of the FPGA was written in the C programming language. An application programming interface (API) from Microsoft was used to enable host-microcontroller communication using the RS-232 interface. The code was compiled and debugged using the Microsoft Visual C++ development environment.

The microcontroller's program was written in the Intel 8051 assembly language and compiled using a share-ware compiler.



**Figure 5.1** Operations performed by microcontroller used for configuring the FPGA for ATM traffic generation and capture

## **5.2 Host-Card Communication**

In order for control and status information and the encoded traffic stream to be transferred between the host and the traffic stream translator module on the ATM traffic generation and capture card, a protocol had to be defined. Before discussing the software which was written to enable communication between these two entities, this section of the chapter will describe the protocol that was developed and how communication between the card and a host was to proceed.

## **Transfer Mechanism**

The S5933's FIFOs were chosen as the mechanism by which the host and the traffic stream translator communicate. Any control or status information or traffic stream data that needs to be exchanged between these two devices is transferred using these FIFOs. The device writing data to a FIFO is responsible for ensuring that it never over-fills the FIFO, while the device receiving the data has to watch for the availability of data in the FIFO and to respond appropriately.

There are several reasons why the S5933's FIFOs were selected to transfer data between the card and the host. First, data transfers using the FIFOs are easier and more efficient for the traffic stream translator to manage than those made with the mailbox registers. Since the S5933 does not provide dedicated status pins on its add-on side for the mailbox registers that are equivalent to the "add-on to PCI FIFO full" and "PCI to add-on FIFO empty" pins, the translator would have to read a status register within the S5933 to determine the presence or absence of data in a mailbox. Before writing data to a mailbox, the translator would have to read this status register to check that the data in a mailbox it had written earlier had been read by the host and would not be overwritten. When reading data, the translator would have to check the status register to determine if the host had written new data to a mailbox. With dedicated pins reporting the status of each of the FIFOs, the translator is able to immediately determine whether a data transfer can proceed (either a write to the FIFO or a read from a FIFO) without wasting any clock cycles reading status registers.

The S5933's FIFOs were selected over the pass-thru registers once again for reasons of simplicity of use. With the pass-thru registers, there is no way for the traffic stream translator to initiate a transfer with the host; when the translator has data that it must transfer, it must wait until the host initiates a read transaction. If the host ini-

tiates a write transaction instead, the translator must buffer the data it wishes to transfer to the host, receive the data supplied by the host, and wait until a read transaction is begun. By using the FIFOs, the translator can load data into the S5933's add-on to PCI FIFO without having to wait for the host to be ready to complete the transfer. Simultaneous write operations to the FIFOs can be made (provided they're both not full, of course) by the translator and by devices on the PCI bus. Similarly, data can be read from each side without the need for synchronization of the two transfers. Alternately, a device on one side of the S5933 can be writing to the FIFO while a device on its other can be reading from the FIFO.

### **Host-Card Relationship**

A master-slave relationship was maintained between the host and the ATM traffic generation and capture card during card configuration as well as during traffic generation and capture. With this arrangement, almost all exchanges between these two devices are initiated by the host; no data is supplied and no operation is performed by the card without first being requested by the host.

The exception to this master-slave relationship occurs when the card is transferring captured ATM traffic to the host. Once the host enables the card for traffic capture, the traffic stream translator loads the S5933's add-on to PCI FIFO with captured traffic as it becomes available and without a specific request from the host. If the card is forced to wait for a command from the host to transfer the traffic, there is the possibility that the request will not come soon enough and captured traffic will be overwritten before being transferred. This unprompted approach to transferring the captured traffic stream reduces the host's responsibilities during traffic capture to simply checking the status of the FIFO and reading the traffic stream as it becomes available.

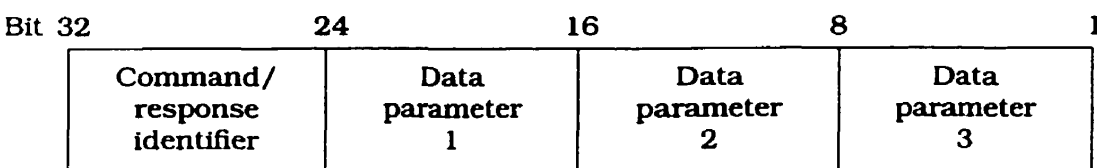
The choice to implement this master-slave arrangement was made in order to

simplify the development of the host's software. With the host initiating the transfers, there is no need for the card to issue interrupts to the host when it needs attention, thereby eliminating the need to write complicated interrupt service routines for the host.

### **Signalling**

Once again, transfers between the host and the ATM traffic generation and capture card are classified as either a command or a response. Commands are requests issued by the host for the card to perform a particular operation or to supply some data. Responses are the card's acknowledgment to a command and are used to return requested data or to confirm that an operation has been performed. This classification corresponds to the master-slave relationship established between the host and the card.

Commands and responses use the same format and are contained within a double-word. The most-significant byte in the double-word identifies the particular command or response issued. The remaining three bytes of the double-word carry any data parameters that need to be transferred. The number of data bytes used range from none to three, depending on the particular command or response. Figure 5.2 defines the command and response format.



**Figure 5.2** Command/response structure

Almost all of the commands and responses defined for the current implementation of the ATM traffic generation and capture card are common to both traffic processing operations. Table 5.3 lists the commands that were defined and Table 5.4 lists the

corresponding responses. The following discussion will describe the use of each of these commands and responses.

**Table 5.3** Commands defined for host-card communication

Command	Data parameters
Reset S/UNI	none
Read S/UNI configuration register	1 - address of register to read
Write S/UNI configuration register	2 - address of register and data value to write
Write RAM	3 - data to write to RAM
Start traffic generation or capture	none
Stop traffic generation or capture	none
Traffic stream to generate	3 - encoded traffic stream

The first command defined in Table 5.3 is for resetting the S/UNI. This command allows the operation of the S/UNI, including its internal status and configuration registers, to be reset through software control. There are no data parameters associated with this command.

**Table 5.4** Responses defined for host-card communication

Response	Data parameters
Generic response	none
Data read from S/UNI	1 - data read from register
Captured traffic	3 - encoded traffic stream

A response called the *generic response* was defined for the translator to use to acknowledge commands (such as the command for resetting the S/UNI) which do not require data to be returned to the host. This response was common to a number of com-



mands and was defined in an effort to simplify the implementation of the system controller module within the traffic stream translator by reducing the logic required from the FPGA.

The next two commands access the S/UNI's configuration and status registers. The first command reads data from one of the S/UNI's registers; a single data byte passed with the command specifies the address of the register to read. The traffic stream translator acknowledges this command with a response that returns the byte of data read from the register. The second command instructs the translator to write data to one of the S/UNI's registers. This command passes the address of the register to write as well as the data to write to the register. The card acknowledges this command through the generic response.

The generic response is also used by the card for acknowledging writing of data to RAM as well as for starting and stopping generation and capture of ATM traffic. The command for writing to RAM packs the data portion of the double-word with three bytes of data which are to be written to the RAM on the card. The start and stop commands instruct the card to start and stop generating or capturing ATM traffic (depending on the configuration of the FPGA); there are no data parameters associated with either of these two commands. No data is expected to be returned in the responses to the commands for writing to RAM or starting and stopping traffic generation or capture.

The last command defined in Table 5.3 is used during ATM traffic generation only. It is used by the host PC to transfer the encoded ATM traffic stream to the card when traffic is being generated. All three bytes of the data portion of the double-word are used for transferring the encoded traffic stream. This command is not acknowledged by the card with a response.

The last response defined in Table 5.4 is issued by the card only during ATM traf-

fic capture. It is not a response in the typical sense in that it is not asserted by the card to acknowledge a command from a host. Rather, the card sends this response to the host in order to transfer the encoded ATM traffic stream that it has captured from the network. All three data bytes in the double-word of this response are used for transferring the encoded traffic stream. The host does not acknowledge this response from the card.

### **5.3 Low-Level PCI Communication Functions**

With the functionality of the FPGA programmed, all further communication between the host and the card can be done using the PCI bus interface. A number of low-level functions were developed to enable communication between these two devices. The modules described in this section are called by higher-level functions that will be discussed later in this chapter.

The first of the PCI support functions acquires the address that is assigned to the card when the host is powered-up. This address is the base to which offsets are added to access the various (configuration, mailbox, FIFO, pass-thru) registers of the S5933 and which is required by the other communication functions that were developed.

The PCI communication functions that are used in the current implementation of the host software for the ATM traffic generation and capture card are those that access registers in the S5933 associated with the add-on to PCI and PCI to add-on FIFOs. In addition to functions which read data from and write data to the FIFOs, functions were written to access a configuration/status register within the S5933 which controls the operation of the FIFOs and provides information on their status. Writing values to specific bits in this register allow each of the FIFOs to be individually cleared. Reading from this register allows the host to determine if the FIFOs are completely empty or com-

pletely full. It also allows the host to determine if at least one space of double-word or at least four spaces or double-words are available in the FIFOs.

Like the host software developed to communicate with the card's microcontroller and the software that will be discussed in the sections that follow, the low-level PCI bus interface functions were written in the C programming language and were compiled using the Microsoft Visual C++ development environment.

## **5.4 Card Initialization**

Once the FPGA is programmed, initialization of the card can be completed through direct communication between the host PC and traffic stream translator using the PCI bus interface and the commands and responses described earlier. The configuration tasks that remain are specific to the ATM traffic processing operation being performed and will be discussed individually in this section of the chapter.

Only a few more steps are required to complete the preparations for ATM traffic generation. The host must be told which traffic log file contains the encoded traffic stream that should be generated by the card and the length of time for which the card is to generate. The host should check that the amount of encoded traffic stored within the selected traffic log file corresponds to the period for which the card is to generate and then open this file in preparation for reading. The host then loads the card with the data necessary for doing the translation between an encoded traffic stream and a stream of ATM cells which can be sent to a network. To ensure that traffic is available when the *start-generation* command is issued, the host pre-loads the card and a buffer of its own (used to cache traffic to be transferred between the hard disk and the card) with short periods of the encoded traffic stream.

There are only a few configuration steps necessary before ATM traffic capture can

begin. In order for the card to maintain the accurate timing information of all cells it has received from a network, the S/UNI must be configured not to discard the idle/unassigned cells it demaps from incoming SONET SPEs. This is accomplished by disabling this function in one of the S/UNI's configuration registers. Next, the host must acquire the name of the file that the encoded captured traffic information should be written to and the duration of time for which traffic should be captured. The software should check that there is sufficient space on the hard disk for this length of capture and then open the file for writing. Finally, the host must allocate some buffer space in its memory to store captured traffic which it has read from the card, but which has not yet been written to disk.

## **5.5 Traffic Stream Transfer**

With initialization of the card complete, traffic generation or traffic capture can begin. The host issues a common command which starts the ATM traffic processing operation for which the card is configured. The responsibilities of the host while traffic generation or capture is in process are similar, but opposite, for each of these functions.

When the card is generating ATM traffic, the host is responsible for ensuring that the card is supplied with encoded traffic stream at all times. To do this, the host must watch the status of the S5933's PCI to add-on FIFO. An empty FIFO condition signals the card's need for traffic and an opportunity for the host to transfer the encoded traffic from its hard disk to the FIFO. The strategy implemented in the host software was to continuously poll the FIFO's status register while maintaining a small cache of traffic on the host; highest priority was given to status checking and writes to the FIFO. During intervals when the S5933's FIFO was full, the host would spend that time refilling its own cache with traffic read from the hard disk.

When the card is configured for ATM traffic capture, the host must read the encoded traffic stream from the card to prevent its buffers from overflowing and traffic from being lost. The host does this by checking for the availability of data in the S5933's add-on to PCI FIFO and reading it from the FIFO as captured traffic appears in it. The host software was written such that priority was assigned to transferring the encoded traffic stream from the card to buffers within the host's memory space. During periods when the software found that the FIFO was empty, it would transfer the captured traffic from its buffer to the hard disk.

Traffic generation or capture continues until the end of the specific traffic processing interval is reached and the host issues the common stop command.

---

# **Chapter 6**

## **Testing and Verification**

With prototype boards fabricated and the host software written, the next phase in the development of the ATM traffic generation and capture card was testing. Testing was comprised of two stages: component testing followed by system testing. This chapter of the thesis will define the purpose of each of these stages and will describe the testing that was performed.

### **6.1 Component Testing**

The first stage of testing performed on the ATM traffic generation and capture card was component testing. The objective of this stage of testing was to individually test and enable each of the components and capabilities in the system and to verify that they functioned correctly prior to testing their integrated operation during ATM traffic generation and capture. Component testing checked hardware modules on the card as well as software routines running on the host PC.

Hardware testing verified that there were no defective components or signal traces on the card. Testing made use of a logic analyzer connected to the header pins

attached to each of the I/O pins of the FPGA. Since the functionality of the FPGA-based traffic stream translator was developed specifically for the ATM traffic generation and capture card, component testing also checked that this module interfaced correctly with other components on the board. Component testing ensured that the translator was asserting the appropriate signals and was not violating any set-up and hold times in its interfaces with the S/UNI, S5933, and RAM.

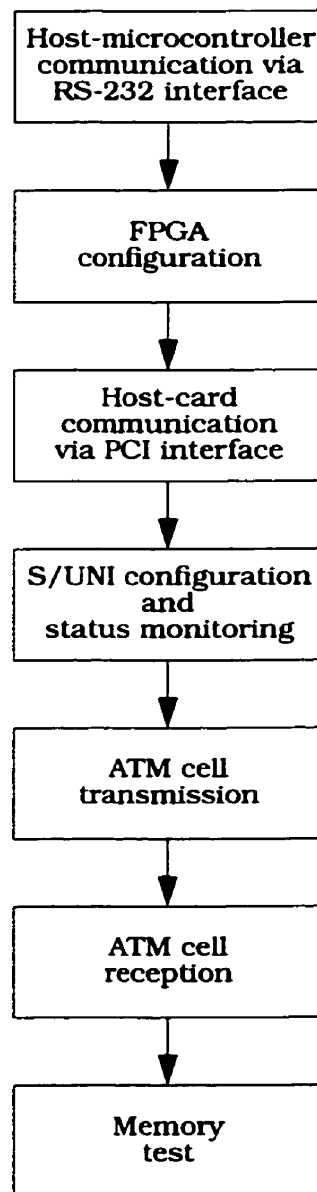
Software testing verified that the host was able to communicate with the ATM traffic generation and capture card in order to control and monitor its operation. The software functions described in Chapter 5 were tested in programs which prompted for user input and displayed the status of the software's execution on the console.

An important consideration during component testing was the order that hardware devices and software functions were tested. The sequence that was chosen allowed components which were successfully enabled and verified to function correctly to be used in the testing of other components. Figure 6.1 illustrates the order that major components and capabilities of the ATM traffic generation and capture system were tested. Discussion in this section will follow the same sequence.

### **FPGA Configuration**

The first part of the system tested during component testing was the procedure used for programming the functionality of the FPGA. Testing began by verifying that the microcontroller on the card and the host were able to communicate via the RS-232 interface. This was done with simple test software running on the microcontroller and the host. The program on the host supplied data to and read data from the microcontroller. The microcontroller was programmed to read data from the interface, modify it (e.g. invert the data), and write it back to the host. Both data values were displayed on the console of the host PC to allow a user to confirm that communication between the two

devices was successful.



**Figure 6.1** Component testing flowchart

With the interface between the microcontroller and the host functional, the microcontroller was then enabled to receive the configuration bitstream from the host and load it into the FPGA. The host was programmed to download to the microcontroller



a bitstream which configured the FPGA to light the status LEDs on the card. Although this was a trivial configuration of the FPGA, it served to verify that the FPGA could successfully be programmed through cooperation between the host and the microcontroller.

Upon successfully testing and enabling FPGA configuration, the FPGA was able to be used to test the other components on the board - namely the S/UNI, S5933, and RAM. To facilitate testing of these other devices, test-specific FPGA configurations were developed to exercise the inputs and process the output signals of the particular device being tested.

### **PCI Interface**

Communication between the host and the ATM traffic generation and capture card via the PCI interface was tested next. To test this interface, the FPGA was configured to respond to the availability of data in the S5933's PCI to add-on FIFO. When the FPGA detected that the host had written data to the FIFO, the FPGA read the data, modified it, and returned it to the host by writing it to the add-on to PCI FIFO. Test software developed for the host wrote data to the FIFO and read the data returned by the card, displaying both data items on the console to allow a user to visually verify correct operation of the card's PCI interface. This step tested the PCI interface software on the host and the operation of the S5933 as well as the FPGA's interface to the S5933. By testing and enabling the PCI interface at this time, host-card intercommunication could proceed for further testing steps as necessary.

### **S/UNI**

Testing of the S/UNI began with the microprocessor interface to its internal configuration and status registers. To test this interface, the host was programmed to issue (to the card via the S5933) one of two instructions. One instruction was for writing data

to a configuration register within the S/UNI. This instruction supplied the address of the register and the data to write to it. The second instruction was for reading data from a register. This instruction supplied the address of the register to read. The FPGA was configured to respond to each of these instructions once the host had written them to the S5933's PCI to add-on FIFO and to perform a register read or write operation, as appropriate. In the case of a read operation, the FPGA returned to the host the data value read from the register by writing it to the S5933's add-on to PCI FIFO. A test program running on the host allowed a user to monitor the register read and write operations, verifying that the correct data values were being written and read and that the FPGA's interface to the S/UNI's configuration/status registers was functioning correctly.

To test the ATM cell transmission capability of the S/UNI, the FPGA was configured to continuously load the S/UNI's transmit FIFO with a stream of ATM cells. To simplify specification of the FPGA's functionality for this step, the stream of cells that the FPGA loaded into the FIFO was set to a fixed pattern which alternated between two ATM cells. Each cell had a different header, but shared common payload values. A GNNetest interWATCH 95000 ATM protocol analyzer was used to capture the ATM traffic transmitted by the S/UNI. A visual examination of the ATM traffic captured by the analyzer verified that the S/UNI was able to correctly map these cells to the outgoing SONET SPEs.

Testing of the S/UNI's ATM cell reception capability made use of the card's PCI interface tested and enabled earlier. For this stage of the testing, the FPGA was configured to read cells from the S/UNI's receive FIFO as they were demapped from incoming SONET SPEs. The first header byte of each of the ATM cells read from the S/UNI was extracted by the FPGA and written to the S5933's add-on to PCI FIFO. The ATM protocol analyzer used earlier was configured to supply a fixed, pre-determined sequence of cells to the S/UNI. A test program running on the host allowed a user to verify that the

header bytes extracted by the FPGA matched those supplied by the protocol analyzer.

## **RAM**

The last component to be tested was the RAM used by the traffic stream translator during traffic generation and capture. To test this module, the FPGA was configured to write a checkerboard pattern of 0's and 1's to the entire memory space. The memory was then read by the FPGA to check that the pattern that was read matched that which was written to it earlier. This write/read sequence was repeated a second time with the complement of the checkerboard pattern used previously. The FPGA was configured to light status LEDs on the board upon successful completion of memory testing or upon the detection of an error where the data read from a location in RAM did not match that which was written to it.

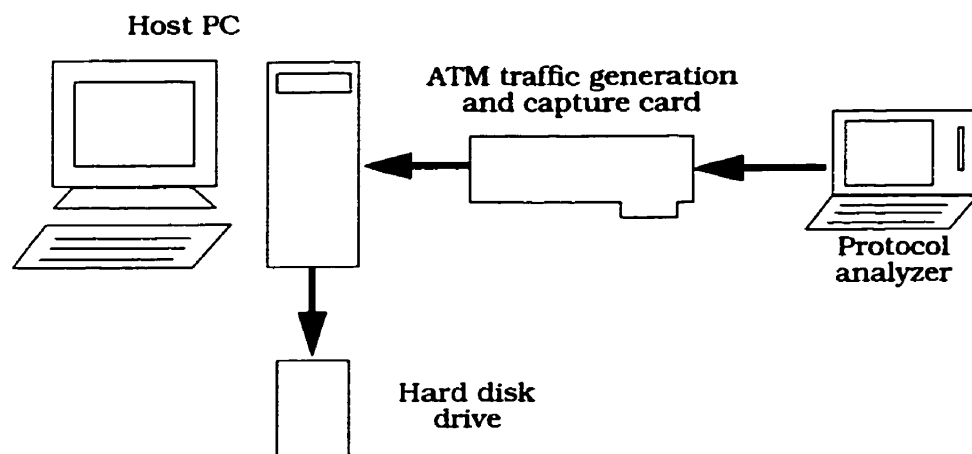
## **6.2 System testing**

Upon the completion of component testing, the next phase of testing performed on the ATM traffic generation and capture card was system testing. By assembling the hardware modules and software functions that were successfully enabled and verified to be operating correctly through component testing, the system testing phase proceeded to test the card's ability to generate and capture ATM traffic. This section will describe the approaches used for individually testing each of the ATM traffic processing operations.

### **Traffic Capture**

The first of the card's functions that was tested was ATM traffic capture. Traffic capture testing analyzed the card's ability to accurately capture the timing information of an ATM traffic stream. Testing checked that a traffic stream was correctly received and encoded by the card and transferred to a host. The strategy used for testing was to

configure the card for capture, have it capture traffic for a period of time, and then analyze the captured traffic log file that was created. Analysis of the log file checked that no ATM cells (from traffic stream that was supplied to the card) were dropped or mislabelled and that there were no extra cells added to the encoded traffic stream. Discussion in this part of the chapter will describe how this testing was performed and the results that were obtained. Figure 6.2 illustrates the environment that was used for testing ATM traffic capture.



**Figure 6.2** Environment used for testing ATM traffic capture

For testing traffic capture, the card was installed in its typical operating environment in a PCI slot of a host PC. The host was required to perform its usual functions of configuring and controlling the operation of the card and storing the ATM encoded traffic stream captured by the card. The host that was used for testing was an IBM-compatible PC which had an Intel Pentium processor running at 100 MHz, 64 mega-bytes of RAM, two giga-bytes of free hard disk space, and running the Windows 95 operating system (OS)..

A GNNetest interWATCH 95000 ATM protocol analyzer was used to supply the

card with a stream of ATM traffic to capture during testing. A protocol analyzer was selected as a traffic source because of its ability to be configured to supply a predictable traffic stream where the timing of the ATM cells that it generated could be accurately specified. The sequence of cells in the source traffic stream had to be known precisely in order to measure and verify the card's ability to accurately capture ATM traffic. Other (non-test) equipment could not have been used as a source of ATM traffic due to the difficulty (impossibility) in controlling or even predicting the timing of cells transmitted from these devices.

ATM traffic capture was tested with the protocol analyzer supplying the card with a fixed, repeating pattern of ATM cells. Selection of this periodic traffic pattern reduced analysis of the log files to simply synchronizing the transmitted pattern with the captured traffic at the start of the file and checking that this pattern was repeated throughout the remainder of the file. Upon detecting the occurrence of an error where one or more cells in the capture file did not match those expected in the source pattern, the incident was recorded, the transmitted pattern re-synchronized with the captured traffic, and the processing of the log file restarted. Testing with a random source pattern would have complicated analysis of the capture file; use of this type of traffic stream would have required the ATM analyzer to record the traffic that it generated for the entire duration of the test session. This would also have required translation between the ATM analyzer's log file format and the format used by the ATM traffic generation and capture card for comparison after the capture session ended. The ATM protocol analyzer that was used for the testing was also unable to create log files (of the traffic generated) for durations long enough to sufficiently test the traffic capture capabilities of the card.

The high-speed computation capabilities of a Sun UltraSparc workstation were exploited for analyzing the traffic log files captured by the card. At the end of a traffic

capture session, the captured traffic log file was transferred from the host to the workstation via a local-area network (LAN) connection. Software developed for the workstation processed the log file, recording and displaying instances where the traffic captured by the card differed from the ATM traffic stream supplied by the protocol analyzer.

The ATM traffic generation and capture card's traffic capture capability was tested with the five cell-sequences shown in Figure 6.3. Each of the sequences used was of a different length and there were some variations in the values of the header fields of cells within a single sequence as well as between sequences. The 48 payload bytes of all cells in the sequences were fixed at a hexadecimal value of 0x5A. Traffic capture was tested with each sequence for a period of approximately eight and a half hours. In each of these capture intervals, approximately 10.8 billion cells were captured by the card to create a captured traffic log file 1.3 giga-bytes in size.

No errors were found in the analysis of each of the eight and a half hour traffic log files captured during testing with the five ATM cell sequences. Analysis of the log files did not find any instance where an ATM cell or a group of cells in the traffic streams supplied to the capture card was either dropped from the encoded traffic stream or mislabelled as a cell not appearing in the source stream. Due to the periodic nature of the ATM traffic streams used for testing, it is possible that the card dropped cells from the encoded captured traffic log file and that the analysis software did not detect this loss. This would have been the case if the number of cells dropped by the card was a multiple of the number of cells in the cell-sequence used for testing. For example, it is possible that, during the capture using the five cell sequence, the card lost cells in blocks whose size was a multiple of five ATM cells. Analysis of the traffic log file captured using this sequence would not have detected the absence of cells in the encoded traffic stream if they were dropped in blocks of 5, 10, 15, etc. cells; capture file analysis would have

detected cell loss in the encoded traffic log file if groups of cells of any other size were dropped. If the card did have the tendency to drop cells in groups of five, for example, this would have been detected during testing with sequences of other lengths (i.e. 7, 9, 10, or 11 cells). Based on the results obtained from testing with each of the five ATM cell sequences, the ATM traffic generation and capture card does not lose or mis-label any cells when performing ATM traffic capture.

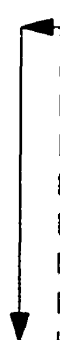
If analysis of the captured traffic log files had discovered traffic loss or mis-labelled cells, their occurrence could not have immediately been attributed to a failure of the ATM traffic generation and capture card to accurately capture traffic. Errors could have originated as a result of ATM cell generation problems at the ATM protocol analyzer or from transmission errors incurred over the fiber optic link from the protocol analyzer to the card, across the host's PCI bus as the encoded traffic stream is transferred from the card to the processor, or as the traffic stream is written to disk from the processor. Had such errors been found in the capture files, efforts would have been undertaken to identify the source of the failure.

Sequence 1						
GFC (bin)	VPI (dec)	VCI (dec)	PT (bin)	CLP (bin)	HEC (hex)	Payload bytes (hex)
0000	10	100	101	0	92	5A
0000	10	100	101	0	92	5A
0000	0	0	000	1	52	5A
0000	0	0	000	1	52	5A
0000	0	0	000	1	52	5A
Sequence 2						
GFC (bin)	VPI (dec)	VCI (dec)	PT (bin)	CLP (bin)	HEC (hex)	Payload bytes (hex)
0000	10	100	100	1	9B	5A
0000	10	100	011	1	B1	5A
0000	20	100	000	0	7C	5A
0000	30	100	101	1	05	5A
0000	10	100	110	0	80	5A
0000	20	100	110	0	58	5A
0000	0	0	000	1	52	5A
Sequence 3						
GFC (bin)	VPI (dec)	VCI (dec)	PT (bin)	CLP (bin)	HEC (hex)	Payload bytes (hex)
0000	0	0	000	0	55	5A
0000	0	0	000	0	55	5A
0000	10	50	100	1	FD	5A
0000	10	52	101	1	D4	5A
0000	0	0	011	1	40	5A
0000	15	100	001	0	8E	5A
0000	95	30	110	0	E8	5A
0000	70	20	000	1	1D	5A
0000	0	0	000	1	52	5A


Figure 6.3 Cell sequences used in testing card's ATM traffic capture capability



Sequence 4						
GFC (bin)	VPI (dec)	VCI (dec)	PT (bin)	CLP (bin)	HEC (hex)	Payload bytes (hex)
0000	10	100	111	0	8E	5A
0000	10	100	011	0	B6	5A
0000	30	100	100	1	0B	5A
0000	20	100	011	0	6E	5A
0000	20	100	011	0	6E	5A
0000	0	0	000	0	55	5A
0000	0	0	100	0	6D	5A
0000	30	100	100	1	0B	5A
0000	0	0	001	0	5B	5A
0000	0	0	000	1	52	5A



Sequence 5						
GFC (bin)	VPI (dec)	VCI (dec)	PT (bin)	CLP (bin)	HEC (hex)	Payload bytes (hex)
0000	10	52	101	1	D4	5A
0000	0	0	011	1	40	5A
0000	15	100	001	0	8E	5A
0000	95	30	110	0	E8	5A
0000	70	20	110	1	1D	5A
0000	0	0	000	1	52	5A
0000	40	50	000	0	AD	5A
0000	40	50	001	0	A3	5A
0000	0	0	000	0	55	5A
0000	0	0	000	0	55	5A
0000	10	50	100	1	FD	5A



**Figure 6.3 (con't)** Cell sequences used in testing card's ATM traffic capture capability

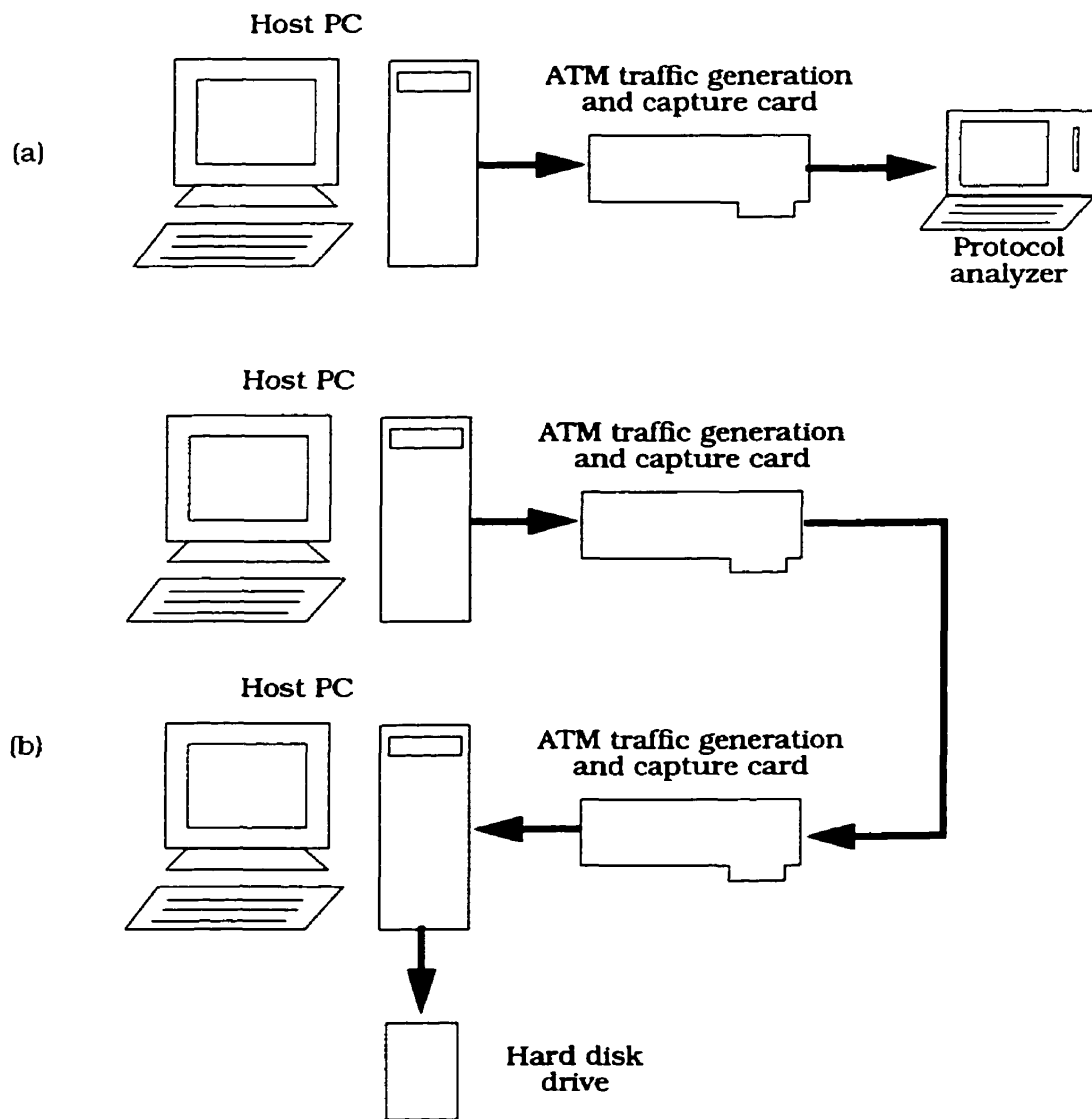
### Traffic Generation

The objective of testing the card's traffic generation function was to verify its ability to accurately generate ATM traffic. Testing checked that the card generated a stream of ATM cells which corresponded exactly to that represented by an encoded traffic stream supplied to it by a host PC. The generated stream was processed to check that there were no SONET signalling errors, that none of the cells from the encoded source

stream were dropped or transmitted as errored, and that no extra cells were inserted into the stream. The approach used for testing was to enable the card for ATM traffic generation, have it generate traffic for a period of time, and to analyze the traffic that was transmitted. This part of the chapter will discuss the testing that was performed and the results that were obtained.

For testing traffic generation, the ATM traffic generation and capture card was installed in a PCI slot of the same 100 MHz Pentium host PC that was used for testing ATM traffic capture. The host had the responsibility of configuring the card for generation and to supply it with the encoded traffic stream representing the sequence of ATM cells that were to be generated.

There were two environments used in the testing of the ATM traffic generation operation. In the first, the GNNetest interWATCH 95000 ATM protocol analyzer used earlier was connected to the fiber optic transmit port of the ATM traffic generation and capture card as shown in Figure 6.4(a). The analyzer was capable of capturing approximately 1.55 million cells, which corresponds to about 4.3 seconds of ATM traffic transmitted at an OC-3c rate. The analyzer was capable of recording the occurrence of SONET signalling errors such as loss of signal, loss of frame, out of frame, STS loss of pointer, and loss of cell delineation; it was also able to keep a count of cells received from the card which contained header errors. At other points during the testing, a second ATM traffic generation and capture card configured for traffic capture was connected to the generation card's transmit output in an arrangement like that illustrated in Figure 6.4(b). The host used to configure and control the operation of the ATM traffic capture card had a Pentium processor running at 120 MHz, had 64 mega-bytes of RAM, and was running the Windows95 operating system. The following discussion will describe the purpose of each of these test set-ups and when each was used.



**Figure 6.4** Environment used for testing ATM traffic generation  
(a) with an ATM analyzer capturing generated traffic, (b) with an ATM capture card capturing generated traffic

Testing of traffic generation began by checking that the ATM traffic stream transmitted from the card did not contain any cells with errored headers, that there were no SONET signalling errors, and that the card did not insert any cells into the generated traffic stream which were not specified in the encoded traffic stream supplied by the

host. For this phase of testing, the ATM protocol analyzer was connected to the fiber optic output of the ATM traffic generation and capture card. The analyzer was used to record the occurrence of SONET signalling errors and the number of ATM cells received with header errors. An ATM cell filter on the analyzer was configured to retain only those cells received from the card which did not match those specified in the encoded traffic stream. The host in which the card was installed configured the card for ATM traffic generation and supplied an encoded traffic stream containing ATM cell sequence (1) shown in Figure 6.5. This sequence was repeated throughout the encoded traffic stream (supplied to the card) to produce an ATM traffic stream with a bandwidth utilization of 100% of the OC-3c rate. During eight hours of testing, the card generated a stream of approximately 10.1 billion cells which did not contain any SONET signalling errors, errored cell headers, or cells not belonging in the traffic stream.

The next phase of traffic generation testing examined the sequencing of cells in the ATM traffic stream generated by the card. The objective here was to verify that the generated sequence exactly matched that specified in the encoded traffic stream supplied by the host. Analysis of the generated traffic streams was performed individually with both the ATM analyzer and with a second ATM traffic generation and capture card. Each of the three cell sequences illustrated in Figure 6.5 was used in the testing.

A visual examination was made on some of the cells recorded in several ATM traffic log files captured by the ATM analyzer. Due to the large number of cells captured by the analyzer, it was not practical to examine all of the cells in the four second capture buffer. Rather, only small groups of cells were checked at different points in the captures performed with the analyzer. The groups of cells that were looked at matched that which the card was expected to generate and were absent of any cells not specified in the encoded traffic stream.

Sequence 1						
GFC (bin)	VPI (dec)	VCI (dec)	PT (bin)	CLP (bin)	HEC (hex)	Payload bytes (hex)
0000	10	100	110	0	80	5A
0000	10	100	110	0	80	5A
0000	5	100	000	0	C8	5A
0000	5	100	000	0	C8	5A
0000	5	100	000	0	C8	5A
Sequence 2						
GFC (bin)	VPI (dec)	VCI (dec)	PT (bin)	CLP (bin)	HEC (hex)	Payload bytes (hex)
0000	20	48	100	0	C2	5A
0000	0	0	000	1	52	5A
0000	0	0	000	1	52	5A
0000	20	48	100	0	C2	5A
0000	20	48	100	0	C2	5A
0000	20	48	100	0	C2	5A
0000	0	0	000	1	52	5A
Sequence 3						
GFC (bin)	VPI (dec)	VCI (dec)	PT (bin)	CLP (bin)	HEC (hex)	Payload bytes (hex)
0000	20	100	101	1	4D	5A
0000	40	50	000	1	AA	5A
0000	20	100	101	1	4D	5A
0000	40	50	000	1	AA	5A
0000	40	50	000	1	AA	5A
0000	40	50	000	1	AA	5A
0000	20	100	101	1	4D	5A
0000	20	100	101	1	4D	5A
0000	40	50	000	1	AA	5A
0000	40	50	000	1	AA	5A

**Figure 6.5** Cell sequences used in testing card's ATM traffic generation capability

In addition to the limitation of the protocol analyzer to be used to examine the generated cell stream thoroughly, use of the analyzer to verify the cell stream was less than ideal for another reason. The limited traffic capture capability of the ATM analyzer

did not permit the card's generation function to be verified for periods nearly as long as for which it was designed to operate; the ATM analyzer's four second traffic capture buffer did not come close to allowing the card's generation function to be tested for intervals that were "hours long" in duration, as suggested in the design specification.

As a result of the shortcomings of using the ATM analyzer to verify the sequence of cells generated by the card, a second ATM traffic generation and capture card was configured to capture the traffic generated by the first card. With this set-up, the card's traffic generation function could be tested for intervals at least a few hours in length and analysis of the generated stream (captured by the second card) could be automated through software similar to that used for the traffic capture testing. Since the card had previously been proven capable of accurately capturing an ATM stream, it could be used with confidence in the testing of the traffic generation function.

Like the approach employed in the verification of ATM traffic capture, a Sun UltraSparc workstation was used in the testing of the card's ATM traffic generation operation. After generating traffic for a period of time, the traffic log file captured by the second card was transferred from the supervising host PC over a LAN to the Sun workstation. The analysis software that was developed for the workstation examined the traffic stream that was captured, searching for instances where it differed from the encoded stream that the host supplied to the card generating ATM traffic.

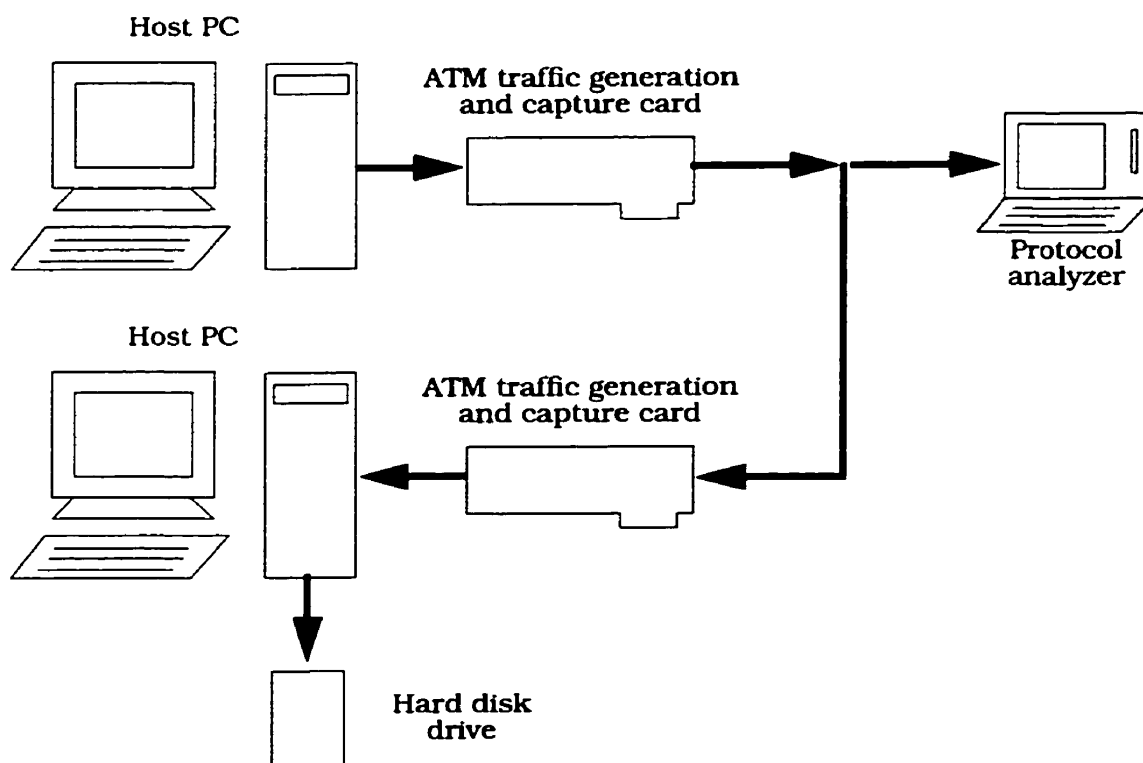
Traffic generation testing was performed (individually) with the three periodic ATM cell sequences shown in Figure 6.5. Testing with each sequence was performed for a period of 6.8 hours. Approximately 8.7 billion cells were generated by the card during each of these intervals and around 26 billion cells during the entire 20 hour test period. The cell sequence that the host supplied to the card during each test was hard-coded in the software which it used to configure and control the operation of the card.

Analysis of the ATM traffic generated by the card found that the ATM traffic generation and capture card was able to generate a sequence of cells which exactly matched the encoded source stream provided to it. In each of the three seven-hour tests, there were no errors (in the log file of the generated traffic captured by the second card) occurring in the form of extra cells inserted or cells dropped from the source stream.

Although testing with the ATM traffic capture card allowed cell sequence errors in the generated traffic stream to be detected, SONET signalling errors occurring in the transmitted stream could not be identified. During intervals when transport errors occur in the SONET signalling, groups of cells can be demapped from the physical transport correctly, incorrectly, dropped entirely, or in some of combination of these. As such, it would not have been possible to differentiate (in the encoded traffic stream captured by the second card) between instances of SONET signalling errors and errors in the sequence of ATM cells generated by the card. In retrospect, an improved approach to analyzing the generated traffic would have been to distribute the generated traffic stream to the capture card and to the protocol analyzer using an optical splitter in an arrangement like that shown in Figure 6.6. The ATM capture card would record the sequence of cells received from the generator card while the protocol analyzer would be used to check for the occurrence of transport errors. This approach would serve as a means of determining whether SONET errors were experienced during a given test interval, but would not provide any way of associating them with the corresponding cell sequence errors in the capture log.

Had cell sequence errors been found in the generated ATM traffic stream during the testing that was performed, their occurrence may not have necessarily indicated that the ATM traffic generator card was at fault. Errors may have resulted due to a fault of the generator card and its host to transmit the proper sequence, failure of the capture

card and its host to accurately receive the cell sequence, or a transmission error on the fiber optic transport itself. Occurrence of any errors during the testing would have required further investigation to identify their cause.



**Figure 6.6** Preferred approach for performing ATM traffic generation testing



---

# **Chapter 7**

## **Discussion**

There were some choices made in the design of the ATM traffic generation and capture card which facilitated its implementation and some choices which complicated the implementation and limited the card's overall performance. This chapter will discuss these choices as well as other aspects of the card's development.

The architecture designed for the ATM traffic generation and capture card allowed a powerful network interface card to be realized. Use of an FPGA to implement the functionality required of the traffic stream translator was appropriate for a number of reasons. With its ability to be reprogrammed, the FPGA permits the functionality of the card to be switched easily between traffic generation and traffic capture as well as additional ATM traffic processing capabilities that may be developed in the future. As mentioned in the previous chapter, the FPGA facilitated testing and debugging of several components on the card through the use of test-specific configurations. With the use of a microcontroller for programming the functionality of the FPGA, the card's behavior and capabilities can be changed quickly and easily with only minimal guidance required from a user. The inclusion of an interface to the PCI Local Bus provides the potential for high-speed data transfers between the card and a host.

The use of an FPGA to implement the traffic stream translator functionality

helped in meeting the low (fabrication) cost objective set for the ATM traffic generation and capture card's design. The FPGA permitted the card to be built at a lower cost than would have been possible had this functionality been targeted to an ASIC instead. Figure 7.1 lists approximate costs of the FPGA and other components on the board as well as fabricating the printed circuit board. The \$830 total estimated cost is considerably less than the price of most commercial network analysis tools and therefore meets the cost objectives for the card. The costs listed in the figure reflect low production volumes of the card; further reductions may be realized as volume is increased.

Component cost	
FPGA	102
S5933	55
S/UNI-LITE	39
RAM	57
Microcontroller	21
Optical transceiver	51
RS-232 interface	3
Crystals/oscillators	14
Clock buffer	3
Resistors, capacitors inductors, and other parts	50
	<hr/>
	395
PCB fabrication	435
	<hr/>
	\$830 (US)

**Figure 7.1** Approximate material cost of ATM traffic generation and capture card  
([1], [8], [11], [12], [27], [30])

Appropriate choices were made in the selection of components used for the implementation of the ATM network interface and host interface modules of the ATM traffic generation and capture card. The S/UNI provides an effective interface to an ATM network's physical layer for the purpose of transmitting cells during traffic generation and receiving cells during traffic capture. It allows idle/unassigned cells to be specified in the generated traffic stream and retained in the captured traffic stream. This feature

is important for maintaining accurate timing information of ATM cells when translating between the ATM cell stream and the encoded traffic stream. With its ability to perform data transfers through three types of transfer mechanisms, the S5933 PCI controller provides a flexible means of interfacing to the PCI Local Bus. The S5933's ability to support asynchronous data transfers on its add-on side was useful as it allowed components on the card to operate at a 20 MHz clock rate rather than at the higher PCI bus clock rate of 33 MHz.

Some problems were encountered during the card's development as a result of the particular FPGA selected to implement the traffic stream translator module. Although only 38% and 57% of the Xilinx XC3195A's logic resources (CLBs) were required for individual realizations of the traffic capture and traffic generation capabilities respectively, implementation of these designs often could not be completed. Due to limited routing resources within the FPGA, the nature of the designs to have wide data paths, and constraining pin assignments early in the design cycle, Xilinx's automatic place and route software was frequently unable to route all of the signals (nets) in the design. Even when signal routing was completed, the designs did not always function correctly because the implemented circuit did not meet the necessary timing requirements. The internal gate delays of this FPGA affected the translator's internal operation and the operation of its interfaces with other components on the board. These limitations were eventually overcome through simplification of the designs' VHDL descriptions and multiple retries of the place and route process. The XC3195A also did not have a sufficient number of I/O pins to allow the data bus between the FPGA and the add-on side of the S5933 to be used in its full 32-bit width. Implementation as an 8-bit bus effectively eliminated the possibility of doing burst transfers of data across this bus. The limitations of this FPGA forced the functionality of the traffic stream translator to be

developed in the most efficient manner possible.

In the early stages of testing the card's ability to capture ATM traffic, analysis of the capture log files found that traffic was regularly being lost. Analysis showed that the traffic that was captured in the log files would match that supplied by the protocol analyzer for short periods of time, but there would be instances where a number of cells from the source stream would be missing in the capture file. With the periodic traffic streams that were used for testing traffic capture, it was not possible to tell exactly how many cells had been dropped. What was seen was that the traffic loss was found to occur at intervals which corresponded to the start of the 3-byte block of captured traffic that was transferred from the card to the host. The ATM traffic stream encoding that was being used during this testing required a constant data transfer rate between the card and the host of approximately 60 kilo-bytes per second<sup>4</sup>. This is equivalent to one double-word transferred every 70 micro-second and is slightly less than 0.05% of the peak transfer rate that the PCI bus is able to provide.

Preliminary indications were that traffic loss was occurring during its transfer from the card to the host. Since the loss was consistently found to occur at the start of the 3-byte blocks of traffic transferred from the card and not at other points in the block, it appeared as though the S/UNI was successfully receiving the ATM traffic stream and the translator was encoding the traffic stream correctly. The integrity of the ATM traffic stream supplied by the protocol analyzer was checked using two ATM traffic generation and capture cards, each configured for traffic capture and each installed in its own host. The ATM traffic stream supplied by the protocol analyzer was distributed to each card using an optical splitter to ensure that both cards simultaneously received an identical

---

4. This data transfer rate includes the response identifier byte that is always passed between the card and the host

traffic stream. Processing and comparison of the two log files from a few test captures found that, although the nature of the traffic loss experienced with both cards was similar, the loss did not occur at the same points in both capture files. This indicated that either the ATM traffic generation and capture card was failing to transfer the entire captured traffic stream to the host or the host was losing parts of the stream received from the card.

Since the captured traffic log files were processed only upon the termination of a capture session, it was not possible to determine the particular events on the card or the host that led up to the occurrence of traffic loss in the log. Although the host could have been programmed to analyze the captured traffic in real-time as it was received from the card, this additional processing by the host would have reduced the rate at which the traffic stream was read from the card and would have likely resulted in further traffic loss.

In an attempt to identify the reason why traffic was being lost in the capture files, the functionality of the system controller module within the traffic stream translator was modified so that a sequence number was attached to each of the captured traffic double-words (in the response identifier byte) that were loaded into the S5933's add-on to PCI FIFO. The host software was modified so that the whole double-word (the response identifier byte and the three bytes of captured traffic) was written to disk during traffic capture. At the end of the capture session, the log file analysis software checked the sequence numbers attached to each of the double-words. Analysis of the capture files found that every double-word that the translator loaded into the S5933's FIFO was being successfully received by the host and written to disk.

The system controller module was also configured to set an error flag if it ever had captured traffic available and was unable to transfer it to the host because it found

that the S5933's add-on to PCI FIFO was full. A few test captures found that this flag was consistently being set, indicating that overflow of captured traffic was occurring at the card because of a failure of the host to read some of the traffic from the S5933's FIFO. This tended to suggest that some task running on the host was preventing it from reading the captured traffic from the card for short periods of time.

A number of optimizations were made to the host and its support software for the card in order to allow it to devote more of its time to reading the S5933's add-on to PCI FIFO. These modifications included removing the host's PCI network and sound cards, reducing the resolution and the number of colors used by the host's video display, optimizing the software that read the captured traffic from the card and wrote it to disk, and even using a host with a faster processor and increased memory. All of these modifications were made to no avail - traffic was still being lost.

A PCI bus analyzer was used to record bus activity to determine the degree to which the host's PCI bus was being used. This bus analyzer card was capable of capturing 512,000 bus events, but did not find any burst transfers or any device making excessive use of the bus. What the analyzer did find was that the host was continuously reading the status register within the S5933 to check for the availability of data in the add-on to PCI FIFO.

Although it was never determined what particular event or task on the host momentarily prevented it from reading captured traffic from the card, the occurrence of traffic loss in the capture log files was eventually eliminated by supplementing the S5933's add-on to PCI FIFO with buffering on the card and on the host. The purpose of these buffers was to provide a means of decoupling the rate at which the card captured and encoded an ATM traffic stream and the rate at which the host transferred the traffic from the card to its hard disk. On the card, a cache for storing captured traffic was

implemented as a circular buffer in the card's on-board RAM. All traffic was written to this cache as it was captured and encoded by the traffic stream translator. Logic within the translator transferred captured traffic from the cache to the add-on to PCI FIFO as space became available in the FIFO. The cache that was implemented allowed almost four and a half seconds of captured ATM traffic (in encoded form) to be buffered for transfer to the host.

Buffering on the host was implemented as an array with the capacity for storing almost 1.4 seconds of captured traffic. All encoded traffic which the host read from the card was temporarily buffered in this array. When the host found that the S5933's add-on to PCI FIFO contained no captured traffic, it would use this idle period as an opportunity to write blocks of traffic from its buffer to the hard disk. Depending upon the amount of traffic available in the host's buffer, captured traffic would be written to disk in blocks of either 1, 4, 8, or 16 double-words, with the objective being to write data in blocks as large as possible for most-efficient disk accesses. The host also tried to read captured traffic from the S5933's FIFO as efficiently as possible. A status register within the S5933 provides an indication of whether the add-on to PCI FIFO contains at least one double-word, at least four double-words, or is completely full with eight double-words of data. Depending upon the status, the host read data from the FIFO in a series of one, four, or eight single-data-phase PCI read transactions. While this approach is a little more efficient than examining the status register after each individual read (of a single double-word), it is still slower than performing burst read transactions.

Based on the results of ATM traffic capture testing, the size of the caches implemented on the card and on the host were proven to be sufficient to eliminate all traffic loss in the log files. Either or both of these caches may in fact be larger than necessary for achieving traffic capture without any loss.

Due to the host's failure to regularly read captured traffic from the card and for overflow on the card and traffic loss in the captured traffic log files to occur, it was anticipated that buffering of the encoded traffic stream would also be needed for the ATM traffic generation operation. Failure of the host to supply the card with the traffic stream to generate would cause idle/unassigned cells to be inserted into the stream of generated traffic. The resulting traffic would consequently differ from that which was to be generated and would have a lower overall bandwidth utilization.

Like traffic capture, buffers were implemented on the card and on the host. A circular buffer implemented in the card's RAM allowed almost nine seconds of encoded traffic to generate to be cached. All traffic that the traffic stream translator received from the S5933's PCI to add-on FIFO was transferred to this cache. The translator read and decoded traffic from the cache as it was required.

When the host supplied the card with traffic (to generate) read from a log file stored on its hard disk<sup>5</sup>, a cache was implemented on the host which allowed it to buffer over 27 seconds of the encoded ATM traffic stream. Before transferring any traffic to the card, the host checked the S5933's PCI to add-on FIFO status register to determine how much space was available. Depending on the status, the host transferred the encoded traffic stream to the card in a sequence of one, four, or eight single-data-phase write transactions. During periods when the FIFO was full, the host read the encoded traffic stream from its hard disk in blocks 13.6 milli-seconds in size, as dictated by the amount of space available in its cache.

Several hours of traffic generation testing was performed with the host supplying the card with an encoded ATM traffic stream which was read from traffic log files stored

---

5. A cache was not implemented and, as the testing has proven, was not needed on the host when the encoded traffic stream that it provided to the card was hard-coded in the support software.



on its hard disk. Testing was done with the three ATM cell sequences shown in Figure 6.5 for a period of 6.8 hours each. Prior to beginning traffic generation testing, each sequence was written to a separate traffic log file.

Analysis of the captured traffic log files found that cell sequence errors occurred during disk-based traffic generation testing with each of the three sequences. Seven errors occurred during testing with sequence (1) from Figure 6.5, 5 errors with sequence (2), and 10 with sequence (3). The error occurrences were distributed throughout the duration of each of the test periods. In each of the instances where an error was found, the error that occurred was one where a number of cells from the source pattern were missing in the capture file. Because the cell sequences used in the testing were periodic, it was not possible to determine exactly how many cells were missing at each occurrence.

The type of cell sequence errors that occurred during ATM traffic generation testing with traffic log files tend to suggest that a fault exists with the management of the host's traffic buffer. As a result of the absence of errors experienced during testing of ATM traffic capture and testing of traffic generation with hard-coded traffic streams in addition to the fact that errors only occurred during generation testing with log files on disk, it is very unlikely that sequence errors were introduced by the card capturing the generated traffic. Also, successful testing of traffic generation using cell sequences hard-coded in the host's software indicates that the card is able to receive, decode, and generate the ATM traffic stream and manage its own traffic cache correctly. The fact that the cell sequence errors were characterized as missing cells from the source stream indicates that the host supervising the generation card is transferring the encoded traffic stream to the card at a high-enough rate. If this were not the case, extra idle/unassigned cells would be found at the error points. For the reasons mentioned above, the

errors in the sequence of cells generated by the card occurred as a result of a failure of the host to properly manage the cache it uses to buffer the encoded traffic stream transferred from its hard disk to the card. This problem was never solved.

---

# **Chapter 8**

## **Recommendations and Future Work**

A number of opportunities exist for future development of the ATM traffic generation and capture card. Some additional work is required to improve the operation of the card when it generates ATM traffic from log files stored on a host's hard disk. There are also several modifications that can be made to the current implementation of the card. Some of these modifications would serve to enhance the performance of the card's existing capabilities, while others would add increased functionality to the card. Some can be realized with the current version of the card, while others may require that a new card be designed and fabricated. This chapter will discuss a number of these improvements and make recommendations for future work on the card.

The most important work required for the current implementation of the ATM traffic generation and capture card is to eliminate the cell sequence errors that result when the card generates ATM traffic from log files stored on the host's hard disk. It is believed that these errors are occurring as a result of improper management of the traffic cache residing on the host. Once the cause of the problem has been identified, several hours of testing should be performed before the card can be used with confidence to

accurately generate ATM traffic streams that match those specified in encoded traffic log files.

In the next revision of the ATM traffic generation and capture card, the Xilinx XC3195A FPGA selected to implement the traffic stream translator module should be replaced with one from a different family of FPGAs. This replacement should be made to resolve some of the limitations of the XC3195A discussed in the previous chapter and to provide additional resources to allow increased functionality to be added to the card. The new FPGA should have improved routing resources and smaller signal delays. A faster part could also allow the clock rate on the card to be increased to the PCI bus clock rate of 33 MHz. The selection of an FPGA with additional I/O pins would allow the data path between the translator and the S5933 to be implemented in its full 32-bit width to increase the efficiency and data transfer rate of exchanges between the host and the card. Ideally, the FPGA that is chosen would be a member of a family which contains larger (increased logic capacity and I/O pin count) and faster pin-compatible parts. Such a choice would minimize the number of modifications to the PCB should it be necessary to make further FPGA replacements. Possible candidate parts include those from the Xilinx Virtex and Altera FLEX 10K families.

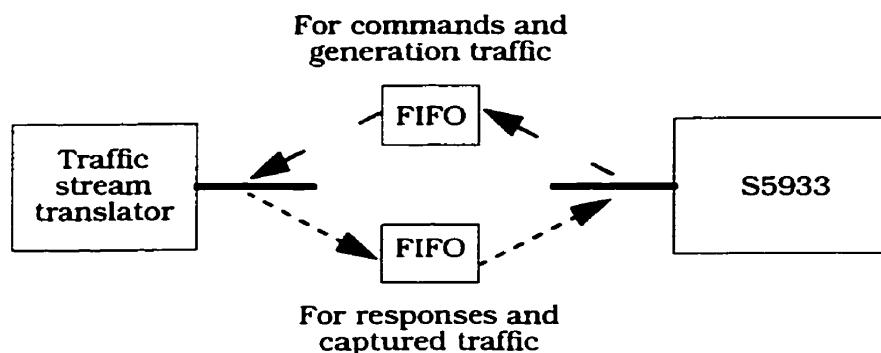
Future work with the card should consider redevelopment of the host software for use under an operating system such as UNIX, VXWorks, or WindowsNT. The purpose of making a change to the host's operating system is to attempt to resolve the problems that occurred when the host periodically failed to transfer the encoded traffic stream to and from the card when the Windows95 operating system was used. Further investigation is required to determine whether a different operating system would read or write the S5933's FIFOs a little more regularly when ATM traffic capture or generation is in progress.

Enabling of PCI bus burst transfers between the host and the card is another modification that should be made in the future. The use of burst transfers would make more efficient use of PCI bandwidth and would serve to increase the rate at which the encoded traffic stream is exchanged between the host and the card. Burst transfers, together with a change of operating system for the host, may eliminate all possibility of captured traffic overflowing at the card, particularly if a higher-bandwidth ATM traffic encoding scheme is ever used. These changes may even allow the host to support the concurrent operation of a number of traffic generation and capture cards installed in a common host.

The ability of the current version of the card to realize burst transfers depends on the particular data transfer mechanism (of the S5933) chosen and how it is used. S5933-initiated burst transfers using the FIFOs could likely be implemented given the internal data buffering that this transfer mechanism provides. During a transfer to the host, the translator can fill the add-on to PCI FIFO and let the S5933 initiate the transfer when the PCI bus becomes available. When receiving data from the host, the S5933 can fill the PCI to add-on FIFO and let the translator read data from the FIFO at its own pace. As a result of the local clock rate on the card and the width of the S5933-translator data path, burst transfers of more than eight double-words (the capacity of each of the FIFOs) cannot be efficiently realized; with a lower clock rate (than the PCI bus clock) and the narrow data path, wait states would occur in the burst transaction as the translator loads data into or reads data from the FIFOs. Burst transfers using the S5933's pass-thru registers are not effective for the same reason.

To improve the card's ability to support PCI bus burst transfers between itself and the host, future versions of the ATM traffic generation and capture card should add two 32-bit wide FIFOs on the add-on side of the S5933. These FIFOs would allow burst

transfers (made using the S5933's FIFOs or pass-thru registers) to directly access data storage space on the card and would eliminate the dependency on the translator to supply and receive data at rates high-enough to support the bursts. The translator would have to write responses and captured traffic that it wishes to send to a host to one FIFO and would read from the other FIFO commands or generation traffic transferred from the host. Figure 8.1 illustrates how the data path between the S5933's add-on side and the traffic stream translator should be modified to incorporate these FIFOs.



**Figure 8.1** External FIFOs on the ATM traffic generation and capture card for buffering data for PCI bus transfers

One application of the card which could be developed is a capability for simulating and recording various network error conditions. This application would use the card to introduce errors into the ATM traffic stream which it transmits to a network and record errors in a traffic stream received from a network, allowing other aspects of network operation to be studied. If appropriate configuration information is written in its internal registers, the S/UNI can be instructed to insert errors such as framing pattern, bit interleaved parity, and illegal pointer errors. By reading the contents of various status registers within the S/UNI, occurrence counts of these same errors on the input from a network can be recorded. The S/UNI is also capable of recording the number of

correctable and uncorrectable header errors occurring in a traffic stream received from an ATM network. As the host and translator are presently able to access the S/UNI's configuration/status registers, this new functionality can be realized with only minor modifications to the host software to enable the desired errors conditions to be inserted and recorded.

By integrating the error insertion and detection functionality described previously with the card's existing capabilities for ATM traffic generation and capture, a further improvement in the card's usefulness for studying network behavior can be realized. During traffic generation, the card could be configured to insert errors into the traffic stream which it transmits to a network. This would increase the realism of the traffic stream used to drive an ATM network or switch. Periodically reading the S/UNI's registers which count the occurrence of errors in the receive stream during traffic capture would serve to provide a measure of the accuracy of the corresponding intervals of captured traffic in the log files. Inclusion of this error information would involve significant modifications to the encoding scheme used to represent the ATM traffic stream and would likely require the fabrication of an improved version of the ATM traffic generation and capture card which implements the recommendations discussed in this chapter.

---

# **Chapter 9**

## **Conclusion**

This thesis has discussed the design and implementation of an ATM network interface card developed for the purpose of generating and capturing ATM traffic at OC-3c rates. The need for this card was motivated by the expense and capabilities of ATM network test equipment available commercially at the time of the project's inception. Due to commercial test equipment's high cost, limited ability to capture network traffic, and inability to generate traffic with certain properties, the testing that is performed on network equipment and networks themselves is often less than ideal. Testing is done in environments which are not realistic and do not provide sufficient data to accurately characterize network behavior.

The ATM traffic generation and capture card that was designed attempted to overcome the shortcomings of testing with commercial equipment by implementing a network interface card designed with the capability for generating and capturing network traffic for extended periods of time, with the flexibility to generate all types of network traffic, at a low cost. Selection of commercial components for the implementation of the card and the use of an FPGA to realize logic functions which could not be found commercially allowed the card to be built at a low cost. The decision to use an IBM-compatible personal computer as the host responsible for configuring and controlling the



operation of the ATM traffic generation and capture card was made to further help the card to be recognized as a low-cost alternative to testing with commercial network test equipment. An encoding scheme used for representing the ATM traffic to be generated to the network and that which was received from the network helped to achieve long ATM traffic processing intervals and minimize the amount of space required for storing these streams. The host's ability to supply the traffic stream to generate and receive the captured traffic stream was the only factor determining the duration of time for which the card could perform either ATM traffic processing operation. By using a traffic representation scheme which specified the traffic as a sequence of cells, the card was able to generate streams of ATM traffic with an extremely large range of characteristics.

Through several hours of testing, the ATM traffic generation and capture card was verified to be capable of accurately generating and capturing ATM network traffic over long periods of time. Testing checked that the card could generate a stream of ATM traffic which did not contain any SONET signalling errors, that none of the cells transmitted had errors in their headers, and that the proper sequence of ATM cells were generated without any extra cells inserted or any required cells dropped from the transmitted stream. For traffic capture, testing verified the card's ability in accurately recording the timing information of an ATM traffic stream supplied to it. Testing of this function checked that the card captured all of the cells in the traffic stream provided to it and that it did not insert into the capture log any cells not originally specified in the source stream. No errors were experienced during testing of either of the card's ATM traffic processing capabilities.

While the traffic generation and capture card discussed in this thesis was able to perform each of the ATM traffic processing operations as required, its capabilities were limited somewhat as a result of some of the choices made during its design. The use of

an FPGA to implement the functionality of the traffic stream translator module was beneficial as it helped in the low-cost realization of the card, allowed the card's function to be switched easily between traffic generation and traffic capture as well as other ATM traffic processing operations, and facilitated testing of other components on the board. Unfortunately, choosing the largest and fastest part in the device family resulted in constraints being placed on the card's capabilities when it was discovered that the FPGA did not have a sufficient number of I/O pins or enough logic capacity and that it had difficulty operating at the required clock rate. The inclusion of the PCI Local Bus interface on the card provided the possibility of high data transfer rates, but proved to be inefficient unless data was transferred in burst transactions. Difficulty in achieving the required transfer rates for the encoded traffic streams complicated the implementation of the card and may preclude the use of other ATM traffic encoding schemes which require more bandwidth. Development of the card was approached as a hardware design project, with little consideration made to the support software running on the host. The approach undertaken for the design of this part of the system should have examined techniques that ensure that the encoded traffic stream is transferred to and from the card in a timely manner. Valuable insight was gained from the implementation of the ATM traffic generation and capture card which can be applied to future revisions of the card or to the design of other similar devices.

---

# Chapter 10

## References

- [1] Accutrace Inc., <http://www.accutrace.com/>, October 1999.
- [2] Applied Micro Circuits Corporation, *S5933 PCI Controller Data Book*, San Diego, CA, Spring 1996.
- [3] ATM Forum, *ATM User-Network Interface Specification, Version 3.1*, Mountain View, CA, 1994.
- [4] ATM Forum, *UTOPIA Specification, Level 1, Version 2.01*, Mountain View, CA, March, 1994.
- [5] J. Beran, R. Sherman, M.S. Taqqu, W. Willinger, "Long-Range Dependence in Variable-Bit-Rate Video Traffic," *IEEE Transactions on Communications*, vol. 43, no. 2/3/4, pp. 1566-1579, 1995.
- [6] C. Blum, *The Serial Port FAQ*, [http://www.repairfaq.org/filipg/LINK/PORTS/F\\_The\\_Serial\\_Port.html](http://www.repairfaq.org/filipg/LINK/PORTS/F_The_Serial_Port.html), Dec. 1995.
- [7] Dallas Semiconductor, *Soft Microcontroller Data Book*, Dallas, TX, 1993.
- [8] Digi-Key Corp., <http://www.digikey.com/>, October 1999.
- [9] GN Nettest, *InterWATCH 95000 Protocol Analyzer User Guide, R2.0*, Markham, ON.
- [10] W.J. Goralski, *Introduction to ATM Networking*, McGraw-Hill, Inc., New York, NY, 1995.
- [11] Insight Electronics, <http://www.insight-electronics.com/>, October 1999.
- [12] InTELaTECH Inc., <http://www.intelatech.com/>, September 1999.
- [13] International Telecommunications Union, *Recommendation I.321, B-ISDN Protocol Reference Model and its Application*, 1991, Geneva, Switzerland.

- [14] International Telecommunications Union, *Recommendation I.363, B-ISDN ATM Adaptation Layer (AAL) Specification*, March 1993, Geneva, Switzerland.
- [15] International Telecommunications Union, *Recommendation I.432, B-ISDN User-Network Interface - Physical Layer Specification: General Characteristics*, February 1999, Geneva, Switzerland.
- [16] P.K. Kurup, T. Abbasi, *Logic Synthesis using Synopsys, Second Edition*, Kluwer Academic Publishers, Norwel, MA, 1997.
- [17] W.E. Leland, M.S. Taqqu, W. Willinger, D.V. Wilson, "On the Self-Similar Nature of Ethernet Traffic," *IEE/ACM Transactions on Networking*, vol 2, no. 1, pp. 1-15, 1994.
- [18] D.E. McDysan, D.L. Spohn, *ATM: Theory and Application*, McGraw-Hill, Inc., New York, NY, 1994.
- [19] Microsoft Corp., *Visual C++ Online Documentation*, Ver. 4.2, January 1997.
- [20] PCI Special Interest Group, *PCI Local Bus Specification, Revision 2.1s*, Hillsboro, OR, June 1995.
- [21] D.L. Perry, *VHDL, Second Edition*, McGraw-Hill, Inc. New York, NY, 1994.
- [22] PMC-Sierra, Inc., *Interfacing and PC Board Layout Suggestions for the S/UNI-LITE*, Application Note, Issue 8, Burnaby, BC, September 1997.
- [23] PMC-Sierra, Inc., *PM5346 S/UNI-155-LITE Data Book*, Burnaby, BC, 1996.
- [24] PMC-Sierra, Inc., *S/UNI-Lite Optical Reference Design, Issue 2*, Burnaby, BC, September 1995.
- [25] T. Shanley, D. Anderson, *PCI System Architecture, Third Edition*, Addison-Wesley Publishing Co., Reading, MA, 1995.
- [26] W. Stallings, *High-Speed Networks: TCP/IP and ATM Design Principles*, Prentice-Hall, Upper Saddle River, NJ, 1998.
- [27] Starvoy Technologies Inc., <http://www.starvoy.com/>, September 1999.
- [28] Synopsys, Inc., *Synopsys Online Documentation*, Release 1998.02, Mountain View, CA, 1998.
- [29] A.S. Tannenbaum, *Computer Networks, Third Edition*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1996.
- [30] Wyle Electronics, <http://www.wyle.com/>, October 1999.
- [31] Xilinx, Inc., *The Programmable Logic Data Book 1998*, San Jose, CA, 1998.

- [32] Xilinx, Inc., *Xilinx Online Documentation, Alliance M1.4*, San Jose, CA, 1998.