Network Latency Measurement for l_2O Architecture

By Logic Analyzer Instrumentation Technique

A Senior Project Report Presented to the
Electrical Engineering Department
California Polytechnic State University San Luis Obispo

In Partial Fulfillment
of the Requirement for the Degree
Bachelor of Science in Electrical Engineering

By
Yu Guang Liang
June 1998

ABSTRACT

This goal of this project is to demonstrate a new technique for the measurement of network performance of a new computer I/O technology called "I_2O." I_2O stands for Intelligent Input and Output. The specific measurement investigated is network latency, the response time of the system to network requests. The project requires not only the understanding of the I_2O architecture, but also issues including the PCI bus, networking protocols such as TCP/IP and Ethernet, the inner structure of Windows NT, and the Intel's i960 Rx I/O processor. This project uses hardware instrumentation with a logic analyzer rather than the development of a software benchmarking tool to measure network performance. The conclusion is that the logic analyzer approach is proven to be effective.

This report first presents the background information, and then the specific test plan, procedure and results. In the beginning, it is explained why I_2O is important to today's computer system. Then a brief overview of the I_2O architecture and the i960 Rx I/O processor is given. Next, the setup of the testbed and the specifics of how the experiment is conducted are presented. Finally, a few test models and suggested test procedures are presented.

Acknowledgement

I would like to take a moment to thank our sponsor, 3 Com, for all their supports. Then I also need to express my sincere gratitude to my advisor, Dr. James Harris, for his guidance, advice, motivation, assistance and patience throughout the project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1. Introduction

1.1 Why add intelligence to the 1/0 interface

One key factor in today's high performance computer network is that the file server must be able to handle all the data flows when many clients access files at the same time. To ensure optimal performance, the processor speed, memory bandwidth, and the I/O throughput must be maximized, and well balanced. Today's high performance server machines use multiple processors to maximize the processing speed. However, the throughput of the 1/0 subsystem is out of balance with the needs of the host processors. Therefore, an intelligent 1/0 interface between the host processor and the network interface card is needed in order to realize the full potential of the fast host processor.

In 1996 the I_20 Special Interest Group (SIG) defined an open architecture for standardizing and managing I/O devices. The goal of the I_20 specification for the I/O device driver architecture is to provide independence at the device driver to the specific hardware device and the specific host operating system software [p5]. By standardizing the device driver architecture, the section of the driver responsible for managing the device is separated from the functions of the host operating system, so the portion of the driver that manages the device becomes portable from one operating system to another. The I_20 also defines a standard architecture for intelligent YO. This approach will transfer the processes of the low-level interrupts from the host CPU to the I/O processor. With the support for message-passing between the I/O Processor (IOP) and host CPU, the I_20 architecture relieves the host of interrupt-intensive I/O tasks, and will greatly improve the I/O performance in high-bandwidth application such as real-time video/audio replay over the Internet, net-meeting, and client/server processing [b4].

1.2 Implementation Of 120 Architecture on 1/0 Processor

To implement the 1_20 architecture, one option is installing an IOP on the motherboard. The IOP along with other peripherals can be used with a standard PCI bridge to bring intelligence to the PCI bus. Another option for implementing the 1_20 architecture is to install a standard plug in PCI adapter card, with an IOP on board. This later approach gives the developer flexibility to interface with different host adapters such as network interface adapters, SCSI/IDE controllers, etc. The system can be populated with one IOP for every type of host adapter plugged into a PCI slot. For example, a server machine can have two intelligent I/O cards plugged into the host PCI bus. One of the I/O cards can handle several network interface cards connected to it, the other one can connect to a SCSI controller that controls several hard drives. The interrupts of all those devices can be handled and managed by the IOPs resulting in a decrease of the processing load of the host CPU [p5].

In order to ensure portability and operating system independence, the I/O adapter card requires not only a fast I/O processor, independent memory, DMA, a compatible bus interface unit, and various other components, but also a sophisticated operating system, which can communicates with the host OS [w3].

1.2 Performance Testing on 1_20 Architecture

Many questions arise from the addition of the IOP to the host CPU and the peripheral devices. Will the IOP affect network throughput? How much is the host CPU utilization changed? Will it add more delay to the data transfer? If so, how much delay does it introduce? To answer these questions, performance testing in throughput, CPU utilization and latency (response time) needs to be conducted. For this particular project, network latency measurement is to be performed to measure the effectiveness of the I_20 architecture for implementing a network driver on an IOP. This report will demonstrate the concept, feasibility and procedure of the performance measurement using a logic analyzer.

Before discussing the measurement test plan and technique, the 1_20 specification and the Intel i960 Rx IOP will be described.

Chapter 2. The 1_20 Architecture and the Intel i960 Rx IOP

2.1 TheI_2OSpecification overview

The I_20 (Intelligent input/output) specification defines a standard architecture for intelligent I/O that is independent of both the specific device being controlled and the host operating system (OS). Developed by the Intelligent I/O Special Interest Group (I_2O SIG); the first version of the specification was delivered in early 1996. It addresses two key problem areas in I/O processing:

1. The performance lost caused by I/O interrupts to the CPU.
2. The necessity to create, test and support unique drivers for every combination of I/O device and OS on the market.

So the I_20 specification defines a "split driver"(See Figure 2. 1) model for creating drivers that are portable across multiple OSs and host platforms. The split 1_2O drivers are composed of two parts [p5]:

1. The Operating System Services Module (OSM), which resides on and interfaces to the host OS.
2. The Hardware Device Module (HDM), which is embedded in the IOP and interfaces with the I/O device such as a Network Interface Card (NIC).

Much like the seven-layer OSI network, these two modules interface with each other through a communication system comprised of two layers:

1. A Message Layer which sets up a communication session.

2. A Transport Layer which defines how information will be shared.

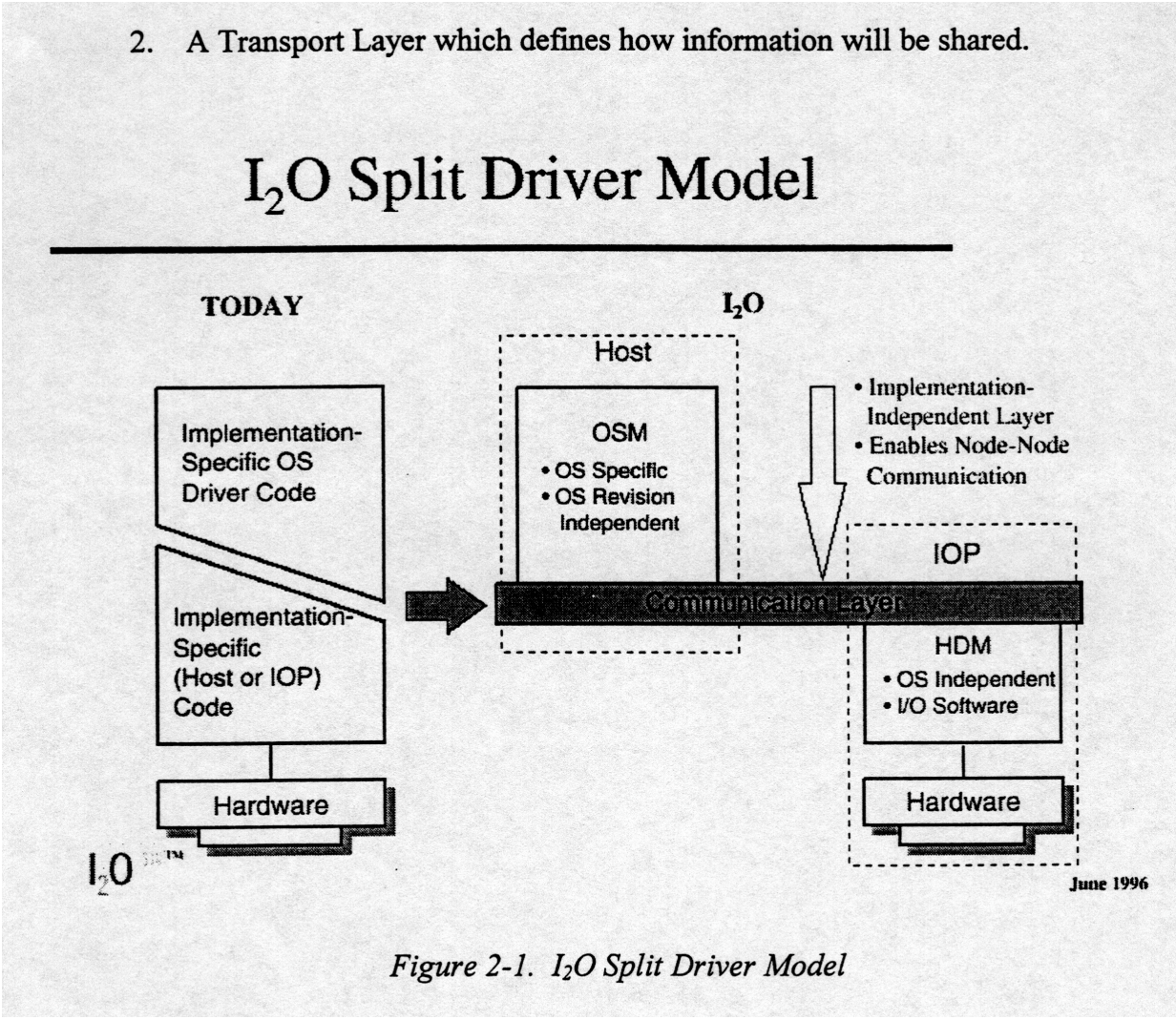

Figure 2-1. I_2O Split Driver Model

The communications system for the I_20 architecture is a message-passing system. It is analogous to a connection-oriented networking protocol, in which two ends exchange messages by using the Message Layer to set up a connection and exchange data and control. When the host OS sends a request to the OSM, the OSM translates the request into an 1_2O message and sends it to the appropriate HDM for processing. Upon completion of the request, the HDM sends the result back to the OSM by sending a message through the 1_2O Message Layer. To the host OS, the OSM appears just like any other device driver (see Figure 2.2).

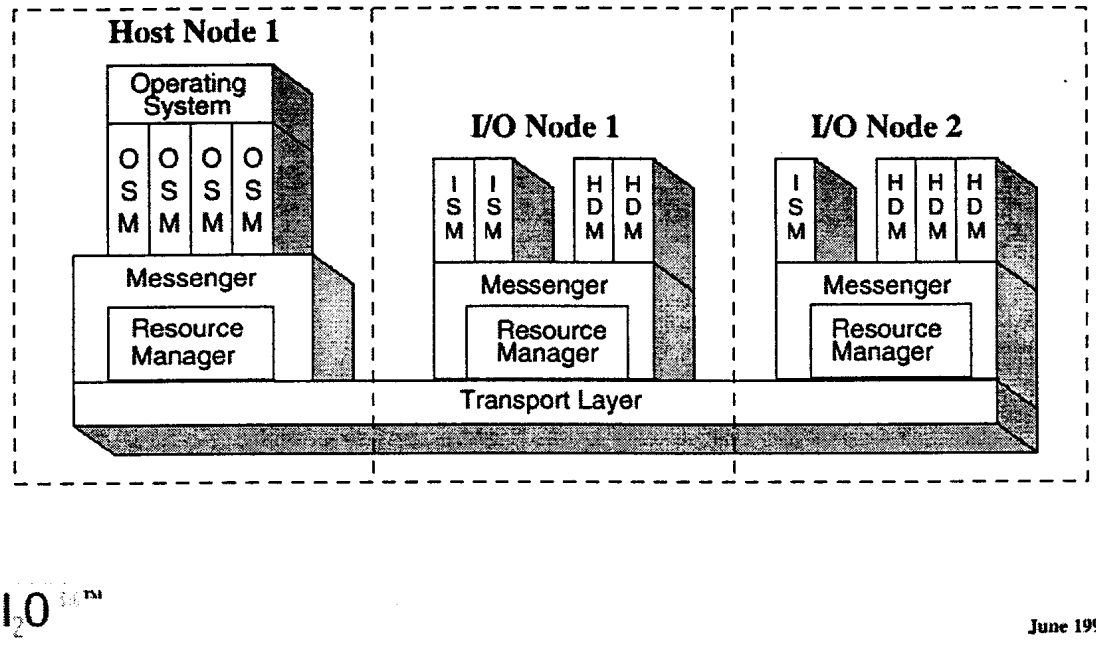

Figure 2-2. I_2O Software Architecture

2.2 The Message Layer

Message Layer defines a standard protocol for communication between the service modules. It acts as the "glue" that connects the framework of the 1_20 driver model, the Message Layer manages and dispatches all requests, and provides a set of Application Programming Interfaces (APIS) for delivering messages, along with a set of support routines that process them. There are three basic components in the Message Layer:

1. The message handle
2. The Message Service Routine (MSR)

3. The message queue

The message handle basically specifies the "address" of the MSR registered in the call. A message handle must be returned for every call to the Message Layer. The message queue provides the link between the requester and the desired service. For instance, when a driver request is made, a message is deposited in a message queue and an MSR is activated to process the request. Much like a network data frame, messages are composed of two parts, a header and a payload, where the header describes the type of request along with the return address of the caller.

2.3 The Operating System Servicing Module (OSM)

The OSM provides the interface between the host OS and the Message Layer. In the split driver module, the OSM represents the portion of the driver that interfaces to the host APIS, translating them to a message-based format that is then sent to the HDM for processing. Then, the HDM information is forwarded back to the host OS through the OSM via the I_20 Message Layer. Developers can also create host OSMs that work with multiple HDMs (see Figure 2.3). By implementing an OSM with a single message handle which services multiple queues from different service modules, a single OSM can send and service requests to and from multiple different devices.

The interface and operating environment for an OSM is host OS specific. Typically, the OS vender supplies this module, which contains no hardware-specific code.

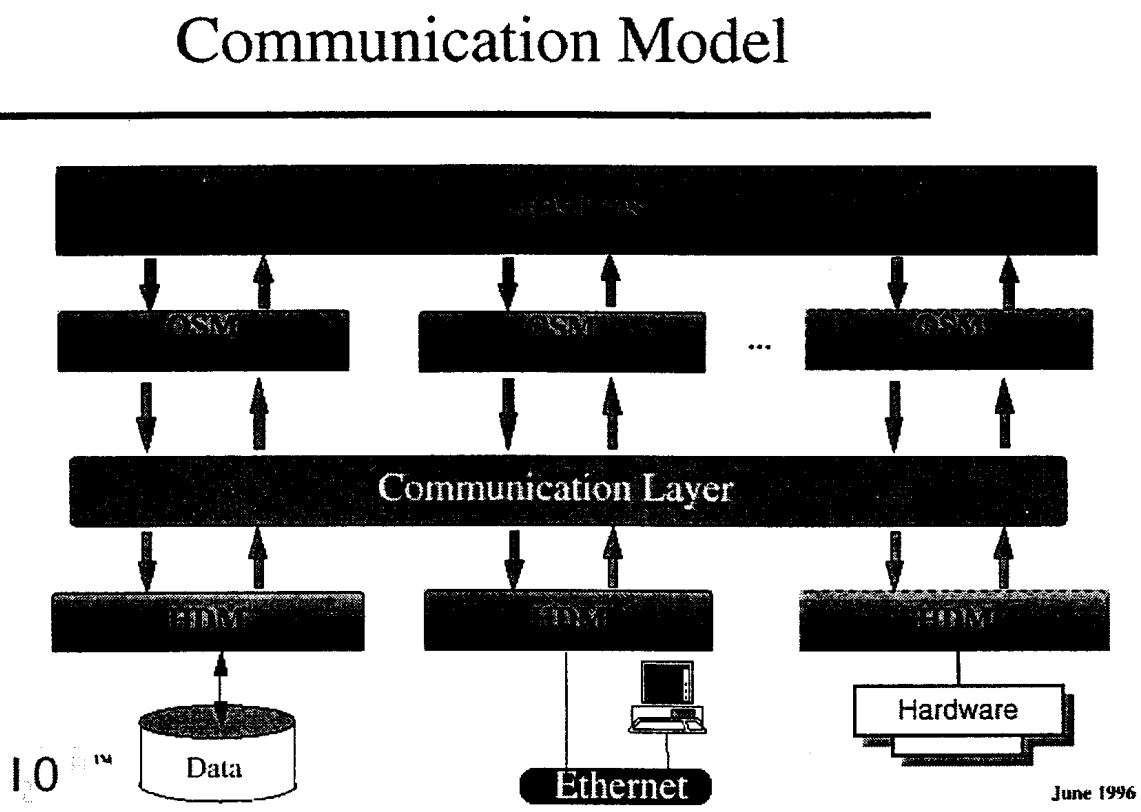

Figure 2-3. Communication Model

2.4 The Hardware Device Module (HDM)

The HDM is the lowest-level module in the I_2O environment. HDMs provide the device-specific portion of the driver that will interface with the particular controller and devices. HDMs can be roughly analogous to the hardware-specific portion of the network and SCSI drivers that exist today. But the HDM translation layer is unique to each individual hardware device and vendor. It supports a range of operation types, including synchronous and asynchronous requests, and interrupt-driven and polled transactions. The HDM is surrounded by the I_2O execution environment (see Figure 2.4), which provides the necessary support for the OS processes, and bus-independent

Execution.

HDMs are typically written in C or C++ and can be structured in a manner that minimizes changes when moving from one hardware platform to another. HDMs are implemented by the device vendor and reside on the IOP. The hardware vendor supplies this module, which contains no OS-specific code.

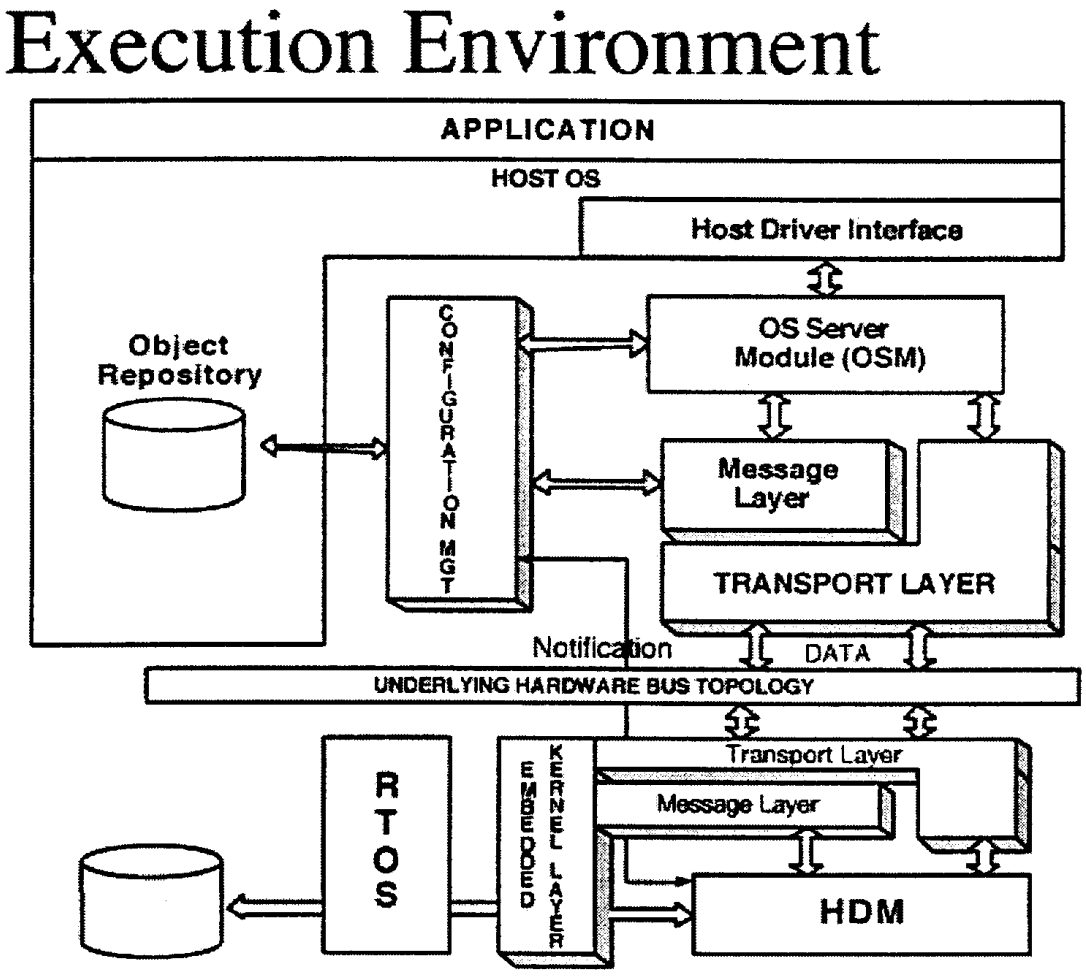

Figure 2-4. Execution Environment

2.5 System Execution Environment

The execution environment for the IOP has additional facilities for loading, initializing, and managing HDMS. In general, these services are provided by

WindRiver System's IxWork Real Time Operating System (RTOS) [p12] (see Figure 2.4).

2.6 Introduction to Intel's I960Rx IOP

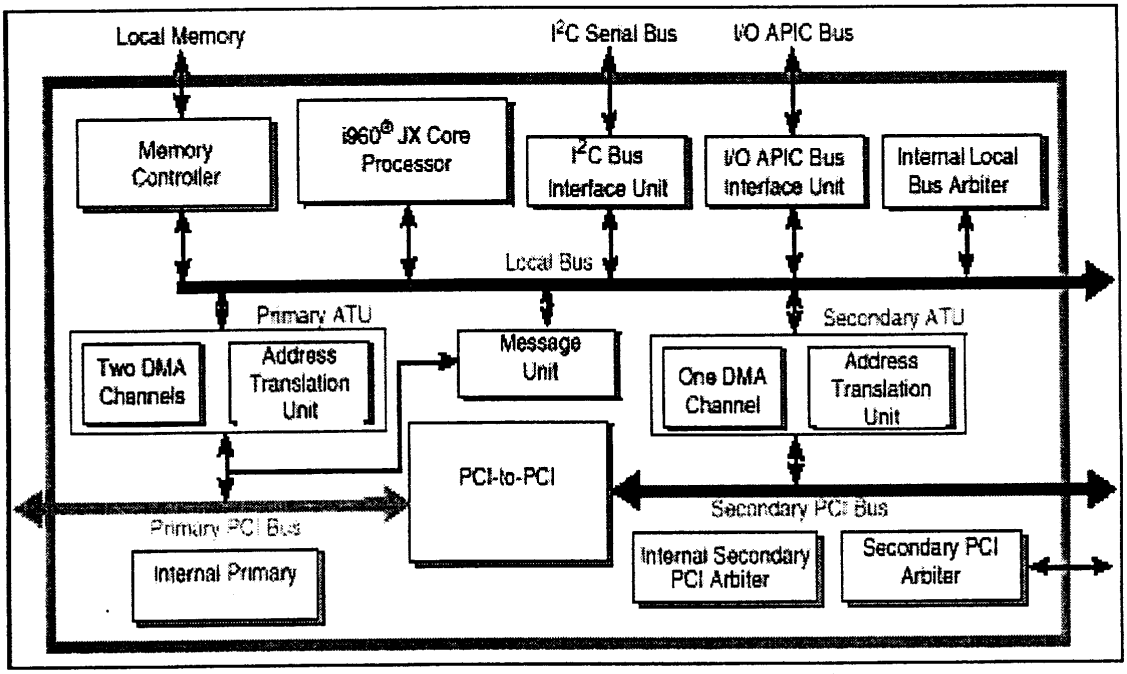

Figure 2-5. I960Rx Processor Block Diagram

Intel's family of I/O processors, the 8096ORx, is built upon the already established performance of Intel's 80960 RISC architecture. It encompass industry initiatives, including PCI and 1_2O technology, which effectively offload host CPUs of I/O processing chores. This highly integrated processor offers a cost-effective approach to implement a PCI-based intelligent I/O subsystem [p I]. It serves as the main component of the IQSDK platform, which allows the developer to connect PCI devices to the i960 Rx processor. The architecture of the i96ORx is shown above in Figure 2.5. As indicated in the figure, the 8096ORx combines many features with the 8096OJF

CPU core to create an intelligent IOP. The PCI-to-PCI bridge unit is the connection path between two independent 32-bit PCI buses, and provides the ability to overcome PCI electrical load limits. The 8096ORx is object code compatible with the i960 core processor. It is capable of executing at the rate of one instruction per clock cycle. The bridge and Address Translation Unit work together to hide the secondary PCI device from the primary PCI configuration software. The Address Translation Unit (ATU) allows direct transfer between the PCI system and the 8096ORx local memory. The Messaging Unit (MU) provides data transfer between the primary PCI system and the 8096ORx. The local bus is a 32-bit multiplexed burst bus, which is a high-speed interface to system memory and I/O. The Secondary PCI Arbitration Unit provides PCI arbitration for the secondary PCI bus. It can also be disabled to allow for external arbitration. The I^2C (InterIntegrated Circuit) Bus Interface Unit allows the 80960 core residing on the I^2C bus to serve as a master and slave device. The I^2C bus is a serial bus developed by Philips Semiconductor consisting of a two-pin interface. The bus allows the 8096ORx to interface to other I^2C component [p2].

2.7 The IQ-SDK

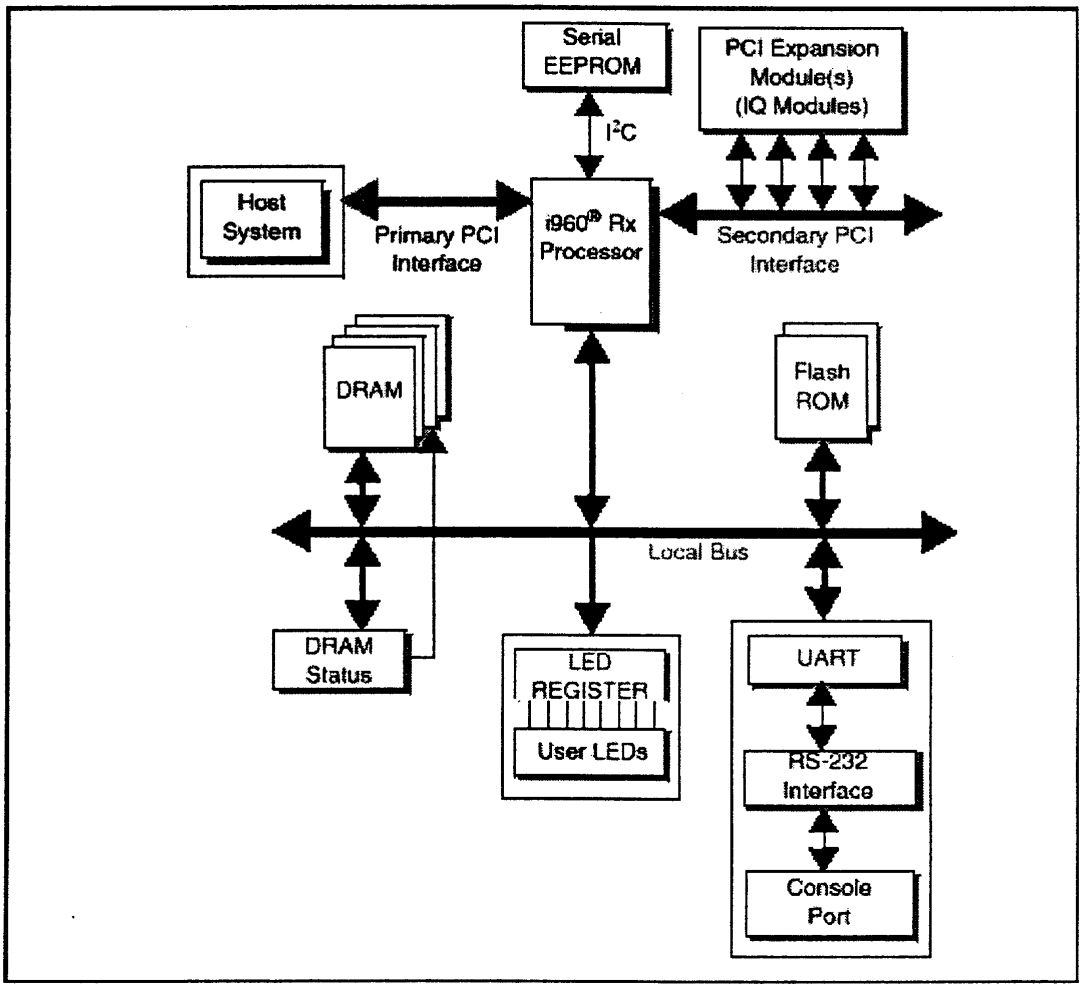

Figure 2-6. IQ-SDK Platform Functional Block Diagram

The IQ-SDK platform (see Figure 2.6) is equipped with 4 Mbytes of DRAM. An additional 4 to 32 Mbytes of DRAM can be added to the SIMM slot to maximize the memory to 36 Mbytes [p2]. The DRAM is accessible from either of the PCI buses on the IQ-SDK platform, and can be used to implement shared communications areas for I_2O's HDM installed on the IQ-SDK platform. The console serial port on the IQ-SDK platform that is based on a 16C550 UART can operate from 300 to 115,200 BPS. The

port is connected to a phone jack-style plug. The DB25 to RJ-45 adapter and cable included with the IQ-SDK can be used to connect the console port to any standard RS232 port on the host system. A 1 Kbytes serial EEPROM is connected to the I^2C bus on the IQ-SDK platform. Intel does not define the contents of this device, so it is available for use by the developer.

In conclusion, from the discussions above, we can see that the Intel 196ORx IOP is the single-chip solution for the I_2O specification. The combination of this IOP and 1_2O architecture provide the framework to allow for significantly enhanced I/O and system performance.

Chapter 3. Cal Poly 3Com Network Testbed Configuration

3.1 3Com Local Area Network (LAN) Testbed

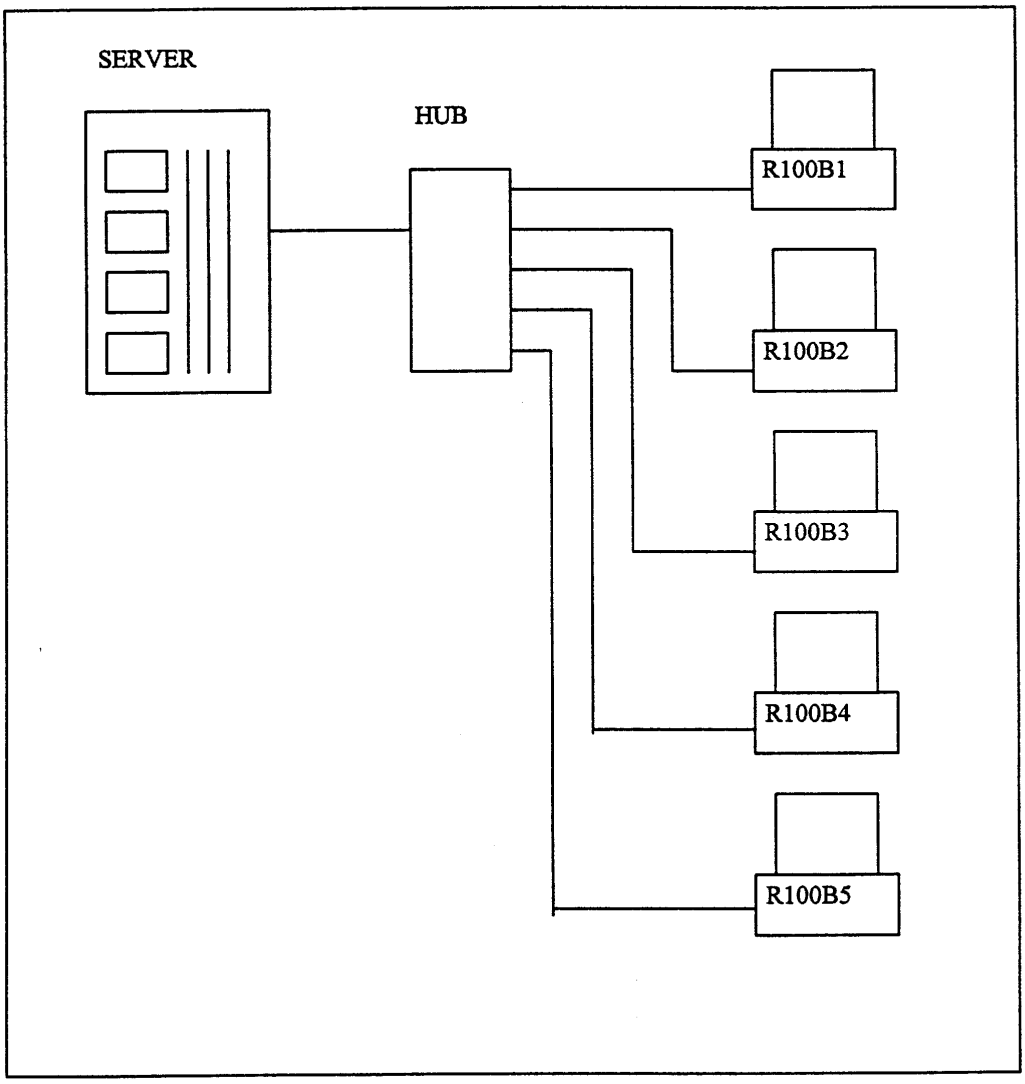

Figure 3-1. 3Com LAN Layout Overview

This Testbed is a one-server and five workstations LAN (see Figure 3. 1). The LAN is either a 10 or 100 Mbps Ethernet [b5]. Twisted pair cable is used to connect all the machines. The server, Outlander, is running Windows NT 4.0 server. Four of the

five workstations are the BP Vectra Pentium-pro 200 MNX with 32 Megabyte RAM; and the RI 0OB5 is a NEC Pentium 166 with 32 Megabyte RAM. All of them are running under Windows NT 4.0 workstation.

3.1.1 Server Configuration

Each of the machines above is uniquely identified with an IP address and an Ethernet address. Table 3.1 shows the IP address and Ethernet address of each machine.

Table 3-1. IP Addresses and Ethernet Addresses on 3Com LAN

NAME	IP Address	Ethernet Address
Outlander	129.65.26.67	00:60:97:2d:b1:a7
R100B1	129.65.26.68	00:60:97:2d:b1:d8
R100B2	129.65.26.69	00:60:97:2d:b1:d9
R100B3	129.65.26.70	00:60:97:2d:b1:e0
R100B4	129.65.26.71	00:60:97:2d:b1:e1
R100B5	129.65.26.72	00:60:97:2d:b1:e2

The server hardware description is given below:

- HP Netserver LH Pro
- Dual Intel X86 family 6 Model I stepping 7 Processors
- AT/AT compatible
- 3Com fast Ethernet 3C905 network adapter (MAC: 00:60:97:2d:bl:a7)
- HP 4.26GB A 80 LXPO Hard Drive
- Microsoft VGA compatible display adapter

3.1.2 Workstation R100Bl- R100B4 Hardware Description

- HP Vectra 200 Pentium-Pro MMX
- 32 Megabyte RAM
- 3Com 3C905 Fast Ethernet XL network adapter
- Matrox Millenium 2MB video adapter
- Hitachi CDR-7930 CD-ROM
- Quantum Fireball TM2 2.4 Gb hard drive

3.1.3 Workstation R10OB5 Hardware Description

- NEC 166 MHz Pentium
- 32 Megabyte RAM
- 3Com 3C905 Fast Ethernet XL network adapter
- Intel IQ-SDK PCI Adapter card with i960 RP IOP on-board
- NEC 8x CD-ROM
- Western Digital 2.1 Gb hard drive

3.1.4 Special Hardware Available

- Altera fixtures and cables
- Microchip fixtures and cables
- HP 16500B Logic Analyzer

- HP Network Adviser
- Iomega zip driver
- FuturePlus FS2000 PCI extender card
- FuturePlus FS2000 PCI Analysis Probe installed in BP logic analyzer

3.2 **Software Configuration**

All software tools installed on the LAN are professional-version development kits, which are accessible by the server or the workstations. The following is a list of all development software tools available:

1. Microsoft Visual Studio 97 Professional Edition
2. Microsoft Office 97 Professional Edition
3. Windows NT Device Development Kit (DDK)
4. Windows NT Software Development Kit (SDK)
5. Wind River's Tornado 1.01 Integrated Development Environment (IDE)
6. Wind River IxWork Real Time Operating System (RTOS) for i960
7. Novell Netware OSM
8. Altera Max Plus II 8.0
9. Performance tools: NetPerf, Perform3, NetBench

Chapter 4. Network Latency Measurement Technique

For this project, network latency is defined as the processing time between a network application request and when the first data frame appears on the PCI bus; in other words, how long it takes for the data to propagate through various network protocol layers [b5]. However, this performance figure is largely dependent on the operating environment of the system such as threads, processes, disks, free memory, network protocols and network traffic conditions. Fortunately, we only need relative measurements between the non-I_2O system configurations and the I_2O system configurations. In addition, all of the above objects are controllable on the user level. In the next three sections, the OSI seven network protocol layers model will be described, and the limitations in conducting performance measurement in Windows NT will be explained. Finally, the measurement technique using the logic analyzer with triggering initiated with a parallel port signal will be presented.

4.1 The OSI Reference Model

The OSI (Open Systems Interconnection) model is shown in Figure 4.1 [b5]. This model was developed by the International Standards Organization to standardize the protocols used in the various layers. The description of the OSI layers is as follows:

1. The physical layer is concerned with transmitting raw bits as electromagnetic energy.
2. The data link layer is responsible for breaking the data into frames, and makes sure the frames are transmitted or received without any error.

3. The network layer is concerned with how a packet of data is routed to the destination. The EP protocol applies to this layer.
4. The basic function of the transport layer is to split data into packets. The TCP protocol applies to this layer.
5. The session layer allows multiple users to establish each of their own sessions from a remote terminal.
6. The presentation layer provides APIs for the various different application programs in the application layer.
7. The application layer contains a variety of network application programs such as FTP, Telnet, HTTP, etc.

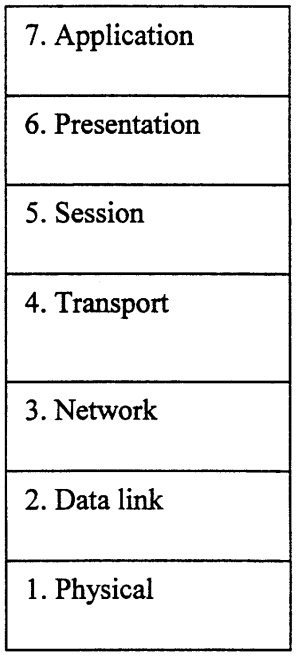

Figure 4-1. The OSI Reference Model

When a machine transmits data out to the network, the data is sent as data packets. As the packet travels from the application layer to the physical layer, it is encapsulated with control information specific to each layers, namely header and trailer. On the other hand, when a machine receives data packet, the headers and trailers will be stripped as the data packet travels upward to the application layer.

4.2 Limitations in Conducting Measurements in NT

To measure the network latency, we must timestamp the application request event, and the data packet delivered event [b3]. Which means we need to capture every keyboard and mouse interrupt, and the data packet as it goes across the PCI bus. In addition, it will be nice if we can set "milestones" as the data packet propagates through the various protocol layers in NT. However, associating a keyboard or mouse input to a high-level event for measurement purposes, e.g. starting a timer, is difficult [p6]. In addition, we also need to associate a packet-delivered event to another high-level event, e.g. stopping the timer. Doing so requires instrumentation of both the entry point and the exit point of the path of execution as an application request is processed. This can't be done without the NT source code. Windows NT has a performance monitor under administrative tools, which performs statistical measurements on objects like processes, processor usage, memory usage, interrupt occurrence, event log, etc. However, the NT performance monitor does not reveal measurements of data paths being executed inside of the operating system.

Often times, extensive latency in a computer system is caused by deadlock of a critical thread. In order to find out where a thread spends all its time, we need a detailed description about the inner workings of the operating system. Kernel profiling can provide such information because a thread executes inside the kernel. And again, kernel profiling requires having the source code of the NT operating system.

Due to the lack of NT source code and problems with its modification [p9], we have chosen to use a hardware approach to conduct measurements, and this is described in detail in the next sections.

4.3 Instrumentation with Logic Analyzer

The idea is to use a FuturePlus PCI extender card to port the data out from the PCI bus. Then, we use a BPI6500B Logic Analyzer to capture the data. The logic analyzer will be triggered by a parallel port signal, which is initiated by the application request on the host computer (see Figure 4.2). After the capture, we can then calculate the relative time between different events. The following table is an example of the data capture from the Logic Analyzer:

Table 4-1. Data Capture from the HP16500B Logic Analyzer

PCI_BUS1 - State Listing

Label	> DATA	Time	GNT	DEVSEL	IRDY	TRDY	FRAME	PCICLK
	> Hex	Absolute	Bin	Binary	Bina	Bina	Binar	Binary
7	00000000	355.4 ms	1	0	0	0	0	0
8	5555AAAA	358.0 ms	1	0	0	0	0	0
9	00000083	358.0 ms	1	0	0	0	0	0
10	5555AAAA	358.0 ms	1	0	0	0	0	0
11	00000083	358.0 ms	1	0	0	0	0	0
12	5555AAAA	358.0 ms	1	0	0	0	0	0
13	5555AAAA	358.0 ms	1	0	0	0	0	0
14	000003CD	360.0 ms	1	0	0	0	0	0
15	000003D4	360.0 ms	1	0	0	0	0	0
16	000003DD	360.0 ms	1	0	0	0	0	0
17	000003E7	360.0 ms	1	0	0	0	0	0
18	00000200	361.2 ms	1	0	0	0	0	0
19	00000000	362.3 ms	1	0	0	0	0	0
20	5555AAAA	362.3 ms	1	0	0	0	0	0
21	00000000	370.0 ms	1	0	0	0	0	0

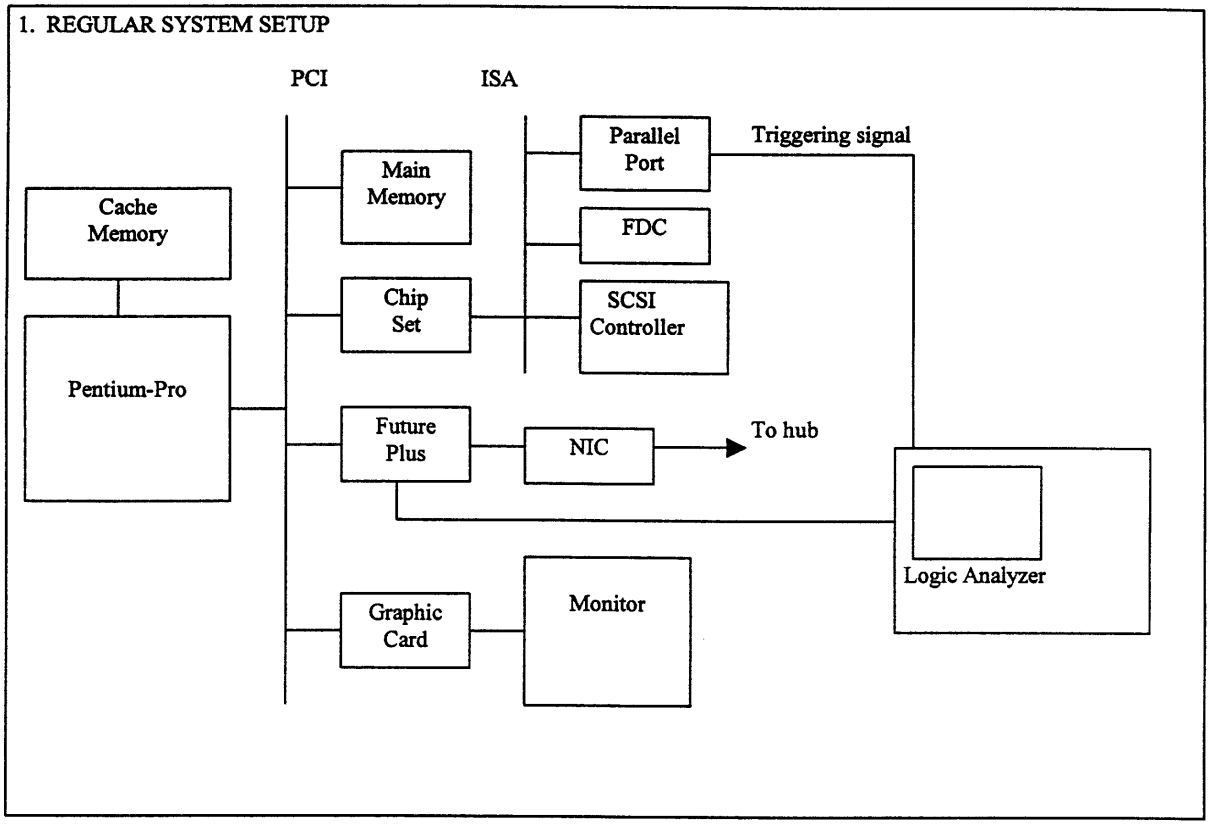

Figure 4-2. Physical Layout of the R100B1 Testbed

4.3.1 Captured Data Analysis and Presentation of Results

The captured data is in hexadecimal form (see Table 4. 1). Each line is 32 bits because the PCI bus uses 32 bit words and transmits one word for each bus cycle. The time field shows either the relative time between two words, or absolute time with respect to the last trigger. The other fields show the control and data signal status of the PCI bus. By using the data capture function of the Logic Analyzer (LA), we can send a

triggering signal to the LA while initiating a File Transfer Protocol (FTP) command for a Local Area Network (LAN). Then the absolute time for the propagation time for the first packet reaching the PCI bus can be used for the measurement of the process latency. Finally, by comparing the measurement results between the I_2O/IOP system and the regular system, we can find out the effectiveness of the 1_2O architecture implemented on an IOP.

4.3.2 Parallel Port Interface with NT

The function of the parallel port is to provide a triggering signal to the LA. However, the parallel port is driven by parallel port driver, which is called by the I/O manager m the NT kernel. To access the parallel port driver, an application programming interface (API), which will guide the use's requests to the driver, must be created. The NT Device Driver Kit (DDK) provides many general I/O port drivers, and they can be applied to the parallel port driver for our measurement purpose. Microsoft also provides a Software Development Kit (SDK), which can be used to create the API. The details in how to create the driver and the API are in Angel Yu's senior project report [p6].

4.3.3 PCI Instrumentation Using FuturePlus Extender Card and HP16500B LA

4.3.3.1 FuturePlus Extender Card Functional Overview

This tool is provided by FuturePlus System. It comes with the FS2000 PCI Analysis Probe and the Extender Card. The Extender Card acts as an extender for a

PCI adapter card beyond the host motherboard front panel. It provides a passive electrical and mechanical interface to Hewlett-Packard logic analyzers for PCI bus analysis (see Figure 4.3).

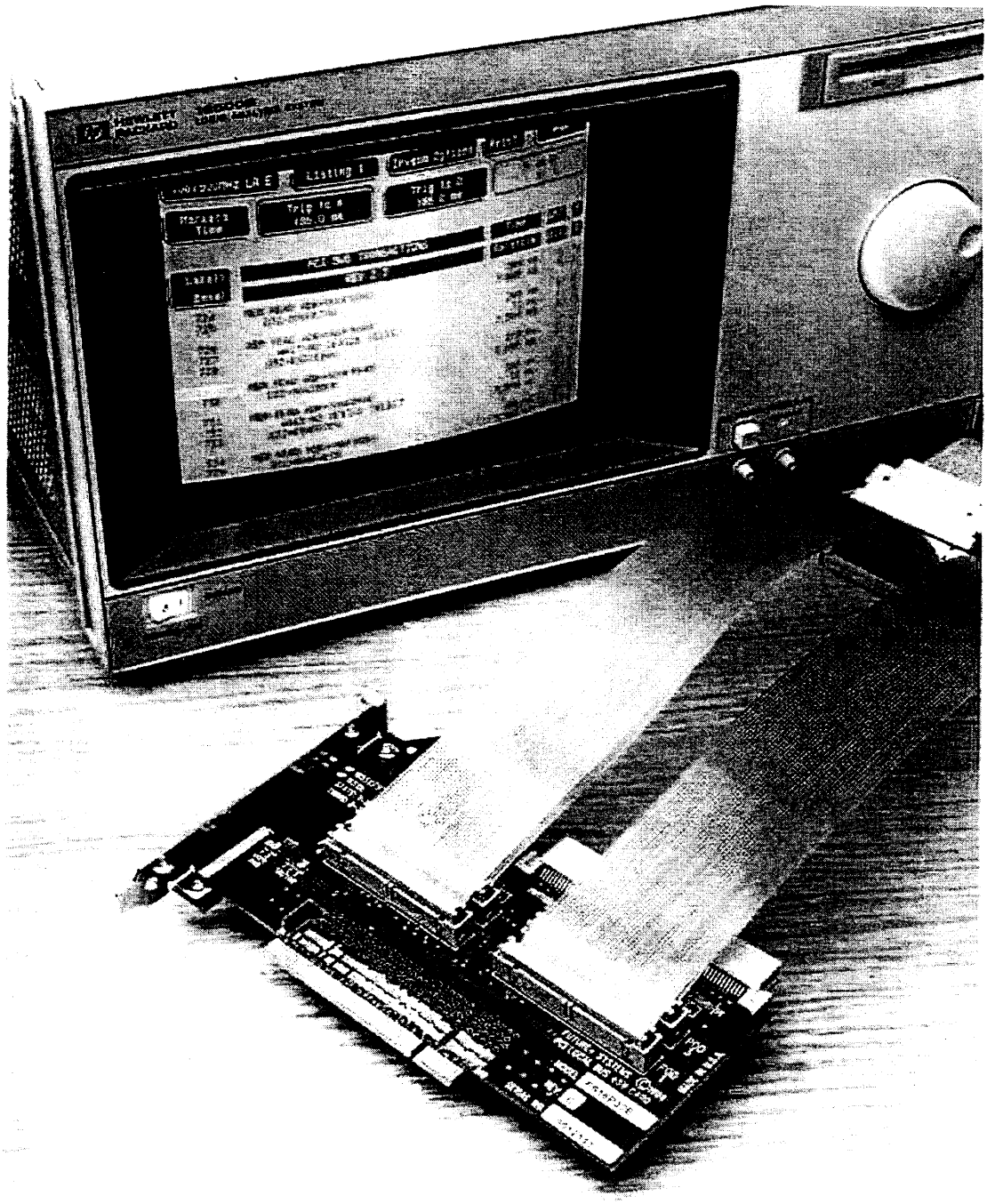

Figure 4-3. FS2000 Extenter Card Interface to HP16500B Logic Analyzer

The FS2000 PCI Analysis Probe is an inverse-assembly software, which executes in the BP logic analyzer. It decodes the key PCI bus signals and presents a readable display that lists the transaction type, address, data and key status conditions such as wait states and retries. This software also provides features such as user-defined symbols that can be easily added to the state listing display, user-select post-processing filters that allow the acquired data to display only chosen transactions. To assist with triggering, pre-defined resource terms have been included that can be used to prevent WAIT and IDLE states from being acquired, thus saving trace memory and providing an easier to read state display. The package also provides a PCI bus triggering application note to all its PCI Analysis Probe customers [w4].

4.3.3.2 Key Features of the HP16500B Logic Analyzer

Referring to the BP manual, some key features that are critical to this project are summarized below [p7]:

- Touchscreen control or mouse control
- Graphical user interface
- 3.5-inch flexible disk drive with DOS and LIF format support
- 85 Mbytes hard disk drive with DOS format support
- 100-@ state and 500- time acquisition speed
- 12 levels of trigger sequencing for state and 10 levels of sequential triggering for timing
- 4 Kbytes deep memory on all channels with 8 Kbytes in half channel mode

- RS-232C interface for hardcopy output to a printer
- Expandable system memory up to 64 Mbytes

4.3.3.3 Pod Connections

The HP16500B has 6 pods. Only 4 of 6 pods (pod 1 - 4) are used by FS2000 extender card (see Figure 4.3). The remaining 2 pods can be used for additional test input. Some of the pod 1 - 4 signal connections are summarized in Table 4.2 below. Note that all variable names are assigned by FS2000 inverse-assembly software; refer to FuturePlus User Manual [p8] and PCI Specification for more detailed description [p 10].

Table 4-2. HP16500B Pod 1-4 Connections (selected signal only)

Variable Name	PCI Function	Pod	Pin
ADDR	Add/data bus	1 and 2	All 16 pins
DATA	Add/data bus	1 and 2	All 16 pins
C/B3_0	Command byte enable	3	6 - 9
DEVSEL	Device select	3	5
IRDY	Insertion ready	4	7
TRDY	Target ready	4	9
FRAME	Frame	4	8
GNT	Grant	4	3

Another input signal called "angel" is added to the LA for triggering signal from the parallel port. It will connect the output pin of the parallel port to pin 1 pod 6 on the LA.

4.3.4 Limitation of the Logic Analyzer's memory

As indicated in Section 4.3.3.2, the HP16500B has only 4Kbytes of memory available. In the timing mode, the LA must sample data at a rate of 33 because the PCI bus is running at 33MHz. In other words, the LA takes in at least 4 bytes of data every 30us. At this rate, the buffer space will get fill up 30us later, which is far too short for the network latency measurement. Therefore, only the state mode can be used in our project due to the insufficient memory.

4.3.5 Logic Analyzer Trigger and Data Capture in State Mode

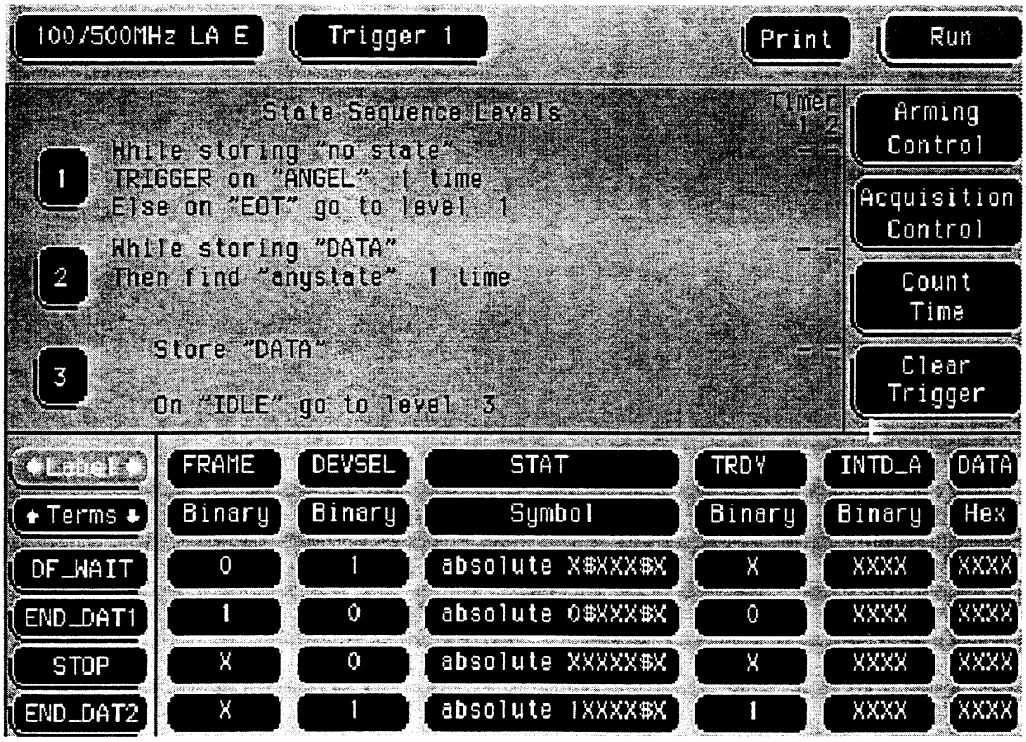

Figure 4-4. Example Triggering Sequence

The HP 16500B has 12 levels of trigger sequences in the state mode [p7]. As indicated in Figure 4.4, at each trigger level, the user can program the LA much like programming a Finite State Machine. For example, at a particular level, if the triggering condition is meet, the LA will go to next level; else if another condition is meet, the LA will jump to wherever the user specifies; else, it will wait at the current level until any of the above conditions are meet. Users can also program the LA to store any incoming signal on the fly. The triggering conditions are also programmable. For example (reference to Figure 4.4), END-DATA 1 is just a dummy variable that represents the state when F is high, DEVSEL is low, TRDY is low and don't cares are specified for all other inputs.

Chapter 5. Prototype Measurement

5.1 Prototype Measurement Procedures

The objective of this experiment is to demonstrate the concept and feasibility of the performance measurement of network latency using the logic analyzer. Manual triggering from a switch which has been debounced (Figure 5. 1) provides a LA triggering signal for this measurement. The trigger is set up such that when "angel" is asserted high, the triggering level will jump to the 2nd level and start to buffer the data, where data is the data inputs on the 32-bit addr/data line; else, the trigger will wait on level 1.

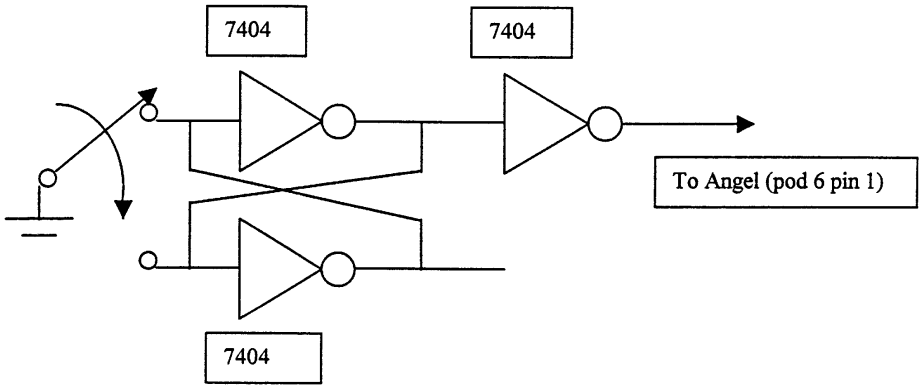

Figure 5-1. Schematic of the Switch Debouncer

The experiment procedure is as follows:

1. Build the debouncer circuit
2. Connect the "angel" to pin l pod 6 of the HPI6500B LA

3. Plug the FuturePlus extender card to a primary PCI slot. Then plug the 3Com NIC to the extender card

4. Connect pod I - 4 to the FuturePlus extender card

5. Turn on the LA and load the saved FuturePlus configuration

6. Set up the trigger such that when "angel" is high, jump to 2nd level and start to buffer the data; else wait on level 1.

7. Connect the BP network adviser (PA) to the 3Com hub

8. Set up the HP network adviser (PA) to decode network stack on 10-base-T mode

9. Generate a FTP socket on R100B1 to Outlander using Winftp

I0. Click "RUN" at the upper right comer on the LA display (see Figure 4.4)

11. Click "run measurement" on the PA window

12. At the same time, hit the switch and initiate a test file write

13. Print both the LA and PA data in ASCII format onto a floppy disk

14. Examine the result using WordPad

The HP network adviser is a network protocol analyzer [p 11]. In this project, it is used for decoding the network frame header.

5.2 Prototype Measurement Data Analysis

Table 5-1. A Sample of Data Capture (datafield only)

PCI_BUS1 -State Listing

Label	> DATA	
Base	> Hex	
7	00000000	Irrelevant data
8	5555AAAA	Irrelevant data
9	00000083	Irrelevant data

10	5555AAAA	Irrelevant data
11	00000083	Irrelevant data
12	5555AAAA	Irrelevant data
13	5555AAAA	Irrelevant data
14	000003CD	Irrelevant data
15	000003D4	Irrelevant data
16	000003DD	Irrelevant data
17	000003E7	Irrelevant data
18	00000200	Irrelevant data
19	00000000	Irrelevant data
20	5555AAAA	Irrelevant data
21	00000000	Irrelevant data
22	0055El94	Irrelevant data
23	FFFFFFFF	lst R10OB1 ARP Broadcast
24	600OFFFF	00-60-97-2D-Bl-D8
25	D8Bl2D97	
26	01000608	(08-06 ARP type)(00-01 Ethernet type)
27	04060008	(08-00 IP Protocol)(06 HWADD length)(04 IP addr length)
28	60000100	(00-01 ARP request)(00-60-97-2D-B1-D8)
29	D8Bl2D97	
30	44lA4181	129.65.26.68
31	00000000	(00-00-00-00-00-00 unknown target HW addr)
32	41810000	(81-41-lA-43=129.65.26.67 1A-43 is not shown??)
33	00000000	data padding
34	0056A608	Irrelevant data
35	0056AOOO	Irrelevant data
36	2D976000	server response to ARP Broadcast
37	600OD8B1	(target HW addr 00-60-97-2D-B1-D8)
38	A7B12D97	(sender HW addr 00-60-97-2D-B1-A7)
39	01000608	(08-06-ARP type)(00-01 Ethernet type)
40	04060008	(08-00-IP Protocol)(06 HWADD length)(04 IP addr length)
41	60000200	(00-02-ARP Reply)(00-60-97-2D-B1-A7)
42	A7Bl2D97	
43	43lA4181	129.65.26.67
44	2D976000	(00-60-97-2D-B1-D8 target HW addr)
45	4181D8B1	(81-41-1A-44=129.65.26.68)
46	441A441A	(1A-44's are the data padding)
47	44lA441A	
48	44lA441A	
49	44lA441A	
50	0056A608	Irrelevant data
51	00000000	Irrelevant data
52	0055F4F8	Irrelevant data
53	2D976000	(target HW addr 00-60-97-2D-Bl-A7)
54	600OA7Bl	
55	D8Bl2D97	(sender HW addr 00-60-97-2D-Bl-D8)
56	00576D60	(Unrecognizable???)
57	41000045	(start of IP header 5 lines)
58	0040135B	IP
59	9A680680	IP
60	44lA4181	IP
61	43lA4181	IP
62	1500DE04	(start of TCP header 5 lines)
63	AB83F408	TCP
64	4FO55DF7	TCP
65	2ClFl850	TCP
66	0052FCAA	last line of TCP missing 39-C9-00-00, AAFC520OF06E ???
67	4F506EFO	(from 504F to line 72: "ASCII PORT
68	31205452	129,65,26,68,4,234")
69	362C393	

70	36322C35	
71	2C38362C	
72	33322C34	
73	00000000	Irrelevant data

Table 5.1 shows the first few frames of the data captured from the PCI bus. The underlined comments are added after the data has been captured and saved. The time field and all the other control information are omitted since they are not important to this demonstration. The following is the explanation for the comments:

- ARP - Address Resolution Protocol
- HW addr - Ethernet address
- HW addr length - the number of bytes in the Ethernet address
- Types - protocol type
- Data pad - repeat the last two bytes of data so that the length of each frame is at least 46 bytes
- ARP Request - the frame is a ARP request frame
- ARP Reply - the frame is a ARP reply frame
- IP protocol - the data transfer uses IP protocol
- IP - IP header
- TCP - TCP header
- "…" -Inside the quotation mark, it is the translation of the ASCII data encapsulated in the header and trailer.

This demonstration clearly shows that the PCI instrumentation using the logic analyzer gives a solid understanding of the computer I/O process. With the parallel port signal used for triggering, a relatively precise response time measurement can be achieved.

Chapter 6. Network Latency Measurement Test Plans

There are four different measurement configurations in this project. They are:

1. TCP/IP, NDIS 3Com driver, PCI
2. Microsoft NetBEUI, NDIS 3Com driver, PCI
3. IPX/SPX, NetWare OSM, PCI, ported 3Com driver on IQ-SDK
4. IPX/SPX, NetWare OSM, Primary PCI, 3Com driver on IQ-SDK, secondary PCI

6.1 TCP/IP, NDIS 3Com driver, PCI

The first test configuration is a normal NT workstation configuration under TCP/IP protocol. The layering configuration is shown in Figure 6.1.

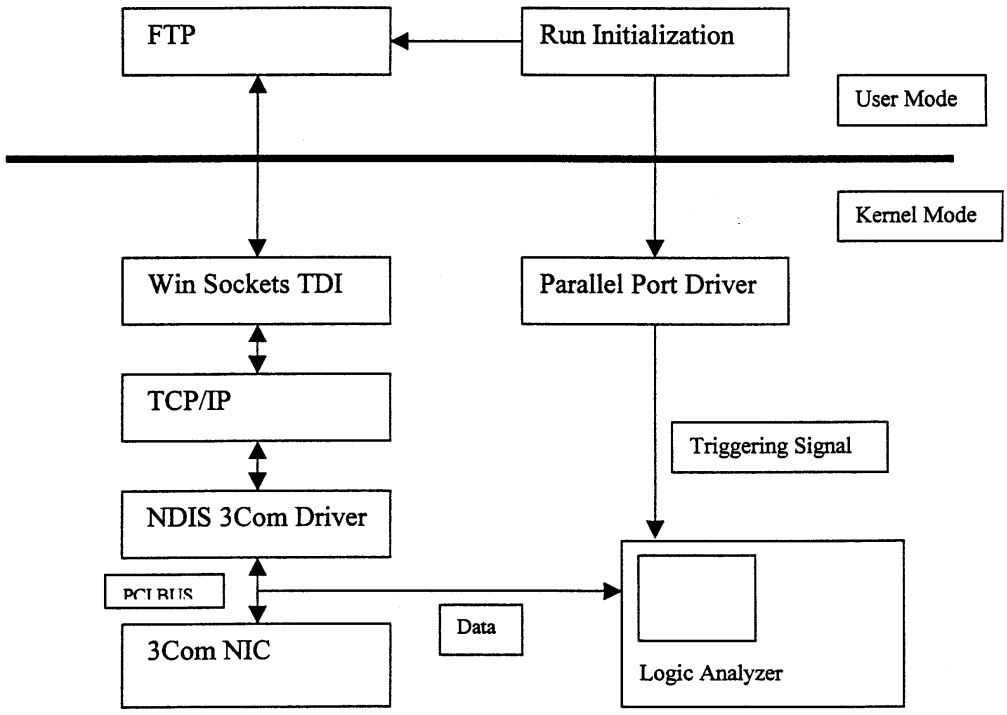

Figure 6-1. Test Model with TCP/IP, NDIS 3Com Driver

The purpose of this test model is to provide a set of reference performance values for the 120 measurements. The actual data flow in the NT is somewhat different. For more information, please refer to *Inside Windows NT* [b3].

6.2 Microsoft NetBEUI, NDIS 3Com driver, PCI

The second test configuration is another normal NT workstation configuration under NetBEUI protocol. The layering configuration is shown in Figure 6.2.

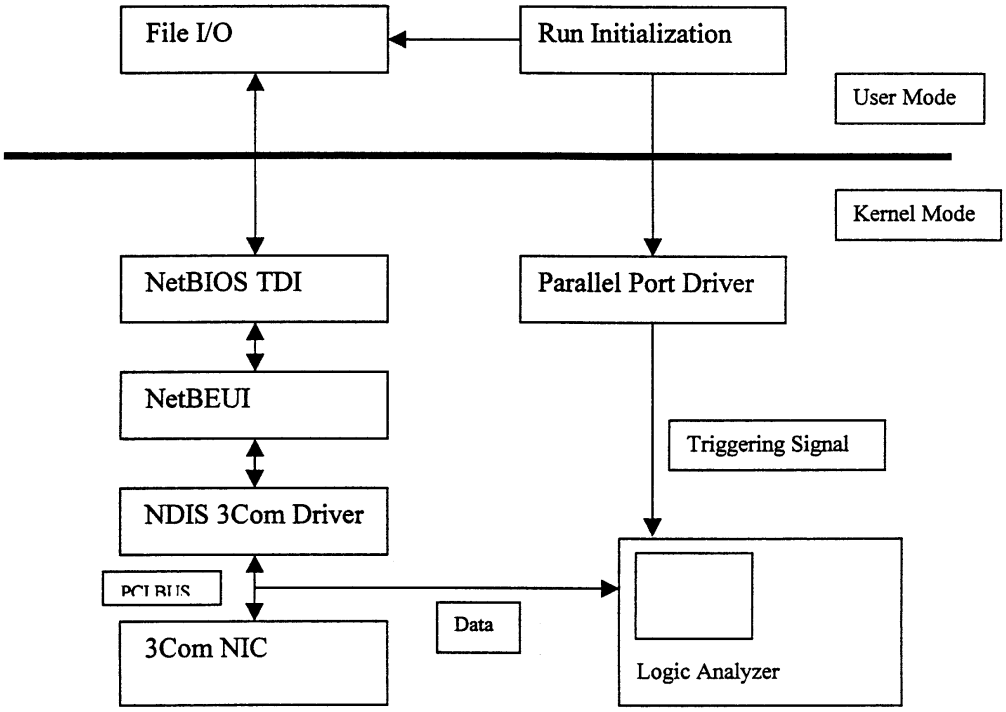

Figure 6-2. Test Model with NetBEUI, NDIS 3Com Driver

The purpose of this test model is to provide comparison between TCP/IP protocol and MS NetBEUI LAN protocol.

6.3 IPX/SPX, NetWare OSM, PCI, ported 3Com driver on IQ-SDK

The third test configuration is an I_2O LAN class configuration with the IPX/SPX protocol stack. The layering configuration is shown in Figure 6.3.

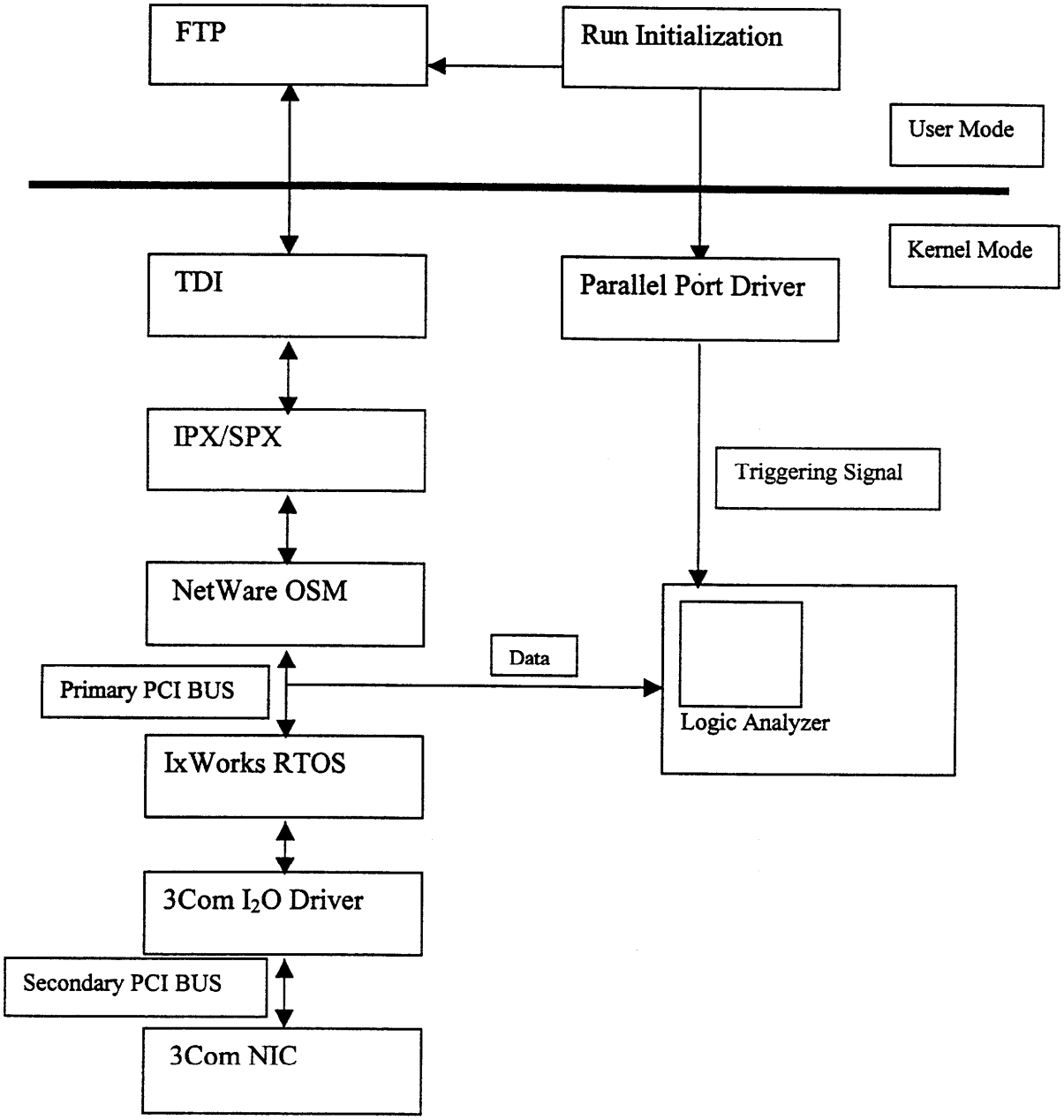

Figure 6-3. I_2O Test Model - Primary Bus

The measurement of this test model takes place on the primary PCI bus. The results can be compared against the first test model results.

6.4 IPX/SPX, NetWare OSM, PCI, 3Com driver on IQ-SDK, secondary PCI

The fourth test configuration is same as the third one except the measurement is taken on the secondary PCI. The layering configuration is shown in figure 6.4.

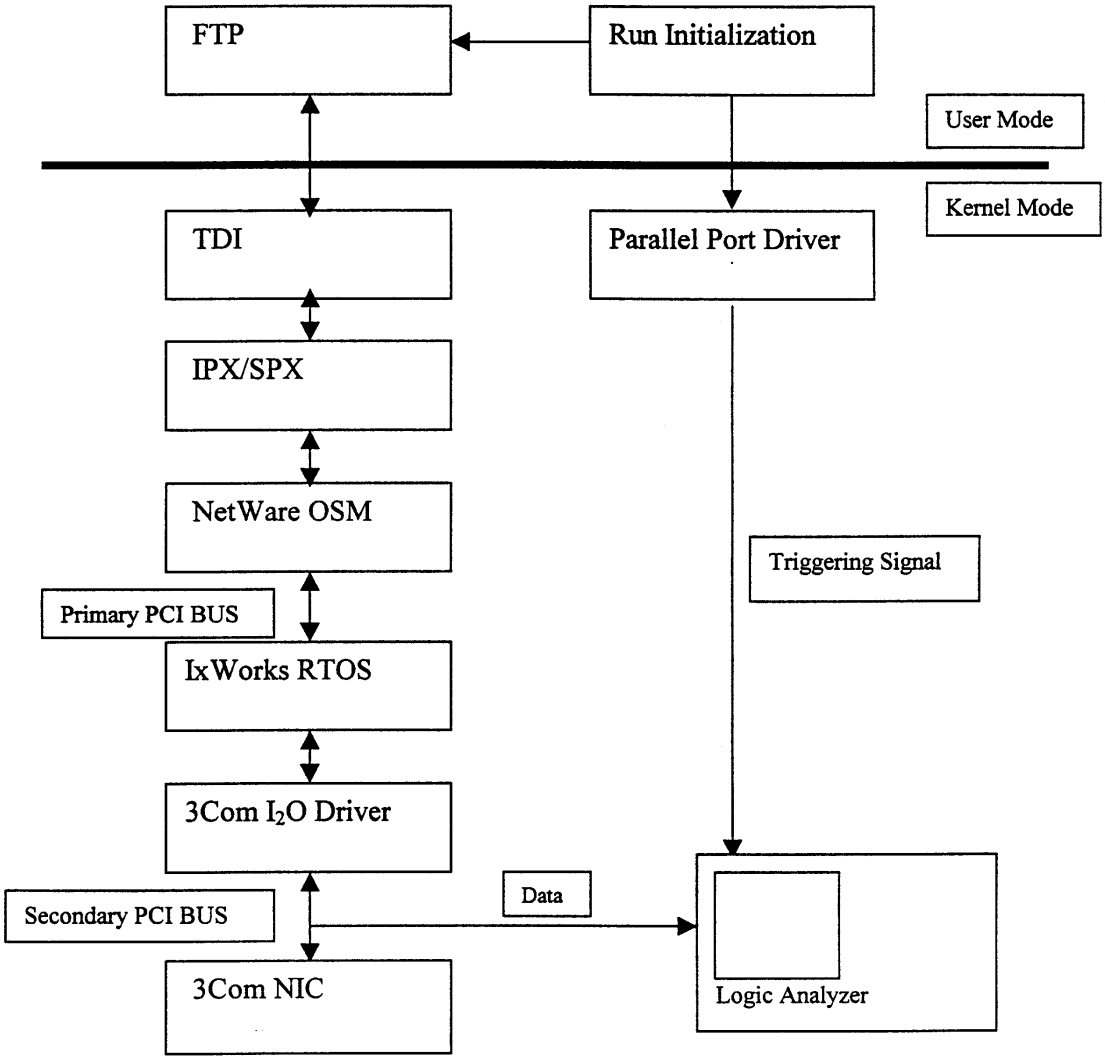

Figure 6-4. Test Model - Secondary PCI Bus

The latency caused by the IOP and HDM can be acquired by subtracting the result of this test model from the previous one.

Chapter 7. Recommendation for Future work

7.1 A Suggestion for Triggering Sequence

In e experiments, parallel port triggering will be the method for triggering the logic analyzer. Except for the source of the triggering signal, the LA triggering sequence could be the same as the prototype experiment presented in Chapter 5. An alternative triggering sequence is suggested below (see Figure 7. 1):

1. In the first level, trigger at "angel" one time and go to the second level. Else remain in the first level.

2. In the second level, trigger at a variable "data", which is setup to be hex 2D976000 (the first 4 bytes of the Ethernet address) on the addr/data bus, and also record its occurrence. Then jump to level 3. Else, remain in level 2.

3. Record all occurrences of "data" on the data bus. Remain in level 3 until the buffer fills up.

This triggering sequence will record the first 4 bytes of the Ethernet address only. The advantages of it are: 1. requires less data to store in limited LA memory. 2. it won't need a protocol analyzer to determent where the frame starts. A data capture of a small file writes to the server is provided in Table 7. 1. Each line corresponds to one Ethernet frame. Although the details in the frame are not shown, the time between the frames is measured. That is sufficient data for the responds time measurement.

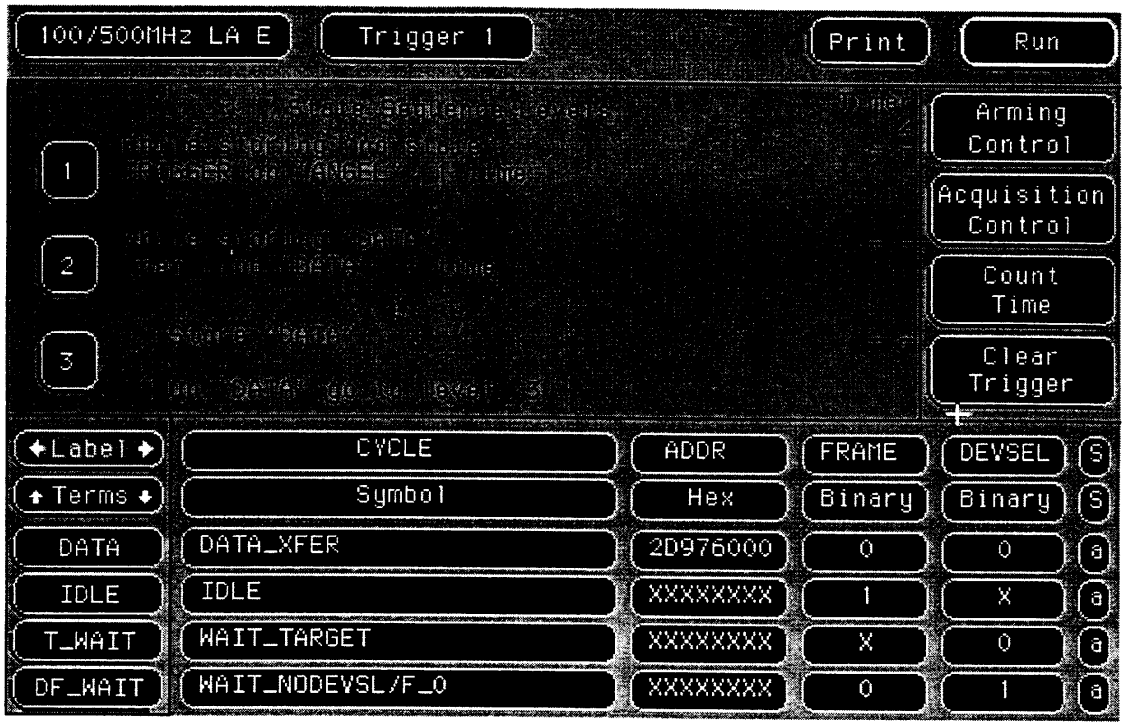

Figure 7-1. Suggested Triggering Sequence

Table 7-1. An Example of Data Frame Header Capture

PCI_BUS1 - State Listing

Label	>	DATA	Time	GNT	DEVSEL	IRDY	TRDY	FRAME	PCICLK
Base	>	Hex	Absolute	Bin	Binary	Bina	Bina	Binar	Binary
1		2D976000	348.1 ms	1	0	0	0	0	0
2		2D976000	348.9 ms	1	0	0	0	0	0
3		2D976000	360.3 ms	1	0	0	0	0	0
4		2D976000	360.9 ms	1	0	0	0	0	0
5		2D976000	363.1 ms	1	0	0	0	0	0
6		2D976000	364.7 ms	1	0	0	0	0	0
7		2D976000	365.8 ms	1	0	0	0	0	0
8		2D976000	366.0 ms	1	0	0	0	0	0
9		2D976000	366.1 ms	1	0	0	0	0	0
10		2D976000	375.2 ms	1	0	0	0	0	0
11		2D976000	505.5 ms	1	0	0	0	0	0
12		2D976000	506.4 ms	1	0	0	0	0	0
13		2D976000	506.6 ms	1	0	0	0	0	0

14	2D976000	507.2	ms	1	0	0	0	0	0
15	2D976000	507.4	ms	1	0	0	0	0	0
16	2D976000	507.5	ms	1	0	0	0	0	0
17	2D976000	523.9	ms	1	0	0	0	0	0
18	2D976000	524.3	ms	1	0	0	0	0	0
19	2D976000	532.5	ms	1	0	0	0	0	0
20	2D976000	532.8	ms	1	0	0	0	0	0
21	2D976000	540.3	ms	1	0	0	0	0	0
22	2D976000	540.6	ms	1	0	0	0	0	0
23	2D976000	543.0	ms	1	0	0	0	0	0
24	2D976000	543.3	ms	1	0	0	0	0	0
25	2D976000	544.3	ms	1	0	0	0	0	0
26	2D976000	544.5	ms	1	0	0	0	0	0
27	2D976000	544.6	ms	1	0	0	0	0	0
28	2D976000	546.9	ms	1	0	0	0	0	0
29	2D976000	547.1	ms	1	0	0	0	0	0
30	2D976000	547.2	ms	1	0	0	0	0	0
31	2D976000	556.5	ms	1	0	0	0	0	0
32	2D976000	556.7	ms	1	0	0	0	0	0
33	2D976000	705.8	ms	1	0	0	0	0	0
34	2D976000	705.9	ms	1	0	0	0	0	0
35	2D976000	906.0	ms	1	0	0	0	0	0

Time Printed: 13 Jun 1998 14:31:07 --- Time Acquired: 13 Jun 1998 14:29:32

7.2 Recommended Test Procedure for Test mode 1

1. Connect the parallel port triggering signal to pin l pod 6 of the HP16500BLA

2. Plug the FuturePlus extender card to a primary PCI slot. Then plug the 3Com NIC to the extender card

3. Connect pod 1 - 4 to the FuturePlus extender card

4. Turn on the LA and load the saved FuturePlus configuration

5. According to the suggested triggering sequence, set up the LA trigger

6. Generate a FTP socket on RI OOB I to Outlander using Winftp

7. Click "RUN" at the upper right comer on the LA display (see Figure 4.4)

8. Initiate the "test run" program requesting use of FTP

9. Print the LA data onto a floppy disk

10. Examine the results using WordPad

7.3 Recommended Test Procedure for Test mode 2

1. Connect the parallel port triggering signal to pin I pod 6 of the HP16500B LA

2. Plug the FuturePlus extender card to a primary PCI slot. Then plug the 3Com NIC to the extender card

3. Connect pod 1 - 4 to the FuturePlus extender card

4. Turn on the LA and load the saved FuturePlus configuration

5. Using the suggested triggering sequence, set up the LA trigger

6. Click "RUN" at the upper right comer on the LA display (see Figure 4.4)

7. Initiate the "test run" program requesting a file write to the server

8. Print the LA data onto a floppy disk

9. Examine the result using WordPad

7.4 Recommended Test Procedure for Test mode 3

1. Connect the parallel port triggering signal to pin l pod 6 of the HP16500B LA

2. Plug the FuturePlus extender card to a primary PCI slot

3. Plug the IQ-SDK to the FuturePlus extender card. Then plug the 3Com NIC to the secondary PCI on the IQ-SDK

4. Connect pod 1 - 4 to the FuturePlus extender card

5. Turn on the LA and load the saved FuturePlus configuration

6. Using the suggested triggering sequence, set up the LA trigger

7. Generate a FTP socket on R100BI to Outlander using Winftp

8. Click "RUN" at the upper right comer on the LA display (see Figure 4.4)

9. Initiate the "test run" program requesting use of FT?
10. Print the LA data onto a floppy disk
11. Examine the result using WordPad

7.5 Recommended Test Procedure for Test mode 4

1. Connect the parallel port triggering signal to pin l pod 6 of the HPI6500B LA

2. Plug the IQ-SDK to the primary PCI

3. Plug the FuturePlus extender card to the secondary PCI slot. Then plug the 3Com NIC to the FuturePlus extender card

4. Connect pod 1 - 4 to the FuturePlus extender card

5. Turn on the LA and load the saved FuturePlus configuration

6. Using the suggested triggering sequence, set up the LA trigger

7. Generate a FTP socket on RI OOB 1 to Outlander using Winftp

8. Click "RUN" at the upper right comer on the LA display (see Figure 4.3)

9. Initiate the "test run" program requesting use of FTP

10. Print the LA data onto a floppy disk.

11. Examine the result using WordPad

Chapter 8. Conclusion

The goal of this project is to measure the network latency of the I_2O architecture. The instrumentation technique, which uses a logic analyzer with triggering initiated by a parallel port signal, is proven to be effective. The result obtained in the prototype experiment shows that this technique also can be used in other measurements that involve I/O processes in a computer system. The only drawback is that the tester must fully understand the I/O process protocol as well as the PCI bus protocol. But this prerequisite is still much better than having to understand the NT source code, and modify it, because it is difficult to deferment whether the modification will introduce bugs into the NT OS. Many of problems encountered were due to the limitation of the LA memory. It is recommended that the memory in the HPI6500B be increased to 64Mbtyes. Then the timing mode can be used for data capture so that a better understanding of the I/O process on the PCI bus can be obtained.

Bibliography

Books

[book 1] Baker, Art. *The Windows NT Device Driver Book.* Prentice-Hall, Inc. 1997.

[book 2] Douglas E. Comer. *Computer Networks and Internets.* Prentice-Hall, Inc. 1997.

[book 3] Helen Custer. *Inside Windows NT.* Microsoft Press. 1993.

[book 4] I. Scott Mackenzie. *The 8051 Microcontroller.* Prentice-Hall, Inc. 1995

[book 5] Andrew S. Tanenbaum. Computer Networks,3rd edition,Prentice-Hall, Inc. 1998

Papers

[paper l] © INTEL CORPORATION. *i960 ® RP/RD I/O PROCESSOR AT 3.3 VOLTS.* 1997

[paper 2] © INTEL CORPORATION. *IQ8096ORx Evaluation Platform.* 1998

[paper 3] © INTEL CORPORATION. i960 ® *Microprocessor Benchmark Report* 1998

[paper 4] © INTEL CORPORATION. i960 ® *Rx I/O Microprocessor Developer's Manual.* 1997

[paper 5] I_2O SIG. *Intelligent I/O Architecture* 1997

[paper 6],Angel Yu. *Senior Project Report.* 1998

[paper 7] Hewlett Packard. *HP 16500B User Manual Vol, 1 & 2*

[paper 8] FuturePlus System Corporation. *FS2000 PCI Extender Card User Manual*

[paper 9] University of Washington. *Instrumentation and Optimization of Win32/Intel Executable Using Etch. 1997*

[paper 10] PCI SIG. *PCI Specifications. 1997*

[paper 11] Hewlett Packard. *HP Network Adviser User Manual, Vol, 1 &2*

[paper l2] WindRiver Systems. *IxWork Real Time Operating System User Guild, Beta 1*

Web Sites

[web 1] I_2O SIG at http://www.i2osig.org/Architecture/

[web 2] Intel i960 Rx at http://developer.intel.com/design/IIO/prodbref/272740.HTM

[web 3] IxWork at http://www.windriver.com/i2o/index.html

[web 4] FuturePlus Systems at http://www.futureplus.com/sidebar.htm

Appendix A: Logic Analyzer Output and Analysis of the Prototype Test

The data capture for the prototype test and its data analysis is provided in a text file called "LA-data.txt" in the PC compatible 3.5" floppy disk.

Appendix B: Protocol Analyzer Output of the prototype Test

The Protocol Analyzer output for the prototype test is provided in a text file called "PA_data.txt" in the PC compatible 3.5" floppy disk.

Appendix C: Logic Analyzer Configuration for the prototype Test

The Logic Analyzer configuration for the prototype test is provided in a HP formatted file called "Prototest_config" in the PC compatible 3.5" floppy disk. If the test result is to be regenerated, this LA configuration can be load in to a HP16500B directly.

Appendix D: Logic Analyzer Configuration for the Frame Capture Test

The Logic Analyzer configuration for the Ethernet frame capture test is provided in a HP formatted file called "test_config" in the PC compatible 3.5" floppy disk. If the test result is to be regenerated, this LA configuration can be load in to a HP 16500B directly.