Design of a Hardware Interface for a High-Speed Parallel Network

by

Scott Jeffery Harper

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Electrical Engineering

APPROVED:

Dr. S. F. Midkiff, Chairman

Dr. I. Jacobs

Dr. N. J. Davis, IV

August, 1994 Blacksburg, Virginia
Design of a Hardware Interface for a High-Speed Parallel Network

by

Scott Jeffery Harper

Dr. S. F. Midkiff, Chairman

Electrical Engineering

(ABSTRACT)

Parallelism can use existing technology in computer communications network design to provide higher data rates and a greater degree of flexibility than monolithic systems. This research investigates the design of a high-speed Parallel Local Area Network (PLAN) interface. It defines the goals of a PLAN interface as low data latency, high data throughput, scalability, and low cost. Three fundamental PLAN interface categories are proposed to meet these goals. These categories are single-bus, dual-bus, and bus-free adaptors. The relative merits of each category are discussed in terms of suitability to several adaptor applications. Each category is further explored by developing a VHDL model of a representative system. The latency, throughput, and component utilization of each model is measured. For medium to large data sets, the dual-bus design provides slightly greater throughput when transmitting encoded data. When transmitting medium to large unencoded data sets, the bus-free design yields marginally higher throughput. In nearly all cases the bus-free design has a greater latency than either of the bus-based design options. Other insights gained from the models regarding physical construction of each adaptor type are also presented.
Acknowledgements

I would like to express my sincere gratitude to my thesis advisor, Dr. S. F. Midkiff for his patient guidance and support during the completion of my thesis. His insights were a valuable asset.

I would also like to thank Dr. I. Jacobs and Dr. N. J. Davis, IV for serving on my advisory committee. Their comments and suggestions were appreciated.

Many thanks to my fellow graduate students and the staff of the Bradley Department of Electrical Engineering. You have made the pursuit of my MS degree most enjoyable. In particular, thanks to Rajesh Kumar and Joe Wiencko for their work on the PLAN system.

Finally, I would like to thank my parents, Albert and Beverly Harper, for their continued encouragement.
Table of Contents

Acknowledgements .................................................................................. iii

Table of Contents .................................................................................. iv

List of Illustrations .................................................................................. viii

List of Tables ........................................................................................... x

Chapter 1. Issues in Parallel Network Interface Design ......................... 1

1.1. Overview and Organization .............................................................. 1

1.2. Background .................................................................................... 2

1.2.1. Need for Parallelism ................................................................. 2

1.2.2. Existing Networks ..................................................................... 3

1.3. Parallelism in Networks .................................................................. 6

1.3.1. Modifying the OSI Reference Model ....................................... 6

1.3.2. The PLAN System .................................................................. 8

1.3.3. Advantages of the PLAN System ............................................ 10

1.4. Research Objectives ...................................................................... 12

1.4.1. Problem and Motivation ......................................................... 12

1.4.2. Design Approach ................................................................... 13

1.4.3. Summary of Results ............................................................... 13

Table of Contents ................................................................................. iv
3.2.3. Protocol Processor Model ........................................ 49
3.2.4. Access Layer Processor Model ................................. 51
3.2.5. Coding Unit Model .............................................. 55
3.2.6. Base Adaptor Model ........................................... 57
3.3. Simulation Results ................................................ 60
3.4. Summary ............................................................. 65

Chapter 4. Multiple-Bus Systems ......................................... 75
4.1. Analysis of a Multiple-bus Systems ............................... 75
  4.1.1. Variables Used in Analysis .................................... 77
  4.1.2. Analysis of a Dual-bus System ................................ 78
  4.1.3. Analysis of a Triple-bus System ............................. 82
  4.1.4. Analysis Results ............................................... 83
4.2. Functional Modeling of a Dual-bus System ....................... 87
4.3. Simulation Results .................................................. 88
4.4. Summary ............................................................... 91

Chapter 5. Bus-Free Systems ............................................. 100
5.1. Analysis of a Multi-ported Queue System ....................... 106
  5.1.1. Variables Used in Analysis .................................... 101
  5.1.2. Multi-ported Queue System Analysis ....................... 102
  5.1.3. Analysis Results ............................................... 106
5.2. Analysis of Other Bus-Free Systems ............................. 111
  5.2.1. Variables Used in Analysis .................................... 111

Table of Contents
List of Illustrations

Figure 1-1. Existing high-speed parallel networking approaches. .......................... 4
Figure 1-2. The ISO/OSI reference model. ......................................................... 6
Figure 1-3. The GPN reference model. ............................................................... 8
Figure 1-4. General PLAN system structure. ....................................................... 9
Figure 1-5. Parallel network with scalability. ..................................................... 11
Figure 2-1. Single-bus system. ................................................................. 20
Figure 2-2. Modified single-bus system. ............................................................ 22
Figure 2-3. Dual-bus system. ................................................................. 23
Figure 2-4. Triple-bus system. ................................................................. 25
Figure 2-5. Multi-ported memory system. ......................................................... 26
Figure 2-6. Interconnection network system. .................................................... 27
Figure 3-1. Single-bus system. ................................................................. 32
Figure 3-2. Modified single-bus based system. ............................................... 37
Figure 3-3. Arbiter state diagram. ............................................................. 45
Figure 3-4. Typical bus request. ............................................................... 46
Figure 3-5. Typical MBus read transaction ...................................................... 47
Figure 3-6. Typical MBus write transaction ..................................................... 47
Figure 3-7. AP send state diagram ............................................................. 48
Figure 3-8. AP receive state diagram ........................................................... 48
Figure 3-9. PPIC state diagram ................................................................. 49
Figure 3-10. PP send state diagram .............................................................. 51
Figure 3-11. PP receive state diagram ........................................................... 52

List of Illustrations  viii
List of Tables

Table 3-1. System Component Bandwidth (bps) ........................................ 39
Table 3-2. Simplified Component Bandwidth Equations (bps) .................. 40
Table 3-3. MBus Physical Signals [SPAR91] ........................................... 44
Table 3-4. Sending Node AP Utilization (percent) .................................... 67
Table 3-5. Sending Node PP Utilization (percent) .................................... 67
Table 3-6. Sending Node CU Utilization (percent) .................................... 68
Table 3-7. Sending Node ALP Utilization (percent) ................................... 68
Table 3-8. Sending Node BA Utilization (percent) .................................... 69
Table 3-9. Sending Node SMU Utilization (percent) ................................... 69
Table 3-10. Sending Node MBus Utilization (percent) ............................ 70
Table 3-11. Receiving Node CU Utilization (percent) ............................... 70
Table 3-12. Receiving Node BA Utilization (percent) ............................... 71
Table 3-13. Receiving Node ALP Utilization (percent) ............................... 71
Table 3-14. Receiving Node PP Utilization (percent) ............................... 72
Table 3-15. Receiving Node SMU Utilization (percent) ............................. 72
Table 3-16. Receiving Node MBus Utilization (percent) ............................ 73
Table 3-17. Receiving Node AP Utilization (percent) ............................... 73
Table 3-18. System Throughput and Latency ......................................... 74
Table 4-1. System Component Bandwidths (bps) .................................... 84
Table 4-2. Simplified System Component Bandwidths (bps) ...................... 85
Table 4-3. Sending Node PP Utilization (percent) .................................... 93
Table 4-4. Sending Node ALP Utilization (percent) .................................. 93
Table 4-5. Sending Node CU Utilization (percent) ........................................ 94
Table 4-6. Sending Node BA Utilization (percent) ......................................... 94
Table 4-7. Sending Node SMU Utilization (percent) ....................................... 95
Table 4-8. Sending Node MBus Utilization (percent) ...................................... 95
Table 4-9. Receiving Node BA Utilization (percent) ......................................... 96
Table 4-10. Receiving Node CU Utilization (percent) ..................................... 96
Table 4-11. Receiving Node ALP Utilization (percent) ..................................... 97
Table 4-12. Receiving Node PP Utilization (percent) ....................................... 97
Table 4-13. Receiving Node SMU Utilization (percent) ................................... 98
Table 4-14. Receiving Node MBus Utilization (percent) ................................... 98
Table 4-15. System Throughput and Latency .................................................. 99
Table 5-1. System Component Bandwidths (bps) ............................................. 107
Table 5-2. Sending Node PP Utilization (percent) .......................................... 124
Table 5-3. Sending Node ALP Utilization (percent) ........................................ 124
Table 5-4. Sending Node CU Utilization (percent) .......................................... 124
Table 5-5. Sending Node BA Utilization (percent) .......................................... 125
Table 5-6. Receiving Node BA Utilization (percent) ....................................... 125
Table 5-7. Receiving Node CU Utilization (percent) ....................................... 126
Table 5-8. Receiving Node ALP Utilization (percent) ..................................... 126
Table 5-9. Receiving Node PP Utilization (percent) ....................................... 126
Table 5-10. System Throughput and Latency ............................................... 127

List of Tables
Chapter 1. Issues in Parallel Network Interface Design

1.1. Overview and Organization

This research investigates the construction of a high-speed Parallel Local Area Network (PLAN). Chapter 1 presents the basis for this system by describing the need for a high-speed network and briefly looking at some of the current approaches to high-speed networking. It then goes on to describe work done at Virginia Polytechnic Institute and State University in the area of parallel networking. The General Parallel Network (GFN) model that resulted from this work is described and presented as the basis for the PLAN system. Finally, the objectives of this research are presented and a summary of the results is given.

Chapters 2 through 6 investigate the development of hardware to support a PLAN. Chapter 2 describes the requirements and objectives for the development of a hardware interface for a PLAN system. Three PLAN architectures are proposed, including single-bus, multiple-bus, and bus-free designs. Specific examples of existing hardware that may be used in realizing the architectures are also given. Chapter 3 investigates the single-bus based interface. It refines the basic single-bus design and presents a Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) model for the refined design. Chapter 4 investigates an adapter design based upon multiple buses. Both dual- and triple-bus architectures are discussed and a dual-bus system model is developed. Chapter 5 investigates the various bus-free designs and contrasts them with the bus-based architectures presented in Chapters 3 and 4. Chapter 6 compares the proposed architectures. Recommendations are made for the development of systems for GPN research and products based upon the various architectures.
Finally, Chapter 7 summarizes the work done in this research. It also describes opportunities for future research.

1.2. Background

Parallelism is used in many aspects of high-speed computing. It can be found within processors, among processors, in memory systems, and in storage devices. In every one of these instances, parallelism is seen as an essential part of increased machine performance. However, computer communications has yet to fully realize the benefits of parallelism.

1.2.1. Need for Parallelism

As computers become faster, they place greater demands on the local area networks to which they are attached. Traditionally, this demand for higher speed networks has been met by increasing the data rates provided by single channel, or monolithic, systems. Each of these increases in monolithic system speed requires a corresponding increase in the speed and complexity of technology employed within the system. At any given time, each increase in system speed is, therefore, accompanied by a corresponding increase in system cost. Finally, as network systems grow faster and costlier, they do not generally become any more flexible in terms of machine connectivity than their lower speed ancestors.

As an example of the additional cost incurred as the capabilities of monolithic technology are increased, consider the cost of 3.5" 2 MegaByte (MB) disks compared to the cost of 3.5" 4 MB disks. A 1994 catalog [ME194] lists the price of the former at $0.769 per disk and the later at $2.359 per disk for a single brand name when purchased in quantity. This translates into $0.384 per MB for the 2 MB disks.
and $0.590 per MB for the 4 MB disks, or a 50 percent increase in unit cost to move to the leading edge of technology.

Parallelism shows promise of providing a number of advantages when used as an integral part of a communication network. The use of parallelism in a network may allow for both a higher data rate using the same technology and a greater degree of flexibility in machine interconnection when compared to monolithic systems. Additional potential advantages include a favorable cost/throughput ratio, scalability, fault tolerance capabilities beyond those achievable with serial networks, and an upward migration path from existing networks.

This research investigates the construction of a high-speed Parallel Local Area Network (PLAN). Initially, a General Parallel Network (GPN) model is described as the basis for this PLAN system. Then, three general PLAN architectures are proposed and investigated. Each of these architectures is refined from a basic case by investigating the advantages and disadvantages of the structure. Finally, recommendations are made concerning the best architecture to use for development of a system based upon the GPN model.

1.2.2. Existing Networks

There are a variety of approaches currently being used to achieve high speeds in computer communication networks. These approaches range from the use of high-speed protocols or hardware in existing network systems to the design of radically new network hardware and software. In an attempt to classify the existing approaches to the design of transport protocols, Zitterbart employed a scheme to list the most interesting approaches to the design of high-speed transport protocols [ZITT91]. A slightly
modified version of this classification scheme may be generalized for high-speed network design as shown in Figure 1-1. Although this scheme is not a full taxonomy, it does provide a basis for discussion of current networking approaches. For a more complete classification of existing protocols for high-speed networking, see [SKOV89].

![High-Speed Networking Diagram]

**Figure 1-1.** Existing high-speed parallel networking approaches.

This classification scheme divides current high-speed networking approaches into two general categories. The category of high-speed protocols is primarily software oriented, while the category of high-speed hardware deals primarily with existing hardware that supports high-speed networking. However, since this classification is not a true taxonomy, the two areas do overlap somewhat. Some of the high-speed hardware may have been specifically designed to implement existing network protocols, and some of the protocols may require the use of special protocol specific hardware.
The category of high-speed protocols is broken down into two subcategories. The category of evolutionary approaches includes those approaches that are based on adapting existing protocols to a high-speed environment. Examples of this approach include protocols that modify the standard TCP/IP protocol by adding features like an expanded window size, a timestamp, or a selective acknowledgement [JACO88, JACO90]. Experiments with these protocol extensions have shown them to be effective in increasing attainable throughput to near gigabit-per-second (Gbps) rates [NICH91].

The subcategory of revolutionary approaches, on the other hand, includes protocols that were designed specifically for high-speed networking. Often, these approaches were developed with a specific class of application in mind. For example, VMTP was designed to provide communication for a distributed operating system [DOER90]. As such, its primary goal is to provide quick responses for small amounts of data. While these types of systems may provide exceptional performance in certain situations, they do not necessarily provide the same performance level for more general networking applications.

High-speed hardware is often built to support an existing protocol. The Network Adapter Board (NAB), for example, is specifically designed to process VMTP messages [ZITT91]. While this type of hardware can certainly increase the realizable performance of a given protocol, it cannot increase the performance of a monolithic protocol beyond the speed provided by the slowest component in the system. Furthermore, for protocols that were not designed with hardware in mind, the design of specialized adapters may be difficult.

The goal of the GPN system is to provide an evolutionary protocol that is readily supportable by parallel hardware. As such, it falls across the categories above. An evolutionary protocol offers the advantage of being able to contain existing networks if it contains, as a subset, the protocol from which it evolved. A PLAN system is the parallel hardware realization of the GPN protocol.

Chapter 1 - Issues in Parallel Network Design
1.3. Parallelism in Networks

1.3.1. Modifying the OSI Reference Model

Any approach to a general parallel network system design must first consider the general structure of a monolithic network. If this structure can in some way be included in the design of a general parallel system, it will allow existing monolithic networks to be more easily brought into the parallel realm. The currently accepted general structure for monolithic network systems is represented by the Open Systems Interconnection (OSI) reference model.

The International Standards Organization (ISO) has proposed a seven layer model for the connection of open systems [TANE88]. That model is called the OSI, or ISO/OSI, reference model and is shown in Figure 1-2.

![Diagram of OSI reference model](image)

*Figure 1-2. The ISO/OSI reference model.*
Each layer in this model provides a different level of abstraction for a networking system. These different levels of abstraction are useful in that they provide a basis for modular network designs.

With the idea of retaining the modular structure of the OSI reference model, we propose the addition of two layers to coordinate parallel activities in a General Parallel Networking (GPN) model as shown in Fig 1-3 [KUMA93]. The Access Layer is responsible for delivering data from a source service access point to a destination service access point. The Application Interface Layer is responsible for partitioning and reassembling data as it moves between the very high-speed Application Layer and the lower speed parallel components of the Presentation Layer [KUMA93].

The Access Layer provides one interface to the Network Layer and a second interface to the Data Link Layer. The services and structure of the Data Link Layer interface may depend upon the monolithic system being replicated as a basis for the parallel network. The services provided by the Access Layer/Network Layer interface, however, are more general and are discussed below.

The Access Layer provides these types of services to the Network Layer [MIDK91]:

- data transfer requests and arrival indications,
- local status information via requests and indications, and
- remote node connectivity requests and indications.

It also supports a set of virtual channel addresses (VCAs) associated with each node that correspond to one or more Data Link Layer address. These addresses allow a remote device to select a given portion of the parallel Data Link Layer to be used in the transfer of a given set of data.
1.3.2. The PLAN System

Figure 1-4 shows the basic structure of a Parallel Local Area Network (PLAN) system developed using the GPN reference model. In this diagram, objects labeled "fabric" are used to represent the logic used to implement the Application Interface Layer and Access Layer as well as the physical connection between nodes. This logic may be any combination of hardware and software that faithfully implements the layers. The fabric is drawn with protrusions toward each connection to indicate the fact that some of the logic may lie within each of the individual components of the system.
The Application Processor (AP) shown here is envisioned to execute a single high-speed application that utilizes the parallel local area network connection. Data that it sends to the network is broken into pieces and read by the set of $N$ Protocol Processors (PPs). These processors attach upper layer protocol, e.g. TCP/IP, headers to the data and generate data requests to the second level of fabric. Part of this second level of fabric is the Access Layer Processor (ALP). When data sufficiency has been determined by the ALP, it sends the data on to the set of $M$ Base Adapters (BAs). The BAs contain any encoding logic that may be used to provide security or reliability for the data that is to travel on the network. The primary function of the Base Adapters, however, is to transmit the data to another station on the network. This functionality is provided by using an existing monolithic network technology such as Ethernet or FDDI.
Data is placed onto the final level of fabric by the BAs. This fabric may be wire, optical fiber, radio, or any other communication medium. After being transferred on this last level of fabric, data is received by the Base Adapters of another station. The BAs of this receiving station signal the receiving station's ALP that data has been received and pass the decoded information up to the station's PPs. The Protocol Processors strip the TCP/IP headers from the data and reassemble it for presentation to the station's Application Processor.

A network node in this PLAN system is described as an $MNI$ node, where $M$ refers to the level of parallelism in the network connection, or the number of Base Adapters, $N$ refers to the level of parallelism used in protocol processing, and $I$ refers to the throughput provided to the application processor [WIEN92].

1.3.3. Advantages of the PLAN System

The PLAN system offers several advantages over a traditional monolithic network. These advantages include scalability, fault tolerance, low cost, and high data rates [WIEN92].

The ability to build a system based on existing technology allows that system to take advantage of the low cost and known reliability of mass-produced components. The PLAN system takes advantage of this both in the Base Adapters and Protocol Processors. As previously stated, the Base Adapters may be implemented using a proven technology like Ethernet or Fiber Distributed Data Interface (FDDI). Furthermore, the Protocol Processors may use existing TCP/IP code to accomplish their tasks. This use of existing technology will save both time in development and debugging and the cost involved in preparing specialized hardware.

Chapter 1 - Issues in Parallel Network Design
Furthermore, by working together in parallel, several components that individually supply modest data rates may provide a very high data rate. Parallelism in general requires that a task be divisible among several separate entities. The PLAN system provides this type of division, and allows lower performance networking components to be combined into a single higher performance network entity.

The $M$ Protocol Processors and $N$ Base Adapters allow a system to contain as much parallelism as is needed at a particular level. For example, if the cumulative speed of the Protocol Processors needs to be increased, $M$ may be increased while leaving the speed of the network connection at the same level by leaving $N$ unchanged. This allows the system to be scalable both within a particular node ($M$) and within the entire network ($N$). Furthermore, a properly designed routing scheme will allow nodes with differing levels of network connectivity to coexist on the same network. Figure 1-5 [WIEN92] shows an example of this type of network. Here, a lower speed station with a single connection is able to communicate with faster stations on the same network.

![Diagram](image)

**Figure 1-5.** Parallel network with scalability.
In addition to scalability, the PLAN system may be designed to include a greater level of fault tolerance than is available with a monolithic system. A technique called Cross Channel Coding [WIEN92] can be employed within the base adaptors to allow data to be recovered in the event that a proper subset of the \( M \) network channels are lost. This technique adds check bits to each packet transferred that, in conjunction with a channel integrity check, allow for the correction of random or burst errors in addition to providing protection against channel loss [WIE92].

Finally, the PLAN system allows for local data distribution of high data rate networks. Data traveling along a high data rate network may be subdivided into the parallel connections of a PLAN. The ability to attach nodes of dissimilar capabilities to the PLAN as shown in Figure 1-5 makes it easier to distribute network data among many different types of users, including individual high performance computers, LANs comprised of lower performance machines, and individual lower performance machines.

1.4. Research Objectives

1.4.1. Problem and Motivation

For a PLAN system to be effective, a high-speed path must be provided between the AP and the BAs. Along this path, the fabric used to connect the various components must be defined in a manner that avoids bottlenecks. Therefore, an investigation into the design of a hardware interface for a PLAN system is warranted.
Furthermore, for a protocol to be effective, it must have a hardware implementation. In addition, for the advantages claimed to exist in the PLAN system to be shown, a specific implementation of such a system must be demonstrated. Therefore, the investigation into the design of a hardware interface for a PLAN system should study alternative designs and make recommendations regarding implementation.

1.4.2. Design Approach

The basic design approach taken is a modular one. Each component of the system is allowed to be independently altered without requiring the alteration of any other system components. This approach is useful in the early design stages since it means that system components may be designed and tested independently. It is also advantageous in the long term since it means that system components may be upgraded as new ones become available.

Three categories of systems are investigated. These categories are identified by the number of buses used to implement the architectures contained within them. This categorization allows recommendations to be made based on a general class of system designs rather than a specific implementation. It also allows for development of several "best" implementations, each having unique properties.

1.4.3. Summary of Results

The investigations performed in Chapters 3 through 5 result in three basic system models. These systems are based on a single system bus, a dual system bus, and a set of multi-ported queues. Simulation models are developed for the three systems and component utilization, system latency, and system throughput are estimated.
Of the three systems modeled, the single-bus system provides the best structure for rapid prototype development and is the best choice for initial testing. However, it presents system limitations that inhibit performance for general use. The dual-bus and multi-ported queue systems are, therefore, better choices for general use.
Chapter 2. The Design of a Parallel Network Interface

Chapter 2 discusses the requirements and objectives for the development of a PLAN interface. In addition, several options for specific PLAN interface architectures are discussed.

2.1. Function of the Interface

The parallel network system described in Chapter 1 requires a high-speed hardware interface to provide an intelligent data path between the set of Protocol Processors (PPs) and the set of Base Adapters (BAs). Intelligent, in this context, refers to a path that is capable of moving a data packet both downward from the Protocol Processors to a specific Base Adapter and upward from a specific BA to the set of PPs. It must be capable of adding header and encoding information to the data travelling along the downward path. The path must also be able to route downward flowing information correctly so that it is sent through a Base Adapter that is connected to the destination node. Furthermore, this path must be able to decode, strip header information from, and reassemble upward flowing data so that it may be presented to the set of PPs.

The data path between the Application Processor and the Protocol Processors must also be able to handle a high rate of traffic. This means that many of the considerations made in the design of the PP to BA interface may also apply to the AP to PP interface. Furthermore, the two interfaces must operate within the same system. For these reasons, the options presented in this document for an interface design include the impact of each design on both interfaces.
2.2. Requirements and Objectives

The design of the interface must be closely tied to the parallel protocol defined for the data transfer. However, it is beneficial in terms of future system development and protocol modification for the adapter to be designed in such a way that its function is easily modified. Toward this end, a modular design approach is used in its development.

Several specific objectives must be met by the interface:

- low data latency,
- high data throughput,
- scalability, and
- low cost.

Each objective is discussed below.

2.2.1. Low Data Latency

Latency is defined here as the time between the last bit of a particular data packet, traveling in a specific direction, entering a component or group of components and that same last bit of the packet leaving the component or component group. Component latency is then the latency introduced by a single component. Adapter latency is the latency introduced by the entire adapter, or the sum of the individual component latencies.
A PLAN system must provide a high data throughput, as discussed in Section 2.2.2, and the interface must not impede this data flow. Minimizing the adapter latency will help increase throughput in situations where small amounts of data are being transferred in bursts. In these situations, each burst of data must incur a startup latency. In addition, if an acknowledgment is sent back from the receiver between bursts, any adapter latency will slow the send/acknowledge cycle. Both of these conditions imply that adapter latency reduces the system throughput for small packets. It is, therefore, beneficial in terms of system throughput to meet the goal of minimizing adapter latency.

To meet the goal of low system latency, it is essential that no single part of the adapter have a high component latency. Furthermore, the sum of the component latencies must be minimized.

2.2.2. High Data Throughput

The interface must be capable of supporting high data transfer rates. As discussed in Section 2.2.1, minimizing the system latency will help to support high data rates. When transferring large amounts of data, however, the adapter latency does not play such an important role in the system throughput. In these cases adapter latency only affects the transfer at the beginning, as startup delays are experienced by the packet, and at the end, as startup delays are experienced by an acknowledgement. In the middle of a large transfer, the adapter limits throughput only by its own sustained data transfer capabilities. Therefore, it is essential that the adapter be designed so that it has a sustained data transfer capacity that is comparable to the rate provided by the sum of the Base Adapters. This design approach guarantees that the adapter presents no bottlenecks to a sustained transfer.
2.2.3. Scalability

One of the goals of the PLAN system is scalability. To support this goal, its interface must be scalable. This interface should be designed to be easily scaled with an approximately linear change in cost and without any major design changes. The ability to scale the system both up and down without major design changes allows users to tailor the PLAN system to their specific needs in terms of throughput and cost.

The use of parallelism in the design of the interface may lead to a system that is readily scalable. For example, if the Protocol Processors are each independent parallel entities responding to some control logic, then only the control logic needs to be changed to add more of them. In fact, if the control logic is designed in such a way that it is capable of handling an arbitrary number of units, then more of the independent Protocol Processors may be added to the system without changing the control logic.

2.2.4. Low Cost

The ability to produce high-performance systems using replicated low-cost components is an advantage of parallelism in general. The interface developed here takes advantage of this. By implementing parallelism in the interface, a number of lower performance components are assembled to form the high-performance system. These lower performance components typically have a lower aggregate cost than a single high-performance component. Therefore, parallelism is used in the design to lower the interface cost.
The goals of producing a scalable interface and doing so in an inexpensive manner may be met in the same fashion, since both involve the development of a parallel system. However, the use of parallelism implies the need for a control mechanism. This mechanism may add extra latency to the system. Furthermore, the control mechanism must be designed so that it does not limit system throughput. Therefore, the goals of producing a low-cost, scalable system must be balanced against the goals of producing a low latency, high throughput system.

2.3. Design Approaches

Several alternatives for the design of the PLAN interface are developed in this document. These alternatives will be categorized by their use of bus structures as listed below.

- Single-bus
- Multiple-bus
- Bus-free

A single-bus system relies on a single bus to provide an interface between the AP, the various PPs, and the Base Adapters. The multiple bus system uses more than one bus to break the system into independent units and reduce the traffic on any given bus. The third option does not rely on buses to connect the various parts of the system. Each of the three alternatives is described below.
2.3.1. *Single-bus System*

The single-bus design is based on a single system bus as shown in Figure 2-1. All components of the interface communicate by using this bus to access a shared memory.

![Diagram of single-bus system](image)

**Figure 2-1.** Single-bus system.

The Shared Memory Unit (SMU) may be either a single large memory or a segmented memory composed of a number of smaller memories. If it is implemented as many small units, each of the units will be separately accessible via the system bus. This segmented structure allows several memory requests to be interleaved, thereby increasing system performance. It also allows memory size to be easily changed by adding or removing individual memory units, rather than upgrading a single large unit.

The Coding Unit (CU) may also be implemented as either a single subsystem or a distributed device. If it is necessary to provide communication between the separate parts of the logic on which the coding is being done, the design complexity is increased by breaking the coding logic into separate units.

---

*Chapter 2 - The Design of a Parallel Network Interface*
However, designing the Coding Unit as a number of subunits provides for a more easily scaled system and may, in fact, increase throughput by allowing coding computations to be performed in parallel.

The single-bus system is not difficult to implement since each processor and the Coding Unit may be designed separately and attached to the common bus. A bus that does not limit the number of attached devices provides a good platform for a readily scalable system. Furthermore, the bus definition specifies all arbitration for access to the various units and maintains any caches that may be part of the attached units. This means that fewer decisions have to be made regarding the design of an individual unit’s interface to the rest of the system. Finally, some of the system components, including the Shared Memory Unit and the Base Adapters, may be readily available for the chosen bus type as "off the shelf" equipment. For example, Ethernet cards are available for most bus structures and memory boards are available for all standard buses. The use of these "off the shelf" components can reduce development time for the PLAN system. This is because the only additional design required to make the system functional is the specification of the control logic that allows the components to work together. Finally, components that are mass produced are less expensive than custom-designed components.

Since each unit attached to the bus may operate independently of other units, this design intrinsically supports parallelism. This parallelism allows the various system components to perform their tasks independently. However, since all communication between the components occurs on a shared bus, the system requires a very high speed bus. A slow bus creates a serial bottleneck in the system and the communication time between components limits system performance.

One step that may be taken to eliminate some of the bus traffic is to incorporate the coding logic and the Base Adapter into a single unit. This modified system is shown in Figure 2-2. Each Base Adapter combined with the coding logic that supplies it with information is now referred to as a Data Pipe (DP).

Chapter 2 - The Design of a Parallel Network Interface
This terminology is a result of the system’s resemblance to a pipelined parallel computer design [STON90]. The DPs in Figure 2-2 are denoted by dotted lines.

![Diagram of a modified single-bus system](image)

**Figure 2-2.** Modified single-bus system.

The modified single-bus design eliminates the need to use the bus when transferring data between the Base Adapter and the Coding Unit. This reduction in the number of bus accesses allows a system to provide fractionally higher throughput and lower data latency than that available in the single-bus design of Figure 2-1. Furthermore, the system loses no functionality with this approach since data must normally pass through the Coding Logic when entering or leaving the Base Adapters.

The design of the coding unit may be more difficult in this design since it is now a dual ported component. However, the fact that the BAs are no longer resident on the system bus creates a more modular system. In fact, they may now be transferred to a physically separate unit and a custom interface may be designed for them.

Chapter 2 - The Design of a Parallel Network Interface
In the case of low data loss rates or non-critical data transfers, it may be desirable to eliminate the Coding Logic functionality. In these cases, the Access Layer Processor (ALP) may turn the logic off via a multiplexed bypass path. By including this bypass path, the coding hardware or software may be replaced or modified while the system continues to function.

2.3.2. Multiple-bus System

Another step may be taken to reduce utilization of any given bus by changing the system to a dual-bus based design. With this approach, two buses and a dual-ported memory unit are employed to reduce the load on any single bus as shown in Figure 2-3.

In this system, the Protocol Processors and Application Processor communicate with the Shared Memory Unit using the Upper-Layer Bus, while the Coding Unit and Base Adapters communicate with the same...
memory unit via a separate Lower-Layer Bus. This Shared Memory Unit is the fabric that serves to connect the Protocol Processors and the Data Pipes. The method by which information is exchanged between the upper and lower buses is providing the functionality of the Access Layer.

The considerations regarding the construction of system components are similar to those previously discussed for the single-bus design. The Protocol Processors and Coding Unit may still be designed as distributed or monolithic units, and commercially available systems may still be employed to reduce costs. Now, however, it is even more likely that commercially available components may be used in the design. This is due to the fact that each bus type may be chosen such that the best choice of commercial equipment can be attached to it. For example, the lower bus may be chosen to be an industry standard EISA bus, allowing low-cost FDDI cards to be purchased for the Base Adapters [AMD90]. The upper bus may then be chosen to be an MBus [SPAR91], allowing the AP to be a readily available workstation [SUN91].

This use of separate bus types for the upper and lower buses complicates the design of the Shared Memory Unit, but not to any great extent. Since the SMU must communicate with both bus types, it is likely that it must be custom designed. However, as long as standard bus types are used for both buses, it is likely that a chip set is available to manage each bus interface. Design of the Shared Memory Unit would then require the construction of a device comprised of two bus interface units and a bank of common memory. If done correctly, this design makes it is possible to change either one of the bus types by simply replacing the interface logic for that bus.

As shown in the previous section, the Coding Unit may be combined with the Base Adapters to relieve congestion on the lower bus. No such combination is possible for the components on the upper bus,
however. Therefore, a third bus may be included to relieve congestion on the upper bus. This triple-bus based system is shown in Figure 2-4.

![Diagram of triple-bus system](image)

**Figure 2-4.** Triple-bus system.

Here, a second Shared Memory Unit is used to interface the Application Processor with the set of Protocol Processors. This memory unit operates exactly as the lower Shared Memory Unit and provides some of the same advantages and disadvantages.

Once again, a custom designed SMU may be required. However, this time it does not require additional design effort. Since it is functionally equivalent to the previously designed lower Shared Memory Unit, it may be built by following the same design using a different set of bus interface logic. The additional design effort required to make this triple-bus system work is focused primarily in the use of dual-ported

Chapter 2 - The Design of a Parallel Network Interface
Protocol Processors. These new processors must be able to interface to two separate buses and operate on the data flowing between them.

2.3.3. Bus-free System

A radical change in the design approach allows all problems associated with bus traffic to be eliminated. This third approach eliminates all buses as interconnection devices. An example of this approach is shown in Figure 2-5.

![Diagram of multi-ported memory system]

Figure 2-5. Multi-ported memory system.
Here, multi-ported Shared Memory Units are used to provide the connection fabric between the layers of the PLAN interface. To produce a scalable system, the Shared Memory Units must provide an arbitrary number of ports to any single memory unit. This type of memory system allows for fast communication between components with the only limiting factor being the speed of the memory itself. However, a memory system with an arbitrary number of ports is not possible using current technology and would take considerable effort to develop. Therefore, its use in this interface design is not practical.

A more practical approach to the elimination of the busses is the use of an interconnection network, as shown in Figure 2-6. The interconnection network shown in this figure is used to eliminate some of the memory ports required to implement a system based upon the multi-ported memory approach. All system components in this design communicate via a single multi-ported SMU. This memory may, however, be broken into smaller single-ported units. Another design option based on interconnection networks is to replace the busses used in the triple-bus based design by separate networks.

![Interconnection Network Diagram](image)

Figure 2-6. Interconnection network system.
Interconnection networks have been used to replace buses in many parallel computing systems [STON90]. The technology for producing these networks is readily available and much is known about the performance of different network schemes. Therefore, this step is the most logical one to take in the reduction of the data traffic congestion problems associated with bus-based systems.

The use of interconnection networks in the design, however, presents new design challenges. It may no longer be possible to purchase commercial components for the various system units. A system built around interconnection networks requires each unit to be either custom designed or adapted to operate with the chosen network. In addition, any limitation in the size of an interconnection network scheme also limits the size of the PLAN system developed around it.

2.4. Base Adaptor Equipment

A fundamental building block of the PLAN system is the Base Adaptor. The set of BAs will be operated in parallel to provide a high-speed network. As such, the type of Base Adaptor used will have a tremendous effect on the performance of the overall system. Although a design goal is to produce a system that can make use of any monolithic network technology as a Base Adaptor, it is necessary to chose a specific network type for use in initial simulation and prototyping. This network must provide a high throughput to cost ratio to fully demonstrate the advantages of the PLAN system.

One commonly available network type is Ethernet. Low-cost Ethernet cards are readily available for most bus types [BLAC93]. Furthermore, many people are familiar with the capabilities and limitations of Ethernet systems. These factors combine to make Ethernet cards an attractive choice for the Base
Adapters. However, the bandwidth of an Ethernet system is limited. At 10 Mbps, it would take a large
number of Ethernet cards to produce a PLAN system capable of transferring data at Gbps rates. Use of
a small number of these cards would not provide data rates that appropriately stress the PLAN adapter
hardware during initial testing.

A better choice for the set of Base Adapters is a set of FDDI cards. A single FDDI system is capable
of providing 100 Mbps data transfer rates [MCCO87]. This means that ten FDDI Base Adapters can
provide transfer rates approaching 1 Gbps. Although FDDI cards are not currently as readily available
as Ethernet cards, they are available for most common bus types [AMD90]. Furthermore, an FDDI
system may be built using a set of four chips produced by Advanced Micro Devices [AMD92]. This
four-chip set allows for relatively rapid development of FDDI systems for the more complex system
design options like an interconnection network based system.

2.5. Bus Equipment

In addition to a specific Base Adapter set, a bus type must be chosen for initial simulation and
prototyping. For the single-bus based systems, this bus will provide all communication between adapter
components. Therefore, it is essential that the bus be able to support high transfer rates. Furthermore,
to fully realize a parallel system on a single bus the bus must be capable of supporting multiple bus
masters so that the various system components may operate in an independent fashion.

A bus that meets these requirements is the SPARC MBus. The MBus specification provides a 64-bit
multiplexed address and data bus capable of supporting multiple masters at fully synchronous clock

Chapter 2 - The Design of a Parallel Network Interface
speeds up to 40 MHz. The bus may address 64 gigabytes of memory and allows individual data transfers of up to 128 bytes. Larger amounts of data may be transferred by "locking" the bus to link several 128-byte transfers. The interface specification also makes provisions for cache coherency. A complete specification of the MBus is given in [SPAR91] and an overview of its operation is provided by [KITI91].

2.6. Summary

Low data latency, high data throughput, scalability, and low cost are all goals of a PLAN system. In this chapter, several basic interface structures were proposed to meet these goals in the implementation of a PLAN interface. These structures include single-bus, dual-bus, and bus-free designs. Chapters 3 through 5 investigate each of these interface options.
Chapter 3. Single-Bus Systems

A basic PLAN system design approach is investigated in Chapter 3. The single-bus system design relies upon a single system bus to provide a foundation for the various PLAN components.

3.1. Analysis of a Single-bus System

The single-bus design is based on a single system bus as shown in Figure 3-1. All components of the interface communicate by using this bus to access a shared memory.

Analysis of the data flow in this design provides insight into the individual component throughputs required to build this type of system. Since the flows of outgoing and incoming data are similar, only the outgoing flow is analyzed in detail. The analysis yields the data rate that must be supported by each of the system components.

This analysis assumes that data is distributed evenly across all parallel components such as the set of Protocol Processors and the set of Base Adapters. Such a distribution may be accomplished by requiring each component to process an equal number of packets or by splitting the data equally across the set of parallel components. For a thorough investigation of packet splitting options and their effects on system performance see [KUMA93]. In addition, it is assumed that data flow rates are constant, i.e. the system is in steady-state, during this analysis. Finally, it is assumed that a memory unit is available for any bus type chosen and that the unit supports data transfers at the full bus speed.
3.1.1. Variables Used in Analysis

The following variables are used in this analysis.

\[ D = \text{Number of requests per second produced by the AP} \]

\[ M = \text{Number of BAs in the system} \]

\[ N = \text{Number of PPs in the system} \]

\[ B_{R} = \text{Length of an AP Data Request in bits} \]

\[ B_{P} = \text{Length of a PP Data Request in bits} \]

\[ B_{I} = \text{Length of a PP Data Indication in bits} \]

\[ B_{T} = \text{Length of an ALP Data Indication in bits} \]

\[ B_{S} = \text{Length of a Data Set in bits} \]

\[ B_{H} = \text{Length of a Network Layer header in bits} \]
\( B_a \) = Length of an Access Layer header in bits

\( B_c \) = Length of a Coding Header in bits

\( T_H \) = Base Adaptor physical medium required throughput in bits per second.

\( R_{BUS} \) = System bus bandwidth in bits per second

\( R_{PPout} \) = Protocol Processor transmission bandwidth in bits per second

\( R_{ALPout} \) = Access Layer Processor transmission bandwidth in bits per second

\( R_{CODEout} \) = Coding Logic transmission bandwidth in bits per second

\( R_{BAout} \) = Base Adaptor transmission bandwidth in bits per second

\( I_{out} \) = The network transmission bandwidth supplied to the AP in bits per second

\( R_{PP} \) = Protocol Processor component bandwidth in bits per second

\( R_{ALP} \) = Access Layer Processor component bandwidth in bits per second

\( R_{CODE} \) = Coding Logic component bandwidth in bits per second

\( R_{BA} \) = Base Adaptor component bandwidth in bits per second

\( I \) = The network component bandwidth supplied to the AP in bits per second

The transmission bandwidth is defined as the bandwidth a component is required to supply for data transmission. The component bandwidth, in contrast, is the total bandwidth required of a component for transmission and reception combined.

Finally, \( S_{XY} \) represents the size in bits of a notification from component \( X \) to component \( Y \). For example, the AP uses a 64-bit data transfer to notify the PPs that a new data set is available to be sent, so \( S_{AP,PP} = 64 \) bits.
3.1.2. Analysis of a Basic Single-bus System

Assume that the Application Processor produces $D$ requests per second and that there are $N$ Protocol Processors and $M$ Base Adapters. Furthermore, assume all data sets produced by the AP are of equal length. Let the length of a data request be $B_r$ bits and the length of the accompanying data set be $B_s$ bits. Finally, let the length of a notification from the AP to a PP be $S_{AP,PP}$. The AP then writes $I_{out} = D(B_r + B_s + S_{AP,PP})$ bits per second (bps) to the Shared Memory Unit via the system bus.

The PPs read these requests at a rate of $DB_r$ bits per second and produce network headers and updated data requests for each data set. To accomplish this task in a steady-state situation, the full set of PPs must produce $D$ network layer headers per second and $D$ updated requests per second. Assuming an equal distribution of the work load, this requires that each of the $N$ PPs be able to produce headers and requests at a rate of $D/N$ per second. In addition, each PP signals the ALP after processing a data request. The headers and updated data requests produced by the PPs are written back to the Shared Memory Unit via the system bus and the ALP is signaled by this same bus. For headers of length $B_h$, updated data requests of length $B_d$, and an $S_{PP,ALP}$ bit notification, this implies that each PP requires $R_{PPout} = D(B_h + B_d + B_j + S_{PP,ALP})/N$ bits per second of the system bus bandwidth.

The Access Layer Processor reads the $D$ data requests produced by the PPs every second and produces corresponding access layer headers. For each request, it also signals both the Coding Unit and the Base Adapters. Let the size of an access layer header be $B_a$, the size of a Coding Unit notification be $S_{ALP-CU}$ and the size of a Base Adaptor signal be $S_{ALP-BA}$. The bus bandwidth required for packet transmission by the ALP is then $R_{ALPout} = D(B_h + B_d + S_{ALP-CU} + S_{ALP-BA})$ bits per second.
Coding of the packet information is done by reading the resulting data set along with its network and access layer headers from the shared memory and producing an additional coding header of length $B_C$ bits. If the coding scheme being used requires the full data set to be modified, as it does in situations where coding is done for encryption, the entire modified data set must be written back to the shared memory along with any header. In a steady-state situation, the coding set must be able to encode $D(B_r+B_H+B_S)$ bits per second, write back coding sets at $DB_C$ bps, and signal the ALP to indicate completion at $DS_{CU-ALP}$ bps. These tasks require a bus connection operating at $R_{CODEout}=D[2(B_r+B_H+B_S)+B_C+S_{CU-ALP}]$ bits per second.

Finally, the Base Adapters read the full, encoded, data set and its associated headers from the memory for transmission. This operation requires a bus connection operating at $D(B_r+B_H+B_S+B_C)$ bits per second for the full set of adapters. Individually, the adapters must absorb $R_{BAout}=D(B_r+B_H+B_S+B_C)/M$ bits per second for transmission.

A similar process takes place during the reception of data. The BAs write the data to the shared memory where it is read by the coding logic, decoded, and written back. The ALP reads the decoded access layer headers to determine the packet handling method and the PPs read the decoded network headers to generate a Data Indication for the AP. Finally, the data is read by the AP. Assuming half of the system bandwidth is consumed by reception and half by transmission, the data rates derived above may be doubled to produce the full system data rates given in Equations 3.1 to 3.3.

\[
I = 2I_{out} = 2D(B_r + B_S + S_{AP-PP}) \quad (3.1)
\]

\[
R_{PP} = 2R_{PPout} = 2D(B_r + B_{R2} + B_H + S_{PP-ALP}) / N \quad (3.2)
\]

\[
R_{ALP} = 2R_{ALPout} = 2D(B_{R2} + B_A + S_{ALP-CU} + S_{ALP-RA}) \quad (3.3)
\]
\[ R_{\text{CODE}} = 2R_{\text{CODEout}} = 2D(2(B_A + B_H + B_S) + B_C + S_{\text{CU-ALP}}) \]  
\[ R_{BA} = 2R_{\text{BAout}} = 2D(B_A + B_H + B_S + B_C) / M \]  
(3.4)  
(3.5)

Each active component attached to the system bus requires a portion of the bus bandwidth. Therefore, summing the individual component requirements given in Equations 3.1 to 3.5 results in the total required bus bandwidth.

\[ R_{\text{BUS}} = 1 + NR_{PP} + R_{\text{ALP}} + R_{\text{CODE}} + MR_{BA} \]

\[ = 2D(B_R + B_S + S_{\text{ALP-PP}}) + 2D(B_R + B_H + B_S + S_{\text{FP-ALP}}) + 2D(B_R + B_A + S_{\text{ALP-CU}} + S_{\text{ALP-BA}}) + 2D(B_A + B_H + B_C + S_{\text{CU-ALP}}) + 2D(B_A + B_S + B_H + B_C) \]

\[ R_{\text{BUS}} = 2D(2B_R + 2B_{R2} + 4B_A + 4B_H + 4B_S + 2B_C + S_{\text{ALP-PP}} + S_{\text{FP-ALP}} + S_{\text{ALP-CU}} + S_{\text{ALP-BA}}) \]  
(3.6)

Since all communication between active bus components takes place via the Shared Memory Unit, the memory unit must support the full system bus bandwidth. Decomposing the Shared Memory Unit into parallel components can reduce the bandwidth that must be supported by any single memory device. Interleaving the data among \( S \) individual memory units reduces the bandwidth that must be supported by a single memory device by a factor of \( S \). However, this type of scheme also requires a memory controller capable of processing memory requests at the full bus data rate and interleaving the data among the individual memory units. In this research, it is assumed that a single memory unit is available that supports the full bandwidth of the chosen system bus.

Chapter 3 - Single-Bus Systems
3.1.3. *Analysis of a Modified Single-bus System*

The analysis presented in Section 3.1.2 demonstrates that the main system data transfer bottleneck is the bus since it must provide throughput equal to the sum of all active system component throughputs. One change that may be made to eliminate some system bus traffic is to incorporate the coding logic and the Base Adaptor set into a single unit. This modified system is shown in Figure 3-2.

![Diagram of modified single-bus based system](Image)

**Figure 3-2.** Modified single-bus based system.

This modified single-bus design eliminates the need to use the bus when transferring data between the Base Adaptor and the Coding Logic. Since data must normally flow between the coding logic and the Base Adapters without intervening operations, this modification does not affect system functionality. It does, however, eliminate the bus use associated with several unnecessary inter-component notifications as well as two unneeded transfers to the shared memory.

The data rates that must be supported by the majority of system components remain unchanged in the modified single-bus design. The components that do need to support different data rates are the system...
bus and the Shared Memory Unit. In addition, the ALP notification overhead is reduced. The new bus and Shared Memory Unit data rates are equal to the sum of the data rates of the active components attached to the bus.

\[
R_{BUS} = 1 + NR_{PP} + R_{ALP} + R_{CODE} = 2D(B_R + B_S + S_{ALP,PP}) + 2D(B_R + B_H + S_{PP,ALP}) + 2D(B_R + B_A + S_{ALP,CU}) + 2D(B_A + B_S + B_H)
\]

\[
R_{BUS} = 2D(2B_R + 2B_R + 2B_A + 2B_H + 2B_S + S_{ALP,PP} + S_{PP,ALP} + S_{ALP,CU}) \quad (3.7)
\]

Another significant change is evident in Equation 3.7. The Coding Unit now requires a lower data rate bus connection. However, the unit must still provide the bandwidth calculated in Section 3.1.2. The ports connected to the BAs consume the remainder of the component bandwidth.

In the case of low data loss rates or non-critical data transfers, it may be desirable to eliminate the Coding Logic functionality. In these cases, the Access Layer Processor (ALP) turns the logic off and data flows along a multiplexed bypass path. Turning off the coding logic in the basic single-bus design reduces the system bandwidth requirements to that given in Equation 3.7. Bypassing the coding logic has no effect on the bandwidth consumed by any system port in the modified single-bus design.

3.1.4. Analysis Results

The results of the analysis presented in Sections 3.1.2 and 3.1.3 are summarized in Table 3-1.
<table>
<thead>
<tr>
<th></th>
<th>Basic Single-bus</th>
<th>Modified Single-bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>$2D( B_R + B_S + S_{AP,FP} )$</td>
<td>$2D( B_R + B_S + S_{AP,FP} )$</td>
</tr>
<tr>
<td>$R_{PP}$</td>
<td>$2D( B_R + B_{R2} + B_H + S_{PP,AP} ) / N$</td>
<td>$2D( B_R + B_{R2} + B_H + S_{PP,AP} ) / N$</td>
</tr>
<tr>
<td>$R_{ALP}$</td>
<td>$2D( B_{R2} + B_A + S_{ALP,CU} + S_{ALP,BA} )$</td>
<td>$2D( B_{R2} + B_A + S_{ALP,CU} )$</td>
</tr>
<tr>
<td>$R_{CODE}$</td>
<td>$2D[ \ 2( B_A + B_H + B_S ) + B_C $ $+ S_{CU,ALP} ]$</td>
<td>$2D[ \ 2( B_A + B_H + B_S ) + B_C $ $+ S_{CU,BA} ]$</td>
</tr>
<tr>
<td>$R_{BA}$</td>
<td>$2D( B_A + B_H + B_S + B_C ) / M$</td>
<td>$2D( B_A + B_H + B_S + B_C ) / M$</td>
</tr>
<tr>
<td>$R_{BUS}$</td>
<td>$2D( 2B_R + 2B_{R2} + 4B_A + 4B_H + 4B_S $ $+ 2B_C $ $+ S_{BP,PP} + S_{PP,ALP} + 2S_{ALP,CU} $ $+ S_{ALP,BA} )$</td>
<td>$2D( 2B_R + 2B_{R2} + 2B_A + 2B_H + 2B_S $ $+ S_{AP,PP} + S_{PP,ALP} + S_{ALP,CU} )$</td>
</tr>
</tbody>
</table>

The equations in Table 3.1 can be simplified by assuming that the number of bits sent in a single transmission is much larger than the number of bits in any request, indication, or header that accompanies that transmission. If it is also assumed inter-processor notifications are much smaller than any requests, indications, or headers, then $B_S$ becomes the dominant term in the equations, allowing them to be simplified to the equations given in Table 3-2.

These simplified equations show that the modified single-bus design reduces the required system bus bandwidth by approximately 50 percent. It should be noted, however, that these savings are based on the fact that the Coding Unit writes the entire data set back to the shared memory after encoding it. The other coding extreme is the case in which the Coding Unit simply writes back a small data set such as a CRC or parity byte. In this case, the Coding Unit requires a bus interface operating at only $D(B_A+B_H+B_S+B_C+S_{CU,ALP})$, or $D(B_A+B_H+B_S)$ for the modified single-bus system. While not affecting the

Chapter 3 - Single-Bus Systems
Table 3-2. Simplified Component Bandwidth Equations (bps)

<table>
<thead>
<tr>
<th></th>
<th>Basic Single-bus</th>
<th>Modified Single-bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>( I )</td>
<td>( 2DB_S )</td>
<td>( 2DB_S )</td>
</tr>
<tr>
<td>( R_{PP} )</td>
<td>( 2D( B_R + B_H )/N )</td>
<td>( 2D( B_R + B_H )/N )</td>
</tr>
<tr>
<td>( R_{ALF} )</td>
<td>( 2D( B_R + B_A ) )</td>
<td>( 2D( B_R + B_A ) )</td>
</tr>
<tr>
<td>( R_{CODE} )</td>
<td>( 4DB_S )</td>
<td>( 4DB_S )</td>
</tr>
<tr>
<td>( R_{EA} )</td>
<td>( 2DB_S / M )</td>
<td>( 2DB_S / M )</td>
</tr>
<tr>
<td>( R_{BUS} )</td>
<td>( 8DB_S )</td>
<td>( 4DB_S )</td>
</tr>
</tbody>
</table>

bus bandwidth equation for the modified design, this does reduce the required bus bandwidth in the basic single-bus system to

\[
R_{BUS} = 2D(2B_R+2B_R+3B_A+3B_H+3B_S+2B_C+S_{ALF,PP}+S_{PP,ALF}+S_{ALF,CI}+S_{ALF,EA}).
\] (3.8)

Simplifying this equation to \(6DB_S\) and comparing it to the modified single-bus equation given in Table 3-1, a 33 percent reduction in the system bus bandwidth requirement is seen. Therefore, the modified single-bus system results in a reduction of the required system bus bandwidth in the range of 33 percent to 50 percent beyond the basic single-bus system.

As an example of the component bandwidths required to support the single-bus system, consider a system with an AP that requires an \( I = 1 \) Gbps network interface. Let the size of a data request be \( B_R = 208 \) bits as specified in [KUMA93]. Assume all inter-component signals require two 64-bit bus transfers,
or are 128 bits. Furthermore, assume all data packets have a length of \( B_s = 35 \) Kbits. If the interface bandwidth is equally distributed between transmission and reception, Equation 3.1 leads to

\[
D = \frac{I}{2(B_r + B_s + S_{ap-data})} = \frac{1 \text{Gbps}}{2(208 + 35000 + 128) \text{bits/sec}} \approx 14.1 \frac{K \text{req}}{\text{sec}}.
\]

Assume the system is supported by \( M = 10 \) BAs and \( N = 5 \) PPs, where the PPs add TCP and IP network headers of 160 bits each [STEV94], or that \( B_H = 320 \) bits. Furthermore, let the ALP add an additional header of length \( B_A = 272 \) bits as specified in [KUMA93]. Let the size of a PP to ALP data request be \( B_{r2} = 256 \) bits. Finally, let the Coding Unit supply a header \( B_C = B_s/M = 35 \) Kbits/10 = 3.5 Kbits without requiring the data to be written back to the shared memory. For a basic single-bus system, the required component bandwidths are then,

\[
R_{pp} = \frac{2D(B_r + B_{r2} + B_H + S_{pp-alp})}{N} = \frac{2(14.2 \frac{K \text{req}}{\text{sec}})(208 + 256 + 320 + 128) \text{bits/sec}}{5} \approx 5.2 \frac{M \text{bits}}{\text{sec}},
\]

\[
R_{alp} = 2D(B_{r2} + B_A + S_{alp-cu} + S_{alp-ba}) = 2(14.2 \frac{K \text{req}}{\text{sec}})(256 + 272 + 2 \times 128) \frac{\text{bits}}{\text{sec}} \approx 22 \frac{M \text{bits}}{\text{sec}},
\]

\[
R_{code} = 2D(B_A + B_s + B_H + B_C + S_{CU-alp}) = 2(14.2 \frac{K \text{req}}{\text{sec}})(272 + 35K + 320 + 3.5K + 128) \frac{\text{bits}}{\text{sec}} \approx 1.1 \frac{G \text{bits}}{\text{sec}}.
\]
\[ R_{BA} = 2D \left( \frac{B_A + B_S + B_H + B_C}{M} \right) = 2 \left( 14.2K \frac{\text{req}}{\text{sec}} \right) \left( \frac{272 + 35K + 320 + 3.5K \text{ bits}}{10 \text{ req}} \right) \approx 110 \frac{\text{Mbits}}{\text{sec}} \]

\[ R_{BUS} = 2D(2B_K + 2B_{x2} + 3B_S + 3B_H + 3B_{AP} + 2B_C + S_{AP-PP} + S_{AP-ALP} + 2S_{ALP-CU} + S_{ALP-BA}) \]

\[ = 2 \left( 14.2K \frac{\text{req}}{\text{sec}} \right)(2 \times 208 + 2 \times 256 + 3 \times 35K + 3 \times 320 + 3 \times 272 + 2 \times 3.5K + 5 \times 128) \approx 3.3 \frac{\text{Gbits}}{\text{sec}} \]

The throughput required of the Base Adapters is similar to that provided by existing FDDI network hardware. However, assuming a 64-bit bus and that one transfer is allowed per clock cycle, the bus must have a clock speed of 52 MHz. This is significantly faster than the 40 MHz clock specified by the 64-bit SPARC MBus [SPAR91].

3.2. Functional Modeling of a Single-bus System

A Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) model of the single-bus system provides insight into the system's design and demonstrates the component utilization and data transfer rates and characteristics that might be expected in an actual PLAN system. One such model is developed here.

The SPARC MBus [SPAR91] was chosen as the basis for this design, as described in Section 2.5. This bus provides a 64-bit data path and 32-bit addressing in addition to supporting up to 16 addressable bus modules. The MBus allows for a system consisting of a single Application Processor (AP), a single Access Layer Processor (ALP), a single Coding Unit (CU), and an arbitrary combination of Protocol
Processors (PPs), Base Adapters (BAs), and Shared Memory Units (SMUs) not to exceed a total of 13.

The system modeled here also includes a single Protocol Processor Interrupt Controller (PPIC) that controls access to the PPs. Bus access is controlled by a single bus arbiter. For this simulation, three PPs and four BAs were added to the bus along with a single 32-bit addressable SMU.

3.2.1. MBus Model

The basic MBus consists of a collection of signals as listed in Table 3-3. These signals connect all MBus modules and provide system communication. Several of the listed signals are not included in the VHDL model, however. In an effort to reduce the model's computational overhead, many signals relating to error conditions are not considered. These signals include MRTY	extsuperscript{*}, MERR	extsuperscript{*}, and AERR	extsuperscript{*}, and INTOUT. The functionality of the signals IRL[3:0] and ID[3:0] is hardcoded into the model, so they are not explicitly included either. In addition to the inter-module signals, the MBus requires a clock module and an arbiter to provide full functionality.

The MBus is clocked by a single 40 MHz clock signal on the MCLK line.\textsuperscript{1} This clock is provided in the model by a 50 percent duty cycle oscillator with an adjustable period.

Access to the MBus is controlled by a single arbiter. This arbiter asynchronously grants use of the bus to masters in a round-robin fashion. While the MBus specification does not require that grants be issued using a round-robin protocol, this choice provides all masters with equal bus access in the PLAN system.

Bus grants issued to masters by the arbiter are considered pending until the masters accept them by

---

\textsuperscript{1} The MBus clock period is kept at 25 ns while performing these simulations. By providing an adjustable clock in the VHDL model, however, the effects of the bus clock speed on system performance may be studied in the future.

Chapter 3 - Single-Bus Systems
Table 3-3. MBus Physical Signals [SPAR91]

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Signal Description</th>
<th>Line Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCLK</td>
<td>MBus Clock</td>
<td>Dedicated</td>
</tr>
<tr>
<td>MAD[63:0]</td>
<td>Address/Control/Data</td>
<td>Bussed</td>
</tr>
<tr>
<td>MAS'</td>
<td>Address Strobe</td>
<td>Bussed</td>
</tr>
<tr>
<td>MRDY'</td>
<td>Data Ready Indicator</td>
<td>Bussed</td>
</tr>
<tr>
<td>MRTY'</td>
<td>Transaction Retry Indicator</td>
<td>Bussed</td>
</tr>
<tr>
<td>MERR'</td>
<td>Error Indicator</td>
<td>Bussed</td>
</tr>
<tr>
<td>MBR'</td>
<td>Bus Request</td>
<td>Dedicated</td>
</tr>
<tr>
<td>MBG'</td>
<td>Bus Grant</td>
<td>Dedicated</td>
</tr>
<tr>
<td>MBB'</td>
<td>Bus Busy Indicator</td>
<td>Bussed</td>
</tr>
<tr>
<td>IRL[3:0]</td>
<td>Interrupt Level</td>
<td>Dedicated</td>
</tr>
<tr>
<td>ID[5:0]</td>
<td>Module Identifier</td>
<td>Dedicated</td>
</tr>
<tr>
<td>AERR'</td>
<td>Asynchronous Error Out</td>
<td>Bussed</td>
</tr>
<tr>
<td>RSTIN'</td>
<td>Module Reset In Signal</td>
<td>Implementation Dependent</td>
</tr>
<tr>
<td>INTOUT'</td>
<td>Interrupt Out</td>
<td>Dedicated</td>
</tr>
</tbody>
</table>

deasserting the bus busy signal. When a grant is accepted, the requesting bus master removes its request and the next requesting master is issued a pending grant. This process is summarized in Figure 3-3.

Once a master is granted bus access, it asserts (lowers) the bus busy signal (MBB') to signal the arbiter that the bus has been accepted and to indicate that the bus is in use. After asserting the bus busy signal, the controlling master performs any desired bus transfers in a locked mode. This means that the controlling master may retain the bus as long as desired. When the master is done with the bus, it deasserts (raises) the bus busy signal. If another master has a pending grant, it may accept the grant by lowering the bus busy signal after one cycle of the bus clock. This arbitration scheme provides

Chapter 3 - Single-Bus Systems
maximum bus availability with a minimum of overhead. The general VHDL procedure for bus access is shown in Figure 3-4.

```
-- Get Bus
nMBR <= '0';
if nMBG /= '0' then
  wait on nMBG until nMBG = '0';
end if;
if nMBR = '1' then
  wait until nMBR = '0';
end if;
if nMBB = '0' then
  wait on MCLK until MCLK = '1';
end if;
```

--- request
--- wait for bus grant
--- drop bus request
--- wait for prior master to finish
--- indicate bus in use
--- put data on at rising bus clock

Figure 3-4. Typical bus request.

A typical bus transaction is done in two phases. The first phase is addressing. During this phase, the current bus master puts an addressable module’s ID on the bus along with the size and type of transfer desired. The size may range from one byte to 128 bytes and the transfer type may be read or write. Also included in the address information is the physical address that the ensuing data will be read from or written to. The second phase of a bus transaction is the data phase. During the data phase, data is
put on the bus in 64-bit units, separated by rising edges of the MBus clock, until the full transfer is finished. Basic VHDL models for the read and write procedures are shown in Figures 3-5 and 3-6, respectively.

```
-- Read Data from Memory
MAD_ID := MEM_ID;
MAD_TYPE := 'X'1';
MAD_SIZE := "100";
MAD_PHYADR := X'0' & HLoc;
MAD <= Drive(MAD_ID & MADUNUSED & MAD_SIZE & MAD_TYPE & MAD_PHYADR);
nMRDY <= 'Z';
-- Tristate data ready line
nMAS <= '1';
wait on MCLK until MCLK = '1';
nMAS <= '0';
wait on MCLK until MCLK = '1';
nMAS <= '1';
MAD <= MAD_TRI;
-- Tristate outputs after 1 cycle
wait on nMRDY until nMRDY = '0';
-- Wait for data ready
wait on MCLK until MCLK = '0';
Data(127 downto 64) := Sense(MAD,0,0);
-- Read 1st set
wait on MCLK until MCLK = '0';
Data(63 downto 0) := Sense(MAD,0,0);
-- Read 2nd set
wait on nMRDY until nMRDY = '1';
-- Wait for memory read to finish
```

**Figure 3-5.** Typical MBus read transaction.

```
-- Write Data to ALP
MAD_ID := ALP_ID;
MAD_TYPE := "001";
MAD_SIZE := "010";
MAD_PHYADR := X'0000000002';
MAD <= Drive(MAD_ID & MADUNUSED & MAD_SIZE & MAD_TYPE & MAD_PHYADR);
nMAS <= '0';
-- Strobe address
wait on MCLK until MCLK = '1';
nMAS <= '1';
wait on MCLK until MCLK = '1';
MAD <= Drive(X'00000000000000000000000000000000');
-- Data set
nMRDY <= '0';
-- Strobe data ready
wait on MCLK until MCLK = '1';
nMRDY <= '1';

```

**Figure 3-6.** Typical MBus write transaction.
3.2.2. Application Processor Model

The Application Processor (AP) is the device that is served by the PLAN system. In the VHDL model, it acts as a data source and as a data sink. These functions are governed within the model by three internal clocks.

The first clock determines the overall AP system speed. An internal counter is updated on each rising edge of this clock to indicate the number of bytes that must be handled to remain at the system speed. This counter is decremented after sending data by the amount of data sent, and after receiving data by the amount of data received. In this way, the AP is held to a known steady data rate.

Two additional clocks determine the frequency at which data availability checks are made. One clock controls the frequency at which the sending process checks the bit counter to determine whether data is available for transmission. The other clock controls the frequency at which the receive process checks the system memory for incoming data availability. Since the receive data availability check must be done via the MBus, the check frequency influences system performance even when no incoming data is available.

Figures 3-7 and 3-8 depict the general operation of the AP model. Data transmission is initiated by sufficient data conditions as described above. When these conditions are met, the AP reads the data request address and data address from the memory. It then writes a data set and a new data request to memory and updates the associated queue pointers. Finally, it signals the Protocol Processors by writing a control word to the Protocol Processor Interrupt Controller.

Chapter 3 - Single-Bus Systems
Data reception is initiated by the internal receive clock. After each rising edge of this clock, the AP reads data indication (DI2) front and back pointers from the Shared Memory Unit. If these pointers
differ, then data is available for reception, and the front indication is read. An incoming data set is read based upon the information in this DI. After reading the data, the front DI pointer is incremented.

3.2.3. Protocol Processor Model

The function of a Protocol Processor (PP) is to construct and handle transport and network protocol headers. In the PLAN system modeled, these headers are TCP and IP packet headers.

A single Protocol Processor Interrupt Controller (PPIC) is used as a signaling point for the full PP array. This single address reduces the amount of PLAN system information that must be known by the AP. It also allows all requests to be handled in a first-in-first-out (FIFO) manner by assigning each one to the next available processor. The operation of the PPIC is depicted in Figure 3-9.

![Diagram of PPIC state transitions]

**Figure 3-9.** PPIC state diagram.
Upon receiving a DR signal from the AP or a DI signal from the ALP, the PPIC increments an internal counter for the type of request. It then passes a request token along a 3-bit daisy chain to the PPs. Each PP monitors the chain, and either accepts the request by changing the request token to an accept token or declines by passing the request to the next processor. This scheme does not result in an even load among the processors, rather it more heavily utilizes the first processors in the chain. The scheme does, however, guarantee that no request is assigned to a processor that is busy and that all requests are handled as quickly as possible.

The difference between the AP system speed and the speed of an individual PP determines the number of PPs that are needed in any given implementation. If too many PPs are included in the implementation, the last ones on the daisy chain are not utilized. If too few PPs are used, all have excessively high utilization levels.

Each PP model consists of one process that monitors the PPIC daisy chain, and another process that handles the requests and indications. A PP that is not busy accepts a request from the PPIC and switches into a busy mode. Any subsequent request made while the PP is busy is passed on to the next PP in the chain. If no PP accepts a request, then it is passed back to the PPIC for reissue.

Upon acceptance of a DR, the PP reads the request from shared memory and uses it to construct TCP and IP headers. These headers are then written to the shared memory along with a new data request (DR2) and the Access Layer Processor is signaled.

Acceptance of a DI results in a similar process. The PP reads the DI and strips the TCP/IP headers from the incoming data before writing a new data indication (DI2) to be read by the AP. In normal operation, the TCP/IP headers contain information that affects the handling of a given packet. For the purposes of

Chapter 3 - Single-Bus Systems
this simulation, however, the contents of the headers are ignored. Data request and indication handling is described in Figures 3-10 and 3-11.

![Diagram](image.png)

Figure 3-10. PP send state diagram.

3.2.4. Access Layer Processor Model

The Access Layer Processor (ALP) accepts data requests from the Protocol Processors and data indications from the Base Adapters. It then determines whether coding needs to be done on the data and
passes the data on the next stage. All request and indication signals are handled by an independent ALP process, so that none are lost while the primary ALP process is busy.

The main ALP process handles data requests and indications as shown in Figures 3-12 and 3-13. Upon receiving a data request signal, the ALP reads the DR2 from memory and builds an access layer header (ALH). This header is written to the shared memory and, if coding is enabled, the Coding Unit is signaled. While waiting for the Coding Unit to finish, the ALP determines the source and destination MAC addresses. If coding is not enabled, this address lookup results in additional delay. When the encoding of the outgoing packet is complete, the ALP scans the Base Adapters for an available adaptor. It then signals this adaptor by sending it the source and destination MAC addresses for the outgoing packet and the addresses of the data and headers to be included in the packet. Finally, the ALP waits for the chosen Base Adaptor to read the packet information from memory before returning to an idle state.
This method of handling outgoing packets may be modified by splitting any given outgoing packet evenly across all of the Base Adapters. The model developed does not, however, support packet splitting. For a more complete investigation of the effects of packet splitting on PLAN performance, see [KUMA93].

Chapter 3 - Single-Bus Systems
A disadvantage of the packet handling method used here is that it results in a serialized operation of the ALP, Coding Unit, and chosen Base Adaptor from the time that a data request is received by the ALP to the time that the packet information is read by the Base Adaptor. This may have an adverse effect on overall system performance and could result in the need for larger outgoing data and protocol header queues within the system's Shared Memory Unit.

The system might be improved by including local queue pointers in each of the ALP, CU, and BA components. The primary reason for the serial operation is that the ALP must wait for the BA to update the data and header pointers before reading the next queue item. These pointers in shared memory keep the AP and PPs from writing new data over the unread data and, therefore, cannot be updated before the BA reads the packet data. If the ALP maintains local pointers to the current data for which it is building a header, however, it can pass this pointer to the Coding Unit rather than requiring that unit to read the pointers from memory. This allows both the Coding Unit and ALP to work ahead of the current outgoing packet. In this new scheme, the Base Adaptor is still responsible for updating the queue front pointers in the shared memory.

The procedure for handling a data indication is similar to that for a data request. Upon receiving an indication signal from one of the Base Adapters, the ALP signals the Coding Unit, if required, to decode the incoming data. After the data has been decoded, the ALP strips the AL header information and uses it to build a data indication. It then signals the PPIC and returns to an idle state. Due to the intrinsically serial nature of this operation, it does not significantly benefit from the use of local queue pointers as described for the data transmission process above.
3.2.5. "Coding Unit Model"

The Coding Unit (CU) encodes outgoing packets prior to sending them to a Base Adapter and decodes incoming packets prior to handling the AL header information. A general Coding Unit might be used to encode (encrypt) all data in a packet for security, add a separate coding set for reliability, or both. Each choice impacts system performance, as it changes the amount of data that must be transferred on the system bus. The Coding Unit included in this model uses the second option and builds a separate coding set without modifying the existing packet data. The model also assumes that coding set generation and checking each take a fixed amount of time for any size packet. This assumption reflects the behavior of a purely combinational hardware encoder and decoder.

A data request means that the Coding Unit must read the packet's data and headers and generate a coding set for inclusion in the transmitted packet. This coding set is written back to the Shared Memory Unit prior to signalling the ALP.

A data indication means the Coding Unit must read the full received packet and compare its attached coding set to a newly generated set. If the two coding sets do not match, the Coding Unit must either correct the incoming data and write the corrected data back to the shared memory or signal a data error. Since no error conditions are generated during this simulation, the CU does not require additional bus access time to write data back to the shared memory in response to data indications. Figures 3-14 and 3-15 depict coding of outgoing packets and decoding of incoming packets.
Figure 3-14. CU send state diagram.

The Coding Unit may be turned on or off for any given simulation. When it is on, the system acts as described above. When the Coding Unit is off, the ALP does not signal the CU to encode or decode data.

System performance attained when the Coding Unit is off is representative of the performance of a modified single-bus system. In a modified single-bus system, coding logic is at the head of each Base Adaptor’s data pipe. This means that the ALP does not need to signal the CU separately, and no separate access to the data is required by any coding logic. The modified single-bus system structure does add additional delay to each Base Adaptor and thus increases system latency over that of a system with the Coding Unit turned off.

Chapter 3 - Single-Bus Systems
3.2.6. *Base Adaptor Model*

The Base Adapters (BAs) drive the links between stations on a PLAN system. As described in Section 2.4, the Base Adaptor equipment chosen for this model is based upon the FDDI standard. The adapters used here are developed around functional models of the Advanced Micro Devices SUPERNET 2 chip set [AMD92]. This chip set consists of four components: the AM79830 fiber-optic media access controller (FORMAC), AM79864 physical layer controller (PLC), AM79865 physical data transmitter (PDT), and AM79866 physical data receiver (PDR). Models of the PLC, PDT, and PDR are incorporated into each Base Adaptor. The adaptor itself provides the functionality of the FORMAC, which is not explicitly modeled.
An individual BA consists of four processes. One process waits for a data request signal from the ALP, then reads a packet for transmission, frames it for FDDI transmission, and passes it down to the PLC. The PLC then uses a separate process to transmit the packet while the BA begins to process the next outgoing packet. These processes are depicted in Figure 3-16. The dashed line indicates the point where the separate PLC process is invoked. The third and fourth processes are used for data reception. As shown in Figure 3-17, the third process accepts FDDI symbols from the PLC and places incoming packets in a local receive buffer. The fourth process independently reads packets from the receive buffer and moves them into the system’s Shared Memory Unit. Again, the separation of these two processes is indicated by a dashed line.

![BA send state diagram](image)

**Figure 3-16.** BA send state diagram.

Upon receiving a data request, the Base Adaptor requests the system bus and reads a data set, its headers, and its coding set, if applicable, from the Shared Memory Unit. It then releases the bus while it sends the packet to its PLC. A packet for transmission must be framed by start delimiter, source address,
destination address, frame check sequence, end delimiter, and frame status fields. The Base Adaptor provides this framing while sending the packet to the PLC.

Packet reception is initiated when a packet start delimiter is detected on the data stream received at the PLC. The destination address on each incoming packet is compared to the local address. If there is a match, then the packet is assembled and written to the local receive buffer. If there is no address match, the packet is sent to the next station on the ring.

Data that has been placed in the receive buffer is read by the third process and written to the system SMU. After writing the data, the process signals the ALP with a data indication (DI2).
If a FORMAC were included in this model, it would provide all packet framing and communication with the PLC. The BA would communicate with it via send and receive packet buffers.

3.3. Simulation Results

The VHDL model described in Section 3.2 is used to simulate PLAN system operation under various load conditions. All simulations are performed in a system that consists of two network nodes, each with a single one megabyte Shared Memory Unit, three Protocol Processors, and four Base Adapters. The simulation factors varied are data size and Coding Unit use (on or off). The performance measures studied are component and system latency and system throughput.

The following assumptions are used in all of the simulations performed.

- All packets in an individual simulation are the same size.
- The Application Processor operates at 500 Mbps.
- The Application Processor is pre-loaded with one data set.
- The Base Adapters have no output buffering enabled.

The assumption that all packets within a simulation are the same size is made so that the effect of packet size on system performance can be studied. The assumptions for AP operating speed and preloading are made to create an environment in which the system operates at its full capabilities for as much of the total simulation time as possible. Finally, the elimination of output buffering on the Base Adapters creates a steady-state environment similar to that found when any BA output buffers are full.
In addition to the above assumptions, it is assumed that one node (the sending node) retains all network tokens for the duration of the simulation. The other node (the receiving node) never has any tokens and is purely a receiver for the duration of the simulations. The AP of the sending node generates bus traffic both by sending packets and by checking the receive queue. Since this node has no incoming data, a full received data set is never read. The combination of these two activities is done at 500 Mbps. The AP of the receiving node only checks for received data; it does not generate any outgoing traffic. Therefore, the receiving node's data rate is determined by the sending node.

The code set size is held to 1 Kbit for this set of simulations. A constant code set size is common for CRC or checksum codes. This choice for the coding size means that the code set has a greater impact on system throughput when using small data sets than it does with larger sets.

The sizes of the data sets are varied in units of 1 kbit from 1 to 30 (128 bytes to 3,840 bytes). This upper limit is determined by the fact that a single FDDI packet has a maximum length of 9,000 symbols [MCC087]. Of these, 38 are used for framing when using 48-bit source and destination addresses. The information field is, therefore, 8,962 4B/5B encoded symbols or 35,848 bits. The TCP and IP headers require 320 bits of this space [STEV94] and a minimal AL header requires 272 bits [KUMA93]. This leaves 35,256 bits or 4,407 bytes. These bits must be divided between data and the code set.

The following delays and clock speeds are used in all simulations.

- PP header build time = 50 ns
- PP header strip time = 50 ns
- CU decode times = 50 ns
- CU decode time = 50 ns
• ALP MAC table lookup delay = 1 ns
• Fiber propagation delay = 1 ns
• AP send clock period = 40 ns
• AP receive clock period = 200 ns

The PP delay is chosen under the assumption that a PP is able to form TCP and IP headers within two Mbus clock cycles. The constant CU delay represents the response of an encoder and decoder constructed of purely combinational logic. The ALP and Fiber delays are simply intended to synchronize system operations. All components attached to the Mbus are limited by the Mbus speed, and it is assumed that the ALP is able to form an AL header inside of a single bus cycle. In optical fiber, an information propagation delay of 1 ns is representative of a short (less than 0.5 meter) connection. Finally, the AP send and receive clock periods are chosen to allow the AP to send data approximately every other Mbus clock cycle, and to check for received data every eight Mbus clock cycles.

Tables 3-4 to 3-17 indicate the utilization of the model components under conditions ranging from small to large data sets with the Coding Unit turned on and off for each packet size. The packet size parameter indicates the amount of data generated by the AP for inclusion in each packet. The Coding Unit parameter indicates the state of the Coding Unit.

Utilization results are based on data for 25 to 30 different packets. Utilization values are rounded to the nearest whole number if greater than one. Otherwise they are rounded to the nearest tenth. A utilization of 0.0 percent indicates that the component spends some small fraction of the simulation time (less than 0.05 percent) in this state. A utilization that is given as a dash (-) indicates the component spends no time in the state.
The tables indicate the percent of time spent in each of three states by system components. A component in an "idle" state is waiting for new packet data to become available. In the "working" state, the component is processing a packet. Finally, while in the "waiting" state, a component is waiting to use system bus.

Table 3-4 includes separate results for the two primary processes, sending and receiving, within the sending AP. While these processes do share a single bus interface, they generally operate in parallel and some of their activities may overlap. The "waiting" results reflect the use of a shared bus port and include any time spent waiting for the other process to finish using the port. In contrast to the sending AP, the results for the receiving node's AP, given in Table 3-17, are only for the receive process since sending is disabled for this component.

Table 3-5 includes utilization results for two of the three Protocol Processors attached to the sending node. The third processor is never utilized. In this table, PP1 represents the first processor on the PP daisy chain and PP2 represents the second processor on the chain. The receiving node only utilizes a single PP, so Table 3-14 contains utilization results for the first PP of the receiver.

Since the Coding Unit is not used at all when it is off, Tables 3-6 and 3-11 only include results for cases in which the Coding Unit is employed.

The results for the sending node's Access Layer Processor, shown in Table 3-7, include two additional states beyond idle, working, and waiting states for the system bus. There is a state that indicates the amount of time spent searching for an available Base Adaptor. This state is entered when all of the adapters are busy and the ALP is polling to find the first one available. Also included is a state that indicates either time spent waiting for the CU to finish coding a packet or time spent waiting for a BA.

Chapter 3 - Single-Bus Systems
to read a packet's information from the SMU. Time spent in this state may be reduced by using separate pointer queues as discussed in Section 3.2.4.

The receiving node's ALP does not explicitly wait for the CU or BAs. Instead, it is first notified of an available incoming packet by the CU. Table 3-13 does not, therefore, contain the additional wait states of Table 3-7.

Table 3-8 contains results for a single Base Adaptor in the sending node. The round-robin data assignment scheme used by the ALP ensures that all BAs are evenly loaded. The results given here are for the second BA and are reflective of the utilization of any single Base Adaptor in the set.

Table 3-12 lists the utilization results for the receiving node's Base Adapters. As is the case with the sending node, these results are for the second BA and are representative of any single adaptor. This table, however, includes two sets of data. The first value is the utilization of the physical media that connects the stations, which, in this case, corresponds to the fiber used to connect the Base Adapters of the two stations. The second value is the utilization of the Base Adaptor's system bus connection.

The MBus utilization results given in Tables 3-10 and 3-16 indicate that both nodes' buses are in use at all times. This continuous use is partly due to the fact that both APs check their respective received data queue every 200 ns. This continual check is a significant contributor to the full use of the bus.

Tables 3-9 and 3-15 give the utilization of the sending and receiving nodes' Shared Memory Units. These results are representative of the amount of time spent transferring data sets and headers to and from the SMUs. Since no component transfers either data or headers directly to another component, the

Chapter 3 - Single-Bus Systems
difference between each node’s SMU and MBus utilization represents additional overhead required for component notifications.

Throughput and latency are calculated using the results for the 25 packets following the first packet. These results are shown in Table 3-18.

The system latency values shown in Table 3-18 are the time between when the last bit of data enters one of the sending node’s PPs and when the last bit of the same data leaves the receiving node’s PPs. This latency consists of two components. The send latency is the time from when the last bit of a data set enters the sending node’s PP set until the last bit of the same packet is placed on the fiber for transmission. The receive latency is the time from when the last bit of a packet is read into a receiving node Base Adaptor until the last bit of that packet’s data is written to the SMU by the receiving node’s PP set.

The packet latency values that were averaged to produce the results in Table 3-18 had large variances. This is due to the fact that the AP is overloading the system to produce steady throughputs. Since the system is overloaded, packets accumulate in the SMU, and each has a slightly longer wait than the one preceding it. This wait produces an increasing packet latency. Therefore, the latency results are for peak load conditions, rather than for "typical" loads.

3.4. Summary

The results given in Tables 3-4 to 3-18 indicate the system does not perform well with a data set size of 128 bytes. This is to be expected since a data size of 128 bytes is on the same order as the inter-
processor notification size and is smaller than most of the accompanying headers. This case does, however, demonstrate the impact of notification overhead more clearly than the cases in which the signal overhead is masked by the size of the data being transferred.

The only components that appear to be heavily loaded during system operation are the buses and the Base Adapters. While the BA load may be reduced by adding more Base Adapters, the bus load requires a new approach to system design. The next logical step toward reducing bus load is to split the bus into two independent buses as described in Chapter 2. This approach is investigated in Chapter 4.
### Table 3-4. Sending Node AP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>48</td>
<td>72</td>
<td>63</td>
</tr>
<tr>
<td>Send</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>Rec</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Send</td>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>Rec</td>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>Waiting</td>
<td></td>
<td></td>
<td>78</td>
<td>24</td>
<td>0.5</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>Send</td>
<td></td>
<td></td>
<td>95</td>
<td>92</td>
<td>74</td>
<td>85</td>
<td>68</td>
</tr>
<tr>
<td>Rec</td>
<td></td>
<td></td>
<td>95</td>
<td>92</td>
<td>74</td>
<td>85</td>
<td>68</td>
</tr>
</tbody>
</table>

### Table 3-5. Sending Node PP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td></td>
<td></td>
<td>49</td>
<td>50</td>
<td>67</td>
<td>88</td>
<td>81</td>
</tr>
<tr>
<td>PP1</td>
<td>65</td>
<td>67</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PP2</td>
<td></td>
<td></td>
<td>14</td>
<td>13</td>
<td>10</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>Working</td>
<td></td>
<td></td>
<td>14</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>5</td>
</tr>
<tr>
<td>PP1</td>
<td>36</td>
<td>37</td>
<td>-</td>
<td>-</td>
<td>23</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>PP2</td>
<td>21</td>
<td>20</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

67
Table 3-6. Sending Node CU Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>60</td>
<td>58</td>
<td>67</td>
<td>77</td>
<td>70</td>
<td>71</td>
<td>68</td>
</tr>
<tr>
<td>Working</td>
<td>19</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>23</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Waiting</td>
<td>21</td>
<td>17</td>
<td>9</td>
<td>0.6</td>
<td>7</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

Table 3-7. Sending Node ALP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>1</td>
<td>19</td>
<td>2</td>
<td>56</td>
<td>6</td>
<td>42</td>
<td>5</td>
</tr>
<tr>
<td>Working</td>
<td>11</td>
<td>20</td>
<td>7</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Searching for BA</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>Waiting for MBus</td>
<td>21</td>
<td>16</td>
<td>22</td>
<td>3</td>
<td>26</td>
<td>14</td>
<td>36</td>
</tr>
<tr>
<td>Waiting for BA or CU</td>
<td>66</td>
<td>42</td>
<td>66</td>
<td>30</td>
<td>55</td>
<td>26</td>
<td>45</td>
</tr>
</tbody>
</table>
### Table 3-8. Sending Node BA Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>36</td>
<td>51</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Working</td>
<td>62</td>
<td>46</td>
<td>96</td>
<td>98</td>
<td>97</td>
<td>97</td>
<td>98</td>
</tr>
<tr>
<td>Waiting</td>
<td>2</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

### Table 3-9. Sending Node SMU Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>31</td>
<td>33</td>
<td>23</td>
<td>31</td>
<td>26</td>
<td>36</td>
<td>27</td>
</tr>
<tr>
<td>Working</td>
<td>69</td>
<td>65</td>
<td>77</td>
<td>69</td>
<td>74</td>
<td>64</td>
<td>73</td>
</tr>
</tbody>
</table>
Table 3-10. Sending Node MBus Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>512 Bytes</th>
<th>1024 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
</tr>
<tr>
<td>Idle</td>
<td>0.2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>Working</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3-11. Receiving Node CU Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>512 Bytes</th>
<th>1024 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
</tr>
<tr>
<td>Idle</td>
<td>69</td>
<td>70</td>
<td>75</td>
<td>77</td>
<td>78</td>
<td>78</td>
</tr>
<tr>
<td>Working</td>
<td>29</td>
<td>29</td>
<td>24</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>Waiting</td>
<td>1</td>
<td>0.8</td>
<td>0.5</td>
<td>0.3</td>
<td>0.3</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Table 3-12. Receiving Node BA Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle Fiber Bus</td>
<td>65</td>
<td>44</td>
<td>10</td>
<td>11</td>
<td>9</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Working Fiber Bus</td>
<td>35</td>
<td>56</td>
<td>90</td>
<td>89</td>
<td>91</td>
<td>92</td>
<td>91</td>
</tr>
<tr>
<td>Waiting Bus</td>
<td>3</td>
<td>0.1</td>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Table 3-13. Receiving Node ALP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>59</td>
<td>75</td>
<td>64</td>
<td>91</td>
<td>70</td>
<td>96</td>
<td>74</td>
</tr>
<tr>
<td>Working</td>
<td>8</td>
<td>19</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Waiting</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0.7</td>
<td>0.8</td>
</tr>
</tbody>
</table>
### Table 3-14. Receiving Node PP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>88</td>
<td>71</td>
<td>93</td>
<td>89</td>
<td>94</td>
<td>95</td>
<td>97</td>
</tr>
<tr>
<td>Working</td>
<td>10</td>
<td>24</td>
<td>6</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Waiting</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0.6</td>
</tr>
</tbody>
</table>

### Table 3-15. Receiving Node SMU Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Working</td>
<td>69</td>
<td>70</td>
<td>74</td>
<td>68</td>
<td>72</td>
<td>64</td>
<td>72</td>
</tr>
</tbody>
</table>
Table 3-16. Receiving Node MBus Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>Working</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3-17. Receiving Node AP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>14</td>
<td>7</td>
<td>11</td>
<td>16</td>
<td>15</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td>Working</td>
<td>27</td>
<td>29</td>
<td>30</td>
<td>42</td>
<td>35</td>
<td>48</td>
<td>37</td>
</tr>
<tr>
<td>Waiting</td>
<td>59</td>
<td>64</td>
<td>59</td>
<td>42</td>
<td>50</td>
<td>30</td>
<td>47</td>
</tr>
</tbody>
</table>
## Table 3-18. System Throughput and Latency

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Throughput (Mbps)</td>
<td>87</td>
<td>203</td>
<td>280</td>
<td>401</td>
<td>332</td>
<td>403</td>
<td>344</td>
<td>401</td>
<td>358</td>
<td>402</td>
<td>364</td>
<td>401</td>
<td>371</td>
<td>401</td>
</tr>
<tr>
<td>System Latency (μs)</td>
<td>126</td>
<td>21</td>
<td>150</td>
<td>63</td>
<td>209</td>
<td>126</td>
<td>281</td>
<td>189</td>
<td>343</td>
<td>251</td>
<td>420</td>
<td>315</td>
<td>490</td>
<td>360</td>
</tr>
<tr>
<td>Send Latency (μs)</td>
<td>118</td>
<td>16</td>
<td>135</td>
<td>55</td>
<td>184</td>
<td>113</td>
<td>253</td>
<td>171</td>
<td>307</td>
<td>228</td>
<td>377</td>
<td>287</td>
<td>440</td>
<td>350</td>
</tr>
<tr>
<td>Receive Latency (μs)</td>
<td>9</td>
<td>5</td>
<td>15</td>
<td>8</td>
<td>24</td>
<td>13</td>
<td>28</td>
<td>18</td>
<td>36</td>
<td>23</td>
<td>43</td>
<td>27</td>
<td>50</td>
<td>32</td>
</tr>
</tbody>
</table>
Chapter 4. Multiple-Bus Systems

Two multiple-bus systems are considered in Chapter 4. The first is a system that uses two buses to divide system traffic. This two-bus design is called a dual-bus system. The second system divides the system again by using a third bus to create a triple-bus system. Both the dual-bus and triple-bus systems require multi-ported components to provide communication between the buses. In addition, both systems allow for the selection of different bus types for different portions of the system.

4.1. Analysis of a Multiple-bus Systems

A dual-bus system relies on a dual-ported Shared Memory Unit (SMU) to provide an interface between an upper bus and a lower bus. This memory unit allows the system bus to be split into two components as shown in Figure 4-1. The upper bus carries only AP and PP traffic, while the lower bus carries traffic to the coding logic and, if the BAs are attached directly to the bus, to the BAs.

This division of the bus structure reduces the bandwidth requirements for each bus below that required for the system bus in a single-bus design. In addition, it allows different bus structures to be chosen to attach to the set consisting of the AP and PPs and the set consisting of the Coding Logic and BAs. This ability to choose different bus structures allows greater flexibility in selecting the system components.

In the spirit of further reducing the load on a given bus, one might consider further bus subdivisions. One such division is shown in Figure 4-2, where a third bus, called the center bus, is added to separate
the AP to PP traffic from the PP to BA traffic. This triple-bus design allows the PPs to reside on a bus that is different than the bus used by the AP. As in the dual-bus design, this modification is done by adding a dual-ported Shared Memory Unit.

The assumptions for the single-bus analysis presented in Section 3.1 are also used in the analyses of dual-bus and triple-bus systems. It is assumed that a memory unit is available that provides two ports, each capable of transferring information at the maximum data transfer rate of the bus to which it is attached. In contrast to the single-bus design, however, it is less likely that such a unit is commercially available for the chosen upper, lower, and center bus types. While it is likely that bus interface logic is available for each bus chosen, the logic will probably have to be combined to build custom memory units for the systems.
4.1.1. Variables Used in Analysis

In addition to the variables defined in Chapter 3, the following variables are used in these analyses.

\[ R_{\text{UBUS}} = \text{Upper system bus bandwidth in bits per second} \]

\[ R_{\text{CBUS}} = \text{Center system bus bandwidth in bits per second} \]

\[ R_{\text{LBUS}} = \text{Lower system bus bandwidth in bits per second} \]

\[ R_{\text{ALP}^{\text{out}}} = \text{Access Layer Processor upper port transmission bandwidth in bits per second} \]

\[ R_{\text{ALP}^{\text{out}}} = \text{Access Layer Processor lower port transmission bandwidth in bits per second} \]
4.1.2. Analysis of a Dual-bus System

Assume that a transmitting Application Processor produces $D$ requests per second, there are $N$ Protocol Processors and $M$ Base Adapters, and all data sets produced by the AP are of equal length. Let the length of a data request be $B_R$ bits and the length of the accompanying data set be $B_I$ bits. Finally, let the length of a signal from the AP to a PP be $S_{AP-PP}$. The AP then writes $I_{out} = D(B_R + B_I + S_{AP-PP})$ bits per second (bps) to the Shared Memory Unit via the upper system bus.

The PPs read these requests at a rate of $DB_R$ bits per second and produce network headers for each data set. To accomplish this task in a steady-state situation, the full set of PPs must be able to produce $D$ headers per second and $D$ updated requests per second. Assuming an equal distribution of the work load, this requires that each of the $N$ PPs be able to produce headers and requests at a rate of $D/N$ headers per second. The headers and requests produced by the PPs are written to the Shared Memory Unit via the upper system bus. In addition, each PP signals the ALP via the upper bus after processing a data request. Let the size of this signal be $S_{PP-ALP}$ bits. For headers of length $B_H$ and updated data requests of length $B_{R2}$, each PP requires a portion of the upper bus bandwidth equal to $R_{PP\text{-}out} = D(B_R + B_{R2} + B_H + S_{PP-ALP})/N$ bits per second.

The Access Layer Processor reads the $D$ data requests produced by the PPs every second and produces corresponding access layer headers. For each request, it also signals both the Coding Unit and the Base Adapters. Since the both the ALP and the SMU are attached to both system buses, they may communicate via either one. Assume the ALP reads data requests via the upper bus and writes ALP headers back via the lower bus. The size of an access layer header is $B_A$, the size of a Coding Unit signal is $S_{ALP-CU}$, and the size of a Base Adaptor signal is $S_{ALP-BA}$. The ALP then requires $R_{ALP\text{-}out} = D(B_R)$ bits per second.
bits per second of the upper bus bandwidth and \( R_{ALP} = D(B_A + S_{ALP-CU} + S_{ALP-BA}) \) bits per second of the lower bus bandwidth for data transmission. Combining the bandwidths for both ALP ports yields the total component bandwidth of \( R_{ALP} = D(B_{R2} + B_A + S_{ALP-CU} + S_{ALP-BA}) \).

Coding of the packet information is done by reading the resulting data set along with its network and access layer headers from the shared memory via the lower bus and producing an additional coding header of length \( B_C \) bits. If the coding scheme being used requires the full data set to be modified, as it does in situations where coding is done for encryption, the entire modified data set must be written back to the shared memory along with any header. In a steady-state situation, the coding set must be able to encode \( D(B_A + B_B + B_D) \) bits per second, write back coding sets at \( D \cdot P_c \) bps, and signal the ALP to indicate completion at \( DS_{CU-ALP} \) bps. These tasks requires a bus connection operating at \( R_{CODE} = D[2(B_A + B_B + B_D) + B_C + S_{CU-ALP}] \) bits per second.

Finally, if the Base Adapters are attached to the lower bus rather than the coding logic, they read the full data set and its associated headers from the memory for transmission. This operation requires a bus connection operating at \( D(B_A + B_B + B_C) \) bits per second for the full set of adapters. As discussed in Section 3.1.3 and depicted in Figure 4-1, the BAs may be attached directly to the coding logic to reduce bus bandwidth requirements. Individually, the adapters must absorb \( R_{BA} = D(B_A + B_B + B_C) \) bits per second for transmission.

A similar process takes place during the reception of data. The BAs write the data to the shared memory via the lower system bus. It is then read by the coding logic, decoded, and written back using the same bus. The ALP reads the decoded access layer headers via the lower bus to determine the packet handling method and signals the PPs via the upper bus. The PPs read the decoded network headers to generate

Chapter 4 - Multiple-Bus Systems
a data indication for the AP using the upper bus. Finally, the data is read by the AP using the upper bus. Assuming half of the system bandwidth is consumed by reception and half by transmission, the data rates derived above are doubled to produce the full system data rates given in Equations 4.1 to 4.6.

\[
I = 2I_{\text{out}} = 2D( B_R + B_S + S_{\text{AP-PP}} )
\] (4.1)

\[
R_{PP} = 2R_{\text{PP-out}} = 2D( B_R + B_{R2} + B_H + S_{PP-ALP} ) / N
\] (4.2)

\[
R_{\text{ALPU}} = 2R_{\text{ALPU-out}} = 2D( B_{R2} )
\] (4.3)

\[
R_{\text{ALFL}} = 2R_{\text{ALFL-out}} = 2D( B_A + S_{\text{ALP-CU}} + S_{\text{ALP-BA}} )
\] (4.4)

\[
R_{\text{CODE}} = 2R_{\text{CODE-out}} = 2D[ 2( B_A + B_H + B_S ) + B_C + S_{\text{CU-ALP}} ]
\] (4.5)

\[
R_{BA} = 2R_{\text{BA-out}} = 2D( B_A + B_H + B_S + B_C ) / M
\] (4.6)

Each active component attached to a system bus requires a portion of the bus bandwidth. Therefore, summing the active component requirements on the upper bus, as given in Equations 4.1 to 4.3, results in the required upper bus bandwidth.

\[
R_{UBUS} = 1 + NR_{PP} + R_{\text{ALPU}}
\]

\[
= 2D( B_R + B_S + S_{\text{AP-PP}} ) + 2D( B_R + B_{R2} + B_H + S_{PP-ALP} ) + 2D( B_{R2} )
\]

\[
R_{UBUS} = 2D( 2B_R + 2B_{R2} + B_H + B_S + S_{\text{AP-PP}} + S_{PP-ALP} )
\] (4.7)

Similarly, summing Equations 4.4 to 4.6 results in an equation for the required lower bus bandwidth.

\[
R_{LBUS} = R_{\text{ALFL}} + R_{\text{CODE}} + MR_{BA}
\]

\[
= 2D( B_A + S_{\text{ALP-CU}} + S_{\text{ALP-BA}} ) + 2D[ 2( B_A + B_S + B_H ) + B_C + S_{\text{CU-ALP}} ] + 2D( B_A + B_S + B_H + B_C )
\]

\[
R_{LBUS} = 2D( 4B_A + 3B_H + 3B_S + 2B_C + 2S_{\text{ALP-CU}} + S_{\text{ALP-BA}} )
\] (4.8)

Chapter 4 - Multiple-Bus Systems
Assuming $B_s$ is the dominant term, the reduced bus bandwidth equations become $R_{UBUS} = 2DB_s$ and $R_{LBUS} = 6DB_s$. Comparing these reduced equations, the lower bus has three times more traffic than the upper bus. Obviously, the proposed bus structure does not provide an equitable division of bus traffic for this system. This situation is reversed, but not improved, by placing the BAs on the lower bus and the Coding Logic on the upper bus. The reduced equations become $R_{UBUS} = 6DB_s$ and $R_{LBUS} = 2DB_s$. This is still not an equitable distribution of traffic.

Attaching the BAs to the coding logic as shown in Figure 4-1 reduces the lower bus bandwidth requirement to that given by Equation 4.9.

$$R_{LBUS} = R_{ALPL} + R_{CODE(Bus Port)}$$

$$= 2D(B_A + S_{ALP-CU} + S_{ALP-Ba}) + 2D(B_A + B_S + B_H)$$

$$R_{BUS} = 2D(2B_A + B_H + B_S + S_{ALP-CU}) \quad (4.9)$$

In this case, both the upper bus equation and the modified lower bus equation reduce to $R_{bus} = 2DB_s$. This indicates that an equitable division of the bus traffic is provided by this system modification.

Another option that results in an equitable distribution of bus traffic is to incorporate the coding logic into the Shared Memory Unit. This approach results in a lower bus equation similar to Equation 4.9 (its reduced form is $R_{BUS} = 2DB_s$), but requires a custom memory unit.
4.1.3. Analysis of a Triple-bus System

The dual-bus system presented in Section 4.1.2 provides an equitable distribution of the bus load between two system buses. The triple-bus design attempts to further reduce the load on any given by introducing a third bus.

This new design results in a somewhat modified system operation in which data sets must be transferred between two separate memory units. The transfer is done by the PPs. During data transmission, the PPs read both the data requests and the data and write an updated data request along with the data to the lower Shared Memory Unit. During reception, they perform the opposite function, reading data indications from the lower memory unit and transferring the updated indications and data to the upper unit.

Bus bandwidth equations are developed following the same procedure used in the dual-bus design. In fact, the lower bus equation remains unchanged from the dual-bus design. Furthermore, since the only active component on the upper bus is the AP, the upper bus bandwidth requirement is simply the AP interface requirement.

The center bus bandwidth requirement is the result of PP and ALP traffic only. However, this traffic increases since the PPs must transfer the full data sets between the upper and lower Shared Memory Units. This means a single PP requires an interface operating at \( R_{PP} = 2D(2B_S + B_R + B_{R2} + B_H + S_{PP-ALP})/N \) bits per second to handle both transmission and reception. The center bus bandwidth equation is equal to the sum of the individual PP interface equations and the ALP upper port equation (Eq. 4.3), or \( R_{BUS} = R_{PP} + R_{ALP}/2D(2B_S + B_R + 2B_{R2} + B_H + S_{PP-ALP}) \) bits per second.
Once again assuming that the size of the data set is the dominant term in this equation, it reduces to 
\[ R_{\text{BUS}} = 4DB_5 \]. This is twice the bandwidth required of either bus in the modified dual-bus system.

Therefore, the addition of a third bus does not reduce the bandwidth required by any system component.

Instead, the need to transfer full data sets between separate Shared Memory Units increases both the system complexity and individual component bandwidth requirements.

4.1.4. Analysis Results

The results of the analysis presented in Sections 4.1.2 and 4.1.3 are summarized in Table 4-1. The simplified equations, assuming the data set length \( (B_d) \) is the dominant term, are given in Table 4-2.

The simplified equations in Table 4-2 show that the triple-bus design does not provide any advantages in terms of component bandwidth requirements. In fact, the center bus in the triple-bus system must provide twice the bandwidth of either bus in the dual-bus system.

The modified single-bus equations presented in Section 3.1.4 indicate a required bus bandwidth of approximately \( 4DB_5 \). Comparing this to the simplified dual-bus equations, one can see that the dual-bus system provides a 50 percent reduction in bus traffic.

As an illustration of the component bandwidths required by a dual-bus system, consider the example used in Section 3.1.4. Once again, consider a dual-bus system with an AP that requires an \( I = 1 \) Gbps network interface. Let the size of a data request be \( B_d = 268 \) bits, the size of a data notification signal between components be \( 128 \) bits, and all data packets be \( B_5 = 35 \) Kbits long. Assuming the interface
Table 4-1. System Component Bandwidths (bps)

<table>
<thead>
<tr>
<th></th>
<th>Dual-bus</th>
<th>Modified Dual-bus</th>
<th>Triple-bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>$2D(B_R + B_S + S_{AP-PP})$</td>
<td>$2D(B_R + B_S + S_{AP-PP})$</td>
<td>$2D(B_R + B_S + S_{AP-PP})$</td>
</tr>
<tr>
<td>$R_{TP}$</td>
<td>$2D(B_R + B_R + B_H + S_{PP-ALP})N$</td>
<td>$2D(B_R + B_R + B_H + S_{PP-ALP})N$</td>
<td>$2D(B_R + B_R + B_H + S_{PP-ALP})N$</td>
</tr>
<tr>
<td>$R_{ALP}$</td>
<td>$2D(B_R + B_A + S_{ALP,CU} + S_{ALP-BA})$</td>
<td>$D(B_R + B_A + S_{ALP,CU})$</td>
<td>$D(B_R + B_A + S_{ALP,CU} + S_{ALP-BA})$</td>
</tr>
<tr>
<td>$R_{CODE}$</td>
<td>$2D[2(B_A + B_H + B_S + B_C) + S_{CL-ALP}]$</td>
<td>$2D[2(B_A + B_H + B_S + B_C)]$</td>
<td>$2D[2(B_A + B_H + B_S + B_C + S_{CL-ALP})]$</td>
</tr>
<tr>
<td>$R_{SN}$</td>
<td>$2D(B_A + B_H + B_S + B_C)/M$</td>
<td>$2D(B_A + B_H + B_S + B_C)/M$</td>
<td>$2D(B_A + B_H + B_S + B_C)/M$</td>
</tr>
<tr>
<td>$R_{UBUS}$</td>
<td>$2D(2B_R + 2B_R + B_H + B_S + S_{AP-PP} + S_{PP-ALP})$</td>
<td>$2D(2B_R + 2B_R + B_H + B_S + S_{AP-PP} + S_{PP-ALP})$</td>
<td>$2D(B_R + B_S + S_{AP-PP})$</td>
</tr>
<tr>
<td>$R_{CBUS}$</td>
<td>$-$</td>
<td>$-$</td>
<td>$2D(B_R + 2B_R + 2B_R + S_{PP-ALP})$</td>
</tr>
<tr>
<td>$R_{LBUS}$</td>
<td>$2D(4B_A + 3B_H + 3B_S + 2B_C + S_{ALP,CU} + S_{ALP-BA})$</td>
<td>$2D(2B_A + 3B_H + B_S + S_{ALP,CU})$</td>
<td>$2D(4B_A + 3B_H + 3B_S + 2B_C + S_{ALP-BA})$</td>
</tr>
</tbody>
</table>

Bandwidth is equally distributed between transmission and reception, Equation 4-1 leads to

$$D = \frac{I}{2(B_R + B_S + B_{AP-PP})} = 1 \text{ Gbps} \frac{1 \text{bits}}{2(208 + 35000 + 128) \text{bits/sec}} \approx 14.1 \frac{\text{Kreq}}{\text{sec}}.$$  

Assume the system includes $M = 10$ BAs and $N = 5$ PPs, where the PPs add TCP and IP headers for a total of $B_H = 320$ bits, the ALP adds a header of $B_A = 272$ bits, and the BAs require an additional header of $B_p = 272$ bits. Furthermore, let the size of a PP to ALP data request be $B_{R2} = 256$ bits. Finally, let the Coding Unit supply a header $B_C = B_H/M = 35 \text{ Kbits/10} = 3.5 \text{ Kbits}$ without requiring the full data set.
Table 4-2. Simplified System Component Bandwidths (bps)

<table>
<thead>
<tr>
<th></th>
<th>Dual-bus</th>
<th>Modified Dual-bus</th>
<th>Triple-bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>$2DB_S$</td>
<td>$2DB_S$</td>
<td>$2DB_S$</td>
</tr>
<tr>
<td>$R_{pp}$</td>
<td>$2D(B_R + B_{R2} + B_H) / N$</td>
<td>$2D(B_R + B_{R2} + B_H) / N$</td>
<td>$2D(B_R + B_{R2} + B_H) / N$</td>
</tr>
<tr>
<td>$R_{ALP}$</td>
<td>$2D(B_{R2} + B_A)$</td>
<td>$D(B_{R1} + B_A)$</td>
<td>$D(B_{R2} + B_A)$</td>
</tr>
<tr>
<td>$R_{CODE}$</td>
<td>$4DB_S$</td>
<td>$4DB_S$</td>
<td>$4DB_S$</td>
</tr>
<tr>
<td>$R_{BA}$</td>
<td>$2DB_S / M$</td>
<td>$2DB_S / M$</td>
<td>$2DB_S / M$</td>
</tr>
<tr>
<td>$R_{USBUS}$</td>
<td>$2DB_S$</td>
<td>$2DB_S$</td>
<td>$2DB_S$</td>
</tr>
<tr>
<td>$R_{CPU}$</td>
<td>-</td>
<td>-</td>
<td>$4DB_S$</td>
</tr>
<tr>
<td>$R_{LBUS}$</td>
<td>$6DB_S$</td>
<td>$2DB_S$</td>
<td>$6DB_S$</td>
</tr>
</tbody>
</table>

to be written back to the shared memory. The required component bandwidths are then

\[
R_{pp} = \frac{2D(B_R + B_{R2} + B_H + S_{ALP-pp})}{N} = \frac{2(14.1K^{req} \times sec)(208 + 256 + 320 + 128)}{5} \frac{\text{bits}}{\text{req}} \approx 5.1 \frac{\text{Mbits}}{\text{sec}} ,
\]

\[
R_{ALP} = 2D(B_{R2} + B_A + S_{ALP-CU} + S_{ALP-BA}) = 2(14.1K^{req} \times sec)(256 + 272 + 2 \times 128) \frac{\text{bits}}{\text{req}} \approx 22 \frac{\text{Mbits}}{\text{sec}} ,
\]

\[
R_{CODE} = 2D(B_A + B_S + B_H + B_C + S_{CU-ALP})
= 2(14.1K^{req} \times sec)(272 + 35K + 320 + 3.5K + 128) \frac{\text{bits}}{\text{req}} \approx 1.1 \frac{\text{Gbits}}{\text{sec}} ,
\]

Chapter 4 - Multiple-Bus Systems

85
\[ R_{BA} = 2D \left( \frac{B_A + B_S + B_H + B_C}{M} \right) = 2(14.1K \text{ \frac{\text{req}}{\text{sec}}}) \left( \frac{272 + 35K + 320 + 3.5K}{10} \text{ \frac{\text{bits}}{\text{req}}} \right) \]

\[ \approx 110 \text{ \frac{Mbits}{sec}}, \]

\[ R_{UBUS} = 2D(2B_R + 2B_{H2} + B_S + B_H + S_{AP-PP} + S_{PP-ALP}) \]

\[ = 2(14.1K \text{ \frac{\text{req}}{\text{sec}}})(2(208) + 2(256) + 35K + 320 + 2(128)) \text{ \frac{\text{bits}}{\text{req}}} \approx 1 \text{ \frac{Gbits}{sec}}, \]

\[ R_{LBUS} = 2D(2B_S + 2B_H + 3B_A + 2B_C + 2S_{ALP-CU} + S_{ALP-BA}) \]

\[ = 2(14.1K \text{ \frac{\text{req}}{\text{sec}}})(2(35K) + 2(320) + 3(272) + 2(3.5K) + 3(128)) \text{ \frac{\text{bits}}{\text{req}}} \approx 2.2 \text{ \frac{Gbits}{sec}}. \]

The majority of the component bandwidths are similar to those of the single-bus design. The two exceptions are the upper and lower bus. Assuming a 64-bit bus upon which one transfer is allowed per clock cycle, the upper bus must have a clock speed of 15 MHz and the lower bus must have a clock speed of 34 MHz. This is a significant reduction below the 52 MHz clock required for the single-bus system in the previous example. These clock speeds, in fact, fall well within the 40 MHz clock of the SPARC MBus.
4.2. Functional Modeling of a Dual-bus System

The VHDL model of a dual-bus PLAN system developed here provides insight into the design of this type of system. It demonstrates the component utilization and data transfer characteristics of the system and allows for comparison with the single-bus implementation developed in Chapter 3.

This dual-bus system model is developed around two SPARC MBuses. The upper MBus supports a single AP, three PPs, and one port each of a dual-ported SMU and a dual-ported ALP. The lower bus supports five BAs, a CU, and the second ports of the dual-ported SMU and dual-ported ALP. Each bus is supported by its own arbiter and bus clock.

Most of the system components are identical to those used in the single-bus model described in Chapter 3. The two exceptions are the ALP and SMU. While both components are based on their single-bus system counterparts, they are both modified to be dual-ported to support communication between the upper and lower buses in the dual-bus model.

The dual-ported SMJ used in this model is the same flat-memory model used in the single-bus VHDL model. Its underlying VHDL processes are essentially the single-ported memory model. These processes must allow for memory access at double an individual bus data rate or faster to support two full-bus-bandwidth ports. In this case, the basic processes are designed to provide 64-bit read and write cycles of 2 ns each. This choice accommodates two 64-bit ports operating on 25 ns bus clocks.

The ALP is converted from the single-bus model by adding a second bus interface and recognizing the bus to which other components are attached. During transmission of data, the ALP accepts DR2 signals
from the PPVs via the upper bus, then writes an AL header and signals the Coding Unit and Base Adapters via the lower bus. During data reception, the ALP accepts D12 signals from the BAs, signals the CU, reads the incoming AL header via the lower bus, and then signals the PPIC via the upper bus.

4.3. Simulation Results

The VHDL model described in Section 4.2 is used to simulate PLAN system operation under various load conditions. All simulations are performed using the same type of two-node system under the same assumptions and conditions described in Section 3.3 to allow for comparison of the results.

Once again, the data sets are varied in increments of 1 Kbit from 1 Kbit to 35 Kbits (128 bytes to 3840 bytes) while the code set is fixed at 1 Kbit (128 bytes). Tables 4-3 through 4-14 indicate the utilization of the various model components under these loads. The packet size parameter in each table represents the size of a data set for each simulation and the coding parameter indicates the state of the Coding Unit during the simulation.

The simulation is controlled by specifying the length of time that it is runs. Each simulation is run for a length of time that results in the transfer of 25 to 30 packets. Utilization results are generated for the whole simulation, i.e. for the 25 to 30 packets that are transferred.

Utilization values are rounded to the nearest whole number if greater than one. Otherwise, they are rounded to the nearest tenth. A utilization value of 0.0 percent indicates that the component spends some
small fraction of the simulation time in this state. A utilization result that is given as a dash (-) indicates that the component spends no time in the state.

As in Chapter 3, most tables indicate the percent of time spent in each of three states by system components. A component in an "idle" state is waiting for new packet data to become available. In the "working" state, it is processing a packet. Finally, in the "waiting" state, a component is waiting to use one of the system buses.

Results are given in Table 4-3 for the first and second Protocol Processors (PP1 and PP2) of the transmitting node and in Table 4-12 for the first PP of the receiving node only since the other PPs are not utilized. In general, only the first PP is heavily utilized, indicating the PP set is not a bottleneck to system throughput.

The system design used results in long waits for the ALPs since they must synchronize the processing of a single packet as it moves through the ALP, CU, and BAs. To provide this synchronization, an ALP must not only wait for access to the MBus, but also for the CU and BAs to finish processing each packet before it moves on to the next packet. The utilization results given for the ALPs include multiple wait statistics to indicate the amount of time spent waiting for each of the system components. In addition to the standard wait for MBus access, the transmitting ALP's statistics, given in Table 4-4, include the percentage of time spent searching for an available BA and the percentage of time spent waiting for either the Coding Unit to finish processing a packet or a BA to read a packet's data. The receiving ALP's statistics, summarized in Table 4-11, include time spent waiting for MBus access and time spent waiting for the Coding Unit.
Tables 4-5 and 4-10 contain utilization results for the transmitting and receiving nodes' Coding Units. Both tables only contain information for those simulations in which a coding set is produced since the Coding Units are not utilized during the other simulations.

The Base Adaptor utilization results are representative of a single adaptor. The results summarized in Tables 4-6 and 4-9 are obtained from the second Base Adaptor (BA2) of each node. All Base Adapters are relatively evenly loaded due to the round-robin adaptor location algorithm used in the ALP.

Tables 4-7 and 4-13 show the utilization of each node's SMU. Three results are included for each memory unit. The first two indicate the utilization of the SMU's ports to the upper and lower buses. These results depict the utilization of the SMU component itself. The third figure, labeled "core," is a measure of the utilization of the core memory process in the VHDL model of the SMU. These core utilization values give some indication of the ability of this component to support additional ports.

The utilization results for both nodes' system buses are summarized in Tables 4-8 and 4-14. These tables indicate that both the upper and lower bus of each node are heavily utilized. This is due in part to the fact that the AP is continually checking the SMU for received data. The degree of utilization shown here, however, indicates the buses are creating a bottleneck to system throughput.

Throughput and latency for the two-node system are calculated by averaging the values for the 25 packets that follow the first packet. The first packet is discarded due to that fact that all of the system components must be actively accessing the buses before the system enters steady state. These results are shown in Table 4-15.
The packet latencies averaged to produce the results in Table 4-15 typically have large variances due to the fact that the AP is overloading the system to produce maximum throughputs. Since the system traffic exceeds its capacity, packets accumulate in the SMU, and each has a slightly longer wait than the one preceding it. This wait produces an increasing packet latency. These variances are not as large as those of the single-bus system, however, since the dual-bus system provides generally higher throughputs.

4.4. Summary

The results given in Tables 4-3 to 4-15 indicate that the system generally performs better with large data sets than with small sets as is the case with the single-bus system. The dual-bus system does, however, provide slightly better throughput across the various loads than the single-bus system. A more thorough comparison of the various system design approaches is presented in Chapter 6.

As in the single-bus system, the only components that are heavily loaded during system operation are the buses, the Base Adapters, and the sending ALP. The Base Adapter load can be diminished by adding more Base Adapters. Since the ALP spends a significant amount of its time waiting for the CU or BAs, its load can be reduced by using a different approach for packet pointer queuing. For example, separate pointer queues in the ALP, CU, and BAs would allow the ALP to do productive work while waiting for the CU or BAs to finish. The savings that this type of scheme might provide are indicated by the amount of time each ALP spends waiting for a CU or BA to finish handling a packet. Tables 4-4 and 4-11 indicate that this savings would be significant.
Reducing the bus load requires another new approach to the system design. The final logical step toward reducing bus load is to eliminate all system buses. This is done by providing an alternative medium for component communication such as an interconnection network or multi-ported memory devices. This approach is investigated in Chapter 5.
### Table 4-3. Sending Node PP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th></th>
<th>640 Bytes</th>
<th></th>
<th>1280 Bytes</th>
<th></th>
<th>1920 Bytes</th>
<th></th>
<th>2560 Bytes</th>
<th></th>
<th>3200 Bytes</th>
<th></th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle PP1</td>
<td>46</td>
<td>39</td>
<td>87</td>
<td>88</td>
<td>93</td>
<td>94</td>
<td>96</td>
<td>96</td>
<td>97</td>
<td>97</td>
<td>97</td>
<td>98</td>
<td>98</td>
</tr>
<tr>
<td>Idle PP2</td>
<td>99</td>
<td>99</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working PP1</td>
<td>45</td>
<td>45</td>
<td>11</td>
<td>11</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Working PP2</td>
<td>0.6</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Waiting PP1</td>
<td>8</td>
<td>16</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0.6</td>
<td>0.5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Waiting PP2</td>
<td>0.4</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 4-4. Sending Node ALP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th></th>
<th>640 Bytes</th>
<th></th>
<th>1280 Bytes</th>
<th></th>
<th>1920 Bytes</th>
<th></th>
<th>2560 Bytes</th>
<th></th>
<th>3200 Bytes</th>
<th></th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>57</td>
<td>5</td>
<td>34</td>
<td>6</td>
<td>29</td>
<td>7</td>
<td>30</td>
<td>7</td>
<td>30</td>
<td>8</td>
</tr>
<tr>
<td>Working</td>
<td>17</td>
<td>36</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Searching for BA</td>
<td>8</td>
<td>10</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>0.6</td>
<td>3</td>
<td>0.4</td>
<td>2</td>
<td>0.4</td>
<td>0.2</td>
</tr>
<tr>
<td>Waiting for MBus</td>
<td>18</td>
<td>4</td>
<td>34</td>
<td>4</td>
<td>39</td>
<td>35</td>
<td>42</td>
<td>44</td>
<td>42</td>
<td>44</td>
<td>44</td>
<td>45</td>
<td>46</td>
</tr>
<tr>
<td>Waiting for BA or CU</td>
<td>56</td>
<td>48</td>
<td>50</td>
<td>29</td>
<td>46</td>
<td>25</td>
<td>46</td>
<td>24</td>
<td>45</td>
<td>23</td>
<td>45</td>
<td>23</td>
<td>44</td>
</tr>
</tbody>
</table>
### Table 4-5. Sending Node CU Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>69</td>
<td>74</td>
<td>76</td>
<td>76</td>
<td>77</td>
<td>77</td>
<td>77</td>
</tr>
<tr>
<td>Working</td>
<td>30</td>
<td>26</td>
<td>24</td>
<td>24</td>
<td>23</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>Waiting</td>
<td>0.5</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

### Table 4-6. Sending Node BA Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>3</td>
<td>21</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Working</td>
<td>97</td>
<td>79</td>
<td>98</td>
<td>97</td>
<td>98</td>
<td>99</td>
<td>98</td>
</tr>
<tr>
<td>Waiting</td>
<td>0.1</td>
<td>0.2</td>
<td>0.0</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
### Table 4-7. Sending Node SMU Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper Port</td>
<td>32</td>
<td>30</td>
<td>43</td>
<td>42</td>
<td>45</td>
<td>45</td>
<td>45</td>
</tr>
<tr>
<td>Lower Port</td>
<td>51</td>
<td>52</td>
<td>55</td>
<td>73</td>
<td>58</td>
<td>77</td>
<td>58</td>
</tr>
<tr>
<td>Core</td>
<td>96</td>
<td>96</td>
<td>97</td>
<td>97</td>
<td>97</td>
<td>98</td>
<td>97</td>
</tr>
<tr>
<td>Working</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper Port</td>
<td>68</td>
<td>70</td>
<td>57</td>
<td>58</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>Lower Port</td>
<td>49</td>
<td>48</td>
<td>45</td>
<td>27</td>
<td>42</td>
<td>23</td>
<td>42</td>
</tr>
<tr>
<td>Core</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

### Table 4-8. Sending Node MBus Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idle</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>0.2</td>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>Lower</td>
<td>0.2</td>
<td>0.1</td>
<td>0.4</td>
<td>0.0</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>Working</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upper</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Lower</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
### Table 4-9. Receiving Node BA Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Working</td>
<td>88/5</td>
<td>60/10</td>
<td>91/5</td>
<td>89/6</td>
<td>93/4</td>
<td>92/5</td>
<td>93/4</td>
</tr>
<tr>
<td>Waiting</td>
<td>0.0/0.0</td>
<td>0.1/0.0</td>
<td>0.0/0.0</td>
<td>0.0/0.0</td>
<td>0.0/0.0</td>
<td>0.0/0.0</td>
<td>0.0/0.0</td>
</tr>
</tbody>
</table>

### Table 4-10. Receiving Node CU Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>51</td>
<td>69</td>
<td>75</td>
<td>76</td>
<td>78</td>
<td>78</td>
<td>80</td>
</tr>
<tr>
<td>Working</td>
<td>49</td>
<td>30</td>
<td>25</td>
<td>23</td>
<td>22</td>
<td>22</td>
<td>20</td>
</tr>
<tr>
<td>Waiting</td>
<td>0.4</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
<td>0.9</td>
<td>0.0</td>
</tr>
</tbody>
</table>
### Table 4-11. Receiving Node ALP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>27</td>
<td>48</td>
<td>64</td>
<td>93</td>
<td>72</td>
<td>96</td>
<td>74</td>
</tr>
<tr>
<td>Working</td>
<td>14</td>
<td>33</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Waiting Bus</td>
<td>13</td>
<td>19</td>
<td>1</td>
<td>0.7</td>
<td>0.6</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>CU</td>
<td>47</td>
<td>-</td>
<td>30</td>
<td>-</td>
<td>25</td>
<td>-</td>
<td>23</td>
</tr>
</tbody>
</table>

### Table 4-12. Receiving Node PP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle PP1</td>
<td>81</td>
<td>48</td>
<td>92</td>
<td>90</td>
<td>96</td>
<td>94</td>
<td>97</td>
</tr>
<tr>
<td>PP2</td>
<td>-</td>
<td>76</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Working PP1</td>
<td>16</td>
<td>30</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>PP2</td>
<td>-</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Waiting PP1</td>
<td>3</td>
<td>22</td>
<td>1</td>
<td>2</td>
<td>0.8</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>PP2</td>
<td>-</td>
<td>12</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 4-13. Receiving Node SMU Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idle</td>
<td>47</td>
<td>30</td>
<td>48</td>
<td>45</td>
<td>48</td>
<td>46</td>
<td>48</td>
</tr>
<tr>
<td>Upper Port</td>
<td>34</td>
<td>53</td>
<td>53</td>
<td>76</td>
<td>59</td>
<td>79</td>
<td>60</td>
</tr>
<tr>
<td>Lower Port</td>
<td>96</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>97</td>
<td>98</td>
<td>97</td>
</tr>
<tr>
<td>Core</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working</td>
<td>53</td>
<td>70</td>
<td>52</td>
<td>55</td>
<td>52</td>
<td>54</td>
<td>52</td>
</tr>
<tr>
<td>Upper Port</td>
<td>66</td>
<td>47</td>
<td>47</td>
<td>24</td>
<td>41</td>
<td>21</td>
<td>40</td>
</tr>
<tr>
<td>Lower Port</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 4-14. Receiving Node MBus Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Idle</td>
<td>0.4</td>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>Lower</td>
<td>0.1</td>
<td>0.1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Working</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Upper</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Lower</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4-15. System Throughput and Latency

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Coding</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Throughput (Mbps)</td>
<td>131</td>
<td>329</td>
<td>283</td>
<td>401</td>
<td>331</td>
<td>404</td>
<td>351</td>
</tr>
<tr>
<td>System Latency (µs)</td>
<td>92</td>
<td>19</td>
<td>139</td>
<td>63</td>
<td>206</td>
<td>126</td>
<td>275</td>
</tr>
<tr>
<td>Send Latency (µs)</td>
<td>82</td>
<td>13</td>
<td>125</td>
<td>55</td>
<td>185</td>
<td>112</td>
<td>246</td>
</tr>
<tr>
<td>Receive Latency (µs)</td>
<td>10</td>
<td>5</td>
<td>14</td>
<td>8</td>
<td>21</td>
<td>13</td>
<td>28</td>
</tr>
</tbody>
</table>
Chapter 5. Bus-Free Systems

Three bus-free PLAN systems are considered in Chapter 5. Two of the systems use alternative structures to connect the components of the single-bus system investigated in Chapter 3. A third system design takes a more novel approach and uses several independent queues to connect the various PLAN system components. While the two bus replacement designs do not provide any significant advantage over a bus-based system, the queue based system does provide an interesting alternative.

5.1. Analysis of a Multi-ported Queue System

A queue system employs several queues to connect components as shown in Figure 5-1. This structure removes the bus-related bottlenecks found in bus-based systems and provides a straightforward design by physically separating the system layers with queues (Q1, Q2, Q3, and Q4 in the figure). In addition, the scalability of this type of system is limited only by the number of ports that can be provided by a single queue, and not by bus constraints. Unlike the bus-based systems described in Chapters 3 and 4, however, this system requires a significant amount of custom-designed hardware. Since it is unlikely that any one of the system components is commercially available, the multi-ported memory units must be custom-built for this system and each of the major system components must either be custom-designed or modified to interface with the queues.

The assumptions presented in the single-bus analysis of Section 3.1 are used in the multi-ported queue analysis as well. Unlike the bus-based systems, however, it is not necessary to assume a memory unit
is available for the chosen bus type. Instead, it is assumed here that it is possible to construct each of the system components.

5.1.1. Variables Used in Analysis

The following variables, in addition to those defined in Section 3.1.1, are used for this analysis.
\( T_{Q1} \) = Application Processor to Protocol Processor queue (Q1) throughput in bits per second

\( T_{Q2} \) = Protocol Processor to Access Layer Processor queue (Q2) throughput in bits per second

\( T_{Q3} \) = Access Layer Processor to Coding Unit queue (Q3) throughput in bits per second

\( T_{Q4} \) = Coding Unit to Base Adaptor queue (Q4) throughput in bits per second

\( R_{PPU} \) = Protocol Processor upper port (Q1) bandwidth in bits per second

\( R_{PPL} \) = Protocol Processor lower port (Q2) bandwidth in bits per second

\( R_{ALPU} \) = Access Layer Processor upper port (Q2) bandwidth in bits per second

\( R_{ALPL} \) = Access Layer Processor lower port (Q3) bandwidth in bits per second

\( R_{CUU} \) = Coding Unit upper port (Q3) bandwidth in bits per second

\( R_{CUL} \) = Coding Unit lower port (Q4) bandwidth in bits per second

\( R_{Xout} \) = Bandwidth of component port \( X \) required for transmission (e.g. \( R_{PPLlow} \)) in bits per second

\( R_{Xin} \) = Bandwidth of component port \( X \) required for reception (e.g. \( R_{PPLinc} \)) in bits per second

\( B_{R1} \) = Length of an Access Layer Processor to Coding Unit Data Request in bits

\( B_{Rt} \) = Length of a Coding Unit to Base Adaptor Data Request in bits

5.1.2. Multi-ported Queue System Analysis

Assume that a transmitting Application Processor (AP) produces \( D \) requests per second, that there are \( N \) Protocol Processors (PPs) and \( M \) Base Adapters (BAs) per node, and that all data sets produced by the AP are of equal length. Let the length of a data request be \( B_{R} \) bits and the length of the
accompanying data set be $B_s$ bits. The AP then writes $I_{in}=D(B_R+B_g)$ bits per second (bps) to the first queue (Q1).

The PPs read this information and produce network headers for each data set. To accomplish this task in a steady-state situation, the full set of PPs must be able to read data from Q1 at the same rate data is produced by the AP. The set of PPs must then produce network headers and provide an updated data request for each data set. Since data sets arrive at a rate of $D$ per second, $D$ headers and $D$ requests must be produced each second. Assuming an equal distribution of the work load, this requires that each of the $N$ PPs be able to read from Q1 at a rate of $I_{in}/N$ and produce both headers and requests at a rate of $D/N$ per second. In addition, each PP must transfer the full data set from the first queue to the second queue (Q2). For network headers of length $B_H$ and updated data requests of length $B_d$, each PP, therefore, requires an upper port to Q1 operating at $R_{PP\text{,load}}=D(B_R+B_g)/N$ bps and a lower port to Q2 operating at $R_{PP\text{,load}}=D(B_R+B_H+B_g)/N$ bps.

The Access Layer Processor (ALP) reads the $D$ data requests produced by the PPs every second and produces corresponding access layer headers along with updated data requests. For each request, it also transfers the accompanying data set and network headers from Q2 to Q3. Let the size of an access layer header be $B_s$ and the size of an updated data request be $B_R$. The ALP then requires an upper port bandwidth of $R_{ALP\text{,load}}=N R_{PP\text{,load}}=D(B_R+B_H+B_g)$ bps and a lower port bandwidth of $R_{ALP\text{,load}}=D(B_R+B_s+B_H+B_g)$ bps for data transmission.

Coding of the packet information is done by reading the data set along with its network and access layer headers from Q3, encoding the data set and headers, and producing an additional coding header of length $B_c$ bits. In a steady-state situation, the Coding Unit (CU) must be able to encode $D(B_s+B_H+B_g)$ bits per
second and transfer this information along with a final updated data request to the fourth multi-ported queue (Q4). This operation requires a port to Q3 operating at $R_{CULow}=R_{ALPLow}=D(B_{Rx}+B_{Ax}+B_{H}+B_{z})$ bps and a port to Q4 operating at $R_{CULow}=D(B_{Rx}+B_{Ax}+B_{H}+B_{z}+B_{C})$ bps.

Finally, the Base Adapters read the full packet, including the data set and its associated headers, from Q4 for transmission. This operation requires a port to Q4 operating at $R_{CULow}=D(B_{Rx}+B_{Ax}+B_{H}+B_{z}+B_{C})$ bps for the full set of adapters. Assuming the load is equally distributed among the adapters, each adaptor must be able to absorb $R_{E_{BA}}=R_{CULow}/M=D(B_{Rx}+B_{Ax}+B_{H}+B_{z}+B_{C})/M$ bps for transmission.

A similar process takes place during the reception of data. The BAs write received packets and data indications to the incoming portion of Q4, from which it is read by the CU. The Coding Unit decodes each packet, strips any coding header, and writes the results along with an updated data indication to the incoming portion of Q3. The ALP reads the decoded packet from Q3 and consults the accompanying access layer header to determine the packet handling method. In general, the access layer header is stripped from the packets and the packets are written to the incoming portion of Q2 accompanied by an updated data indication. The FPs read the packet from Q2 and generate a final data indication for the AP. This data indication and the data set are written to the incoming portion of Q1 from which they are finally read by the AP. This analysis demonstrates that for any given component $X$, $R_{BA}=R_{CULow}$, implying that the outgoing port data rates derived above may be doubled to approximate the full system data rates.

The bandwidth for each system component port is determined by combining its bandwidth requirements for both transmission and reception. Assuming half of the system bandwidth is consumed by reception and half by transmission, the data rates derived above for transmission are doubled to produce the full system data rates given in Equations 5.1 to 5.8.

Chapter 5 - Bus-Free Systems
\[ I = 2I_{out} = 2D \left( B_A + B_S \right) \]  \hspace{1cm} (5.1)

\[ R_{PPU} = 2R_{PPOut} = 2D \left( B_K + B_S \right) / N \]  \hspace{1cm} (5.2)

\[ R_{PPL} = 2R_{PPLout} = 2D \left( B_{R2} + B_H + B_S \right) / N \]  \hspace{1cm} (5.3)

\[ R_{ALPU} = 2R_{ALPOut} = 2D \left( B_{R3} + B_H + B_S \right) \]  \hspace{1cm} (5.4)

\[ R_{ALPL} = 2R_{ALPPL} = 2D \left( B_{R3} + B_A + B_H + B_S \right) \]  \hspace{1cm} (5.5)

\[ R_{CULU} = 2R_{CULOut} = 2D \left( B_{K3} + B_A + B_H + B_S \right) \]  \hspace{1cm} (5.6)

\[ R_{CUL} = 2R_{CULout} = 2D \left( B_{R4} + B_A + B_H + B_S + B_C \right) \]  \hspace{1cm} (5.7)

\[ R_{BA} = 2R_{BAout} = 2D \left( B_{R4} + B_A + B_H + B_S + B_C \right) / M \]  \hspace{1cm} (5.8)

The variable \( R_{BA} \), less the data request portion, defines the amount of data that must be transferred via the physical network medium. Therefore, \( R_{BA} \) closely approximates the required network throughput. Furthermore, data flows from an input to an output port of each system component. This means the throughput required of each component is essentially equivalent to sum of the bandwidths of its input ports. The above equations reflect the effects of both input and output data on each port. Since data flow is assumed to be equivalent in both directions, an individual port’s input bandwidth is half of its total bandwidth requirement.

Each multi-ported queue that supports this system consists of both a transmission queue and a reception queue. While these two portions of each multi-ported queue may be physically separate, the transmission and reception bandwidths are combined here to form a single equation for the throughput required of each queue. The throughput requirement for each queue is equal to the sum of the bandwidths of all ports writing to that queue. The throughput of Q1 is, therefore, the sum of the AP’s transmission bandwidth and the PPs upper port reception bandwidth. The throughput of Q2 is the sum of the bandwidths of the lower PP transmission ports and the upper ALP reception port. The throughput of Q3
is the sum of the bandwidths of the lower ALP transmission port and the upper CU reception port.

Finally, Q4’s throughput is mandated by the bandwidth of the lower CU transmission port and the upper reception ports of the BAs. These throughput requirements are summarized in Equations 5.9 to 5.12.

\[
T_{Q1} = R_{APout} + NR_{PPLim} = 2D( B_R + B_S )
\]
\[
T_{Q2} = R_{PPLout} + R_{ALPim} = 2D( B_R + B_H + B_S )
\]
\[
T_{Q3} = R_{ALPout} + R_{CUlim} = 2D( B_R + B_A + B_H + B_S )
\]
\[
T_{Q4} = R_{CUout} + MR_{Blim} = 2D( B_R + B_A + B_H + B_S + B_C )
\]

Assuming \( B_S \) is the dominant term, the reduced queue throughput equations become \( T_0 = 2DB_S \) for all queues. This reflects the fact that data sets must pass through each of the queues during both transmission and reception.

5.1.3. Analysis Results

The results of the analysis presented in Section 5.1.2 are summarized in Table 5-1. The simplified equations given in the table assume that the data length, \( B_S \), is the dominant term in the bandwidth equations.

The simplified equations show that all of the non-parallel system components must be able to handle data at the same rate it is produced by the AP. This means each one of these components, including the ALP, CU, and multi-ported queues, is a potential bottleneck to system performance. The data rate requirements for many of these components may be reduced, however, by physically separating the transmitting and receiving portion of the component. The queues are already separated in this fashion,
Table 5-1. System Component Bandwidths (bps)

<table>
<thead>
<tr>
<th></th>
<th>Bandwidth Equations</th>
<th>Simplified Equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I$</td>
<td>$2D(B_n + B_d)$</td>
<td>$2DB_s$</td>
</tr>
<tr>
<td>$R_{PPL}$</td>
<td>$2D(B_n + B_d)/N$</td>
<td>$2DB_s/N$</td>
</tr>
<tr>
<td>$R_{ALP}$</td>
<td>$2D(B_{nR} + B_{dH} + B_d)/N$</td>
<td>$2DB_s/N$</td>
</tr>
<tr>
<td>$R_{ALFL}$</td>
<td>$2D(B_{nR} + B_{dH} + B_d)$</td>
<td>$2DB_s$</td>
</tr>
<tr>
<td>$R_{CUU}$</td>
<td>$2D(B_{pR} + B_{dH} + B_d)$</td>
<td>$2DB_s$</td>
</tr>
<tr>
<td>$R_{CUL}$</td>
<td>$2D(B_{pR} + B_{dH} + B_d)$</td>
<td>$2DB_s$</td>
</tr>
<tr>
<td>$R_{BA}$</td>
<td>$2D(B_{pR} + B_{dH} + B_d)/M$</td>
<td>$2DB_s/M$</td>
</tr>
<tr>
<td>$T_{Q1}$</td>
<td>$2D(B_n + B_d)$</td>
<td>$2DB_s$</td>
</tr>
<tr>
<td>$T_{Q2}$</td>
<td>$2D(B_{pR} + B_{dH} + B_d)$</td>
<td>$2DB_s$</td>
</tr>
<tr>
<td>$T_{Q3}$</td>
<td>$2D(B_{pR} + B_{dH} + B_d)$</td>
<td>$2DB_s$</td>
</tr>
<tr>
<td>$T_{Q4}$</td>
<td>$2D(B_{pR} + B_{dH} + B_d)$</td>
<td>$2DB_s$</td>
</tr>
</tbody>
</table>

so it is not difficult to provide separate ports for transmission and reception on each queue. Furthermore, for most coding schemes the Coding Unit is separable into an encode and a decode section. Both of these sections may be provided with independent ports. In addition, although some data, e.g. node address lookup tables, may need to be shared by transmitting and receiving portions of the ALP, much of its operation may be split into separate transmit and receive portions. Finally, the PPs have a send and receive functionality similar to that of the ALP. They too must provide some shared data to their send and receive processes. Since they are already functioning in a parallel fashion, a method may have
already been devised to share data and the PPs may easily be split into send and receive components. These changes are reflected in the modified multi-ported queue system design depicted in Figure 5-2.

Figure 5-2. Modified multi-ported queue system.

As an illustration of the component port bandwidths required in a multi-ported queue system, consider the example used in Section 3.1.4. Once again, assume the AP requires an $I = 1$ Gbps network interface and all data packets are $B_s = 35$ Kbits long. Let the size of all data requests be $B_{R1}=B_{R2}=B_{R3}=B_{R4}=176$ bits. This request size is smaller than the request size used in the example of Section 3.1.4 due to the fact that the request is placed in the queue along with its data so that it is no longer necessary to include
a pointer to the data in the request. The fact that the request is kept with the packet also means the data request remains a constant size as it passes through the system since each component simply adds its information to the packet and does not need to add an additional pointer in the data request.

Assuming the interface bandwidth is equally distributed between transmission and reception, Equation 5-1 leads to

\[ D = \frac{I}{2(B_R + B_S)} = \frac{1\, \text{Gbps}}{2(176 + 35000) \frac{\text{bits}}{\text{sec}}} \approx 14.2 \frac{K\text{req}}{\text{sec}}. \]

Assume the system includes \( M = 10 \) BAs and \( N = 5 \) PPs, where the PPs add \( B_H = 320 \) bits of TCP and IP headers and the ALP adds a header of \( B_A = 272 \) bits. Furthermore, let the Coding Unit supply a header of

\[ B_c = B_A/M = 35 \text{ Kbits} / 10 = 3.5 \text{ Kbits}. \]

The required component bandwidths are then

\[
R_{PPU} = \frac{2D(B_R + B_S)}{N} = \frac{2(14.2K\text{req}) (176 + 35000) \frac{\text{bits}}{\text{req}}}{5} \approx 200 \text{ Mbits/s},
\]

\[
R_{PPL} = \frac{2D(B_R + B_H + B_S)}{N} = \frac{2(14.2K\text{req}) (176 + 320 + 35000) \frac{\text{bits}}{\text{req}}}{5} \approx 202 \text{ Mbits/s},
\]
\[ R_{ALPU} = 2D(B_{R_2} + B_{H} + B_3) = 2(14.2K \frac{\text{req}}{\text{sec}})(176 + 320 + 35000) \frac{\text{bits}}{\text{req}} \approx 1 \text{ Gbps}, \]

\[ R_{ALU} = R_{CUU} = 2D(B_{R_3} + B_A + B_2 + B_3) = 2(14.2K \frac{\text{req}}{\text{sec}})(176 + 272 + 320 + 35000) \frac{\text{bits}}{\text{req}} \approx 1.02 \text{ Gbps}, \]

\[ R_{CU} = 2D(B_{R_3} + B_A + B_2 + B_3 + B_{C}) = 2(14.2K \frac{\text{req}}{\text{sec}})(176 + 272 + 320 + 35K + 3.5K) \frac{\text{bits}}{\text{req}} \approx 1.12 \text{ Gbps}, \]

\[ R_{BA} = \frac{2D(B_{R_3} + B_A + B_2 + B_3 + B_{C})}{10} = \frac{2(14.2K \frac{\text{req}}{\text{sec}})(176 + 272 + 320 + 35K + 3.5K) \frac{\text{bits}}{\text{req}}}{10} \approx 112 \text{ Mbps}. \]

Most of these components have higher bandwidth requirements than the corresponding components in a single-bus or dual-bus system. These higher requirements are due to the fact that all of the components must transfer the full data set from one location to another. Components like the AP, CU and BAs that must transfer the full data set in the bus-based systems have the same bandwidth requirements here.

Assuming all multi-ported queues are identical, they must all support the largest queue throughput requirement. Therefore, for all queues,

\[ T_Q = 2D(B_{R_3} + B_A + B_2 + B_3 + B_{C}) = 2(14.2K \frac{\text{req}}{\text{sec}})(176 + 272 + 320 + 35K + 3.5K) \frac{\text{bits}}{\text{req}} \approx 1.12 \text{ Gbps}. \]

If the multi-ported queues provide 64-bit ports and only one port operation is performed per clock cycle, each queue must be clocked by a 17.5 MHz clock to provide a throughput of 1.12 Gbps. This is similar to the lowest clock speed required by any given bus in the bus-based systems. Memory used to construct the queues must have a 57 ns access time to support this clock rate. This is within the realm of current technology and implies that the multi-ported queues required for this example can be constructed using currently available technology.
5.2. Analysis of Other Bus-Free Systems

Some additional bus-free system designs use alternative components as direct replacements for the system bus. As a general rule, these system structures result in the same component bandwidth equations derived for the single-bus system. The only exception to this is the fact that the bus bandwidth equation becomes the bandwidth requirement for the device used as a bus replacement. Two of these designs are briefly discussed in Sections 5.2.2 and 5.2.3.

5.2.1. Variables Used in Analysis

In addition to the variables used in Section 5.1, the following variables are used in these analyses.

\[ R_{SMB} = \text{Multi-ported Shared Memory Unit bandwidth in bits per second} \]
\[ R_{NET} = \text{Interconnection Network bandwidth in bits per second} \]

5.2.2. Analysis of a Multi-ported Memory System

Another alternative to a bus-based structure is the multi-ported memory system. Systems based on multi-ported memory rely on a single multi-ported Shared Memory Unit to provide an interface between all system components as shown in Figure 5-3. Like the multi-ported queue system discussed in Section 5.1, this architecture eliminates buses and their associated limitations from the system. However, such a design requires a high speed memory unit. This type of memory unit is not currently available, but may be made practical by advances in technology. Even if available, the memory limits the scalability...
of the system since the required number of ports will change as parallel components are added or removed.

![Multi-ported memory system diagram](image)

Figure 5-3. Multi-ported memory system.

Assuming that the required SMU can be constructed, the AP, PP, ALP, CU, and BA component bandwidths required by this system are identical to those required for the modified single-bus system analyzed in Section 3.1.3. This is made clear by considering the fact that the SMU is simply substituting for the combination of the bus and SMU of the single-bus system. Since the SMU is used as a replacement for the bus, its aggregate bandwidth requirement becomes that of the bus as given in Equation 3.7.

\[
R_{SMU} = R_{BUS} = 2D (2B_R + 2B_{R2} + 2B_S + 2B_H + 2B_A + S_{AP,PP} + S_{PP,ALP} + S_{ALP,CU})
\]  

Equation 5.13

Chapter 5 - Bus-Free Systems
In addition to its aggregate bandwidth, the SMU's highest rate port is of interest. All of the ports to the SMU have bandwidth requirements as defined in the single-bus system analysis of Chapter 3. Considering these equations, it is evident that the highest rate port of the multi-ported memory system's SMU is the CU port which operates at the rate defined by Equation 3.4, or

\[ R_{CU} = 2D \left[ 2 \left( B_a + B_S + B_H \right) + B_C + S_{CU,ALP} \right]. \]

5.2.3. Analysis of an Interconnection Network System

A system based on an interconnection network might be constructed as shown in Figure 5-4. This system relies on an interconnection network to provide a link between the active components that process data and the passive components that simply store data. The Shared Memory Unit is the only passive component in the system and resides on one side of the interconnection network, allowing the other system components, attached to the opposite side of the interconnection network, to communicate. In this case, the Shared Memory Unit can be either a single multi-ported system or several smaller single-ported memory units.

Analysis of component data rates is exactly the same as the analysis done for the single-bus system, except that an equation must be developed for the interconnection network. By recognizing the similarity of the network to the bus in the basic single-bus system developed in Section 3.1.2, one can conclude that the aggregate bandwidth required for the interconnection network, \( R_{NET} \), is the same bandwidth required of the single bus.

\[ R_{NET} = 2D \left( 2B_a + 2B_{R2} + 4B_S + 4B_H + 4B_C + 2B_{CR} + S_{ALP-CPU} + S_{ALP-ALP} + 2S_{ALP-CU} + S_{ALP-BA} \right) \]  (5.14)

Chapter 5 - Bus-Free Systems
Similarly, moving the BAs off of the interconnection network and onto the Coding Logic results in the reduced requirement given by Equation 5.15.

$$R_{NET} = 2D(2R_R + 2B_{FL} + 2B_S + 2B_H + 2B_A + S_{AP-PP} + S_{PP-ALP} + S_{ALP-CU} )$$ (5.15)

Like the multi-ported memory system, the highest data rate port of this system is of interest. Unlike the multi-ported memory system, however, that component is no longer the CU. Instead it is the SMU, whose port bandwidth must be equal to the interconnection network aggregate bandwidth.

5.3. Functional Modeling of a Bus-Free System

Of the three system designs presented in this chapter, the multi-ported queue system discussed in Section 5.1 is both the most viable and the most unlike the bus-based systems presented in Chapters 3 and 4.
A VHDL model is developed for the multi-ported queue system to provide insight into its design, demonstrate its data transfer characteristics, and allow comparison with the bus-based systems.

The basis for the multi-ported queue system is the queue. The queue model developed for this simulation consists of a single memory unit with an arbitrary number of read/write ports. Each port consists of 134 signals. There are 66 signals used for reading and 66 signals used for writing. The read and write portions each consist of 64 bits for data transfer, a request bit, and an acknowledge bit. In addition to the read and write lines, one-bit data ready and queue full signals are provided.

Queue access is done using a request/acknowledge protocol as shown in Figures 5-5 and 5-6.

```
DATAIN <= DATA;  -- Put data on port write line
WR <= '1';        -- Signal queue
wait until WACK = '1';  -- Wait for acknowledge
WR <= '0';        -- Drop request
```

**Figure 5-5. Typical queue write transaction.**

```
RD <= '1';  -- Request read
wait until RACK = '1';  -- Wait for acknowledge
RD <= '0';  -- Drop request
DATA <= DATAOUT;  -- Read data from queue port
```

**Figure 5-6. Typical queue read transaction.**

The basic functionality of the AP, PPs, ALP, CU, and BAs is provided by the same VHDL code used to model these components for the bus-based systems of Chapters 3 and 4. Instead of using bus transactions to move data into and out of the components, however, the read and write routines given
in Figures 5-5 and 5-6 are used. In addition, most component control signals are routed through the queues rather than being directly applied via the bus.

One example of a modified signalling scheme is the method by which the PPs are notified of outgoing data availability. In the bus-based systems, the AP signals the PPIC to indicate the presence of a new packet. In this system, however, the PPs use the data ready signal from Q1 to determine data availability.

Queue access among parallel system components is controlled by a set of control tokens that pass along a daisy chain. In the case of the PPs, access must be controlled to both Q1 and Q2. This control is required to assure that a full packet is read from a queue or written to a queue as a single unit. Each PP is provided with an input control signal from one neighboring PP and provides an output control signal to its other neighboring PP. This logic as used in the PP models is shown in figures 5-7 and 5-8. The two VHDL processes used to provide this control is shown in Figures 5-9 and 5-10. The BAs use this same type of token passing scheme to control access to Q4.

Read tokens for each queue pass along the control chain until they arrive at a PP that is not busy. This PP holds the token and waits for the associated queue's data ready signal. When that signal is asserted, the PP reads a packet from the queue, releases the token to the next PP, and then processes the packet. Once it has performed any necessary packet modifications, it waits for the write token for the queue opposite the one from which the data was read. After it acquires the write token it writes the packet to the queue, releases the token, and returns to an idle state.
Figure 5-7. PP control input state diagram.

Figure 5-8. PP control output state diagram.
CONTROL_READ: process
begin
  wait on PPIN;
  if PWORKING and (PPIN /= "000") then
    -- If input control is active and PP is waiting
    -- for a write token then hold the token.
    if (PPIN = UWT) and (W_WAIT) then
      PPUWT2 <= TRUE;
    elsif (PPIN = LWT) and (L_WAIT) then
      PLLWT2 <= TRUE;
    -- If input control is active and PP is working
    -- but not waiting for a queue, then pass the token.
    elsif (PPIN = UWT) then UMTSIG1 <= TRUE;
    elsif (PPIN = LWT) then LMTSIG1 <= TRUE;
    elsif (PPIN = URT) then URTSIG1 <= TRUE;
    elsif (PPIN = LRT) then LRTSIG1 <= TRUE;
  end if;
elsif PPIN /= "000" then
  -- If input control is active and PP is not busy,
  -- then hold read tokens and pass write tokens.
  if PPIN = URT then PPURT2 <= TRUE;
  elsif PPIN = LRT then PLLRT2 <= TRUE;
elsif PPIN = UWT then UMTSIG1 <= TRUE;
elsif PPIN = LWT then LMTSIG1 <= TRUE;
end if;
end if;
end process CONTROL_READ;

Figure 5-9. Protocol Processor input control process.

CONTROL_WRITE: process
  variable PPNEW: BIT_VECTOR/2 downto 0;
begin
  -- Set output control signal at inactive state
  PPOUT <= "000";
  wait for 1 ns;
  -- Wait for internal PP signal to pass token
  if not (URTSIG or UMTSIG or LMTSIG or LRTSIG) then
    wait until URTSIG or UMTSIG or LMTSIG or LRTSIG;
  end if;
  -- Pass token to next PP
  if URTSIG then PPNEW := URT; URTSIG2 <= FALSE;
  elsif LMTSIG then PPNEW := LMT; LMTSIG2 <= FALSE;
  elsif UMTSIG then PPNEW := UWT; UMTSIG2 <= FALSE;
  elsif LRTSIG then PPNEW := LWT; LRTSIG2 <= FALSE;
  else assert FALSE report "Bad Control Write Signal";
  end if;
  PPOUT <= PPNEW;
  wait for 1 ns;
end process CONTROL_WRITE;

Figure 5-10. Protocol Processor output control process.
5.4. Simulation Results

The VHDL model described in Section 5.3 is used to simulate PLAN system operation under various load conditions. All simulations are performed using the same type of two-node system under the same assumptions and conditions described in Section 3.3.

The simulations performed vary the data set size from 128 bytes to 3,840 bytes while the code set is fixed at 128 bytes. Tables 5-3 to 5-9 indicate the utilization characteristics of the various model components under these loads. Each column in the table represents a single simulation. The column’s packet size parameter is the size of the data set used for that simulation and the coding parameter indicates the state of the Coding Unit during the simulation. Utilization results are generated over the time required for 30 packets to arrive at the receiving node’s AP.

All tables depict the percent of time each component spends in working and idle states for each simulation. An idle component is waiting for new packet data to become available for it to process, while a working component is processing a packet. The wait states included in the results given in Chapters 3 and 4 translate into time spent waiting for a queue to become available for writing. Since these times are negligible in this design, no wait states are included in the tables.

Tables 5-2 and 5-9 indicate the utilization of a typical PP in the transmitting and receiving nodes, respectively. All but one of the PPs in each node have a similar load since the token passing scheme employed guarantees an equal load balance. The two exceptions to this are the PP that initially captures the Q2 read token in the transmitting node and the PP that captures the Q1 read token in the receiving node. The transmitting node’s PP waits for incoming data and the receiving node’s PP waits for
outgoing data to become available. Since the transmitting node does not receive data and the receiving node does not transmit data during this simulation, these PPs remain idle.

Utilization results for the transmitting ALP are given in Table 5-3 and for the receiving ALP in Table 5-8. Unlike the results given for the bus-based systems, these tables do not include any percentage of time spent waiting for other system components. This is due to the fact that the components in the multi-ported queue structure are essentially independent of one another. The ALP does provide a signal to the CU to indicate the current coding state (on or off) and a signal to the BAs to control the channel through which a packet is to pass. In this simulation, all of this information is included in the data request and placed into Q3 along with the packet for transmission.

Unlike the bus-based simulations of Chapters 3 and 4, the results listed in Tables 5-4 and 5-7 for the transmitting and receiving Coding Units include the component utilization for situations in which coding is turned off. This is due to the fact that each component in a multi-ported queue system must pass the full packets between queues. For example, the transmitting node’s CU must transfer packets from Q3 to Q4 whether they are transmitted with or without coding.

Table 5-5 lists the utilization results for the transmitting node’s Base Adapters. All adapters are evenly loaded since they use the same token passing scheme employed by the Protocol Processors to determine data availability and manage queue access. Nearly all of the simulations fully utilize the Base Adapters, driving their time spent working to 100 percent. This means that for these simulations, the transmitting BAs are the primary bottleneck to data transfer.

Finally, Table 5-6 provides the utilization results for the receiving node’s Base Adapters. Two sets of state information are given for these adapters. The first set, labeled "fiber," is generated by the process
that receives incoming data on the FDDI fiber. This set of utilization data represents both the utilization of a typical receiving BA’s FDDI input port and the utilization of the fiber-optic connection between the sending and receiving nodes. The second set of data is the utilization of the receiving BA’s port to Q4.

Latency is a measure of the time required to transfer a single packet from one point to another in a system. In this system, a packet is considered to have entered the transmitting node when its last bit has been read from Q1 by one of the transmitting PPs. It is considered to have left the transmitting node when its last bit has been placed on the physical medium connecting the stations. The send latency reported in Table 5-10 is the difference between these two times.

Similarly, a packet is considered to have entered the receiving node when its last bit is read from the physical medium by one of the receiving node’s BAs. Finally, it is considered to have arrived at its destination when its last bit is placed into Q1 by one of the receiving node’s PPs. The time differential between the point that the packet enters the receiving node and the time that it is placed in Q1 of that node is reported as the receive latency in Table 5-10.

The system latency reported in Table 5-10 is the difference between the time that a packet enters the transmitting node and the time that it arrives at its destination in the receiving node.

Throughput is a measure of the amount of data that is transferred from one point to another in a given amount of time. The throughput results given in Table 5-10 are calculated by dividing the amount of data transferred in 30 packets by the time required to transfer them from the sending node to the receiving node.
All of the values reported in Table 5-10 are the result of averaging the values produced by moving 30 packets through the system. Unlike Chapters 3 and 4, the first packet is not discarded. In the prior chapters, the systems required a packet in each active component to bring the bus to its full utilization and the average bus wait times to a steady state. Unlike bus-based systems, wait times in a multi-ported queue system do depend on any components but those that access a particular queue. For example, the amount of time a PP spends waiting for queue access in a single-bus system depends on BA access to the system's SMU. In multi-ported queue systems, Q1 and Q2 access times and utilization do not depend on BA activity. This means that the system does not have to be fully loaded for queue wait times to reach a steady state.

5.5. Summary

The multi-ported queue system modeled here has both advantages and disadvantages when compared with the dual-bus system of Chapter 4. This design provides slightly higher throughputs for cases in which coding is required, while the dual-bus system provides slightly higher throughput for the cases in which coding is not required. This design also provides an even loading of all parallel components without requiring a centralized controller to coordinate the parallel activities, e.g. a PPIC. Finally, while the dual-bus system requires fewer custom-designed components, its expandability may be restricted by the components chosen. Since the multi-ported queue system components are custom designed for the system, they do not have this limitation.

A more thorough comparison of the bus-based systems developed in Chapters 3 and 4 and the multi-ported queue system developed here is presented in Chapter 6. There, conclusions are drawn regarding
the suitability of certain systems to particular applications and recommendations are made regarding the
development of a PLAN system using existing technology.
### Table 5-2. Sending Node PP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>56</td>
<td>56</td>
<td>59</td>
<td>58</td>
<td>60</td>
<td>60</td>
<td>59</td>
</tr>
<tr>
<td>Working</td>
<td>44</td>
<td>44</td>
<td>41</td>
<td>42</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
</tbody>
</table>

### Table 5-3. Sending Node ALP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>17</td>
<td>19</td>
<td>15</td>
<td>14</td>
<td>18</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>Working</td>
<td>83</td>
<td>81</td>
<td>85</td>
<td>86</td>
<td>82</td>
<td>82</td>
<td>81</td>
</tr>
</tbody>
</table>

### Table 5-4. Sending Node CU Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>2</td>
<td>4</td>
<td>20</td>
<td>22</td>
<td>32</td>
<td>34</td>
<td>28</td>
</tr>
<tr>
<td>Working</td>
<td>98</td>
<td>95</td>
<td>80</td>
<td>78</td>
<td>68</td>
<td>66</td>
<td>71</td>
</tr>
</tbody>
</table>
### Table 5-5. Sending Node BA Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Working</td>
<td>100</td>
<td>85</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

### Table 5-6. Receiving Node BA Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>8</td>
<td>13</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Fiber</td>
<td>10</td>
<td>37</td>
<td>8</td>
<td>12</td>
<td>8</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Bus</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Working</td>
<td>92</td>
<td>87</td>
<td>92</td>
<td>90</td>
<td>92</td>
<td>91</td>
<td>92</td>
</tr>
<tr>
<td>Fiber</td>
<td>90</td>
<td>63</td>
<td>92</td>
<td>88</td>
<td>92</td>
<td>91</td>
<td>92</td>
</tr>
<tr>
<td>Bus</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 5-7. Receiving Node CU Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>Idle</td>
<td>45</td>
<td>7</td>
<td>30</td>
<td>14</td>
<td>28</td>
<td>79</td>
<td>28</td>
</tr>
<tr>
<td>Working</td>
<td>55</td>
<td>93</td>
<td>70</td>
<td>86</td>
<td>72</td>
<td>81</td>
<td>72</td>
</tr>
</tbody>
</table>

### Table 5-8. Receiving Node ALP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>56</td>
<td>8</td>
<td>37</td>
<td>16</td>
<td>31</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td>Working</td>
<td>44</td>
<td>92</td>
<td>63</td>
<td>84</td>
<td>68</td>
<td>80</td>
<td>70</td>
</tr>
</tbody>
</table>

### Table 5-9. Receiving Node FP Utilization (percent)

<table>
<thead>
<tr>
<th>Packet Size</th>
<th>128 Bytes</th>
<th>640 Bytes</th>
<th>1280 Bytes</th>
<th>1920 Bytes</th>
<th>2560 Bytes</th>
<th>3200 Bytes</th>
<th>3840 Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>Idle</td>
<td>80</td>
<td>54</td>
<td>69</td>
<td>59</td>
<td>28</td>
<td>60</td>
<td>65</td>
</tr>
<tr>
<td>Working</td>
<td>20</td>
<td>46</td>
<td>31</td>
<td>41</td>
<td>72</td>
<td>40</td>
<td>35</td>
</tr>
<tr>
<td>Packet Size</td>
<td>128 Bytes</td>
<td>640 Bytes</td>
<td>1280 Bytes</td>
<td>1920 Bytes</td>
<td>2560 Bytes</td>
<td>3200 Bytes</td>
<td>3840 Bytes</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-----------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>Coding</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
<td>Off</td>
<td>On</td>
</tr>
<tr>
<td>System Throughput (Mbps)</td>
<td>151</td>
<td>355</td>
<td>302</td>
<td>396</td>
<td>343</td>
<td>400</td>
<td>359</td>
</tr>
<tr>
<td>System Latency (µs)</td>
<td>77</td>
<td>18</td>
<td>142</td>
<td>81</td>
<td>225</td>
<td>163</td>
<td>308</td>
</tr>
<tr>
<td>Send Latency (µs)</td>
<td>71</td>
<td>13</td>
<td>123</td>
<td>64</td>
<td>190</td>
<td>129</td>
<td>257</td>
</tr>
<tr>
<td>Receive Latency (µs)</td>
<td>6</td>
<td>5</td>
<td>18</td>
<td>18</td>
<td>34</td>
<td>34</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 5-10. System Throughput and Latency
Chapter 6. Comparison of Options and Recommendations

The simulation results of Chapters 3 through 5 are compared in Chapter 6 to determine the relative merits of each system. Based on these comparisons, recommendations are made concerning system construction.

6.1. Comparison of Results

6.1.1. Throughput

Of primary interest is the throughput that can be achieved by a PLAN system. The theoretical maximum data throughput for a PLAN system is calculated by summing the maximum throughputs of all attached Base Adapters and scaling the result to eliminate header overhead. For any given system, this maximum data throughput ($T_{\text{max}}$) is calculated using Equation 6.1.

$$T_{\text{max}} = M T_b \frac{B_s}{B_s + H}$$

(6.1)

In this equation, $M$ is the number of Base Adapters, $T_b$ is the maximum throughput of a single Base Adaptor, $B_s$ is the number of data bits per packet, and $H$ is the number of header bits per packet sent to a BA for transmission.
As an example, consider a system in which there are $M = 5$ Base Adapters, each of which has a maximum throughput of $T_B = 100$ Mbps. Let the length of a data set be $B_d = 1,280$ bytes and the length of a packet header be $H = 202$ bytes. Then the maximum throughput for this system is

$$T_{\text{max}} = MT_B \frac{B_d}{P_d + H} = 5(100 \text{ Mbps}) \left( \frac{1280 \text{ bytes}}{1280 \text{ bytes} + 202 \text{ bytes}} \right) = 432 \text{ Mbps.}$$

The simulations performed in Chapters 3 through 5 provide throughputs for single-bus, dual-bus, and multi-ported queue system structures. The systems simulated all include $M = 5$ FDDI Base Adapters, where the FDDI standard provides $T_B = 190$ Mbps [MCC087]. In addition, all simulations add TCP, IP, and ALP headers for an aggregate header length of $H = 74$ bytes. Finally, the simulations in which coding is used add an additional 128-byte coding header to make the total header length $H = 202$ bytes. These parameters are used to determine the maximum system throughput for each data set size simulated and are graphically compared to the measured simulation results in Figures 6-1 and 6-2.

These graphs show a significant increase in the theoretical throughput as the amount of data becomes much larger than the size of the headers included in a packet. This increase in throughput is also seen in the system simulations as the packet size is changed from 128 bytes to 1280 bytes. In the instances in which coding is turned off, however, the theoretical throughput continues to rise as packet size is increased beyond 640 bytes while the throughput attained in all three simulated systems remains relatively constant. Since no significant gain or loss in system throughput is observed for data set sizes of 1280 bytes and above, the systems’ data set size is not important as long as it is kept much greater than the aggregate header size.
Figure 6-1. Throughput comparison (coding on).

Figure 6-2. Throughput comparison (coding off).
Figure 6-3. Throughput comparison zoom (coding on).

Figure 6-4. Throughput comparison zoom (coding off).
The flat response at 400 Mbps that is evident in the simulation results is indicative of a saturation of the protocol implementation. In other words, the response is not due to any specific component within the model, but is instead due to the manner in which the components work together. This response indicates the protocol implementation itself is creating a bottleneck to system performance.

Figures 6-3 and 6-4 show more detail of the system responses with packet sizes above 540 bytes. Figure 6-3 compares the three simulated systems for instances in which coding is not being used. It is evident from this graph that the multi-ported queue system provides the worst performance of the three systems simulated. This is due to the fact that the multi-ported queue system must transfer the full data set between each system component while the bus-based systems only transfer the data set between the BAs, the SMU, and the ALP. When coding is turned off, as in Figure 6-4, all three systems perform equally, with the multi-ported queue system giving slightly better performance than the bus-based systems.

6.1.2. Latency

In addition to throughput, latency is a concern in selecting a PLAN system architecture. Figures 6-5 and 6-6 graphically compare the latencies of the three systems simulated.

These figures show that packet latency increases as packet sizes increase. This is expected, since larger packets take longer to transfer through the various system components. Also expected is the fact that the multi-ported queue system generally has larger packet latencies than the bus systems. This is due to the fact that the multi-ported queue system modeled must transfer the full data set between each system component. The bus-based systems only transfer the entire data set between the shared memory, the AP, and the BAs. The extra transfers of a large data set require additional time.
Figure 6-5. Latency comparison (coding on).

Figure 6-6. Latency comparison (coding off).
Packet latencies for both of the bus-based systems are similar. In fact, they appear coincident in Figure 6-6. Based on Figures 6-5 and 6-6, it is evident that the bus-based systems are a better choice for a PLAN if minimal latencies are of utmost importance.

6.13. Utilization

All of the systems modeled heavily utilize a few system components while leaving others idle. In all systems, the Base Adaptors are actively transferring a packet over 95 percent of the time. In addition, the bus-based systems heavily utilize the system buses. These components are potential bottlenecks to system performance. In the case of the BAs, the bottleneck may be relieved by adding additional BAs. In the case of the system buses, however, measures must be taken to move traffic off of the bus. This was the general strategy that lead to the development of the dual-bus and multi-ported queue systems.

Even if additional Base Adaptors are used and bus-related bottlenecks are removed, eventually other system components will reach a saturation point in terms of data throughput. The primary candidates for this type of saturation are the ALP and the CU. A way must be found to build these system elements either from high-speed components so that they can handle any potential data flow rates, or from lower-speed parallel components. The second choice is preferable, and more in keeping with the spirit of the PLAN system, but requires that any coding scheme used can be partitioned between parallel coding elements and that a scheme can be found to parallelize the function of the ALP.
6.2. **Recommendations**

Based on the simulation results and discussion in Section 6.1, recommendations may be made for implementing specific instances of a PLAN.

The single-bus system is suited to experimentation and quick prototyping. Since it can be built by adding commercially available components to an existing system bus structure, little development time is required to construct the physical system. All efforts may, therefore, be directed toward developing system software and measuring the resultant system parameters.

The single-bus system is also suited to situations in which a relatively low, fixed, throughput is desired. If a network is to be developed that only requires a few times the throughput of a single Base Adaptor, then this system structure is ideal.

The single-bus system, however, does heavily utilize the bus to which the PLAN components are attached. If a computer system contains only a single system bus, then general memory and component access will be restricted by the PLAN operation. In addition, the system bus may limit the number of BAs that can be included in a system by imposing limits on both the number of devices that can be connected to the bus and the maximum throughput that can be attained by the bus.

Dual-bus systems do not appear to have any significant performance advantages over similar single-bus systems. However, they do unload some PLAN processing from the bus to which the AP is attached. They are, therefore, suited to systems in which greater throughputs are desired or lower throughput buses
are used. In addition, by moving some components to a second bus additional parallel components may be attached to systems that employ buses that limit the number of attached devices.

The multi-ported queue structure completely dissociates the AP from the PLAN implementation and requires only minimal software support in the host system. It removes all internal PLAN traffic from the host's system bus and is the least intrusive of the system options investigated. Multi-ported queue PLAN implementations are, therefore, best suited to systems that do not have additional AP bus bandwidth to support network activity. These systems do, however, require a significant amount of initial hardware development since each system component must be custom designed.

Finally, it should be noted that a PLAN does not require that all nodes be constructed in the same manner. In fact, a typical PLAN may contain nodes that are a mixture of the three structures discussed above. For example, the PLAN might contain several small computers that are connected using only a single BA in a single-bus structure, some mid-sized computers that use a few BAs in a dual-bus configuration, and a large computer connected to the network through a multi-ported queue system that uses several BAs.

6.3. Other Issues

In addition to latency, throughput, and utilization under various load conditions, the VHDL models developed provide insight into the PLAN interface structure and implementation techniques.

For example, it may be observed that the dual-bus system provides a clear break point for dividing a system into two components. The lower bus and upper bus may be contained within separate enclosures.
and connected via the SMU and ALP. This physical separation of the system components makes it possible to develop and test both bus units independently. In fact, for systems in which the Protocol Processors are implemented in software by the host, a dual-bus system allows most PLAN hardware to be moved outside of the AP case.

A multi-ported queue system also allows for physical separation of the various PLAN stages. Each stage may be constructed independently, then assembled into a single unit. If the PPs, ALP, CU, and BAs are assembled in a unit that is externally attached to an AP, the interface may be designed and tested prior to attaching it to an AP.

6.4. Summary

Chapters 3 through 5 investigated several options for PLAN system structure. These options were compared in Chapter 6 and recommendations were made regarding the suitability of each of the investigated structures to specific application goals.

The single-bus system is the most likely candidate for initial development and further testing of PLAN systems since it can be quickly assembled and provides throughput comparable to the dual-bus and multi-ported queue systems at AP data rates up to 500 Mbps. In the long run, limitations of the single-bus system will restrict the PLAN.

For non-experimental systems that will be used to provide high data rate network service, the dual-bus and multi-ported queue systems appear to be the best candidates for system structure. These systems interfere the least with normal system processing and provide the greatest degree of expandability.
Chapter 7. Conclusions

This research investigated the implementation of a high-speed Parallel Local Area Network interface. Several design options were considered and hardware models were developed to investigate the construction and evaluate the capabilities of a selected few designs. The models were compared and recommendations were made concerning the suitability of each design option to specific tasks. This work is summarized and opportunities for further research are discussed below.

7.1. Summary of Work

Parallelism is used in many aspects of high-speed computing. However, computer communications have yet to fully realize its benefits. Several approaches to high-speed networking do exist, but many require radically new protocols or hardware. The General Parallel Network (GPN) model [WEIN92, KUMA93] provides an evolutionary, rather than revolutionary, protocol that can be supported by existing hardware. The hardware realization of the GPN model is the high-speed Parallel Local Area Network (PLAN) system. This system offers several advantages over traditional monolithic networks, including scalability, fault tolerance, low cost, high data rates, and distribution of high-speed network traffic to lower speed hosts.

A PLAN system depends on a high data rate path between an Application Processor (AP) and the Base Adapters (BAs) that provide network connections. Chapters 2 through 5 described several alternative designs for the AP to BA path and the general implementation of a PLAN system. Three basic designs
were investigated, including single-bus, dual-bus, and bus-free systems. VHDL models were developed for each of these designs to provide insight into interface construction techniques and to evaluate system performance.

The three models were compared in Chapter 6 and recommendations were made concerning the suitability of each design option to specific tasks. These comparisons indicate the single-bus system design is best suited to initial research since it is readily implemented using existing hardware and has the ability to provide throughputs similar to those of the dual-bus and bus-free systems for a small number of Base Adapters. The single-bus system does, however, have the potential to monopolize the host’s system bus and interfere with general AP processing tasks. For general networking applications, the multi-ported queue bus-free system is the recommended system structure. This design alternative removes all network processing overhead from the AP and its system bus and allows a larger number of PPs and BAs to be attached to a single interface.

7.2. Contributions of this Research

The PLAN system is a unique way of looking at network system design. It allows existing network hardware to be combined to form new, more powerful networks. This research was an initial investigation into the design of network interfaces for these systems. It forms a basis for continued study of this network structure and demonstrates its feasibility.

By dividing the design of a PLAN system interface into three general categories, this research forms a basic taxonomy that provides a framework for PLAN interface implementation. All three branches of
this taxonomy were shown to be valid options by developing interface models within them. Each of the
models developed can be fabricated using existing technology and may be expanded as technology
improves.

The VHDL system models developed in Chapters 3 through 5 provide a solid basis for additional study
of PLAN systems. The models are readily modifiable to study the effect of varying system parameters
and components. The choice of VHDL as a modeling tool and the design of the system models allow
each PLAN system component to be individually developed and tested down to the chip level prior to
actual hardware construction.

Finally, measurements taken from the VHDL models allow recommendations to be made regarding the
suitability of each system structure for further research and development. All three interface are shown
to provide increased throughput over the monolithic system from which they are built. Each option is
worthy of further study. Several areas that could benefit from further investigation are given in Section
7.3.

7.3. Future Research Opportunities

Additional investigation is needed into the effects of varying the number and type of Base Adapters. In
particular, the amount of bandwidth contributed to the system by each BA, as a function of the BA’s
total bandwidth, should be determined.

Chapter 7 - Conclusions
Investigation should also be done into techniques for performing both Access Layer Processor (ALP) and Coding Unit (CU) operations in parallel. These two entities are the only serial components left in a PLAN system and are the most likely to present bottlenecks to system throughput. A parallel CU depends upon a divisible coding scheme. Coding schemes that are able to encode information in a parallel fashion with minimal communication between the parallel components are ideally suited to PLAN systems. These schemes should be readily implemented within the existing PLAN structures. Division of ALP tasks may be more difficult, however, since synchronization of access layer header creation is required. This may be done by providing a global network address map to the ALP components and by providing a method for synchronizing packet ID generation.

In addition to the above investigations, a method for moving data through the multi-ported queue system should be developed that does not require the transfer of full data sets through all system components. This would dramatically reduce both the system's latency and the amount of memory required by the system components.

Finally, hardware development and implementation of PLAN systems should be pursued. A fully operational PLAN will most effectively demonstrate the advantages and disadvantages of the PLAN structure. In addition, the development of PLAN hardware will provide an excellent medium for the encouragement of continued PLAN system development.
References


[BLAC93] Black Box Corp., Black Box Catalog, Black Box Corp., Pittsburgh, PA, Spring/Summer 1993.


Vita

Scott Harper was born in Huron, South Dakota on December 30, 1964. He received his high school diploma from Chippewa Falls Senior High School, Chippewa Falls, Wisconsin in May 1983. His Bachelor of Science degree in Electrical Engineering was granted by the University of Wisconsin - Platteville in December of 1989.

He began work toward the Master of Science degree in Electrical Engineering at Virginia Polytechnic Institute and State University in August of 1990 and completed the requirements for the degree in August of 1994.

Scott Harper