MERLIN: A Multiprocessor Design

For a MicroChannel Architecture

by

Neal M. Lackritz

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical and Computer Science and Engineering

at the Massachusetts Institute of Technology

January, 1989

Author_____
Department of Electrical Engineering and Computer Science
January 6, 1989

Certified by_____
Professor Carl Hewitt
Thesis Supervisor

Accepted by_____
Leonard A. Gould
Chairman, Department Committee on Undergraduate Theses

MERLIN:
A Multiprocessor Design For A MicroChannel Architecture

by

Neal M. Lackritz

Submitted to the
Department of Electrical Engineering and Computer Science

January 6, 1989

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Engineering and Computer Science

Abstract

Computers of the 1990's will need to take advantage of highly parallel computing models in order to keep pace with the increasing hardware and performance demands of software systems. One possible design for future desktop computers would be a chassis containing only input and output facilities that allowed multiple heterogeneous processors to be installed onto a connecting channel. Such a design allows for an easily expandable, but powerful workstation.

This thesis discusses the design of a multiprocessor expansion card (MERLIN) for the MicroChannel based IBM PS/2 family. The MERLIN card provides plug-in processing capability for the PS/2 family by allowing multiple 80386 based systems to be installed on the MicroChannel. The resulting workstation contains multiple, object code compatible processors working in parallel.

This thesis describes the MicroChannel architecture of the IBM PS/2 family and presents the design of the MERLIN card along with a comparison with other existing multiprocessors.

Thesis Supervisor: Carl Hewitt
Title: Associate Professor of Computer Science and Engineering

Acknowledgments

This work was done for Professor Carl Hewitt of MIT's Laboratory for Computer Science in conjunction with the IBM Cambridge Scientific Center. The thesis has not been reviewed by any IBM publication board and represents the ideas of the author.

Thanks are given to David Saul at IBM for providing a stimulating environment for individual research, Professor Carl Hewitt for supervising the project, and John Webster for his support and guidance.

Table of Contents

List of Illustrations

1.0 Introduction

In April of 1987, International Business Machines (IBM) announced a family of microcomputers. These computers, officially known as the IBM Personal System/2[1] or PS/2 family, were designed to appeal to both low-end personal computer users and high-end workstation users. The family members, while differing in processing capability, share the upwardly compatible Intel 80x86 microprocessor series and the IBM MicroChannel Architecture.[2] The MERLIN board specified in this thesis was designed specifically for the IBM PS/2 Model 70 and IBM PS/2 Model 80. Both these machines have 32-bit Intel 80386 microprocessors, video output, serial and parallel interfaces, up to 16 Megabytes of random access memory, large hard disk drives, and peripheral slots giving access to the MicroChannel.

The designs of the Model 70 and Model 80 make them appropriate for scientific and high-speed computing applications. When an interface allowing network communication (eg. Token Ring or Ethernet) is added to these systems, they are well suited to act as network disk servers or computational servers in a distributed processing environment. Adding processors to these systems greatly enhances their capability. For example, in a multitasking environment, each processor can be given a separate process. This arrangement increases overall system throughput over single processor systems that timeslice between several processes.

---

[1] Personal System/2, MicroChannel, and OS/2 are trademarks of the International Business Machines Corporation.

[2] The IBM PS/2 Models 25 and 30 and the IBM PC Convertible do not use the MicroChannel. The MERLIN design applies only to those PS/2 members that implement the MicroChannel.

Operating systems such as IBM's OS/2 divide processes into threads of computation. In a single processor PS/2, the microprocessor performs bursts of computation on each thread, constantly alternating between threads. A multiprocessor PS/2 could assign threads to separate processors. The computation of individual process components may then proceed in parallel, leading to reduced process execution time.

In the coming years, the use of multiprocessor systems will become commonplace as software engineers begin to exploit the processing capabilities of parallel computers. MERLIN serves as a stepping stone for bringing parallel computers into the workstation market.

1.1 Thesis Objective

This thesis was motivated by work being done at MIT's Message Passing Semantics Group on a distributed computing environment. Since the system was being developed on IBM PS/2 systems, it was a natural decision to choose IBM's MicroChannel as a development vehicle for MERLIN. This decision greatly simplifies the software conversion task necessary to convert the environment to effectively utilize MERLIN's multiprocessing capability. This decision also motivated the choice to use Intel's 80386 as the processor engine for the MERLIN card.

MERLIN was designed with the features and limitations of the MicroChannel in mind (see "IBM MicroChannel Architecture" on page 6 for a description of the MicroChannel). MERLIN was architected for use in the IBM PS/2 Model 80 and further enhanced for use in the IBM PS/2 Model 70. These machines, while similar in overall architectural features, differ in their use of the MicroChannel due to the Model 70's cache memory.

Physical constraints, such as the number of 32-bit MicroChannel slots and power supply ratings, limited the design of the MERLIN card. However, MERLIN was designed with future technology

in mind. As a result, MERLIN can be integrated easily into future members of IBM's PS/2 MicroChannel family.

In comparing MERLIN's design with existing multiprocessors, it should be recognized that MERLIN was developed as an enhancement card for the MicroChannel. The Model 70 and Model 80 were not designed with multiprocessing as their central focus. As a result, these machines are not optimized for the requirements of a multiprocessor system.

1.2 Multiprocessor versus Co-Processor

Often the terms *multiprocessor* and *co-processor* are used interchangeably; however, in this discussion, a clear distinction is made. Every microcomputer is powered by a microprocessor, like Intel's 80386 or Motorola's 68030. A *co-processor* refers to a piece of hardware that functions as a slave device for the main microprocessor. For example, Intel's 80387 math co-processor operates as a slave to the 80386. On command from the microprocessor, the co-processor performs a computation. The result is then directed by the microprocessor to the correct destination. Co-processors, acting as slaves, are used to supplement the speed or capability of a microprocessor.

A *multiprocessor* refers to an additional, independent microprocessor that operates in parallel with all other processors in the system. Each node in a multiprocessor system may have primary and secondary storage, input and output facilities, and co-processors (see Figure 1 on page 4). Each node in a multiprocessor system has nearly equal capability. What clearly distinguishes a co-processor from a multiprocessor system is the slave like operation of a co-processor. Co-processors are used to provide special purpose computations on demand from a master microprocessor. Multiprocessor nodes provide general computational functions.

The channel (often referred to as a bus) provides the communication facilities between the microprocessors. Thus, a multiprocessor based computer might have 5 microprocessors, each with a

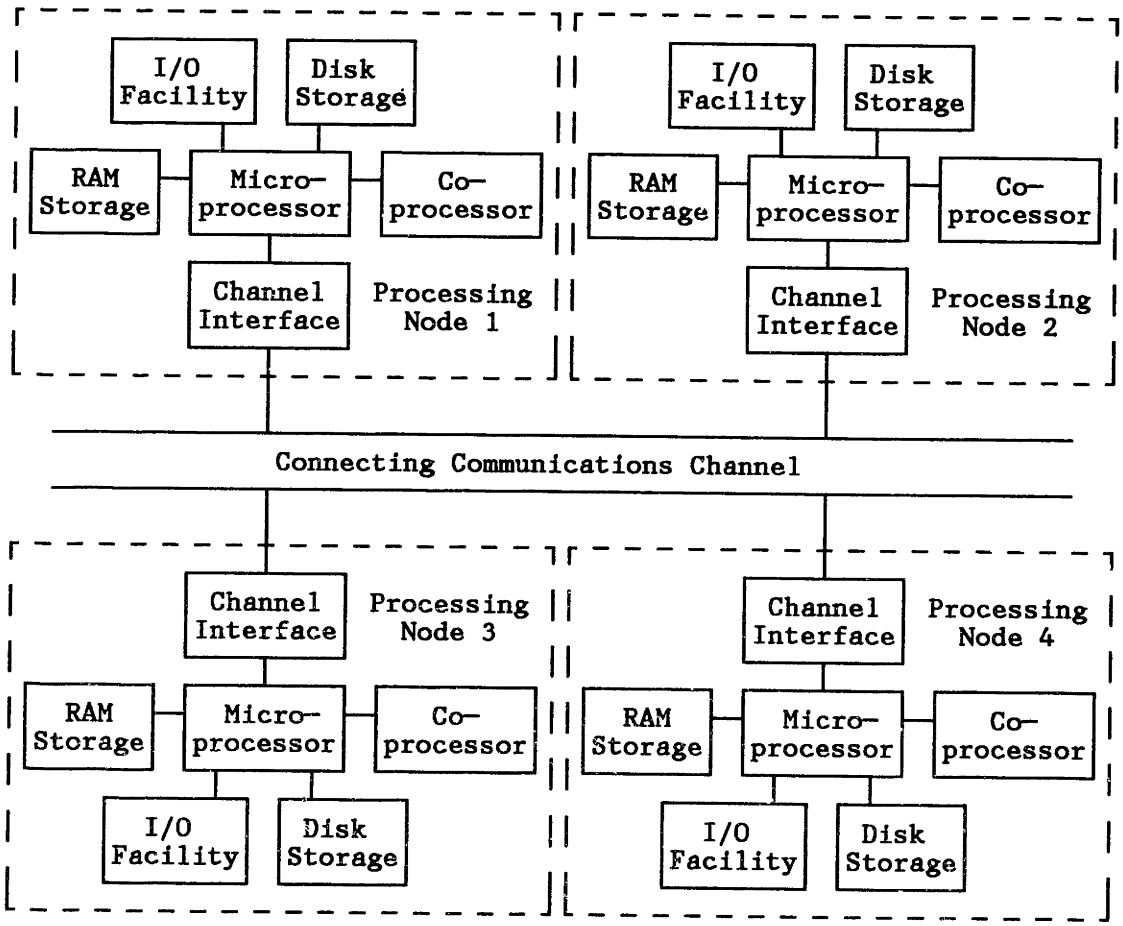

Figure 1. A Four Node Multiprocessor System

co-processor, random access memory, and disk storage. The 5 microprocessors could be linked via a communication channel such as IBM's MicroChannel or Intel's MultiBus II.

The MERLIN board is an example of a multiprocessor. Each MERLIN unit contains a high performance microprocessor, a numeric co-processor, and a large amount of dynamic RAM. Several MERLIN boards can be placed into an IBM PS/2, creating a more powerful workstation. In this situation, the MERLIN boards intercommunicate via the MicroChannel.

1.3 Terminology and Notation

The following terms are used throughout the thesis:

Doubleword 32-bits or 4 bytes

Gigabyte 1,073,741,824 bytes

Megabyte 1,048,576 bytes

MIPS (instructions/clock cycle) * (clock cycles/second) * 10e-6

The following notation is used throughout the thesis: hexadecimal (base 16) numbers are denoted by a leading '0' and a trailing 'h'. For example, 0FFh represents the decimal number 255. When naming signals, a '-' before the name indicates the signal is active low. For example, -X represents a signal, X, which is negatively asserted. Signal names of the form X/-Y indicate that the signal represents the 'X' state when asserted and the 'Y' state when deasserted.

The remaining chapters of this document examine the MERLIN design. The second chapter explains the MicroChannel interface used by the MERLIN card. Chapters 3 and 4 address MERLIN's architecture for two members of the PS/2 family. The appendices provide comparisons of MERLIN with Encore's Multimax and IBM's Research Parallel Processor Prototype.

2.0 IBM MicroChannel Architecture

The MicroChannel inside the IBM PS/2 Model 70 and Model 80 provides the link between the main processor and the peripheral cards. Peripheral cards include memory additions, communication ports, graphics cards, and multiprocessor cards. The MicroChannel not only provides for communication between the system unit and the peripheral cards, it also allows communication amongst the various peripheral cards installed in the system.

The development of the MicroChannel was fueled by inadequacies in the data bus used in IBM's Personal Computer. The original Personal Computer data bus was a synchronous extension of the central processor. Because all functions on the bus had to be synchronized to the system's clock (which suffered serious skew problems as it was distributed through the data bus), timing was dependent on the overall processor speed. An interface board that worked in a 6MHz IBM PC/AT would not necessarily work in an 8MHz IBM PC/AT. In addition, the concept of a peripheral card controlling the data, address, and control lines of the bus - bus mastery - was primitive in its implementation (IBM Seminar Proceedings, 1987). As a result, IBM developed the MicroChannel for use in the PS/2 family of computers. Like other modern channels, it addresses the problems of synchronous protocols and bus mastery. The PS/2, therefore, provides a good development system for multiprocessors such as MERLIN.

The MicroChannel is assembled from a 32-bit address bus, a 32-bit data bus, a transfer control bus, an arbitration bus, and multiple support signals (IBM Technical Reference, 1987). In general, the MicroChannel signals reflect the memory and I/O requests of the system microprocessor. The

MicroChannel specification defines a precise asynchronous protocol for control and data transfer. This asynchronous protocol frees the peripheral card from any system specific timing, making the card portable between systems of varying speeds. Using an arbitration protocol, any peripheral card can gain control of the MicroChannel and control the other devices in the system by issuing its own memory and I/O requests. To ease installation and support, the MicroChannel provides Programmable Option Select (POS) registers that allow high flexibility during system configuration.

2.1 IBM PS/2 Architecture

On the system board of an IBM PS/2 Model 70 or Model 80 is an 80386. The microprocessor, possibly paired with an 80387, accesses memory that may be located either on the system board or in a peripheral card connected to the MicroChannel. Thus, every memory read or write or I/O read or write request is presented to both the system board and the MicroChannel. In this way, the MicroChannel is an extension of the system microprocessor's local bus.

2.2 Bus Mastery

Bus mastery implies the ability to win control of a channel from the system microprocessor (or another bus master) and then proceed to drive the data, address, and control lines with information. A bus master, unlike passive MicroChannel slave devices, performs memory and I/O transfers by executing MicroChannel protocols. These memory and I/O transfers can be addressed to either system board memory or MicroChannel peripheral devices. Bus masters arbitrate for the channel, and upon gaining the grant, take over control of operations for one or more channel cycles.

A card such as MERLIN needs the ability to control the MicroChannel during certain memory cycles. In order to accomplish this feat, it must first be assigned a unique 4-bit arbitration level. Later, during the arbitration phase, this level determines which device wins the channel. MERLIN

uses bus mastery to transfer data between other MERLIN cards, memory cards, and the system unit's main memory. For example, when a MERLIN card wishes to write the 32-bit word 01FE7A945h to memory location 8 of the main system board, it becomes a bus master and drives the appropriate signals to perform the write.

To become a bus master, a card must assert the MicroChannel's -PREEMPT line. The central arbiter initiates an arbitration cycle as soon as the present device releases the channel. The central arbiter indicates the beginning of an arbitration cycle by driving the ARB/-GNT signal to the arbitrate state. The requesting devices then drive their assigned 4-bit arbitration level onto the arbitration bus (lines ARB0 through ARB3). Any device seeing a more significant bit low on the arbitration bus than those driven low by that device stops driving its lower order bits onto the arbitration bus. The device driving the lowest arbitration level thereby wins control of the system channel when ARB/-GNT goes to the grant state. Since the arbitration value assigned to each card must be unique, the outcom. of the arbitration protocol is never ambiguous. If a device wishes to perform multiple transfers (as in the case of a DMA memory to memory transfer), the device also asserts the -BURST line. This allows the central arbiter to perform a fairness function, not allowing any device to swamp the bus. The central arbiter tracks burst mode bus transfers in order to force higher priority bus masters to release the channel for lower priority requests.

A block diagram of the central arbiter is shown in Figure 2 on page 9. The main advantages of a central system-wide arbiter are that it:

1. Allows up to 15 DMA devices,
2. Allows burst data transfers,
3. Allows prioritization of control between DMA devices,
4. Supports multiple masters (IBM Seminar Proceedings, 1987).

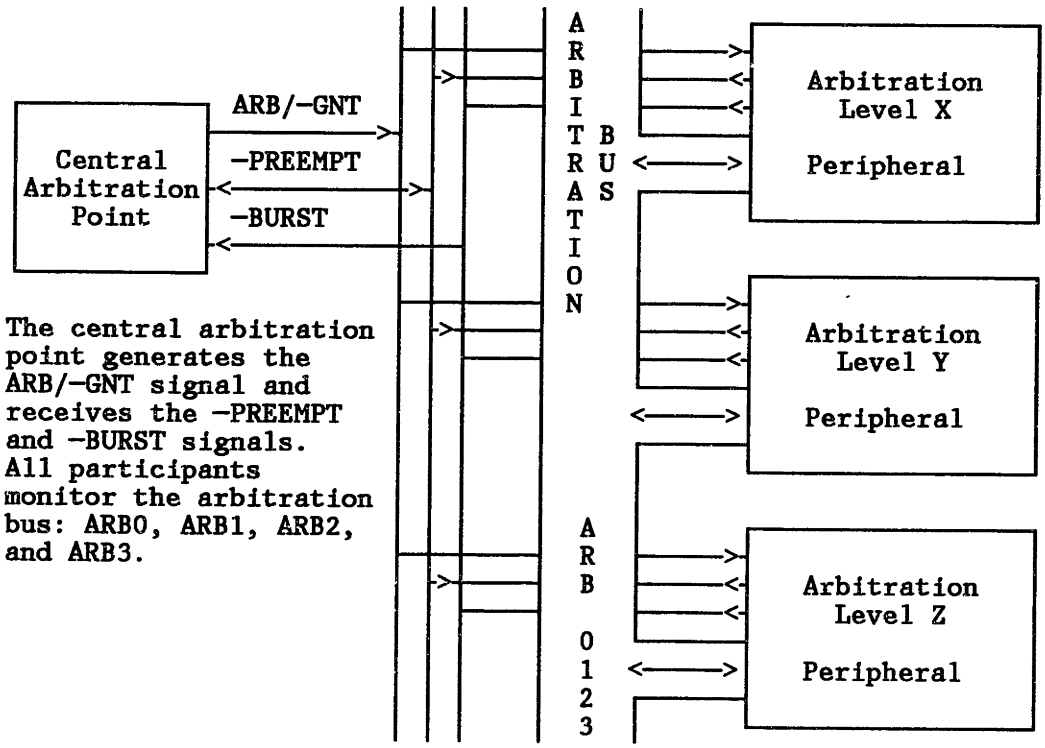

Figure 2. Block Diagram of Central Arbiter

2.3 *MicroChannel Data Transfer Rate*

Two key factors in evaluating a channel architecture are the communication latency and communication throughput. The latency of communication is the time required to initiate a transfer. This time includes the arbitration and winning of the MicroChannel which must be incurred each time a new transfer is begun. The throughput of communication, usually measured in Megabytes per second, indicates how quickly large blocks of data can be transferred once bus mastery is granted. Typical throughputs for channels in microcomputers are on the order of tens of Megabytes per second.

Each time the MERLIN card requests to become a bus master, it initiates an arbitration cycle. The minimum arbitration cycle (ARB/-GNT going from low to high to low) is 300ns. Thus, any transfer suffers a communication latency of 300ns. Because of the high cost of initiating a transfer,

if MERLIN recognizes a DMA block transfer request it asserts the -BURST line and holds the MicroChannel for the duration of the transfer. Use of the -BURST signal allows a large transfer to occur without a 300ns penalty between each doubleword. Burst mode transfers must be aborted within 7.8 microseconds of another MicroChannel card asserting the -PREEMPT line. Allowing another card to preempt the current transfer keeps any one device from continuously holding the MicroChannel. If no other card requests the MicroChannel by asserting the -PREEMPT signal, a burst mode transfer may continue for an arbitrarily long period of time. As a programmer, it is definitely beneficial to use MERLIN's DMA controller to transfer data blocks between MERLIN and other devices. MERLIN classifies all DMA transfers onto the MicroChannel as burst mode transfers. Thus, when MERLIN's memory router observes the DMA controller requesting control of both the local memory bus and the MicroChannel, an arbitration cycle is initiated with the -BURST line asserted. If the -PREEMPT signal is asserted before the transfer is complete, MERLIN holds the DMA transfer, relinquishes the MicroChannel for one arbitration cycle, asserts -PREEMPT, regains control of the MicroChannel at some later arbitration cycle, and continues the transfer. As a result, interrupted DMA transfers are automatically restarted when interrupted by other MicroChannel bus masters.

Once MERLIN has won control of the bus, it may begin data transfer. The MicroChannel has a defined bus cycle time of 200ns, which is a 5MHz operating frequency. Thus, a 32-bit (4 byte) MicroChannel connector can transfer data at a maximum rate of 5MHz*4 bytes = 20 Megabytes/second. In actuality, this data transfer rate is rarely achieved due to I/O delay times or memory wait states. For the Model 80, every memory access requires one wait state. For the Model 70, consecutive accesses to the same page of DRAM require zero wait states and an access to a different page requires two wait states.

It should be noted that the MicroChannel architecture also defines a Matched Memory Cycle (MMC) protocol for memory add-on peripherals. As described above, every MicroChannel memory access requires 200ns to complete. For both the Model 70 and Model 80, however, memory cycle times are significantly less than this figure. A 20MHz system operating with one

memory wait state accesses memory every three 50ns clock periods, or 150ns. The MMC feature allows channel cycles to be shortened in order to service these higher speed processors. In this case, the MicroChannel bus cycle time is reduced to three processor clock cycles.[3] Again, memory wait states are usually added which cause the realized transfer rate to be somewhat less than optimal. Memory cards are free to implement MMC (which the card indicates by asserting an MMC signal early in the channel cycle) or to operate at the 200ns cycle time required by the MicroChannel.

2.4 *Programmable Option Select Registers*

The Programmable Option Select eliminates switches from the PS/2 system board and interface cards by replacing their function with programmable registers. POS registers are used to configure memory or I/O port locations for peripheral cards, set arbitration levels, and configure peripheral dependent options.

Through the use of Adapter Description Files (ADF) for each interface card, the System Configuration program can automatically create a valid configuration for the system board and each interface card. Configuration is achieved by reading a unique adapter identification number from each interface card, matching it with an ADF file, and configuring the system accordingly. The resulting data is stored in battery-backed CMOS RAM along with the adapter number. When the system is powered on, Power-On Self-Test (POST) retrieves the configuration information from the CMOS RAM and configures the POS bytes of each interface card (IBM Technical Reference, 1987). A valid system configuration requires that each card attached on the MicroChannel be assigned a unique arbitration value and be mapped to a unique memory or I/O segment.

---

[3] MMC provides the highest data transfer rates available for the PS/2 family. For a 25MHz 80386 system, the clock cycle is 1/25MHz = 40ns. Thus, a Matched Memory Cycle, three clock cycles, is 40ns*3 = 120ns, yielding an operating frequency of 8.33MHz. A 32-bit (4 byte) MicroChannel connector can transfer 8.33MHz*4 bytes = 33.3 Megabytes/second.

Each interface card has 8 POS registers at I/O locations 0100h through 0107h. The first two locations are read-only bytes dedicated to the adapter identification number, a 16-bit value. The other 6 POS bytes are free for use. In the MERLIN card, these locations are used to configure the starting memory location for MERLIN's RAM and determine the bus arbitration level. These values are set by POST during system configuration using the values stored in CMOS RAM. The Configuration Program supplied with every IBM PS/2 is used to create the configuration using MERLIN's ADF file.

Because each connector on the MicroChannel has a unique -CDSETUP signal, each card is configured individually. Thus, every card on the MicroChannel responds to I/O locations 0100h through 0107h when their unique -CDSETUP line is asserted.

2.5 Limitations Of The MicroChannel

The IBM PS/2 MicroChannel is limited in available communication bandwidth. To illustrate this fact, assume a MERLIN board is installed in a 20 MHz Model 80. The 80386 in this system is pipelined and operates with one wait state, yielding a bus utilization of about 86 percent. RAM refresh for DRAM peripherals on the MicroChannel takes less than 5 percent of the bus bandwidth, leaving about 10 percent of the bus bandwidth available for DMA transfers when the processor is 100 percent busy.

Thus, for a 100 percent busy processor, 10%*1 second = 100 milliseconds of every second are available for DMA transfers. At 200ns per DMA transfer, this gives a data rate of 100ms/200ns*4bytes = 200 Kilobytes per second using 32-bit transfers.

Under normal operation, a system microprocessor runs at 75% to 90% bus utilization, so the above numbers are overly pessimistic. These numbers are greatly improved through the use of a processor cache that reduces memory accesses presented to the MicroChannel by the system

microprocessor. For example, if a Model 70 is used with its 64 Kilobyte cache, the system microprocessor uses less than 33% of the bus bandwidth (Shiell, 1987). As a result, the data transfer rates and bus availability are much higher. In any case, with several MERLIN boards installed in a system, bus contention will begin to dominate communication latency time and slow processor intercommunication. Given the above calculations, it is reasonable to expect the Model 80 to support up to 1 or 2 MERLIN boards and the Model 70 to support up to 2 to 4 MERLIN boards.

While the IBM PS/2 MicroChannel's bandwidth suffices for a few MERLIN cards on a single channel, it was not designed to service a massively parallel computing system. In systems with hundreds or thousands of processors, the design task must focus on communication issues. These systems tend to use interconnection networks, such as meshes, rather than simple one-dimensional communication paths such as channels. Indeed, the MERLIN card was specifically designed to be placed in a machine with only a few processors.

MERLIN's design was limited by the engineering and physical constraints of the MicroChannel. Power, heat, and size requirements are specified for every MicroChannel adapter. These factors affect board area, layout and implementation technology. In addition, the physical number of 32-bit MicroChannel slots available on an IBM PS/2 is limited by chassis size. As a result, only three MERLIN cards can be installed in a Model 80 and only two MERLIN cards can be installed in a Model 70.

2.6 *MicroChannel and NuBus II*

Many of the fundamental design decisions made while developing MERLIN centered around the choice to use IBM's MicroChannel as an interconnection network. IBM's MicroChannel, however, is only one of several popular 32-bit busses used in microcomputers. A second bus, used in Apple's Macintosh II, is the NuBus II.

Historically, busses were structured to optimize processor to memory bandwidth. As a result, busses tended to be processor dependent and tightly coupled to processor speed. The cost of the performance was loss of flexibility and compatibility.

Unlike older system busses, today's channels are designed to maximize hardware subsystem-to-subsystem transfers. These busses define general protocols or transfer methods for the system CPU and peripherals. This is accomplished by treating the bus as a resource, not as a logical extension of the microprocessor address and data lines.

Apple's NuBus is a full system bus, independent of the Macintosh II's host processor. In the Macintosh II, the motherboard is treated as a NuBus slot. Both the NuBus and the MicroChannel provide bus mastery capabilities in addition to standard slave services. Both include a form of bus locking which allows a processor to lock a bus for exclusive access (until another device preempts the current transfer in order to initiate a new arbitration cycle).

The Apple NuBus is a synchronous bus, operating at 10 MHz. Both the MicroChannel and the NuBus pass a common clock through the bus to minimize synchronization problems among the bus entities. However, there is a clock mismatch between the Macintosh II's local bus and the NuBus which requires a level of synchronization before a transfer can occur. Normally, a transfer from the NuBus to the local bus takes a full 68020 instruction cycle (about 400 to 500ns for a 15.7 MHz 68020 clock) to synchronize. Going the other way, a Macintosh II request can take a typical NuBus transaction of 2 bus cycles (about 200ns at 10 MHz) to synchronize. These times are the typical time penalty paid by the communications protocol between the CPU bus and the system bus (Cornejo, 1987). The MicroChannel does not suffer these overheads since the channel acts more as an extension of the processor bus. Both busses, however, have specified arbitration times to which bus masters must adhere before gaining the communications channel.

Although the NuBus and MicroChannel have many similarities, they differ in important respects. The NuBus multiplexes data and address lines while the MicroChannel does not. The

MicroChannel provides a matched memory cycle to ensure fast CPU to memory access for the 80386. A MicroChannel designer must accept the difficulties of using an asynchronous protocol while NuBus designers are faced with extremely tight timing constraints and clock skew problems. Synchronous busses, like those used in the IBM PC and IBM PC/AT, must distribute a clock signal to every card. In return, the card must ensure that every signal it generates adheres to certain setup and hold times. However, as the clock is distributed throughout the system, the timing of the clock is changed due to wire propagation delays. NuBus designers must accept the problems of dealing with clock skew. Finally, each NuBus slot is allocated a 16 Megabyte section of memory. The MicroChannel does not define the memory mapping of cards, allowing greater flexibility when designer an interface card. A card can allocate a much larger segment of memory of many tens of Megabytes or a much smaller segment of tens of bytes. In addition, the MicroChannel supports I/O bus cycles. Supporting I/O ports allows the MicroChannel to more accurately reflect the behavior of the PS/2's microprocessor since Intel separates memory and I/O space in their microprocessors. The NuBus assumes that memory and I/O will be combined into the same address space.

Had MERLIN's design centered around the NuBus, few changes would have to made. This results from the fact that both busses provide a single dimensional interconnection network that is acceptable for a low-level shared memory multiprocessor. Since today's microcomputer workstations use linear busses, the MERLIN design is well suited to bring parallel processing to these machines.

3.0 MERLIN's Design For a Model 80

The Model 80 uses a 32-bit MicroChannel interface. This chapter analyzes the design of a MERLIN card for the Model 80 and concentrates on the engineering constraints that controlled the architectural decisions.

3.1 MERLIN Hardware Overview

MERLIN is based upon an IBM personal computer environment. Therefore, it is essential that the MERLIN board execute the same code. Thus, it was decided that an 80386 based multiprocessor is required since the 80386 is capable of executing any program that runs on an 8088, 8086, 80186, or 80286 based system (Intel 80386, 1987). During MERLIN's design process, great pains were taken to insure complete hardware compatibility with IBM's existing series of computers. MERLIN was designed to be installed in a 32-bit MicroChannel slot inside an IBM PS/2 Model 70 or Model 80.

MERLIN's hardware includes the following:

- Intel 80386 32-bit microprocessor with integrated memory management
- Intel 80387 80-bit numeric processor extension
- Intel 82385 32-bit cache controller
- 32 Kilobytes of 20ns SRAM for cache
- Intel 82380 32-bit DMA controller with integrated system support peripherals
- 8 Megabytes of 80ns DRAM used for main memory

- MicroChannel interface subsystem.

A simple block diagram of the MERLIN card is provided in Figure 3 on page 18. The 80386 and its numeric co-processor, the 80387, interact primarily with the cache controller. The 82385 cache controller interfaces with the memory router control system which routs memory requests to either the DRAM or the MicroChannel. In addition, the router system watches the MicroChannel for requests to MERLIN's memory. The 82380 provides timers for use in multitasking operating systems and high speed Direct Memory Access (DMA).

The prototype MERLIN card was designed to operate as a 20MHz system; however the final version is limited only by the speed of the microprocessor and its supporting family of chips. As Intel releases faster version of the 80386 microprocessor, MERLIN's performance will increase. A user can replace a slow MERLIN card with a faster MERLIN card and immediately notice the speed improvement. MERLIN cards of different speeds and capabilities may be freely installed together in a system due to the asynchronous nature of the MicroChannel. Thus, the computational capability of the MERLIN card is independent of the processing speed of the host system (see "IBM MicroChannel Architecture" on page 6).

3.2 MERLIN Use of I/O Ports

The 80386 supports 8-bit, 16-bit, and 32-bit I/O devices that can be mapped into either the 64 Kilobyte (16-bit) I/O address space or the 4 Gigabyte physical memory address space. The MERLIN board, in order to be compatible with existing IBM PS/2s and to simplify circuitry, supports 8-bit I/O ports located in a 64 Kilobyte I/O address space.

The MERLIN board uses the I/O ports provided by the 80386 to configure certain features of the MERLIN card (see Figure 4 on page 19). One port is dedicated to controlling the cache. One bit of this port allows the cache to be enabled and disabled. This is useful during performance measurements (to see the effect of the cache) and during system initialization. A second bit allows

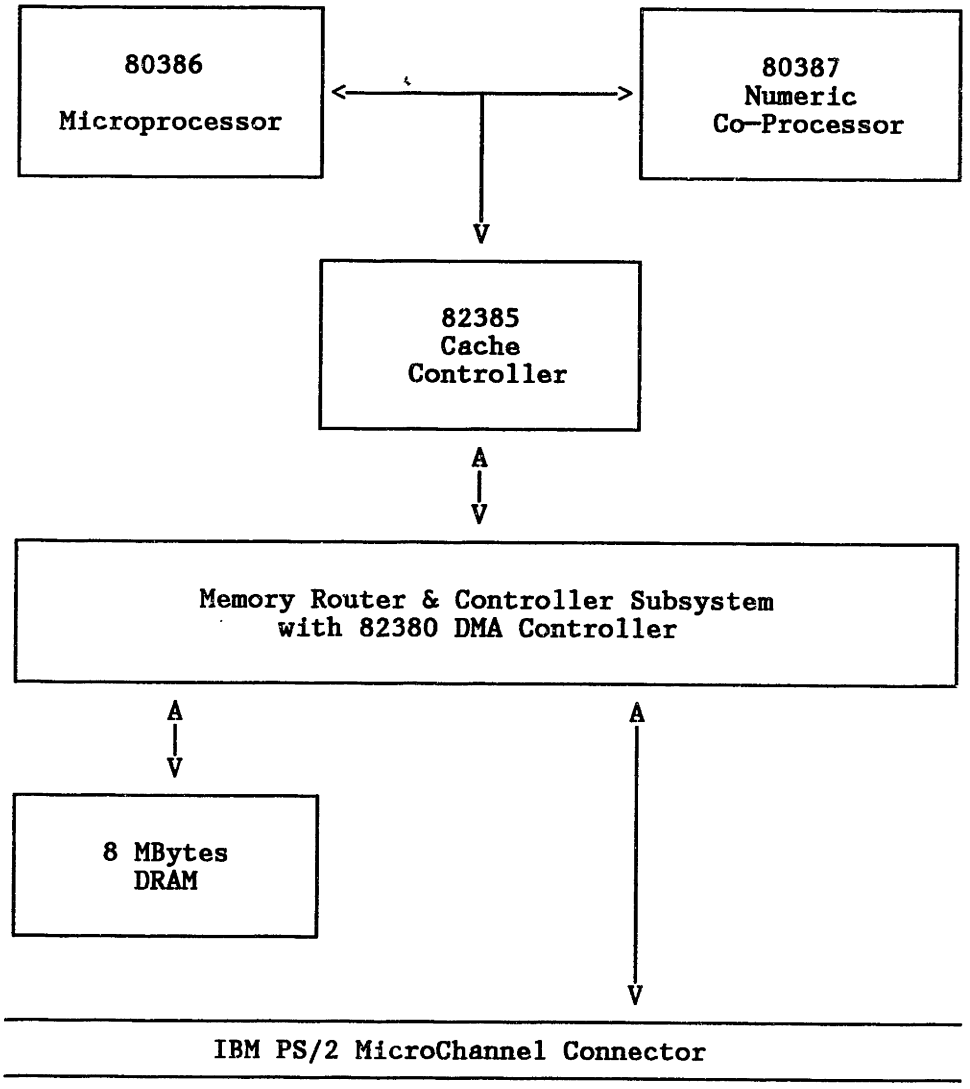

Figure 3. Overview of MERLIN Hardware

the cache to be flushed. This feature is useful during performance measurements and for process switching in an operating system. A second port allows the MERLIN card to sense its location in the 4 Gigabyte address space. The value, set during system initialization, reflects MERLIN's POS bits.

Port	Bit(s)	Access	Function
0E2h	0	R/W	Cache Enable – When this bit is set, the memory cache is enabled. When this bit is clear, the system cache is disabled.
0E2h	4	R/W	Cache Flush – When this bit is set, the memory cache is flushed, invalidating every cached location. When this bit is clear, the cache functions normally.
0F2h	0–7	RO	Memory Map – Provides the 16 Megabyte offset of the MERLIN card in the 4 Gigabyte address space. This value is set during system configuration.

Figure 4. MERLIN I/O Ports: The access column describes if each port is R/W = readable and writable or RO = read only.

3.3 *MERLIN Use of POS Bits*

Each interface card is assigned 8 bytes of POS registers located at I/O locations 0100h through 0107h. Locations 0100h and 0101h contain MERLIN's adapter identification number. This number is assigned by IBM to prevent conflicts between adapters. It is suggested that bus masters like MERLIN have values in the range of 00001h to 00FFFh. MERLIN uses location 0102h to store the DMA arbitration value it has been assigned by the System Configuration utility.

3.4 *80386/80387 Processor Core*

Like every IBM personal computer, the MERLIN board uses an Intel 8086 family processor as its engine. This provides object code compatibility between MERLIN cards and all IBM PS/2s. This feature greatly simplifies the software development process needed to convert IBM PS/2 applications to run on MERLIN.

The MERLIN card depends upon the 80386 microprocessor for its computational power. The 80386 is a 32-bit microprocessor produced by Intel Corporation. The 80386 features object code compatibility with all family microprocessors, making it an ideal choice for systems that must execute applications that currently run on IBM personal systems. In addition, the 80386 has a 4 Gigabyte physical address space and a 64 Terabyte virtual address space, allowing for the creation of huge programs.

Using pipelined instruction techniques and on-chip address translation, the 80386 operates at approximately 5 Million Instructions Per Second (MIPS).

While the choice of processor family was dictated by software compatibility constraints, MERLIN boards are free to use any processor from the Intel 80x86 family. Future Intel 80x86 family chips (such as the 80486 and 80586) can be incorporated into the MERLIN design without major engineering changes. These chips will bring increased computational power and performance to the MERLIN card.

3.4.1 80386 Memory Organization

Memory on the 80386 is divided into 8 bit quantities (bytes), 16 bit quantities (words), and 32 bit quantities (doublewords). Words are stored in two consecutive bytes with the low-order byte at the lowest address and the high order byte at the high address. Doublewords are stored in four consecutive bytes in memory with the low-order byte at the lowest address and the high-order byte at the highest address. The address of a word or doubleword is the byte address of the low-order byte.

In addition to these basic data types the 80386 supports two larger units of memory called pages and segments. Memory can be divided into one or more variable length segments, which can be swapped to disk or shared between programs. Memory can also be organized into one or more 4 Kilobyte pages. Finally, both segmentation and paging can be combined, gaining the advantages of both systems. Segmentation and paging are complementary.

The 80386 has three distinct address spaces: logical, linear, and physical. A logical address (also known as a virtual address) consists of a selector and an offset. A selector is the contents of a segment register. An offset is formed by summing all of the addressing components (Base, Index, and Displacement) into an effective address. Since each process on the 80386 has a maximum of 2^{14}-1 selectors, and offsets can be 4 Gigabytes, this gives a total of 2^{46} or 64 Terabytes of logical address space per process.

The segmentation unit translates the logical address space into a 32 bit linear address space. If the paging unit is not enabled then the 32 bit linear address corresponds to the physical address. The paging unit translates the linear address space into the physical address space. The physical address is what appears on the address pins. This is a key fact when dealing with the issues of system cache coherence and virtual memory. All virtual memory translation is done before an address is presented at the address pins. The 82835 cache controller, therefore, caches physical address rather than virtual addresses. Figure 5 on page 22 shows the relationship between the address spaces. "Virtual Memory Coherence" on page 43 examines the virtual memory coherence problem that arises in a MERLIN system.

The 80386 supports the Intel 80387 chip as a numeric co-processor. This chip features 80 bit floating point operations as well as extended integer and BCD support. It includes a full range of transcendental operations. The 80387 implements and fully conforms to the ANSI/IEEE Standard 754-1985 for Binary Floating Point Arithmetic.

3.5 MERLIN Memory Architecture

The MERLIN system is based upon a 32 bit microprocessor, the 80386. The 32 bit address space of the 80386 gives it access to up to 4 Gigabytes of real memory. Memory may be viewed as a 4 Gigabyte linear array that is allocated to both MERLIN cards and the host processor (see figure Figure 6 on page 23).

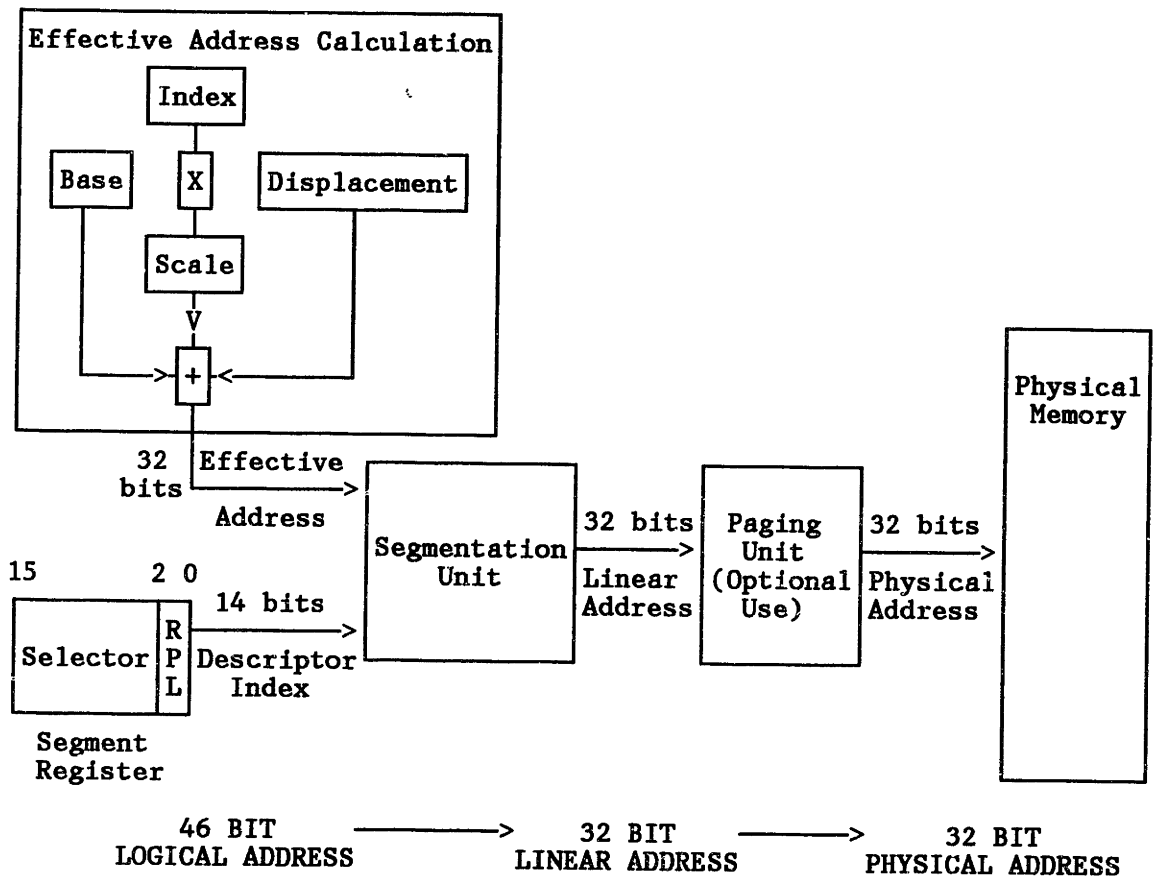

Figure 5. Address Translation

To understand the shared memory architecture of the MERLIN card, first envision Figure 6 on page 23 as a system with one processor (the IBM PS/2 processor) and several MERLIN boards installed. Ignore the processors on the MERLIN boards and examine them strictly as RAM additions to the system. Programs operating in the PS/2 80386 processor generate virtual addresses. These addresses are translated by the process described in the section "80386 Memory Organization" on page 20 to physical addresses. The physical addresses are presented on the 80386 address pins. This physical address is used to access a location of physical memory. Physical memory can be provided by either system board RAM, memory expansion cards on the MicroChannel, or MERLIN boards. The MicroChannel allows the RAM on both the memory

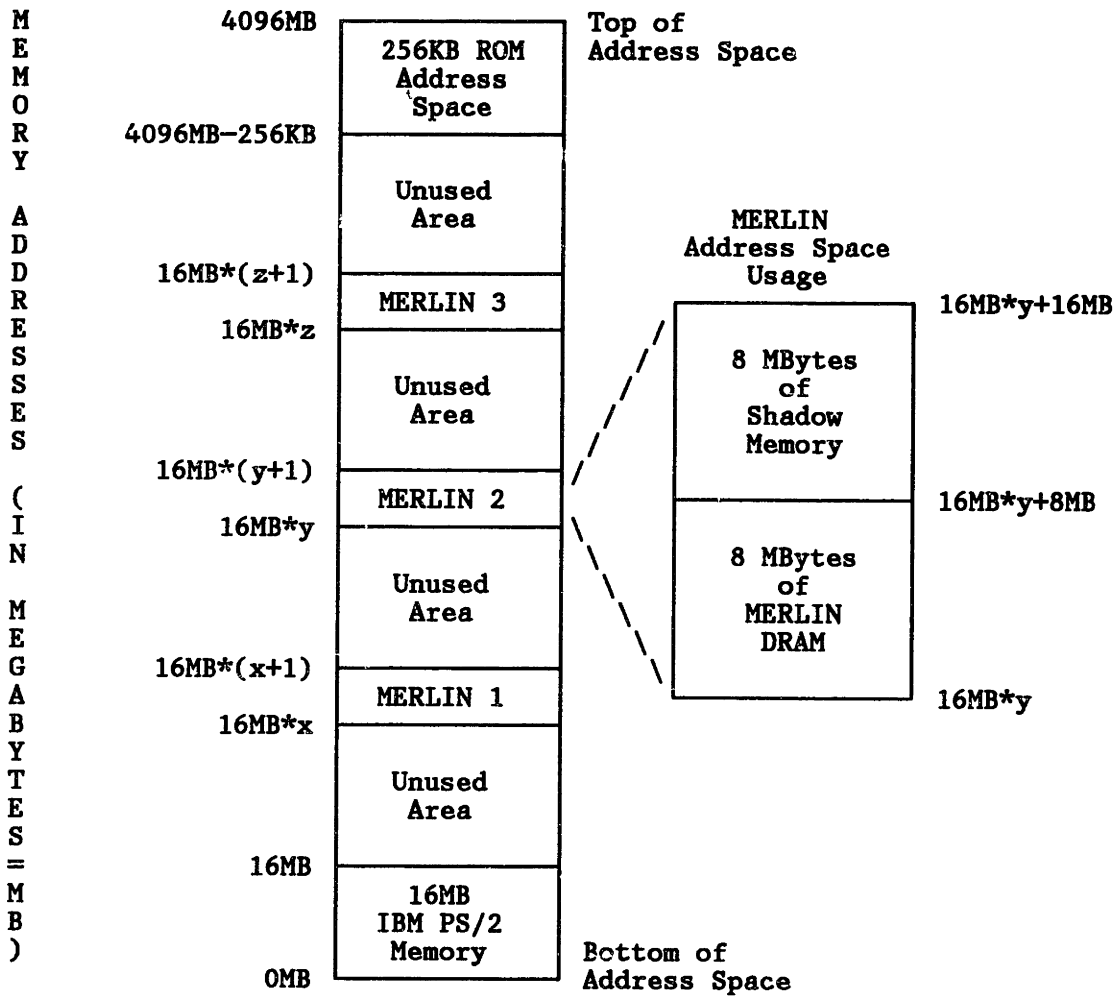

Figure 6. Memory Map of 4 Gigabyte Address Space

expansion boards and the MERLIN boards to mapped anywhere into the 4 Gigabyte address space of the 80386.

In general, memory is mapped into contiguous segments starting from location 0. Thus, in a system with 8 Megabytes of system board memory, 8 Megabytes of memory expansion board memory, and two 16 Megabyte MERLIN boards, memory would be allocated as follows: 0 to 8 Megabytes to system board RAM, 8 to 16 Megabytes to memory expansion board RAM, 16 to 32 Megabytes to MERLIN 1, and 32 to 48 Megabytes to MERLIN 2. Thus, when the system is first initialized, POST will perform memory checks on the 48 Megabytes of memory in order to insure proper

operation. The System Configuration utility determines the amount of available RAM and stores the value in the battery-backed CMOS SRAM. POST uses this value on system restart to determine the amount of memory installed.

In the above system, the processor always accesses memory through the MicroChannel. In a system with several processors, every processor must use the MicroChannel to access memory. If several processors are operating in parallel, the MicroChannel cannot provide memory accesses to each processor quickly enough and will become a performance bottleneck. As a result, MERLIN boards include caches to reduce MicroChannel bus traffic. The use of caches is described in "Cache System" on page 30.

The choice to place the 8 Megabytes of system board RAM in the lowest address range is motivated by the PS/2 memory architecture. The Model 70 and Model 80 contain DMA controllers that support 24 bit wide addresses. Thus, these machines may use DMA only when accessing the first 16 Megabytes of memory. However, the MicroChannel does provide the full 32 bit address the 80386 processor can support. Thus, while the PS/2 cannot DMA into memory above the 16 Megabyte line, it can access this memory using the system processor. As a result, in the 4 Gigabyte address space, the IBM PS/2 is always mapped into the first 16 Megabytes.

A MERLIN card may be placed anywhere in the remaining 4 Gigabyte address space (at 16 Megabyte boundaries). Since each card maps its 8 Megabytes of memory into the 4 Gigabyte address space, each processor may communicate with another by writing directly into its memory space. In other words, in Figure 6 on page 23, if MERLIN 1 wished to transfer data to MERLIN 2, MERLIN 1 initiates a transfer (using assembly language instructions or the DMA controller) to transfer data from MERLIN 1's address space into MERLIN 2's address space. Similarly, if MERLIN 1 wished to transfer data to the IBM PS/2 host memory, it initiates a transfer and writes data directly to the PS/2's memory. The only exception to this scheme is that the IBM PS/2 may only use the system processor to transfer data to the MERLIN cards since the PS/2's DMA controller supports only 16 Megabytes of linear memory.

The 8 Megabytes of DRAM memory on each MERLIN board is mapped into the 4 Gigabyte address space at a location specified during system configuration. An 8 bit value, stored at I/O port 0F2h selects the upper 8 bits of address space for the MERLIN card.[4]

Whenever an 80386 is reset, it begins fetching code from memory location 0FFFF FFF0h. Thus, each MERLIN card must have ROM that is mapped into the uppermost portion of the memory map. This implies that each MERLIN card locally replaces the upper 256 Kilobytes of the memory map with its own system ROM.

It should be noted that the hardware provides no intrinsic memory protection between MERLIN cards. Any card is free to read and write data to any location in the 4 Gigabyte address space. Indeed, the hardware is designed to support and enhance this feature. It is up to the operating system controlling the installed MERLIN boards to provide for memory protection between processors. Since the 80386 has a complete virtual memory system with page fault and segmentation violation protection, memory protection is fairly easy to install (see "Virtual Memory Coherence" on page 43).

When no adapter card has mastery of the bus, the MicroChannel presents the memory and I/O requests of the main microprocessor. If a MERLIN card wishes to initiate a data transfer to a location of memory that does not reside in its local DRAM, in arbitrates and wins the MicroChannel, executes a memory read or write, and releases the channel. In return, each MERLIN card must constantly be watching the bus to recognize memory requests to its own DRAM. If a request is found, the memory router system automatically services the request without interrupting the processor.

In the above system, local cache coherency is a major issue. However, since the 82835 cache controller allows for synchronous snooping and posted write cycles, MicroChannel interactions may

---

[4] This port is specified as being read-only. Thus, the processor may not change the value stored in the port at run time.

be handled without halting the local microprocessor. An analysis of the cache coherency issue is provided in "Cache Coherency" on page 37.

3.5.1 MERLIN Memory Hardware

Each MERLIN card is equipped with 8 Megabytes of DRAM. The DRAM, packaged in 2 Megabyte SIMMs, has an 80ns access time. Since MERLIN uses the page mode feature of the DRAM, it is possible to access continuous blocks of memory (on the same page) without incurring any memory wait states. As a result, a 25MHz MERLIN card can perform block transfer to and from memory without additional wait states.

The choice of 8 Megabytes of DRAM on every MERLIN card was influenced by the power and size constraints of a MicroChannel adapter cards. Since MERLIN provides a shared memory architecture, however, additional memory installed in the system can be used by a MERLIN card.

3.5.2 MERLIN DMA Support

MERLIN uses an Intel 82380 32-bit DMA controller with integrated system support peripherals chip to perform memory transfers. The 82380 DMA controller can transfer data within MERLIN local memory or between MERLIN local memory and memory available through the MicroChannel.

The 82380 operates directly on the 80386 processor bus. In slave mode, it monitors the state of the processor at all times and acts or idles according to the commands of the host. It monitors the address pipeline status and generates the programmed number of wait states for the device being accessed. The 82380 operates in slave mode except when it is performing DMA transfers.

The 82380 provides a high-performance, 8-channel, 32-bit DMA controller. It is capable of transferring any combination of bytes, words, and doublewords. The addresses of both source and destination can cover the entire 32-bit physical address space of the 80386.

The DMA controller is programmed using 24 status and command registers. Each DMA channel has three programmable registers which determine the location and amount of data to be transferred. These register are:

- Byte Count Register - Number of bytes to transfer
- Requester Register - Source address for transfer
- Target Register - Destination address for transfer

The DMA controller uses the protocol shown in Figure 7 on page 28 to arbitrate and win the local processor bus in order to perform a transfer. For a demand mode transfer, the DMA controller relinquishes the bus after every doubleword transfer in order to allow the MERLIN processor access to the bus. Thus, demand mode allows the parallel operation of the 82380 and 80386. For block mode transfers, the DMA controller holds the bus until the DMA request has been completely serviced. Block mode transfers provide the fastest method for moving a block of data. The number of bytes transferred is determined by the channel's 24-bit Byte Count Register. Up to 16 Megabytes of information may be transferred by one DMA request. The type of transfer is determined by programming an 82380 control port. With the installation of a processor cache (see "Cache System" on page 30) between the 80386 and the processor bus, the processor can continue at full speed (except in the case of a cache miss or memory write) regardless of the transfer mode.

The DMA controller on the MERLIN card is configured to use its highest priority DMA channel for DRAM refresh. Thus, each MERLIN card is responsible for refreshing its own DRAM. By giving refresh highest priority, it can interrupt any other DMA transfer. For very large DMA transfers, several refresh cycles will be inserted. The DMA controller automatically resumes a lower priority transfer after servicing a higher priority transfer.

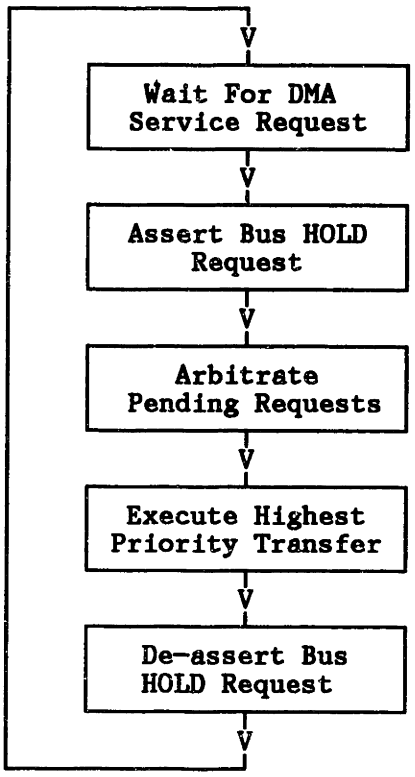

Figure 7. Flow of DMA Controller Operation

3.5.3 Memory Router Subsystem

The memory router subsystem has two responsibilities. First, it must route MERLIN memory accesses to either local DRAM or ROM or use the MicroChannel in bus master mode to write data to a memory slave. Secondly, it must perform as a memory slave, watching for memory accesses to the local MERLIN card. If such a reference occurs, the router must halt local accesses and immediately service the MicroChannel request.

When a local memory access is presented to the router, it uses the upper address bits of the desired location and the MERLIN memory map configuration bits to decide the destination of the request. If the address is to a location in the upper 256 Kilobytes of the processor address space, the request is sent to the MERLIN card's ROM. If the address is to a location mapped by the MERLIN card's

DRAM, the memory request is sent directly to the DRAM. In neither case is the memory request presented to the channel. If the access is to a remote location, the router gains control of the MicroChannel and performs a bus cycle to access the memory location.

The router constantly watches the MicroChannel for memory access to MERLIN's DRAM. When such an access occurs, the router uses a high priority DMA channel to interrupt current memory cycles and service the request. Once the request has been serviced, any interrupted memory cycles are restarted and completed.

The memory router subsystem has two substantial performance optimizations. Between the 80386 and the memory system is a high performance cache. As a result, the memory router is required to service local requests only on cache misses. It should be a rare occurrence that a remote MicroChannel request occurs at the same time a cache miss happens. In this case, however, the MicroChannel request is given preference since there is a MicroChannel timeout requirement that must be met.[5] The second optimization concerns the use of the -BURST signal of the MicroChannel. When MERLIN's memory router realizes a block mode DMA transfer is about to occur to memory on the MicroChannel, it arbitrates and wins the bus in burst mode. As a result, the MERLIN board is free to transfer large blocks of data without relinquishing the MicroChannel. The resulting data transfer rate is the optimal value computed in "MicroChannel Data Transfer Rate" on page 9.

The memory router system is given the second highest priority DMA channel (RAM refresh has the highest priority) in order to complete its operations. Thus, even if a block mode DMA transfer is occurring on one channel, a MicroChannel service request will interrupt the transfer. This allows MicroChannel requests to be serviced immediately, without waiting for the transfer to complete. Since the MicroChannel has timeout restrictions on all memory cycles, it is imperative that a long DMA transfer be interrupted in order to service the remote request.

---

[5] The MicroChannel timeout constraint is 3.5 microseconds (using the CDCHRDY signal - Card Channel Ready).

3.6 *Cache System*

Between the 80386/80387 processor core and the memory router system is an Intel 82385 32-bit cache controller. In addition to providing a high cache hit ratio, the controller reduces memory system contention between the processor and MicroChannel requests.

3.6.1 Cache Performance

Every processor must access memory in order to compute since memory holds both the data and the instruction stream. In a typical system, the processor can use registers, primary memory (DRAM), and secondary memory (disk storage) to store information. Registers, while extremely fast (capable of being accessed every processor cycle), are expensive and difficult to make very large. Secondary storage, while extremely large, is much too slow to be used for general computation. Primary memory resides somewhere between the two extremes of extremely fast and extremely large. In today's technology, DRAM can be made relatively fast (80ns access times) and relatively large (several Megabytes). Processors, however, need higher performance memory than DRAM is capable of providing. Thus, a fourth level of memory is introduced into the hierarchy: caches. Caches are fast enough to handle a processor access every memory cycle and are much larger than register files.[6] Thus, if the cache can be made to store much of the data and code for an executing program, program execution time will be greatly reduced as compared to a system with only DRAM.

Caches rely on a program's temporal and spatial locality to increase system performance. Temporal locality encompasses the notion that if a location of memory is touched at time T, there is a high probability that it will be touched again at time T + delta for small delta. Spatial locality is similar

---

[6] Typically, the processor clock cycle is faster than a processor memory cycle. On the 80386, a memory cycle is two clock cycles. Thus, registers may be accessed every clock cycle while computing a result. Cache memory may be accessed every other clock cycle. As a result, it is beneficial to use registers for computation whenever possible.

to temporal locality, but applies to a stream of executing code. Specifically, spatial locality states that if a location of memory, L, is referenced, there is a high probability that location L+delta will be referenced next for small delta. Typically, delta is one.

In practical terms, temporal and spatial locality are empirical facts collected from observing executing programs. Most application programs spend much of their time in small loops, executing the same code over and over again. The data referenced by these loops also tend to be referred to multiple times. As a result, if a cache provides extremely fast access to both the instruction stream and the data, program execution speed will be greatly increased.

When a cache is placed between the processor and main memory, memory cycle time can be greatly reduced. The effective memory cycle time can be computed as:

$$t_{eff} = h*t_{cache} + (1-h)*t_{main}$$

where t_{cache} is the time required to access the cache, t_{main} is the time required to access main memory, and h is the hit ratio of the cache. The hit ratio represents the percentage of time a requested location is in the cache (in steady state program execution). It should be noted that the above formula neglects the effects of virtual memory page misses and process context switches. In any case, however, a high hit rate and a low cache access time will substantially lower the effective memory cycle time. In typical systems, cache hit ratios are in the 90% to 99% range while t_{main} ranges from 4 to 20 times t_{cache}.

When a cache miss occurs (i.e. when the location requested is not in the cache), the new location is cached and an old cached location is removed from the cache. An enormous amount of literature has been dedicated to the topic of cache replacement strategies. Many schemes have been designed that have slightly better performance for some test case. Any cache replacement scheme, however, can be fooled by a sadistic address stream and yield an arbitrarily long period of zero cache hits. The Intel 82385 cache controller uses a one bit Least Recently Used (LRU) algorithm to determine

	Time 1	Time 2	Time 3
1st	A	Z	C
2nd	B	A	Z
3rd	C	B	A
4th	D	C	B
5th	E	D	D
6th	F	E	E

Figure 8. LRU Replacement Example: Time 1 shows the initial order. Time 2 shows the order after a reference to line Z. Time 3 shows the order after a reference to line C.

the correct set for replacement (see "Direct Mapped versus Two-way Set Associative" on page 32). Figure 8 on page 32 shows an LRU replacement strategy being applied to an address stream for a 6 element cache. LRU chooses to remove from the cache the object that has not been referenced for the longest amount of time.

Cache designers try to have a very high steady-state cache hit ratio. Regardless of the cache replacement policy used, however, there are times when the hit ratio can drop to zero. For example, when the processor switches processes, the cache contains information for the old program. Thus, almost every memory access of the new program will cause a cache miss.

3.6.2 Direct Mapped versus Two-way Set Associative

The Intel 82385 provides the system designer with the choice of implementing a direct mapped or a two-way set associative cache. The choice dictates the cache memory circuit and varies the hit ratio perceived by various programs.

A direct mapped cache stores a memory location in fast cache memory. The offset in the cache memory is determined by computing the primary memory address modulo N, where N is the size of the cache (measured in doublewords). For the 82385, the 4 Gigabyte address is mapped into

pages of 32 Kilobytes (8192 doublewords) using 8192 as the value for N. Each 32-bit doubleword stored in the cache is called a line and represents the unit of transfer between primary memory and cache memory. Associated with each cache line is a tag and a tag valid bit. The tag indicates the main memory page number of the address being cached at a given location. For example, if line 15 of page 8 currently resides in the cache, then an 8 is stored in the line's tag field and the tag valid bit is set. An empty line has its tag valid bit cleared.

A two-way set associative cache can be though of as two direct mapped caches operating in parallel. Again, each primary memory location is stored at its address modulo N. In this case, however, the size of the cache is 2*N doublewords since two direct mapped caches are being used. In a two-way set associative cache, each memory address can be stored in one of two cache lines. Each of the two cache lines has its own tag valid bit and tag bits. MERLIN uses the two-way set associative configuration of the 82385.

Figure 9 on page 34 demonstrates how MERLIN's 32 address bits are divided by the cache controller. The upper 18 bits specify the cache page and are stored with the cache line as its tag. The lower 12 bits select the line address in the cache. These bits are divided into a 9 bit set field and a 3 bit line field. The 82385 groups eight lines together to form a cache set. The 82385 implements two groups of 512 sets, called directories, with each set having eight lines. In this configuration, the cache stores 2 * 512 sets * 8 lines/set * 4 bytes/line = 32 Kilobytes of data. While the size of the cache remains constant, as compared with the direct mapped cache, the size of the pages is halved and the number of sets is doubled.

Figure 10 on page 35 and Figure 11 on page 36 exhibit how data is cached by the 82385. The internal cache directory, stored entirely within the 82385, stores an 18-bit tag and 9 bits of tag valid information for each set. In addition, a 1-bit LRU flag is kept for each set address. The LRU flag is used during replacement to decide which directory should be updated with new information. On every cache access, the LRU bit is toggled to indicate which directory was last touched. The actual cache data is stored in external high-speed SRAM. The SRAM stores 8 lines per set for the

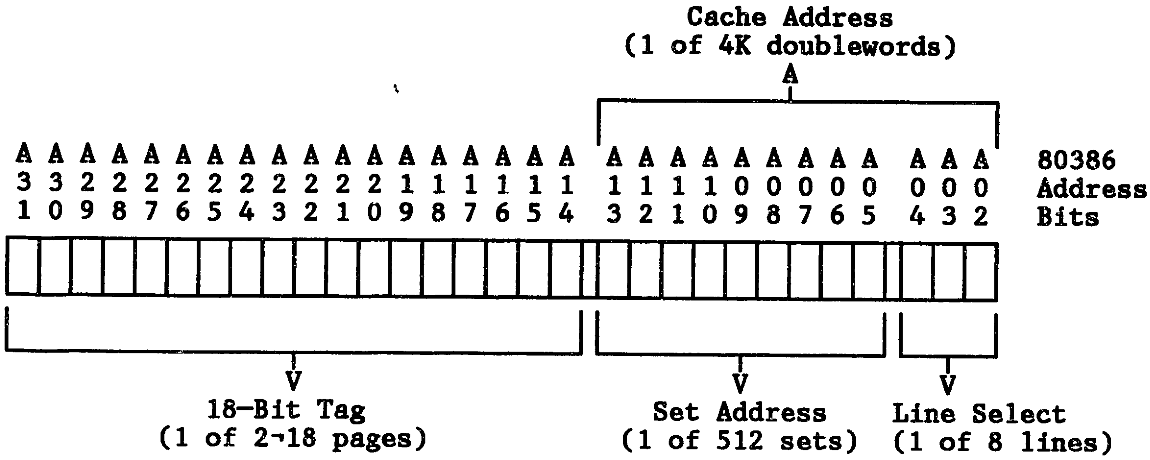

Figure 9. Cache Use of 80386 Address Bus Bit Fields

1024 sets comprising the cache. The data in SRAM is managed using the information stored in the internal cache directory.

The 82385 groups data into sets in order to reduce the amount of internal storage required by the cache. Ideally, each line would have its own tag. This would require additional tag storage and additional LRU bits. Since chip space was limited in the implementation of the 82385, Intel choose to utilize sets to reduce storage needs.

In most cases, a 32 Kilobyte two-way set associative cache will have higher performance than a direct mapped cache (also known as a one-way set associative cache). Since all identical page offsets map to two cache locations in the two-way set associative cache, rather than one, the potential for cache thrashing is reduced. This idea can be extended for N-way set associative caches which have higher performance but are more difficult to build.

3.6.3 Write-through versus Write-back

When the processor writes data to memory, both the cache and primary memory system must be updated. There are two techniques used to accomplish this: write-through and write-back. While

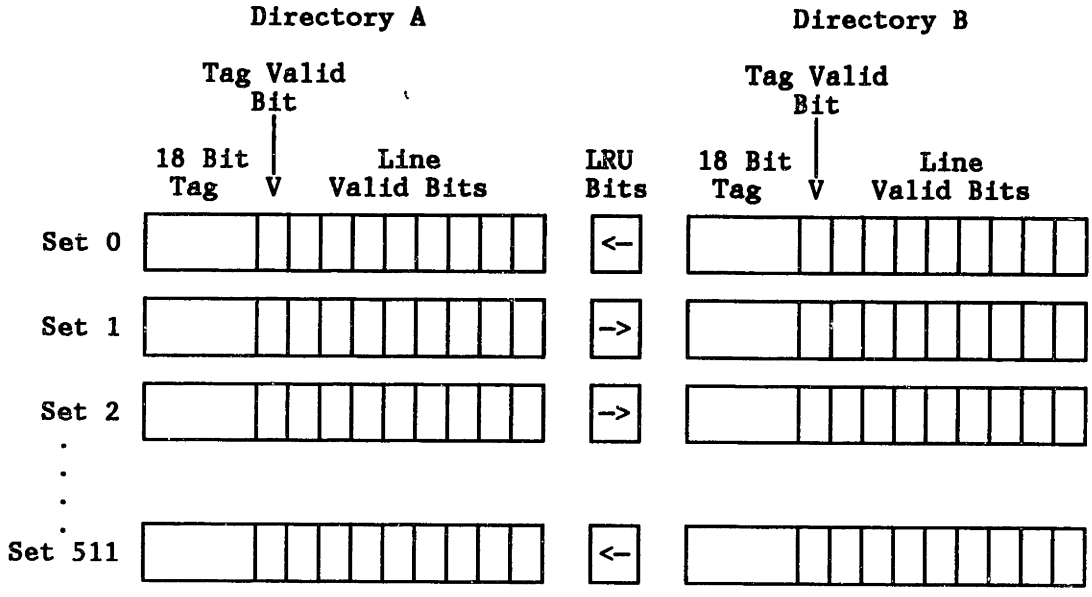

Figure 10. Internal Cache Directory - Two Way Set Associative: Each set in the two-way set associative cache stores 8 lines. If a tag for the set is marked as valid, the 8 individual line valid bits are used to indicate which lines store valid data.

both techniques achieve the task of keeping the cache and main memory coherent, write-through proves to be much simpler to implement in a shared memory multiprocessor system.

Write-through refers to cache designs that choose to immediately pass data written by the processor through the cache directly to the main memory system. Thus, any performance benefits seen when reading from the cache are lost during memory writes. As data is being written through the cache to main memory, the cache has two choices for handling the data. First, the cache controller may choose to replace a current cache line with the updated data. If the processor accesses this location again soon, the cache will already have the data. Secondly, the cache controller may choose to ignore the data being written, only invalidating a cache entry if the address matches a tag and offset. This scheme, while lower in performance, is much simpler to implement. Write-through caches that invalidate data on writes can only add new data to the cache on cache misses.

For write-through caches, primary memory and the cache always contain the same data. For a multiprocessor system, any processor that accesses main memory is guaranteed to access valid data.

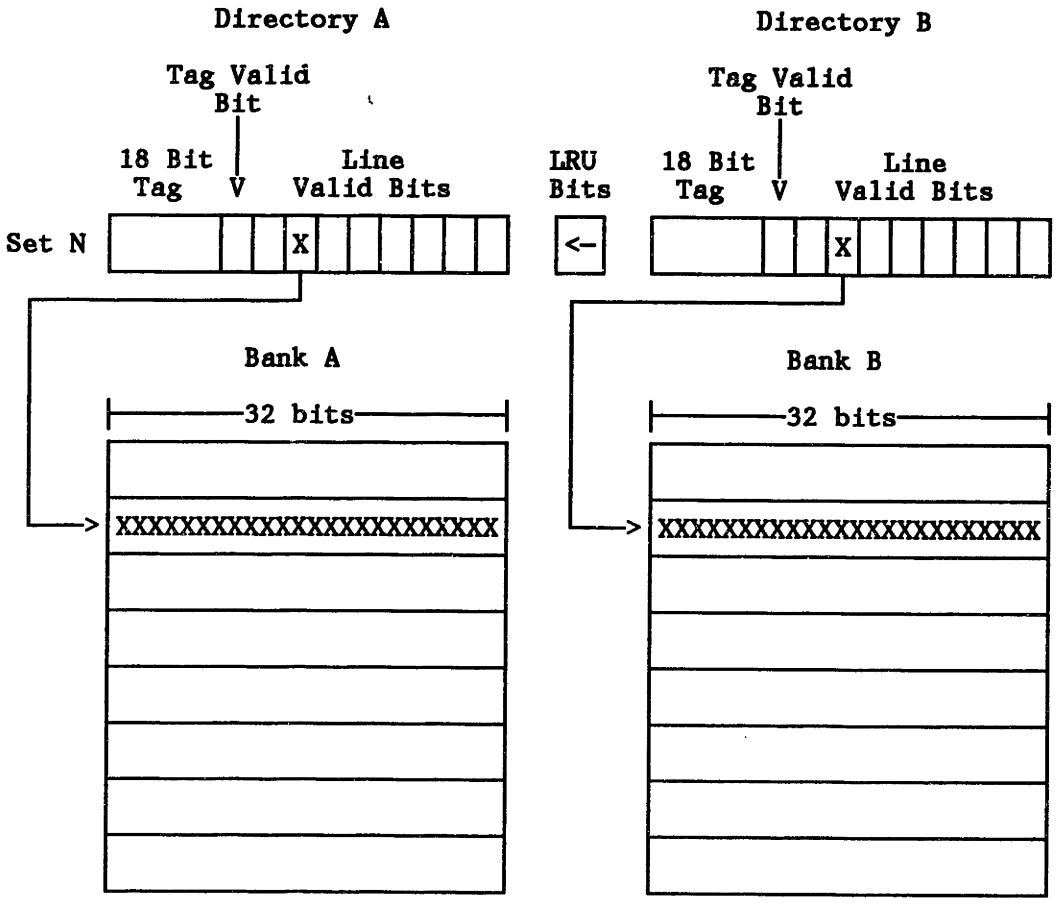

Figure 11. External Data Cache - Two Way Set Associative

Write-back caches, however, take a different approach. When data is written by a processor to a write-back cache, the data is updated in the cache - primary memory is not updated. As a result, the processor does not have to wait for the slow primary memory to complete a write cycle. At this point, the cache contains data that is different than main memory. Latter, when the line in the cache containing previously written data is to be replaced, the updated data is written to memory. This strategy moves the time penalty incurred by cache writes from processor write time to cache miss time. The result is a higher performance cache for most programs. The cost is the increased amount of information that must be stored for cache lines and the complexity of the cache miss control section. Note, however, the additional complexity of write-back caches in a multiprocessor

configuration. No longer can a processor only access primary memory to read a data value. It must access every cache in the system in order to be insured the most recent value of the location.

The Intel 82385 cache uses a buffered write-through scheme that provides a compromise between write-through and write-back caches. The buffered write-through method allows the processor to continue after a data write without waiting for the memory system to complete. As the processor begins its write cycle, the cache begins a write cycle to main memory. The processor may return immediately while the cache completes the write. As the cache waits for the memory system to complete the write, it may still service the processor read requests. Essentially, write operations are queued by the cache to main memory without ever halting the processor. At some point, however, the queue will be full and the processor must wait for at least one write to complete before returning from the write. The 82385 provides a one element queue. Thus, if the 80386 tries to write two locations, one right after the other, the processor will be halted during the second write while the first completes. Buffered write-through provides higher performance without the added cache complexity of write-back techniques.

3.6.4 Cache Coherency

The key to using an interconnection network in shared memory multiprocessor systems is to send data over the network as rarely as possible. This tends to reduce channel contention. As the channel use per processor decreases, the effective number of processors that can be installed on the communication channel increases. At first sight, adding caches to the system might seem to greatly reduce the amount of time each processor needs the bus. However, in order to keep all cached data coherent, a great deal of additional bus traffic must occur.

Cache coherency is a problem that arises in every shared memory system. When several processors are writing to different memory locations, it is extremely difficult to keep each cache updated. For example, if processor 1's cache (write-through or write-back) updates location X, then every other cache must be informed of this update. If processor 2's cache held location X, the cache controller

must either fetch a new value for location X or invalidate the cache line. In either case, processor 1's cache must post to every other cache in the system the fact that it is updating location X. In a communication channel bound system, this posting is prohibitively expensive.

Every MERLIN board contains a 32 Kilobyte cache. In addition, MERLIN's memory architecture demands that every MERLIN board be able to access any location in the 4 Gigabyte physical address space of the processor. A cache coherency problem arises while determining which data should be cached. Specifically, each MERLIN board has 8 Megabytes of local memory. Obviously, the cache controller should cache this data. However, should the cache controller cache locations not on the MERLIN board? Should the cache contain locations from anywhere in the 4 Gigabyte address space? The answer depends on the PS/2 in which the MERLIN board is installed. For a Model 80, it is infeasible for a MERLIN card to cache the entire address space. However, the Model 70's processor cache makes this a practical solution. The next sections address a MERLIN's use in a Model 80. The next chapter examines MERLIN's use in a Model 70.

In a Model 80, the MicroChannel reflects the memory cycles of the main PS/2 microprocessor. As the main processor operates, each MERLIN board is updating its local memory. When a MERLIN board must access data that is not present in its local memory, it arbitrates and wins the MicroChannel in order to perform a memory access cycle. Thus, MERLIN cards only post memory cycles when they access non-local memory. In order to allow every cache in the system to cache any location from the 4 Gigabyte address space, however, every memory write cycle a processor performs would have to be posted on the MicroChannel. For example, if the MERLIN 1 board in Figure 6 on page 23 wished to cache the local memory of the MERLIN 2 board, every memory write that MERLIN 2 performed to its local memory would have to be communicated to the MERLIN 1 board in order that the data in MERLIN 1's cache remain coherent.

Posting every memory write of every MERLIN card onto the MicroChannel would cause a serious communications bottleneck in a Model 80. Thus, MERLIN uses a simple solution to the cache

coherency problem. Only the local 8 Megabytes of DRAM data are cached by MERLIN's 82385. All other memory in the 4 Gigabyte physical address space of the processor is not cached.[7]

The above problem can be addressed by configuring the cache as a code-only cache. Code-only caches utilize the fact that instruction streams are read-only. As a result, they cannot be updated. In this case, the entire 4 Gigabyte address space could be cached. This design choice was not taken for two reasons: first, a code-only cache has lower performance than a data and code cache and secondly, the 80386 allows a program to modify itself. As a result, programs may change themselves during execution and these changes (occurring in local memory) would have to be posted on the MicroChannel.

There is a second aspect to cache coherency for a MERLIN based system. While MERLIN's 80386 is executing instructions from the cache, a remote MicroChannel access could occur. Another processor can read or write to MERLIN's memory while the 80386 is accessing the cache. In order to remain coherent, the cache must recognize all writes to local memory and act accordingly. At the same time, however, the cache must be servicing the processor. The 82385 solves this problem using a snoopy bus. The snoopy bus watches all writes to local memory and notifies the cache controller if a cached location has been updated by another processor. In this case, the cache controller invalidates the cache line of the updated data.

Memory snooping is accomplished in the 82385 by alternating between servicing the 80386 and snoopy bus. Since every memory cycle on the 80386 is two cycles, the 82385 can use one cycle to feed data to the processor and one cycle to watch operations on the snoopy bus. Thus, the memory router system can service MicroChannel request in parallel with processor operation since the snoopy bus keeps the cache coherent.

---

[7] The 82385 cache controller's -NCA (Non-Cacheable Access) signal is used to control addresses that are to be cached. If -NCA is asserted, the 82385 forwards the processor's non-cacheable cycles to the 82385 local bus (memory bus) and runs them. The cache and cache directory are unaffected during these operations.

3.6.5 Cache and DMA Interactions

From the block diagram in Figure 3 on page 18 it can be seen that the 82385 cache controller acts as a buffer between the processor and the memory bus. As a result, DMA operations occur in parallel with the operation of the processor. In this case, the processor fetches data from the cache while the DMA controller uses the memory system. When a cache miss occurs, the processor is either held (block transfer mode) until the DMA transfer is finished or is timesliced with the DMA controller (demand transfer mode).

The snoopy bus is also used when MERLIN's 82380 DMA controller is operating. Since the DMA controller can operate in parallel with the processor and has complete access to local memory, the snoopy cache feature ensures that any write performed by the 82380 to local memory is correctly updated in the cache. Thus, if a cached location is updated during a DMA transfer, the 82385 will utilize the snoopy cache feature to invalidate the cached line.

3.7 Other MIMD Architectures

MERLIN's design results from a series of decisions made according to engineering constraints. This section examine other architectures that could have been implemented.

3.7.1 Caching Entire Address Space

The most serious drawback of the design presented above is the fact that a MERLIN board is only allowed to cache its local 8 Megabytes of memory. This decision was motivated by the bandwidth constraints of the Model 80 MicroChannel. Since the Model 80 does not have a cache, every memory cycle is presented to the channel. As a result, there is not enough channel bandwidth for each MERLIN card to post memory writes in order to keep all caches coherent. The MERLIN design for the Model 70, described in the next chapter, addresses this issue.

3.7.2 Cache Constraints

The MERLIN design uses Intel's 82385 cache controller. This controller was chosen since it adds caching to an 80386 system with minimal chip count.

The 82385 provides a two-way set associative 32 Kilobyte cache. As memory system become larger, spanning tens of Megabytes, caches must also become larger. Thus, a larger cache would be preferable. As technology emerges, larger caches will be developed. Adding larger caches to the MERLIN card does not require any major design changes. Indeed, if caches can be incorporated onto the microprocessor chip, the MERLIN design becomes even simpler, removing the caching system from the hardware design.

A second major improvement to the 82385, in addition to increasing cache size, would be an improved cache management policy for multiprocessor systems. There are several possible states for a location in memory. A location could be unused implying that no cache contains that location. A location can be private, or stored in only one cache. A location may also be shared if several caches contain copies of that location. A cache management policy that recognizes these situations could more effectively utilize bus bandwidth.

Several "smart" cache management algorithms are commonly used including one that recognizes private memory locations. When a cache reads a location, it posts this read to every cache in the system. If any other cache signals that it contains this data, the location is marked as shared and any future updates to this location by any processor are posted onto the channel. If no other cache signals that it contains the location, the location is marked as private and future writes to that location are kept only in the local cache. In this case, writes do not need to be posted onto the bus. For some applications, such a cache management technique greatly reduces the needed bus bandwidth by reducing the number of writes that must be made. Of course, since this scheme must post reads to memory, cache misses must be minimized in order to insure efficient operation.

In order to implement a smart cache management algorithm on top of the 828385 Cache Controller would require a significantly increased chip count that the MERLIN design can not afford. Future versions of the 82385, however, will surely take multiprocessor considerations into account and provide "smart" cache management algorithms.

3.7.3 Separating Processors and Memory

MERLIN cards include both processor and memory. Since the memory is equally shared between all processors, however, there is no reason why processors and memory cannot reside on separate cards.

In a system where processors and memory are separated, two kinds of boards would need to be installed. The processor board, possibly containing two or more processors, would be one of the two boards installed. Memory would be added in a separate MicroChannel connector. All reads and writes to memory, due to cache misses, would be sent over the MicroChannel. This would provide a design similar to the Multimax design described in "Appendix A. Encore's Multimax" on page 52.

There are several advantages to the Multimax memory design. One of the most significant is the ability to interleave memory cards. Thus, while one memory card is recovering from servicing a memory request, a second could be handling the next request. This scheme takes advantage of the fact that memory cycle times are larger than memory access times.

There are two problems with the above scheme for a MicroChannel architecture. First, the MicroChannel has a limited number of 32-bit slots. Thus, MERLIN chooses to give the user one processor and 8 Megabytes of memory on one card rather than two processors and 16 Megabytes on two separate cards. The net effect is the same, but for systems with other 32-bit MicroChannel needs, it is more convenient to use one slot at a time rather than two. Secondly, using the 82385 cache and memory router described above, the MERLIN card does not have to present a read cycle

to the MicroChannel in order to read from the card's local 8 Megabytes of DRAM. Thus, if a cache miss occurs to a location that physically resides on the MERLIN board, it can be read without affecting the channel. For some applications, this could be a large performance gain. Splitting processor and memory would negate this gain.

3.8 Virtual Memory Coherence

Figure 12 on page 44 and Figure 13 on page 45 demonstrate how two processors share virtual memory locations in the MERLIN system. All tables are stored in memory and all processors have access to this memory. Thus, once the operating system properly initializes the processor registers to point to these tables, memory can be shared. Since all processors view the same set of tables, when the operating system updates a location, every processor can see the change. Each processor is required to update its translation lookaside buffer (TLB) whenever changes to the page table occur. Thus, each time the page tables are changed, every processor must execute code to update the TLB. As with any single processor multitasking operating system, the tables are stored in RAM that is protected by the operating system kernel.

Once all tables are initialized, the operating system can page memory to and from disk. Each MERLIN's processor TLB and cache memory will be coherent during program operation.

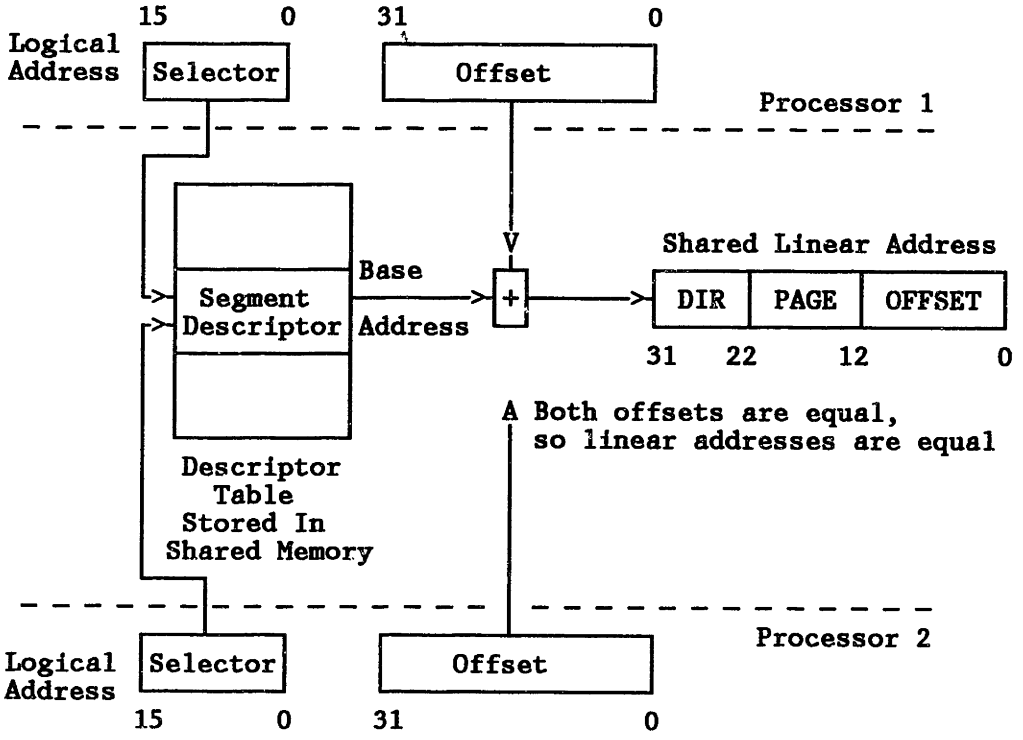

Figure 12. Segment Translation For Shared Location: Linear address translation for shared memory location. Both Processor 1 and Processor 2 use the shared descriptor table. The selector and offset values are equal for the shared location.

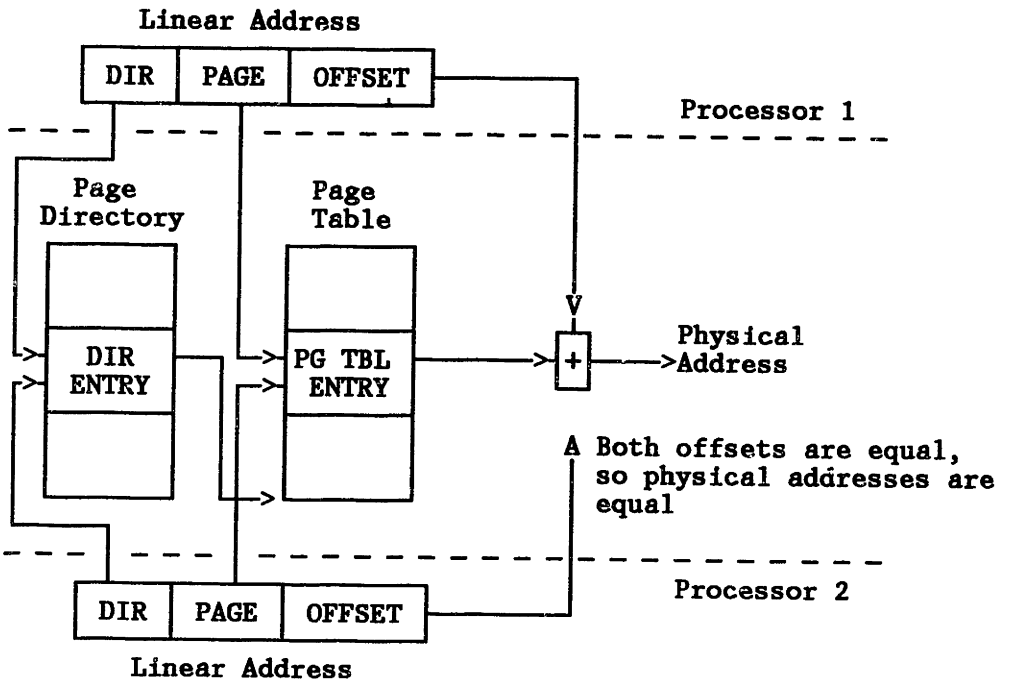

Figure 13. Page Translation For Shared Location: Page translation for shared memory location. Both Processor 1 and Processor 2 use the same page directory and page table. The linear addresses are equal for the shared location.

4.0 MERLIN's Design For a Model 70

The IBM PS/2 Model 70, unlike the Model 80, has a 64 Kilobyte cache (based on the Intel 82385 cache controller) between the processor and the MicroChannel memory system. MERLIN's design can be adapted to the Model 70 in order to provide higher overall system performance than is available on the Model 80. Since it is likely that future members of the IBM PS/2 family will use processor caches, the MERLIN design described in this chapter is best suited to the next generation of IBM personal computers.

Previous sections have made the point that every memory cycle on the Model 80 must be presented to the MicroChannel. This leaves approximately 10% of the available bus bandwidth for all the MicroChannel adapter installed in a system to use. As a result, it is impractical for a MERLIN card to post all memory writes in order to keep global caches coherent. Several MERLIN cards operating in parallel would spend too much time waiting for a bus grant, negating the benefits of multiprocessing. The Model 80 design caches only the local 8 Megabytes of memory for this reason.

Since the Model 70 has a cache servicing the system board's processor memory requests, memory cycles are only presented to the MicroChannel on cache misses and memory writes. Measurements show that less than 33% of the MicroChannel bus bandwidth is used in typical applications. Because there is a great deal of bandwidth not being used by the processor, it becomes feasible for MERLIN cards to post their memory writes. Thus, the Model 70 MERLIN design caches the entire 4 Gigabyte physical address space of the 80386.

The hardware of the MERLIN card (80386/80387 processor core, 82380 DMA controller, 82385 cache controller, and DRAM) remain unchanged when converting MERLIN to be used in a Model 70 architecture. In addition, MERLIN's use of I/O ports and MicroChannel POS bits remains the same. Changes do occur, however, in the memory system of the Model 70 MERLIN card.

4.1 Memory Router Subsystem

The Model 80 based MERLIN card had a memory router subsystem that was responsible for routing MERLIN memory accesses to either local DRAM or ROM or using the MicroChannel in bus master mode to write data to a memory slave. The router also had to watch for and service MicroChannel accesses to the local MERLIN card. The Model 70 based MERLIN card's memory router subsystem operates slightly differently.

Like the Model 80 based router, the Model 70 router accesses either local memory or MicroChannel memory on processor read cycles. The router determines if the location desired is mapped into local memory and if it is, proceeds to read from local memory without interrupting the MicroChannel. If the location desired is not in local memory, the router arbitrates for the channel and performs a remote memory read cycle to obtain the data.

On a memory write, the Model 80 router decided whether data should be written into local memory or onto the channel. Every memory location can be written either to local memory or onto the channel, never to both. The Model 70 router, however, always writes the memory location onto the channel, regardless of its location in physical DRAM. Thus, if the processor writes to a location not present in the local MERLIN's DRAM, the write cycle is presented to the MicroChannel. However, if the processor writes to a location that is present in the local MERLIN's DRAM, the write cycle is presented to *both* the local DRAM and the MicroChannel. In this way, every device listening to the MicroChannel is kept informed of updates to every location of memory. This is a critical feature needed to obtain global cache coherency (see "Cache System" on page 48).

4.2 *Cache System*

For the Model 70 based MERLIN, the memory router system posts every processor write cycle onto the MicroChannel. Every cache can watch the channel, therefore, to recognize if a cached location has been updated. If a cached location has been updated, the cache may choose to invalidate the line or update the entry.

As described above, the Intel 82385 processor includes a snoopy bus feature. The Model 70 based memory router presents to the snoopy bus every processor write (regardless of whether it addresses a local location in memory). If the snoopy bus recognizes an address as being cached, it invalidates the appropriate cache line. Thus, when the 80386 processor tries to fetch this location, a cache miss occurs and the cache is forced to read the most current value from shared memory. In this way, caches are kept coherent throughout the system.

For a Model 70 based MERLIN system, a "smart" cache algorithm that distinguishes shared and private memory locations would have excellent performance. Adding a "smart" cache to the MERLIN design, however, would have substantially increased the card's chip count. Since it is likely that future cache controllers for use with the 80386 will provide better algorithms, engineering a separate cache solution would not have been a justifiable choice.

5.0 Conclusion

The MERLIN card provides a shared memory multiprocessor for the IBM PS/2 MicroChannel. Because most members of the PS/2 family do not contain processor caches, two designs have been proposed. Every MERLIN card provides an 80386 based processor system with a 32 Kilobyte memory cache. The memory router system presents MERLIN memory cycles to the MicroChannel and watches for remote access to local RAM. The Model 80 based MERLIN chooses to cache only local RAM in order to minimize MicroChannel traffic. The Model 70 based MERLIN caches the 4 Gigabyte physical address space of the 80386 microprocessor. Cache consistency is maintained using posted write cycles.

The design of the MERLIN card described in this thesis demonstrates how a multiprocessor system can be supported in IBM's MicroChannel architecture. However, the completion of the design of the hardware does not end the project. It is still necessary to develop an operating system capable of taking advantage of the powerful computing capabilities of a shared memory MIMD architecture. Modifications to IBM's OS/2 could prove the simplest and most effective way to implement the operating system.

The MERLIN board was designed with the engineering constraints of the MicroChannel in mind. Slots, power, and bandwidth are all limited quantities. MERLIN tries to optimize its use of each. The multiprocessors systems described in the appendices demonstrate other shared memory MIMD architectures that take approaches different than the MERLIN design. These architectures, however, were not designed as cards for an IBM PS/2 workstation, and did not suffer the same engi-

neering constraints. In addition, these systems are substantially more expensive than an IBM PS/2 with several MERLIN cards. A MERLIN card costs less than one-half of the price of a PS/2 Model 70.

MERLIN's design motivates a new way to design high performance desktop workstations. Rather than concentrate on the power of a single processor and the flexibility of the interconnection bus for providing memory peripherals, I believe that workstations should be sold as units containing motherboards that support a high speed network. Both the I/O and the processing capabilities of the system should be installed as cards attached to the network. The resulting box would have a high degree of parallelism and allow a heterogeneous processing system to operate together smoothly.

6.0 Bibliography

Cornejo, Ciro and Lee, Raymond *Comparing IBM's MicroChannel and Apple's NuBus.* Byte: McGraw Hill Publishing, Vol. 12, Number 12, 1987.

Encore Computer Corporation *Multimax Technical Summary.* Encore Computer Corporation, 1987.

Intel Corporation *80386 Hardware Reference Manual.* Intel, 1987.

International Business Machines Corporation *IBM Personal System/2 Model 80 Technical Reference.* IBM, 1987.

International Business Machines Corporation *IBM Personal System/2 Seminar Proceedings.* Volume 5, Number 3. IBM, 1987.

Pfister, G. F. *The Architecture of the IBM Research Parallel Processor Prototype (RP3).* IBM Research Report RC11210, IBM T. J. Watson Research Laboratory, 1985.

Pfister, G. F. and Norton, V. A. *"Hot Spot" Contention and Combining in Multistage Interconnection Networks.* IEEE Transactions on Computers, Volume C-34, Number 10, October 1985.

Shiell, Jon *The 32-bit MicroChannel.* Byte: McGraw Hill Publishing, Vol. 12, Number 12, 1987.

Appendix A. Encore's Multimax

Encore Computer Corporation's Multimax systems are shared memory MIMD multiprocessor computers with 2 to 20 32-bit processors. Total system performance reaches a peak rate of 40 MIPS. The Multimax system is built around a bus one foot in physical length, the Nanobus. Onto the Nanobus plugs one System Control Card, up to 10 processor cards, and up to 8 shared memory cards.

A.1 Nanobus Communication Channel

The Nanobus is a wide synchronous bus operating at 12.5MHz. The Nanobus provides a 32-bit wide (with 4 parity bits) address bus and a separate 64-bit wide (with 8 parity bits) data bus. A 14-bit vector bus provides a path for interrupt vector distribution throughout the Multimax. In addition, the Nanobus provides a control bus.

The Nanobus supports several simultaneous transfers, allowing many requests to be posted by memory requesters before the memory system can respond. In addition, bus interfaces can pipeline multiple bus transactions requests by buffering them at different stages of processing. A memory card, for instance, can send data to a memory requester while simultaneously accepting an unrelated request for data from another requester. The Nanobus protocol guarantees synchronized read-modify-write cycles without compromising atomicity and without delaying system activity while synchronization activities take place.

A.2 System Control Card

Every Multimax is equipped with one System Control Card (SCC). Like all Multimax cards, the SCC plugs into a slot of the Nanobus. The SCC provides several key diagnostic and bootstrapping functions for the Multimax including:

- Supervising hardware fault diagnostics
- Performing environmental monitoring
- Providing interface to front panel switches and indicators
- Providing local and remote console terminal interface
- Mediating bus arbitration
- Generating bus timing signals
- Providing interval timing and time-of-year clock
- Controlling system start-up, building a configuration map of existing resources, sizing memory, and assigning optimum interleaving characteristics (Encore, 1987).

The SCC's interface to the Nanobus provides two-way traffic between the Nanobus and the SCC shared bus. The SCC provides separate address and data bus arbiters as well as a vector bus arbiter. The address bus arbiter decides priority on a round-robin basis. The data bus arbiter uses a fixed priority algorithm that gives highest priority to the SCC. In addition, the SCC Nanobus interface generates all bus timing signals for the Nanobus.

A.3 Advanced Dual Processor Card

The Multimax 320 uses National Semiconductor's NS32332 as its main processing engine. This chip is a member of the National Semiconductor 32000 family. The NS32332 architecture supports the primitive data types of: integer, floating point, Boolean, BCD, and bit fields. In addition, the structured types of: matrices and arrays, records, string, and stacks are supported. The NS32081 Floating Point Processor can be coupled with the NS32332 in order to provide high speed double precision floating point support.

Each processor card for the Multimax provides two independent 15MHz NS32332 processors, each with its own private 64 Kilobyte cache (see "Cache System" on page 55 for a discussion of the Multimax cache system). Using the NS32382 Memory Management Unit, the processors can generate a 32-bit virtual address that is translated to a 32-bit real address.

A.4 Shared Memory Card

Each Shared Memory Card (SMC) provides 4 or 16 Megabytes of RAM (with an error correction system) in two independent banks. Each card supports 2-way interleaving between banks and 4-way interleaving between boards. This permits 8-way interleaving of memory on systems that have at least four SMCs. The base address and interleaving characteristics of each SMC are set by the SCC at system startup.

The memory cycle time of each SMC is four Nanobus cycles, or 320 nanoseconds. During this time, an SMC can compose up to eight bytes of data for transfer (on the 64-bit data bus) to the Nanobus or accept and store up to 8 bytes received from the Nanobus. Since the bus architecture allows another interleaved board to begin a new 8-byte memory transfer with each successive bus clock cycle, the four boards involved in 8-way interleaving can transfer two doublewords of data at an aggregate rate of 100 Megabytes per second (Encore, 1987).

Every byte stored on an SMC can be used as a multiprocessor lock. These locks can be set, reset, and tested across the Nanobus using atomic read-modify-write bus cycles. Other processors, when testing the state of a lock, will first read the byte's contents from the SMC into their cache, and subsequently read from the cache until the value of the lock changes (the change is posted as described in "Cache System" on page 55). As a result, no load is imposed on the Nanobus or the SMC during the time spent waiting for the lock to change state. However, processor cycles are wasted by constantly checking the state of the variable.

A.5 Cache System

As on the MERLIN card, each processor on a Dual Processor Card has a 64 Kilobyte cache. The processor spends most of its time reading data and instructions from this cache memory. The Nanobus implements a memory write posting scheme in order to keep processor caches coherent.

Each processor cache continuously scans the Nanobus for memory writes involving locally cached addresses. When such writes are detected, the valid bit for the indicated cache address is switched to its invalid state. Later, when the processor needs the data for that cache address, the cache controller fetches the data from main memory. Bus snooping is performed in parallel with processor servicing in order not to degrade cache performance. It is the responsibility of each cache to post all processor writes on the Nanobus.

A.6 Comparison To MERLIN

There are several key architectural differences between MERLIN and the Multimax. The primary difference is the use of a posted write scheme for keeping caches coherent on the Multimax. In an IBM PS/2 Model 80, the MicroChannel is swamped with service requests for the system processor. Every read and write cycle of the system processor is presented to the MicroChannel. With the remaining bus bandwidth, the MERLIN cards would be hopelessly inefficient if they had to post every write cycle onto the channel in order to keep the caches coherent. Since every processor on the Nanobus has a large local cache, memory requests are only presented to the Nanobus on cache misses and memory writes. Thus, no one processor swamps the bus.

A MERLIN card operating in a IBM PS/2 Model 70 could behave like a Multimax processor card. Since each Model 70 contains a 64 Kilobyte cache between the processor and the system bus, there is substantial bandwidth to support posting write requests on this bus. A MERLIN card could

cache the entire 4 Gigabyte address space and keep other cards coherent by supporting write posting.

The Nanobus is further optimized for multiprocessor designs by the inclusion of a interrupt vector bus and double width data bus. The MicroChannel makes no provisions for either of these two features.

Appendix B. IBM's Research Parallel Processor Prototype (RP3)

RP3 is a 512 node MIMD processing system with a unique memory organization that encompasses both shared memory and message passing MIMD architectures. A full RP3 configuration can reach a peak performance of 1300 MIPS and is predicted to demonstrate a steady state performance of 1000 MIPS for several scientific applications. RP3 uses two intercommunication networks to connect Processor- Memory Elements (PME): a high speed Omega network for memory accesses and a lower speed combining network for process synchronization tasks.

B.1 RP3 Communication Network

Both the high speed memory network and lower speed combining network use a packet switching technique, combining circuit switching and packet switching methodologies. A message is pipelined across switch stages as if circuit switched; but when blocked some or all of the message is queued within a switch stage as if packet switched. Since messages are pipelined, paths are dynamically allocated by a dynamic routing system.

The high speed network is used by the PMEs to gain access to shared memory locations. Thus, the network provides a two-way communication path: from PME to memory system and from memory system to PME. There are no direct processor to processor connections.

The combining network supports the RP3 synchronization operations (described in "Processor-Memory Element" on page 59). When packets arrive at a switch, their destination PME is com-

pared. If both packets are addressed to the same PME, the switch tries to combine the requests into one request. When the single result is returned, the switch separates the results and returns separate results to each requester. The combining network also provides a two-way communication path.

The combining network was created to address the shared memory location hot spot issue. Shared memory locations, especially those used for process synchronization, tend to be accessed by several processors simultaneously. The PME containing the memory location is suddenly flooded with requests for access to this location and request packets are stalled in the network as they wait to be served. Because packets are queued in the network, other packets (to different memory location) cannot be routed through the network. As a result, even a small percentage of processor references to a hot spot location can cause a major decrease in overall system performance by effectively freezing network routing while hot spot references are being serviced. The RP3 designers address this problem using a combining network with queues at each node (Pfister and Norton, 1985). By combining messages, hot spot memory accesses are reduced resulting in reduced network queueing problems. The resulting network performance is comparable to a network operating without any hot spot memory locations.

B.2 Address Structure

The RP3 address translation is performed in two levels: a segment/page mapping and an interleave transformation. Because two levels of translation are used, three address representations are required: virtual (before any translation occurs), real (after segment/page translation, but before interleave), and absolute (the result of the entire translation process).

A virtual address is composed of a segment index, page index, and a page offset. The segment/page mapping is applied to the virtual address in order to produce a real address along with additional transformation information, the interleave amount. The interleave amount and the real address are inputs to the interleave transformation. The absolute address is computed by right-rotating the bits

of the real address by an amount specified by the interleave amount. The absolute address is composed of a PME number and a memory offset. The PME number identifies the memory to which the request is to be transmitted. The memory offset specifies the location to be accessed within memory.

The interleave amount ranges from 0 to the base 2 logarithm of the maximum number of PMEs in the system. For the RP3, the interleave range is 0 to 9. An interleave amount of N results in consecutive addresses within a page to be interleaved across 2^N memories. When the interleave amount is zero, the real and absolute addresses are identical.

Prior to the interleave transformation, the real address can be optionally hashed (the option is set during system configuration). The hashing is a page-dependent one-to-one reordering of addresses within a page. The purpose of the hashing function is to improve system performance by uniformly distributing interleaved addresses across the PME memories regardless of the access pattern within the virtual address space.

B.3 Processor-Memory Element

The RP3 Processor-Memory Element includes an 801 RISC microprocessor, memory-mapping unit, 32 Kilobyte cache, vector floating-point unit, I/O interface, 4 Megabytes of RAM, and a performance monitoring device. A fully configured RP3 contains 512 PMEs. Smaller systems may be created using 8 or 16 PMEs for experimentation.

Since the 801 processor was not designed to be used in parallel system, its instruction set was augmented to include special coordination, serialization, and cache control (see "RP3 Cache" on page 60) functions. Coordination functions include fetch-and-ops (eg. fetch-and-add). The PME supports interprocessor interrupts for coordination and synchronization of processes. A processor can

cause an interrupt to be generated at any other processor by invoking an interprocessor-interrupt function.

Since the processor prefetches and since the network supports up to 16 concurrently outstanding stores, program order may be violated which, in turn, violates sequential consistency. To guarantee sequential consistency, the PME defaults to restricting the number of outstanding requests to shared data to one. The limitation of one outstanding request to shared data is sufficient in general, but is overly restrictive in a number of cases. Thus, the RP3 implements functions to permit the programmer or compiler to control the serialization of access to shared data.

The floating point unit of every PME serves both as a numeric co-processor and as a DMA controller for the PME. The floating point can be used to move blocks of data asynchronously to processor operation.

B.4 RP3 Cache

Two different solution techniques can be used to ensure cache coherence in the RP3 memory hierarchy: run-time checks or compile-time checks. MERLIN's design employs run-time coherence checks. The designers of RP3 felt that run-time checks introduce serialization problems and are not suitable for large parallel systems. Compiler-time checks tag every memory location as either cacheable or non-cacheable. Although compile-time checks are static (the cacheable/non-cacheable attribute is determined during compilation), the attributes are not restricted from being changed between program segments. Between two program segments, the usage of a location may change from shared read-write to read-only, changing the location from non-cacheable to cacheable. Thus, the RP3 maintains cache coherency in software and associated with every memory location is its ability to be cached.

Hardware caches are normally invisible to the programmer (except for performance differences) where requiring software to maintain cache coherence removes this convenience. The RP3 designers chose to use software coherency since it is was considered infeasible to keep caches coherent using posted writes for a 512 processor system. In addition, since optimizing compilers on a parallel system must be aware that data is shared read/write, they must selectively inhibit optimizations such as keeping the data totally in internal processor registers. Thus, classifying a location as cacheable or non-cacheable results from compiler variable decisions (Pfister, 1985) and requires no additional compiler overhead.

B.4.1 Cache Organization

Since multiple PME devices (the processor, floating point unit, and I/O interface) can issue data requests to the cache, the cache design must minimize the penalty of a cache miss. Thus, the RP3 cache is designed to be lockup free - the cache continues to satisfy other requirements while a cache line is being fetched to satisfy a cache miss. A lockup free design increases the utilization of each device by reducing effective memory access time. The cache also permits devices to prefetch requests without the penalty of locking out other device loads and stores if the prefetch causes a miss. The RP3 cache supports up to eight concurrent misses.

When a cache miss occurs, two doubleword memory requests are issued to fetch the cache line (16 bytes). Since interleaving is used, the requests are generally directed to different PMEs. The RP3 cache uses a write-through policy. Although write-through produces a larger amount of network traffic than write-back, the RP3 interconnection has sufficient bandwidth to handle the increased traffic with only a slight loss in system performance. The designers traded performance degradation for cache simplicity.

B.5 RP3 Memory

Each PME has two or four Megabytes of storage. Thus, the total amount of memory in a 512 node RP3 is 1 or 2 Gigabytes. The memory system is sophisticated in that it performs the atomic arithmetic functions needed to support the various fetch-and-op functions. Currently, fetch-and-add, fetch-and-and, fetch-and-or, fetch-and-min, fetch-and-max, fetch-and-store (swap), and fetch-and-store-if-zero are supported. In addition, the memory provides support of the interprocessor interrupt system described above.

When the PME cache needs memory, a memory router is presented with the absolute address (PME number and offset). If the PME number corresponds to the node number, the access is performed locally, without involving the network.

B.6 Comparison To MERLIN

The MERLIN and RP3 designs share several features. Both designs chose to combine memory and processor into one functional unit attached to the network. As a result, if a memory request is performed to local memory, the router does not need to access the network yielding greater network availability.

RP3 designer's concentrated on the system's intercommunication network. MERLIN's intercommunication network was a given: the MicroChannel. Because the RP3 design uses 512 nodes it is necessary to include high speed networks and utilize combined memory reference techniques. Systems for which MERLIN cards are designed would have only a few processors (less than 20). As a result, a linear communication channel suffices for the MERLIN design. In the RP3, however, a linear channel would serve as a performance bottleneck and negate the benefits of using several processors in parallel to complete a task. Since the RP3 network can support multiple simultaneous messages, synchronization problems can occur. Multiple messages also provide the

ability to combine similar messages destined to the same memory location. The MicroChannel cannot support multiple messages and need not take advantage of combining messages.

MERLIN chose to make its cache transparent to the programmer. While it may be true that software coherency is the only efficient method for ensuring cache coherency in a 512 node system, the additional programming difficulties will make the process of converting applications to use the RP3 architecture more challenging.