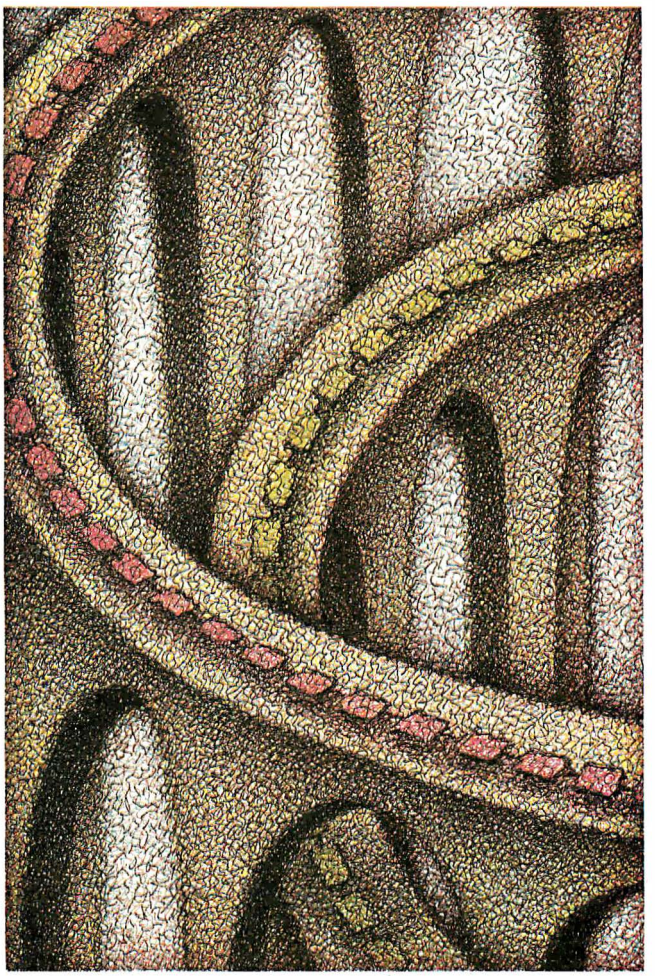
Buses provide the base for the next-generation personal computer/workstation

Comparing IBM's Micro Channel and Apple's NuBus

Ciro Cornejo and Raymond Lee



The 32-bit bus has finally arrived for the personal computer in the form of Apple's Macintosh II and IBM's Personal System/2. Central to each of these machines is a 32-bit bus capable of high-speed operations at the bandwidth required for today's 16-megahertz processors.

As you might expect, though, Apple and IBM have adopted dissimilar bus architectures. NuBus, developed by MIT and Texas Instruments, has been adapted by Apple as the Macintosh II system bus. It supplements the Macintosh II's private 68020 processor bus, and six slots open the microcomputer for expansion. IBM, on the other hand, has revamped the older IBM PC and AT bus to handle higher speeds and 32-bit processing. The new IBM bus, the Micro Channel, serves as both a CPU bus and a system bus.

It's no accident that these new computers contain new bus architectures. Today's new microcomputers require more than just increased processor power and expanded memory. Investing in these products means committing to computing platforms that must be stable up to and perhaps through the mid-1990s. This requires a bus-based architecture capable of adapting to expanding processing rates, coprocessing or multiprocessing, and adapting to new peripherals such as advanced graphics terminals. I'll examine the two buses with regard to these capabilities.

Why a Bus?

Why have a special bus at all? Most processors actually define a bus structure of address and data paths called a *local*, or *CPU*, *bus*. The reason for this is straightforward: to build generality into systems.

A local bus is structured to optimize the processor-to-memory bandwidth. It is therefore highly processor-dependent: It is tightly linked to its processor, memory, and specific support peripherals. The cost of this performance is a loss of flexibility. A local bus might be unable to take advantage of newer technologies if they differ significantly from the local bus's design.

The existing IBM PC or AT buses are examples of an *expanded local bus*. An expanded local bus is a local bus with extensions that provide a set of generalized signals. These additional signals offer a general architecture that is easy to interface with. Since they use many of the processor's signals, expanded local buses are still processor-specific. For example, the IBM PC and AT buses are designed around the Intel 80x8x microprocessor architecture, and they have problems accommodating large memory expansions. The PC is limited to 1 megabyte of RAM (the 640K-byte limit is imposed by the layout of the PC BIOS), and the AT to 16 megabytes.

Unlike local buses, *system buses* are designed to maximize hardware subsystem-to-subsystem transfers. System buses offer a general protocol, or transfer method, for system CPUs or peripherals to interchange data. This is accomplished by treating the bus as a resource. To get control of the resource, a peripheral or processor must request its use formally, in competition with others. With this general approach, you can add peripherals, special functions, and even full computer subsystems easily

continued

Ciro Cornejo is an engineer with AST Research (2121 Alton Ave., Irvine, CA 92714). He was born in Chile, and his interests are nature, computers, math, and physics. Raymond Lee, a technical advisor at AST Research, is interested in computer architecture.

to a system bus. The bus integrates hardware, cards, and subsystems into one smoothly running machine, much as an operating system integrates applications programs. The more general the integrating mechanism, the easier it is to add functionality and avoid obsolescence. Moreover, these buses are processor-independent. For example, NuBus defines a generalized address space that requires no processor-specific signals for peripheral or I/O accesses.

Overview of the Buses

The new IBM Micro Channel has evolved from the earlier PC and AT buses. Like them, it is a CPU or local bus once removed. As a local bus, it optimizes the host CPU-to-memory bandwidth, using a special transfer method termed "matched memory cycles," which I'll discuss later. It is also a system bus, in that it is treated as a system resource.

Taking an opposite tack, the Apple NuBus is a full system bus. It is independent of the Macintosh II's host processor; in fact, in the Mac II the motherboard is treated as a NuBus slot.

Both buses create a memory-mapped system. Each card or hardware entity is addressed within this bus address space. The 16-bit PS/2 systems, the Models 50 and 60, address a 24-bit space, or 16 megabytes; the PS/2 32-bit Model 80 and the Macintosh II NuBus address a full 32-bit space, or 4 gigabytes. Like its predecessors, the Micro Channel also has a 64K-byte I/O space.

Table 1: *A comparison of the two buses. Not all bus signals are included.*

	NuBus	Micro Channel
Utility signals		
Reset	RESET*	-CHRESET
Clock	CLOCK*	OSC
Interrupt	NMRQ* [Note 1]	-IRQ (3-7, 9-12, 14-15)
Audio	-----	AUDIO, AUDIO GND
Control signals		
Start bus cycle	START*	A0-A31, M/-IO, MADE24 [Note 2]
End bus cycle	ACK*	-CMD, or -MMC CMD
Cycle definition	TM0*, TM1*	M/-IO, -S0, -S1
Burst control	[Note 3]	-BURST, -TC
Address size	[Note 4]	-MADE 24
Data size	TM0*, TM1*, AD0*, AD1*	-BE0 through -BE3, TR32, -SBHE -CD DS 16 [Note 1], -CD DS 32 [Note 1]
Matched memory cycle	-----	MMC, -MMCR
Signal returns		CHRDY RTN, -DS 16 RTN, -DS 32 RTN
Slot occupancy	[Note 5]	-CD SFDBK
Address/data		
Address	AD0* through AD31*	A0 through A31
Data	AD0* through AD31*	D0 through D31
Arbitration		
Request bus	RQST*	-PREEMPT
Arbitration lines	ARB0* through ARB3*	-ARB0 through -ARB3, ARB/-GNT
Slot ID	ID0* through ID3* [Note 6]	[Note 7]

Note 1: Separate line for each slot or card.
Note 2: For a memory refresh cycle, -REFRESH will also be used.
Note 3: Although a burst mode is defined in NuBus, it is not used in the Apple version of NuBus.
Note 4: All addresses on NuBus use 32 bits of address space.
Note 5: The declaration ROM must respond to a read at the top of the slot space.
Note 6: These lines are not bused.
Note 7: ID is stored on card but is not used for arbitration. A separate arbitration level is stored on the card when it is configured into the system.

Each bus entity can be defined as a *master* or a *slave*. A master entity can request and get control of the bus. A master must own the bus to send or receive data from another target entity on the bus, which can be another master or a slave unit. A bus slave unit cannot own the bus, but it can request service through an interrupt signal to one of the bus masters.

The masters contend for ownership of the bus resource via an arbitration protocol, which I'll describe later. Both buses allow multiple masters. However, only the Apple NuBus provides mechanisms for true multiprocessing: bus and resource locks. Bus locking allows a processor to lock a bus for exclusive access. With resource locking, a shared resource, such as RAM on a card with its own local processor, is locked so that the local processor can't access it. Both types of locks are necessary to prevent one processor from interfering with or corrupting memory that another processor is using.

While the IBM Micro Channel does permit multiple masters, there is not much to be gained in going to multiple host processors. This is because the Micro Channel is also an extension of the CPU bus. Processor memory operations tie up the Micro Channel, making the bus a bottleneck for the concurrent operation of two host processors. It should be noted that IBM, for efficiency, allows the host processor to access system motherboard memory without passing through the Micro Channel bus.

Also hindering multiprocessing on the Micro Channel is the absence of any direct provisions for bus or resource locking, although the 80386 in the IBM PS/2 Model 80 has hardware for bus locking. While not intended for true host-level multiprocessing, the Micro Channel does offer a general interface for drop-in coprocessing. The PS/2 host processors can be easily supplemented by powerful coprocessors, such as array and floating-point processors, or AI compute engines.

Timing the Critical Element

As a logic designer once said, "There are three important aspects of a digital design that must be carefully monitored: timing, timing, and timing." This is still true, especially for computer and bus designs.

Difficulties usually start when one block of logic has to talk to another block, especially if they each rely on different clock signals. This requires that the signals be synchronized to be passed from one logic block to another. A transmitting signal from a flip-flop strobed with one clock must be picked up and strobed into a receiving flip-flop using a second clock. The two clocks, transmitting and

receiving, are asynchronous; no fixed relationship exists between them. Thus, it can take one receiving clock period to synch up to the transmitting data.

Buses, like logic, define synchronous or asynchronous interactions. In a synchronous bus, all interactions are defined in terms of a fixed bus clock or cycle. The bus clock edges define when data is valid and when to strobe it. Moreover, all transactions are in multiples of these bus cycles. The Apple NuBus is a synchronous bus.

Instead of relying on a fixed clock, an asynchronous bus is controlled by handshaking signals. A command signal is sent to a target adapter or card that responds with an acknowledge signal upon completion of a data transfer. All bus timing is dependent on the signals themselves. The IBM Micro Channel is an asynchronous bus, although it supports certain synchronous transfers.

Both the IBM Micro Channel and the Apple NuBus pass a common clock through the bus to minimize the synchronization problem among bus entities. However, there is a clock mismatch between the Macintosh II's local bus and NuBus, requiring synchronization before a transfer can occur.

The Macintosh II's 68020 runs with a 15.7-MHz clock, while NuBus runs with a 10-MHz clock. Synchronization delays between these bus clocks is minimized by using high-frequency clock signals. The NuBus 10-MHz clock is divided down from a 40-MHz crystal; the 68020 15.7-MHz clock is divided down from a 31.4-MHz crystal. The cost of clock synchronization is thus held to one clock period, either 25 or 31.5 nanoseconds. Clock synchronization is accomplished through the application-specific integrated circuit (ASIC)—the "GLU" custom gate array on the Mac II motherboard—and the NuBus timing control logic.

Synching up between the bus *processes* (i.e., bus reads or writes) also exacts a time penalty. The requesting bus must wait for the other bus to complete its current transaction cycle before it can attempt a transfer. All NuBus operations are defined with respect to its 10-MHz system clock. This clock has a 25 percent duty cycle: It is false (or high) for 75 ns and true (or low) for 25 ns.

Normally, a transfer from NuBus to the local bus takes a full 68020 instruction cycle (about 400 to 500 ns) to synch up. Going the other way, a Macintosh II request can take a typical NuBus transaction of 2 bus cycles (about 200 ns) to synch. It must be noted that this type of delay is not out of the ordinary; it is the time penalty paid by the communications protocol between the CPU bus and the system bus.

The IBM Micro Channel is an asynchronous bus, and all operations are gauged by the transmitted and returned signals. A common 14.3-MHz clock, OSC, is provided on the bus, eliminating the problem of signal synching. Moreover, a delayed signal will be picked up by the next clock, providing a built-in safety net for bus operations.

Bus to Bus

To distinguish between the two sets of bus signals, I'll stick to each bus's naming conventions. NuBus active low signals

continued

Micro Channel Timing

The IBM Micro Channel is an asynchronous bus. Handshake signals are used to initiate processes, signal availability of addresses and data, and completion of operations. It is the signal changes and the logic's response to them that drives this asynchronous bus.

The keys to this cycle are the -ADL and -CMD lines that define when the address is valid and when the data is valid. Their trailing edges can be used to strobe addresses and data as well. CD CHRDY is the mechanism for extending bus cycles. When this signal goes high, it triggers -CMD, which ends the bus cycle. See figure A for the sequence of a Micro Channel Basic Write cycle. Here is a short description of the sequence:

1. The cycle begins with the address and definition lines (-S0, -S1, M/-IO, MADE 24, TR32) defining the bus operation (read, write, memory, or I/O cycles), addressing mode (24-bit), and 32-bit transfers.
2. -ADL is asserted and defines a stable address.
3. The addressed card responds by asserting -CD 16, -CD 32, and -CD SFDBK. -CD 16 is asserted for 16- and 32-bit operations, -CD 32 for 32-bit operations. The card asserts its -CD SFDBK line to acknowledge being addressed. CD CHRDY is deasserted to extend the cycle, if necessary.
4. The data appears on the bus.
5. The -CMD line is asserted, indicating that valid data is on the bus. -ADL is deasserted.
6. If CD CHRDY was deasserted, the card drives CD CHRDY active after it has read the data.
7. -CMD is deasserted, ending the bus cycle.

Other masters can be contending for the bus ownership during the bus transaction. For more details on the Micro Channel, see the article "The 32-bit Micro Channel" by John Shiell on page 59.

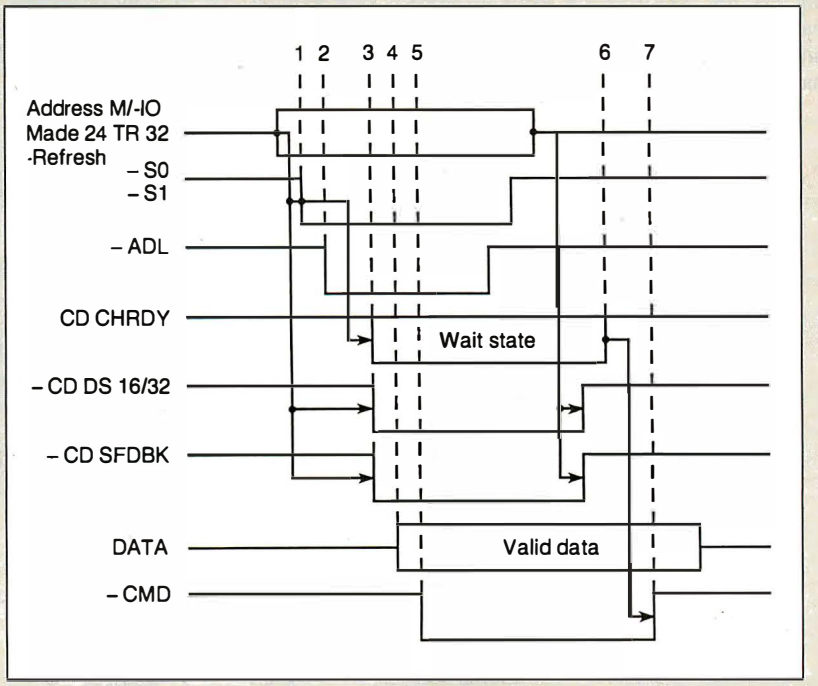


Figure A: *A Micro Channel Basic Write cycle.*

are labeled as signal_name*, while IBM uses its own convention for labeling an active low signal: -signal_name.

The NuBus is a simple and elegant bus that matches Apple's minimalist approach toward hardware. The NuBus has only 51 signals, including two parity signals not used by Apple. The IBM Micro Channel has 77 and 111 signals for the 16- and 32-bit versions, respectively. All Micro Channel signals are TTL-logic-compatible. Table 1 compares signals between the NuBus and the Micro Channel, and you can see a great deal of similarity between the two buses. The arbitration and utility signals almost match.

But there are differences. NuBus is multiplexed, sharing data and address on common lines, while the Micro Channel is nonmultiplexed, providing lines for both address and data. The IBM Micro Channel defines a number of discrete interrupts (-IRQ 3–7, 9–12, and 14–15) that can be shared among the boards. The Apple implementation, on the other hand, defines an interrupt (NMRQ*) per slot that is fed separately into the Macintosh II interrupt logic for processing.

The Micro Channel has a number of signals for coordinating asynchronous handshakes: The signals -ADL, -CMD, and -MMC CMD provide the basic bus handshake edges. Hardware signals are also used to delineate bus sizing (-BE0 through -BE3), 32-bit operation (-CD DS 32(n)), and 24-bit addressing (MADE 24). See the text box "Micro Channel Timing" on page 85 for more information on the bus cycles.

A special set of signals (-MMC, -MMCR, and -MMR CMD) is used in *matched memory cycles* to ensure fast CPU-to-memory accesses for the 80386. A matched memory cycle is started by the target slave returning an -MMCR request signal after being addressed by the system CPU. The 80386 responds by driving the faster -MMCR CMD handshake signal instead of the -CMD during a bus cycle. Matched memory cycles provide a bus read transaction in three clocks at 16 MHz, or 187.5 ns, while standard cycles using the -CMD handshake signal run four or more system clocks for a minimum of 250 ns. Matched memory cycles can be run with both 16- and 32-bit channel devices.

In contrast, the NuBus synchronous operations are relatively simple, requiring no special signals or exception processing. NuBus timing, however, is more stringent than the Micro Channel's, fitting sending and strobing of signals and data within 75 ns in the 100-ns clock cycle. See the text box "Apple NuBus Timing" below for more details on NuBus bus cycles.

NuBus defines a byte/word structure that matches the Intel 80x8x addressing schemes (byte order 0, 1, 2, 3), not the Macintosh's 68020 scheme (byte order 3, 2, 1, 0). The bus transceivers are wired to map the data from NuBus order into the Macintosh byte order. Bus sizing is handled automatically; the bus handles byte (8 bits), half-word (16 bits) and word (32-bits) sizes.

The NuBus specification defines a block, or burst mode, that can move up to sixteen 32-bit words in a transaction, but Apple has not implemented it in the Mac II NuBus design. IBM, however, has implemented a burst mode in the Micro Channel in conjunction with direct memory access. This DMA burst capability allows large blocks of data to be moved while minimizing bus overhead. In fact, each peripheral on the channel can be viewed as a DMA channel.

When accessed by the DMA controller, a card can assert -BURST, guaranteeing bus ownership for block transfers. Thereafter, data is transferred using only the -CMD signal to define data valid for both the read and write stages. The block transfer ends when the card deasserts the -BURST line for the last cycle. For predefined transfers, the DMA controller

continued

Apple NuBus Timing

The Apple NuBus is a synchronous bus; all operations are defined with respect to its basic clock cycle. The clock runs at 10 MHz, with a 100-ns period and a 25 percent duty cycle. Two edges of the clock serve the bus. The rising edge at the start of the period is the driving edge, strobing signals and address onto the bus, and the falling edge, 75 ns later, is the sampling edge for taking information off the bus.

Bus transactions are made up of bus cycles or clock periods. A transaction can be a single cycle or multiple cycles, especially if a slower peripheral is involved. Delays are added by inserting additional bus cycles. The timing diagram in figure B shows the basic write transaction, which consists of a START cycle, any intervening bus cycles, and an ACK cycle. Here is the sequence:

1. START* is asserted, indicating the start of a bus transaction. The master places addresses on the AD31* through AD0* lines; and the TM0*, TM1* lines define the type of transaction.
2. All cards read the addresses. The slave is identified by the address.
3. The master drives the data onto the AD31* through AD0* lines.
4. The slave reads the data off the bus.
5. The slave asserts ACK* to signal the end of the transaction and places the appropriate status codes onto TM0* and TM1*.
6. The master releases the AD31* through AD0* lines, and the slave releases the ACK* and status lines.

Other masters can be competing for the bus during the bus transaction.

For more information on NuBus, see "The Apple Macintosh II" by Gregg Williams and Tom Thompson in the April BYTE.

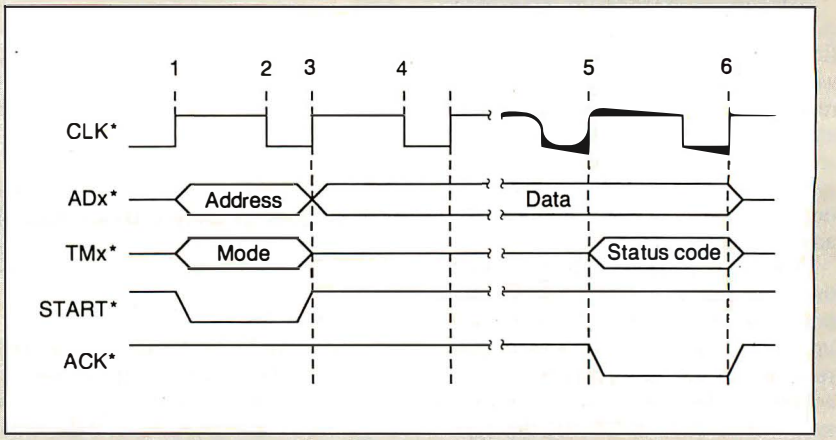


Figure B: *A Nubus write cycle.*

marks the last cycle by asserting the terminal count line (-TC).

A DMA controller can transfer 64K bytes of data between a peripheral and memory, the same as in an IBM PC. The PS/2 DMA controller can handle 24-bit read and write addresses, unlike the PC's 20-bit address limit. Unfortunately, this DMA capability is limited to transfers of 8- or 16-bit data.

Bus Address Space

Both the NuBus and the Micro Channel map bus addresses into a full bus address space that includes system memory and ROM, setup ROM, and device buffer space. Analogous to a CPU bus, these buses provide access to locations in that space.

The IBM implementation maps into a 16-megabyte or a 4-gigabyte address space. The bus address space is the same as the CPU address space. In this respect, the Micro Channel acts as a local CPU bus. The system board RAM, either 512K bytes or 640K bytes, starts at 00000 hexadecimal. The 128K-byte video RAM and channel ROM are mapped into the lower address pages. Topping off the memory space at E0000h through FFFFFh is the 128K bytes of system board ROM or RAM, depending upon how the computer's resources have been allocated. RAM memory mappings above address FFFFFh are managed in 1-megabyte chunks. See figure 1 for a memory map of an IBM PS/2 Model 80. Bits in a memory-encoding register and a split-address register determine how and where memory will be allocated.

Bus memory space for Apple NuBus implementation doesn't match the Macintosh II's 68020 processor address space. The upper one-sixteenth, or 256 megabytes, of the NuBus 4-gigabyte address space is called the *slot space*. This slot space is divided into 16 sections, one for each NuBus slot, and each slot owns 16 megabytes of the space. The top of each slot address space is reserved for a slot-declaration ROM that is accessed at that address. The slot a card occupies on NuBus determines its slot identification, which in turn determines its arbitration level and its location in the slot address space.

NuBus defines 16 slots, but the Macintosh II provides six. The six slots have IDs of 9h through Eh. Slot 0 is the Mac II motherboard, and slot F (which does not have a physical slot) is reserved. One slot becomes the video buffer for the machine, depending upon which slot the video card is placed in. Slots 1 through 8 are unused, because no room exists in the 24-bit address space for them. For this reason, the existing slots are limited to 1 megabyte of slot space instead of 16 megabytes.

Apple's implementation of NuBus allows a slot to own a "superslot" space of 256 megabytes, as well as its 16-megabyte slot space at the top of NuBus memory. We won't discuss superslots further, since they aren't accessible by the Mac II, although you should note that other cards on NuBus could use these areas. See figure 2 for a detailed look at the Macintosh II memory map and its arrangement in the NuBus address space.

The 24-bit address space for the Macintosh II starts at 0h with 8 megabytes of RAM, followed by 1 megabyte of ROM, then 6 megabytes of slot space, and topped by a 1-megabyte region of memory-mapped I/O devices. The Mac II's 24-bit address space is mapped into the 32-bit NuBus address space by placing the RAM, ROM, and I/O areas at the bottom of the NuBus address space. However, from the NuBus side, the Mac II's ROM appears at addresses F0800000h to F0FFFFFFh, and the I/O area maps to F0000000h through F07FFFFFh.

Under this scheme, the maximum RAM that can be accessed on the local bus is 8 megabytes, using 1-megabyte single in-line memory modules (SIMMs). The Mac II's motherboard RAM can be expanded to 128 megabytes if and when higher-density SIMMs are available. However, you can add more RAM to the system through the NuBus slots, and vendors are now supplying NuBus memory cards.

The Macintosh II is currently restricted to 24-bit addressing or 16 megabytes when running with the current operating system. An Apple Unix implementation (A/UX) is in the works that will handle 32-bit addressing and requires a memory-management unit for virtual-memory processing.

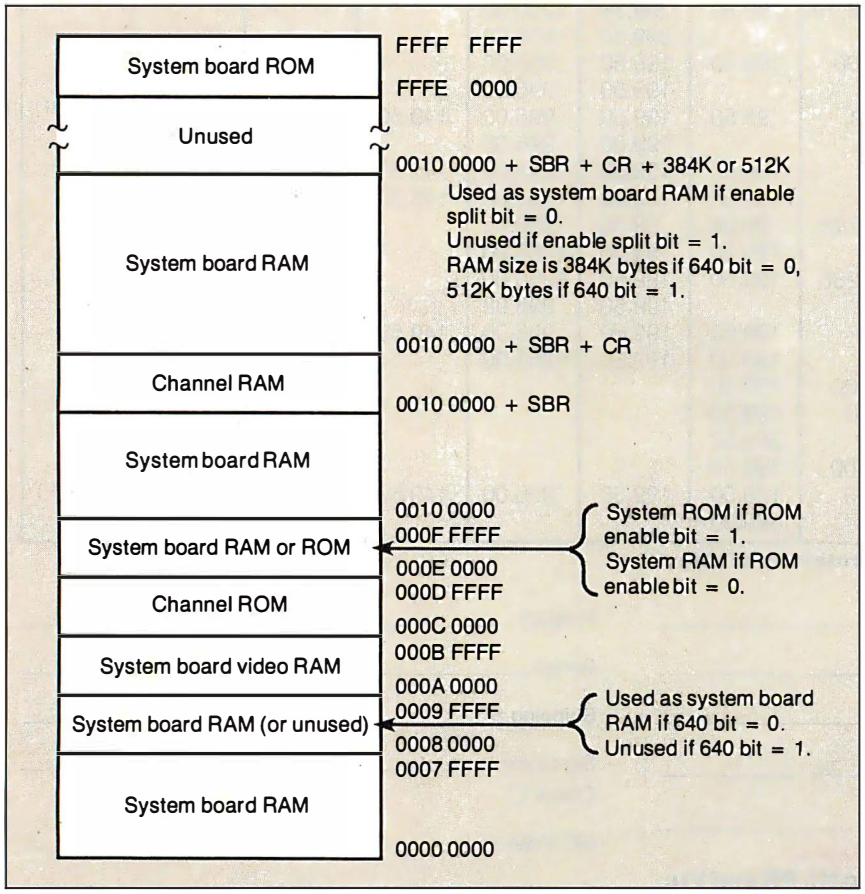


Figure 1: *The PS/2 Model 80 memory map. The memory arrangement is determined by the contents of the memory-encoding and split-address registers. SBR is system board RAM; CR is channel RAM. System board RAM and channel RAM are allocated in 1-megabyte chunks above address FFFFFh, with the exception of the split-system RAM. The system ROM at addresses E0000 through FFFFF is a copy of the system ROM at addresses FFFE0000 through FFFFFFFF.*

Bus Ownership

Both buses use arbitration to allocate ownership of the bus to a single master when several masters request use of the

continued

bus. Arbitration typically takes place concurrently with bus transactions on both buses, but the Micro Channel allows a system configuration that restricts arbitration to nonconcurrent operation.

NuBus arbitrations take two full bus cycles, or 200 ns, to select the next bus owner. On the Micro Channel, arbitrations typically take 300 ns.

Each bus uses distributed arbitration to select the next bus owner; that is, logic on each card outputs the arbitration level on four arbitration lines (either ARB0* through ARB3*, or -ARB0 through -ARB3) and determines the winner of each arbitration contest based on the signals on these lines. The arbitration level is determined in NuBus by the card's slot ID, with 0 being the lowest priority and Fh being the highest. For the Micro Channel, the arbitration level is stored on the card when it is configured into the system. The highest priority a card can have is level 0, and the lowest is Fh. See table 2 for a comparison of the arbitration levels. The Micro Channel also has a Central Arbitration Control Point, which is some logic on the PS/2 motherboard, that controls the start and winner of an arbitration contest.

To compete for ownership, the master asserts its request line (RQST* for NuBus, -PREEMPT for Micro Channel). For the Micro Channel, the Central Arbitration Control Point drives the ARB/-GNT line to the arbitrate state, allowing the arbitration contest to begin. Each master then places its arbitration level onto the 4-bit arbitration bus. If a competing master has output a higher level, the master will cease to compete for ownership for the next bus transaction. It will, however, hold its asserted request line to compete for the following bus transaction. On NuBus, at this point, the winner of the contest owns the bus. On the Micro Channel, the Central Arbitration Control Point lowers the ARB/-GNT line to the -GNT state, allowing the winner to own the bus.

Both buses ensure fairness by preventing a higher-priority-level card or channel from continuously withholding ownership of the bus from lower-priority-level entities. Card or channel logic prevents the card just serviced from requesting bus ownership until all pending requests are honored. In a sense, there are no arbitration priority levels for NuBus cards, since the NuBus strictly enforces fair bus access. However, for special cases, a channel can be configured on the Micro Channel without fairness to ensure continued ownership of the bus.

The NuBus has explicit mechanisms

continued

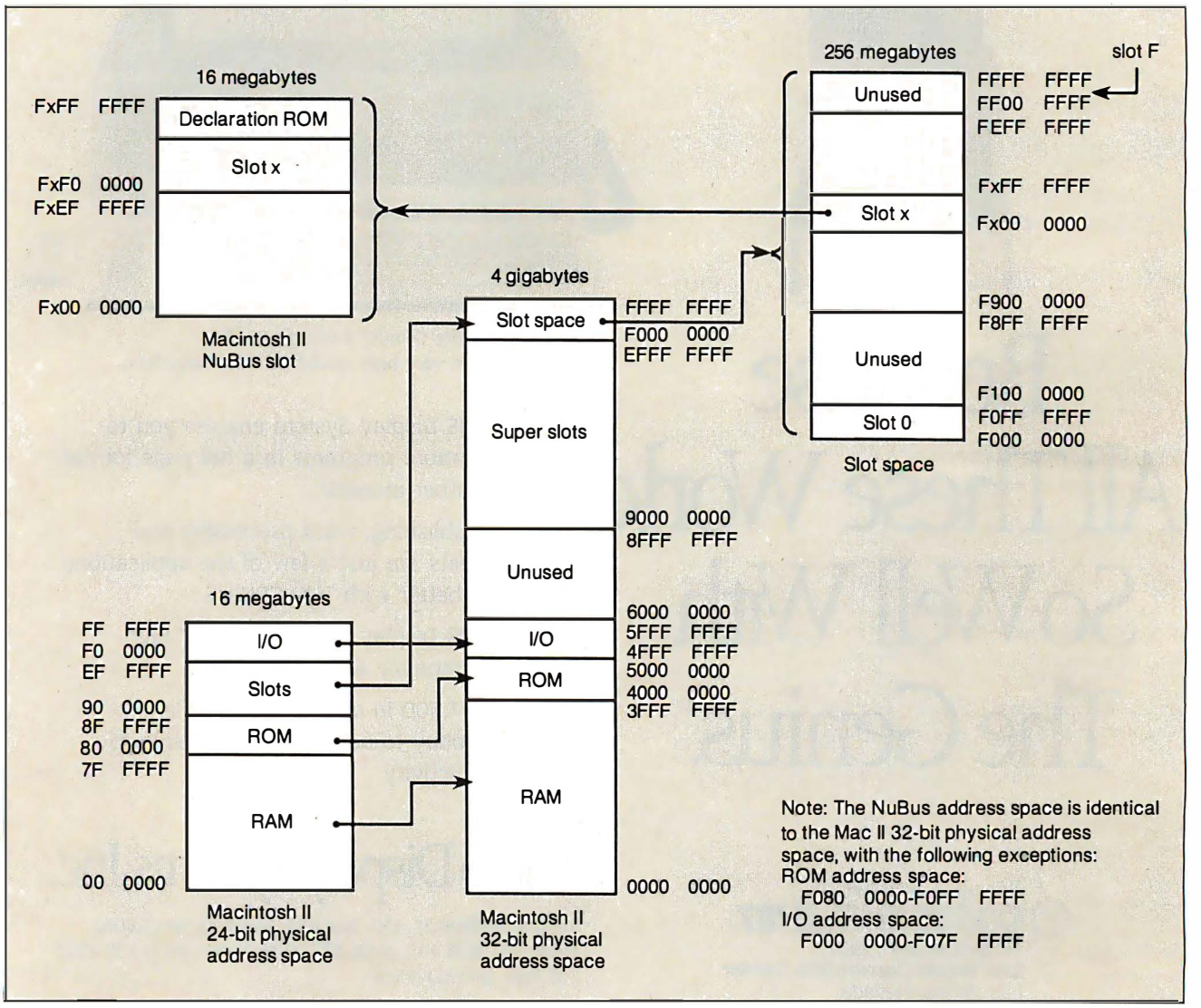


Figure 2: *The Macintosh II memory map. The memory in the superslot space and 15 megabytes of the slot space are not available to the Macintosh II, but NuBus cards can access these regions.*

for continued bus and resource ownership. Using an attention cycle (START* and ACK* both asserted), a master can request continuing bus ownership. It can also request a resource lock. A resource such as a memory card can be locked, denying access to any other master.

Both locks are extremely useful for multiprocessing; they allow a processor to do an uninterrupted test and set, as well as control access to a critical resource. For example, the Macintosh II motherboard uses bus locking to lock out the NuBus for critical local processing, including disk transfers and interrupt processing.

Card Configuration

Both the IBM Micro Channel and the Apple NuBus define high-level mechanisms to integrate cards or devices into the bus system. This eliminates the need for jumpers or switches to set either a card's interrupt level or its address space, which is the cause of a lot of bus problems on typical microcomputer systems.

The Micro Channel's Programmable Option Select (POS) eliminates switches from the system board and adapters by replacing them with programmable registers. Automatic configuration routines store the POS data into a battery-powered CMOS memory for system configuration and operations. The configuration utilities rely on adapter description files that contain the configuration data for a card. Configuration files define system operation, including system memory maps, video-processing options, and the individual adapter configurations.

At boot-up, the PS/2 Model 80 first validates the contents of the POS memory by examining a check character stored there. If the memory passes this test, the system then selects a card using the -CD SETUP lines. The card responds with its ID number. The system then loads the appropriate configuration data from CMOS memory into the card, as determined by the card's ID. This data sets the card's arbitration level and fairness, the address range of the card's I/O ROM, and the I/O address range. Cards that fail to configure properly are disabled by the system.

The Macintosh II relies on a slot manager to configure and maintain NuBus cards. Each card is required to have a special declaration ROM that holds the card-specific configuration information. Information in the declaration ROM includes byte lanes (which bytes of the NuBus data path are used), a test pattern, a revision level, a ROM cyclic redundancy check for validating the contents of the declaration ROM, and a resource directory. The resource directory points to various resource lists, such as the device icon, the device boot record, and the driver directory, which in turn points to blocks of code for the driver. The slot manager reads the declaration code at boot-up to configure the card into the system and installs any drivers or interrupt routines into system memory. The slot manager can also recognize a card as a bootable device and transfer control to the card when the system starts up. A card that fails to configure properly will be ignored, or a system error is posted.

Table 2: *The priority levels for the two buses and their device assignments. The priority levels are programmed into Micro Channel cards when they are configured into the system; NuBus priorities depend upon the slot the card is in.*

Arbitration level (Micro Channel)	Micro Channel		Apple NuBus		Arbitration level (NuBus)
	Value	Device assignment	Value	Device assignment	
Highest	-2	Memory refresh	--	--	Lowest
	-1	NMI	--	--	
	0	DMA channel 0	0	Motherboard	
	1	DMA channel 1	1	No slot	
	2	DMA channel 2	2	No slot	
	3	DMA channel 3	3	No slot	
	4	DMA channel 4	4	No slot	
	5	DMA channel 5	5	No slot	
	6	DMA channel 6	6	No slot	
	7	DMA channel 7	7	No slot	
	8	Reserved	8	No slot	
	9	Reserved	9	Slot 9	
	A	Reserved	A	Slot A	
	B	Reserved	B	Slot B	
	C	Reserved	C	Slot C	
	D	Reserved	D	Slot D	
	E	Reserved	E	Slot E	
Lowest	F	System CPU	F	Reserved	Highest

A Future with a Past

As you can see, both buses break new ground to optimize bus performance and minimize the user's effort to add a new card to the system. However, these buses must also deal with their past: providing compatibility with the existing market of software and hardware.

IBM faced the dilemma of maintaining compatibility with existing AT bus cards and limiting bus throughput to about 8 MHz, or redesigning the bus to optimize throughput at the expense of hardware compatibility. Looking toward a future of higher-speed processors and computing needs that require the handling of vast amounts of data, IBM chose to redesign the bus. However, the Micro Channel is, in a sense, still a CPU bus; throughput is optimized, since few bus clocks are lost synchronizing dissimilar components in the system. Its asynchronous nature allows future cards, operating at those higher speeds, to be installed with little to no change to the PS/2 system, while bus operations on NuBus are bound to its 10-MHz clock.

However, since the Micro Channel is a CPU bus, it's difficult to allow for multiple processors on the bus without interfering with the 80386's operation. NuBus, being a system bus, readily allows other processors to operate on it. Cards on NuBus can communicate and share data with one another without interfering with operations on the Mac II's local bus. In fact, AST Research offers a NuBus card that is essentially an IBM PC AT that runs independently in the Macintosh II but can share data with the 68020 CPU when necessary. Finally, the slot manager in the Mac II allows a NuBus card to be a boot device. You could drop a NuBus card with the next-generation CPU into a Mac II and let it take control of the machine—the ultimate in hardware expandability.

Both machines still have some of their past built into them. A look at the memory maps shows that both systems were designed to be compatible with their current operating systems, while providing a gateway to the next generation of software. The Macintosh II is the first machine in the Macintosh line to have slots, so Apple at least did not have to confront the problem of bus compatibility. But there's a certain irony in the fact that Apple must migrate from a 24-bit to a 32-bit operating system, similar to what IBM faces in the move to OS/2. ■