SpurBus Specification
SPUR, Symbolic Processing Using RISC, Project

Garth A Gibson
Computer Science Division
Electrical Engineering and Computer Sciences
University of California, Berkeley, CA 94720
January 1989

SpurBus Specification
SPUR, Symbolic Processing Using RISC, Project

Garth A Gibson

1. Introduction

This documents specifies the design of a shared system bus for a synchronous multiprocessor based on shared memory. The bus is called the SpurBus and it is part of the SPUR, Symbolic Processing Using RISCs, project [Hill86,Ouster88,Wood87].

SPUR is a multiprocessor workstation. Each processor is a RISC, Reduced Instruction Set Computer, with a tagged architecture for supporting the LISP programming environment [Taylor86] and an instruction buffer to reduce instruction traffic across chip borders [Hill87]. Each node in the multiprocessor contains a processor, a large cache (128 KB), a floating point coprocessor and a cache controller. The caches are kept consistent by a "snooping cache protocol" [Katz85a] implemented in the cache controller and across the bus.

The goal of SPUR is to provide a low cost, fast uniprocessor with additional processors available for research efforts into shared memory multiprocessing. The number of processors is small (6 to 12) so that a low cost interconnect, the system bus, will be able to supply the required memory bandwidth. Conceptually, a system bus is a simple, flexible, reliable, convenient point of serialization for synchronization and monitoring and, most important, its design can be borrowed from existing microcomputer system buses.

The Texas Instruments NuBus was selected for SPUR [TI83,IEEE86,IEEE88]. The NuBus is described in detail in this document. It is extended to provide for cache coherency and virtual memory mechanisms in SPUR. Because these extensions are transparent to the NuBus subset, the SPUR project can use commercial memory and I/O boards based on the NuBus protocol. The resulting design is called the SpurBus. A programmer's view of the SpurBus and memory system in general can be found in [Wood87].

This specification is organized so that the SpurBus is presented as modifications to the NuBus. Section 2 presents the NuBus in detail. The SpurBus' NuBus compliance is discussed in Section 3. Additional features of the SpurBus are introduced and detailed in Section 4. Finally the details of the mechanical backplane are presented in Section 5 and references are given in Section 6.

This document supercedes a draft specification [Katz85b] largely as a result of difficulties in the cache coherency implementation and in the electrical characteristics of a multiple master NuBus backplane. Modifications targeted at the latter are based on Draft 2.0 of the ANSI/IEEE NuBus specification [IEEE86]. The SpurBus was finalized before ANSI/IEEE completed its standard [IEEE88]; in Section 3, discrepancies are discussed.

2. NuBus Specifications

The description of the NuBus in this document was initially drawn from the Texas Instruments document, "NuMachine NuBus Specification," [TI83]. Revisions based on the ANSI/IEEE standardization draft "NuBus - a Simple 32-Bit Backplane Bus, P1196 Specification Draft 2.0," [IEEE88], have been made to deal with slow backplane termination pullups in a multiple master system.

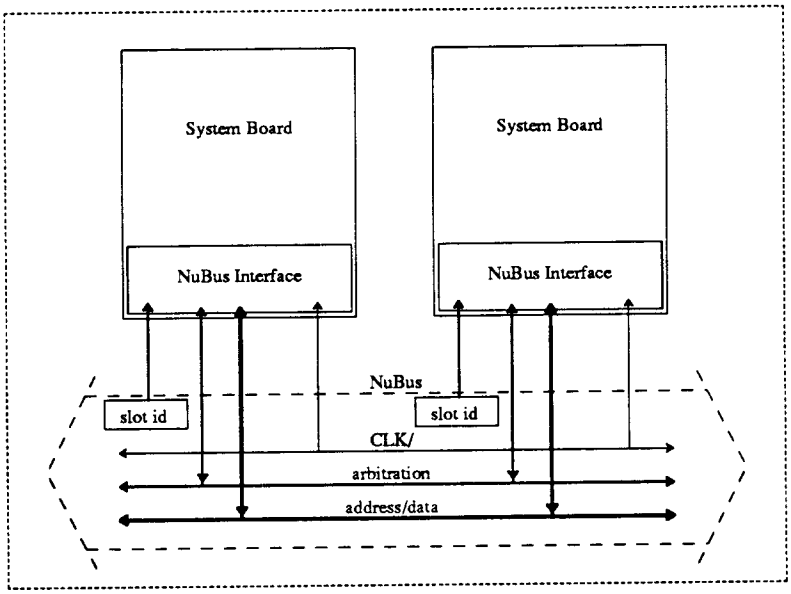

Figure 2.1: NuBus Block Diagram

The NuBus provides 32 multiplexed address/data lines, some control, arbitration, clocking and unique identifiers to each backplane slot.

After the SpurBus specification stabilized, ANSI/IEEE completed its standardization [IEEE88]. The NuBus subset of this SpurBus specification is not entirely compliant with the standard and is discussed in Section 3.

2.1. Overview

Figure 2.1 shows an overview of a NuBus configuration. Each node has a NuBus interface connected to the backplane. The NuBus provides each interface with a group of data transfer lines (address/data and control), a group of arbitration lines, a central clock and an unique identifier.

Some of the major characteristics of the NuBus are:

Synchronous
: All events are synchronized by a central clock; however, the number of cycles between events is usually unspecified. In this way events are asynchronous in bus cycle units.

Bandwidth
: The 10 MHz nominal cycle time coupled with block transfers gives a peak transfer rate of 37.5 Mbytes/sec, although available memory parts limit memory traffic to about 30 MBytes/sec.

Simple
: The only operations on the bus are reads and writes. Interrupts and IO transfers are memory mapped into a large physical address space (4 GBytes)

Small consumption of backplane lines
: Multiplexing address and data lines helps limit signal lines to a total of 49. With power and ground lines included, only one 96 pin backplane connector is required by the NuBus.

Distributed configuration
: Each of the 16 possible bus ports are provided with backplane identification. This eliminates board

"DIP" switches and allows distributed, parallel arbitration rather than daisy chain bus grants.

Multiprocessor support

The fair arbitration policy and the capability to lock the bus across distinct transfers provides basic and effective multiprocessor support.

2.2. Connector Pins Description

All NuBus signals use negative logic; that is, the active state has a low voltage on the line and the inactive state has a high voltage on the line. This is indicated in Table 2.1 by the suffix "/" on each signal name. This means that a high value is a logic zero and a low value is a logic one. This document will use the terminology "active" and "inactive" to indicate movement of data. An activated signal will usually be a signal driven low; however, the address/data lines are activated whenever they are driven high or low.

The NuBus defined backplane lines can be grouped as Clock, Control, Address/Data, Arbitration, Parity, Utility, Reserved, and Power and Ground. Table 2.1 shows these groups, their member signals, the connector pins they require and the electrical protocol of each. The Clock is described in the next section, Utility is describe in its own section, Control, Address/Data and Parity are described in the section on data transfer, and Arbitration is described in its own section. The electrical protocols are input-only, tri-

GROUP	SIGNAL	# OF PINS	PROTOCOL
Clock	CLK/	1	input-only
Control	START/	1	tri-state
	ACK/	1	tri-state
	TM0/	1	tri-state
	TM1/	1	tri-state
Address/Data	AD<31..0>/	32	tri-state
Parity	SP/	1	tri-state
	SPV/	1	tri-state
Arbitration	ARB<3..0>/	4	open-collector
	RQST/	1	open-collector
Utility	RESET/	1	open-collector
	ID<3..0>/	4	input-only
	Total Signals	49	
Reserved	RSVD/	2	open-collector
Power/Ground	+5	11	
	-5.2	8	
	+12	2	
	-12	2	
	GND/	22	
	Total Lines	96	

Table 2.1: NuBus Backplane Lines

The NuBus defines the use of one 96 pin backplane connector for 49 signal lines plus reserved, power and ground. The protocol of a signal refers to its driving characteristics. The two RSVD/ signals are defined in the ANSI/IEEE NuBus [IEEE88] as NMRQ/ and PFW/.

state (drive on output, receive on input and high impedance otherwise) and open-collector (value is the logic AND of all ports driving the signal as output).

2.3. System Timing

The NuBus clock, shown in Figure 2.2, has a nominal cycle time of 100 nsec. It has a 75% duty cycle; that is, 75 nsec inactive and 25 nsec active phases. The low to high transition (clock assertion edge) triggers changes in bus signals. The high to low transition (clock deassertion edge) triggers bus signal sampling. The duty cycle of the central clock is asymmetric to provide increased time for propagation and setup time while preserving enough time between signal sample and change to avoid skew problems. Figure 2.2 also shows the setup, hold, propagation, assertion and release times relative to the clock phases.

2.4. Physical Memory Map

The NuBus defines a large physical address space (4 GBytes). Each bus port, called a slot, has some of this memory mapped to it. In total the top 16th (256 Mbytes) of the physical address space, called slot space, memory maps the 16 possible bus ports. Figure 2.3 shows this organization of physical memory as it applies to a node in slot 13.

The node at each bus port interprets reads and writes to its slot without constraint[1]. Processor nodes

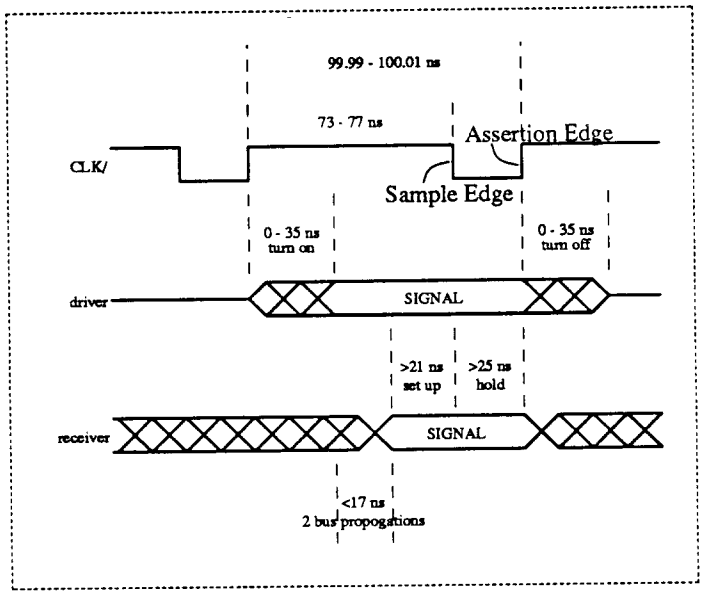

Figure 2.2: Signal Timing

The NuBus has a synchronous design. The central clock, CLK/, has a 75% duty cycle separating its sample and assertion edges. The selection of the 75% duty cycle provides for greater relative emphasis on propagation and setup times while preserving freedom from bus skew problems.

[1] This does not comply with ANSI/IEEE NuBus standard [IEEE88] in that a word read of the highest address in the slot space must return either successful or error status and a configuration ROM immediately below the highest addressable word is required to be present.

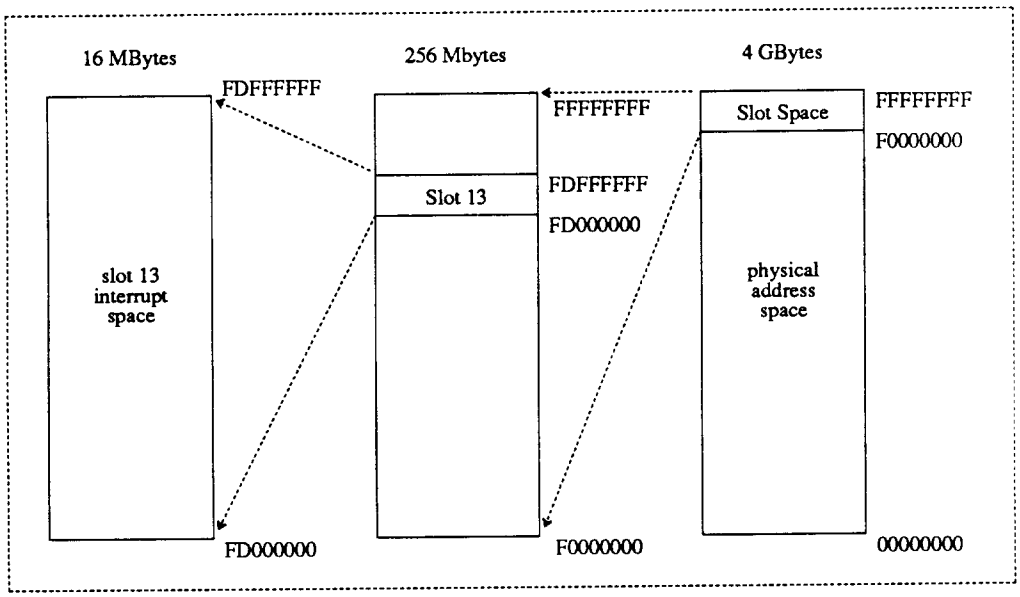

Figure 2.3: Physical Memory Layout

NuBus physical memory maps the top 256 MB to its 16 slots, 16 MB each. This slot space is used to memory map interrupt and I/O transfers.

desiring interrupt capabilities associate interrupt registers with slot space addresses. They may also provide debug or status information through slot space addresses. Memory nodes may provide their RAM through slot space and IO nodes can provide control, status and data ports through slot space. All nodes should provide identification (type, revision, name, etc.) and configuration control registers through slot space. The remaining physical memory (non-slot space) is not constrained by the NuBus specification. Typically, memory nodes will allow their RAM addresses to be remapped to arbitrary locations.

2.5. Utility Signals

RESET/

This open-collector signal is used to synchronize the state of the system. If it is activated for one cycle all NuBus interfaces are initialized. This is referred to as a Bus Reset[2]. If it is activated for more than one clock period, then it should be activated for at least a millisecond (10000 cycles) and the entire system is initialized to the power-up state. This is called a System Reset.

ID<3..0>/

These four bits name, in binary, the backplane slot that the reading node occupies. In addition to uniquely naming each slot these lines are used for NuBus arbitration (see Section 2.7). These lines are also used to determine whether a physical address is mapped to the local slot (addresses F(ID)XXXXXX, X any hex digit, are mapped to the local slot).

[2] In the ANSI/IEEE NuBus standard [IEEE88], Bus Reset has disappeared. This is a major non-compliance because SPUR's SpurBus implementation uses Bus Reset instead of System Reset to deal with many error conditions.

2.6. Data Transfer

The NuBus defines read and write data transfers. These may move bytes, halfwords, words, doublewords, quadwords, octwords and doubleoctwords. All units must be aligned in physical memory according to their size (more precisely, starting address modulo size is zero). For small units, byte and halfword units, data is returned as if the word enclosing them was read. It is up to the processor receiving the data unit to justify the data as it wishes. For these sizes and the word size, one transfer across the backplane is all that is needed. These are referred to as word reads and word writes. For the larger sizes, multiple transfers across the backplane are needed. These are referred to as block transfers. This section describes word and block transfers in detail.

Before a NuBus interface may begin a data transfer it must become the bus owner. This is the process of arbitration. Once an owner, the interface must still wait for the possible current transfer to complete. When it has obtained ownership and seen the end of the last owner's transfer then the interface may initiate a new transfer.

To avoid system deadlock when backplane errors occur, the system is required to provide a "watchdog timer." This timer is started at the beginning of each transaction, reset each time a word is successfully transferred and cancelled at the end of a transaction. If it reaches 256 cycles, then an error is very likely and the watchdog generates a transfer termination (ACK/) in error.

2.6.1. Control Signals

The signals START/, ACK/, TM0/, TM0/, AD<5..0>/ and SPV/ are data transfer control signals.

START
: The START signal originates data transfers and triggers arbitration. When active, it has a one cycle duration. This cycle is referred to as a START cycle. During a START cycle all possibly addressed nodes, slaves to the current bus master, examine the remaining control lines and the address lines to determine whether they are effected. Each START cycle is paired with exactly one ACK cycle.

ACK
: The ACK signal terminates data transfers and triggers the current arbitration winner to take over bus ownership. It also has a one cycle duration referred to as an ACK cycle. During an ACK cycle the last word of data transferred is valid and the remaining control lines contain transfer status. There is exactly one ACK cycle for each START cycle.

TM0 and TM1
: These are multi-purpose control lines. During a START cycle TM1 carries the transfer type (inactive=read, active=write) and TM0 determines whether the transfer unit size is a byte (active=byte transfer) as shown in Table 2.2. During an ACK cycle they carry transfer status information as shown in Table 2.4 in Section 2.6.4. Between a START and ACK cycle of a block transfer TM0 is used to strobe word transfers (activated means data lines valid for non-final word transfer).

AD<1> and AD<0>
: During a START cycle these are used for control instead of addressing. Conveniently, if the transfer unit is a byte, these can be interpreted as the byte offset. If the transfer unit is not a byte they describe units of word, low halfword, high halfword and block transfer as shown in Table 2.2. When block transfer is specified, AD<5..2> may also carry control information.

AD<5..2>
: During a START cycle of a block transfer at least some of the low order address bits carry no information because blocks are aligned. For example, the smallest block is a doubleword and is doubleword aligned. So its low order address bits (AD<2..0>) are 000. NuBus block transfer encoding, shown in Table 2.3, specifies that if AD<2> is inactive then the transfer unit is a doubleword and AD<5..3> are valid address bits. However, if AD<2> is active, then the transfer unit must be bigger

TM1/	TM0/	AD<1>/	AD<0>/	Transfer
L	L	L	L	Write Byte 3
L	L	L	H	Write Byte 2
L	L	H	L	Write Byte 1
L	L	H	H	Write Byte 0
L	H	L	L	Write Halfword 1
L	H	L	H	Write Block
L	H	H	L	Write Halfword 0
L	H	H	H	Write Word
H	L	L	L	Read Byte 3
H	L	L	H	Read Byte 2
H	L	H	L	Read Byte 1
H	L	H	H	Read Byte 0
H	H	L	L	Read Halfword 1
H	H	L	H	Read Block
H	H	H	L	Read Halfword 0
H	H	H	H	Read Word

Table 2.2: Transfer Mode Encoding

The type and size of a transfer unit are encoded in the Transfer Mode (TM) lines and the low order address lines during a START cycle.

AD<5>/	AD<4>/	AD<3>/	AD<2>/	# of Words	Address
X	X	X	H	2	AD<31..3> 000
X	X	H	L	4	AD<31..4> 0000
X	H	L	L	8	AD<31..5> 00000
H	L	L	L	16	AD<31..6> 000000

Table 2.3: Block Size Encoding

Transfer units must be aligned according to their size in memory. This forces some of the low order bits of an address to zeroes. The encoding of block sizes larger than one word takes advantage of these address bits with known values.

---

than a doubleword and AD<3> should be examined. This algorithm is applied to each of AD<3> and AD<4> in turn. If AD<4> is active, then AD<5> is examined, but it must be inactive because the largest transfer unit defined is 16 words long.

SPV

The signal SP carries an even word parity bit on any cycle that SPV is active. The AD lines are the only lines covered by this parity mechanism.

In draft 2.0 of the ANSI/IEEE specification [IEEE86], it was made clear that the termination pullups on tri-state lines can not be relied on to pull a released low signal to a high state in time for the sample edge immediately after release (they can be relied on for the sample edge after the immediately following sample edge and they can be relied on to hold a released high signal in the high state). This introduces a problem of falsely active tri-state control signals (this does not happen with open-collector signals). To avoid this problem the current bus owner (most recent winner of an arbitration contest) is required to drive START/ and ACK/ (usually to the inactive state) on the cycle after they have been activated. During START cycles, a bus master (owner) must also drive all other control lines. During ACK cycles, the bus slave must drive all control lines except for START/. Between START and ACK cycles on block transfers, a bus slave that cannot transfer every cycle must be sure to drive TM0/ inactive after each word

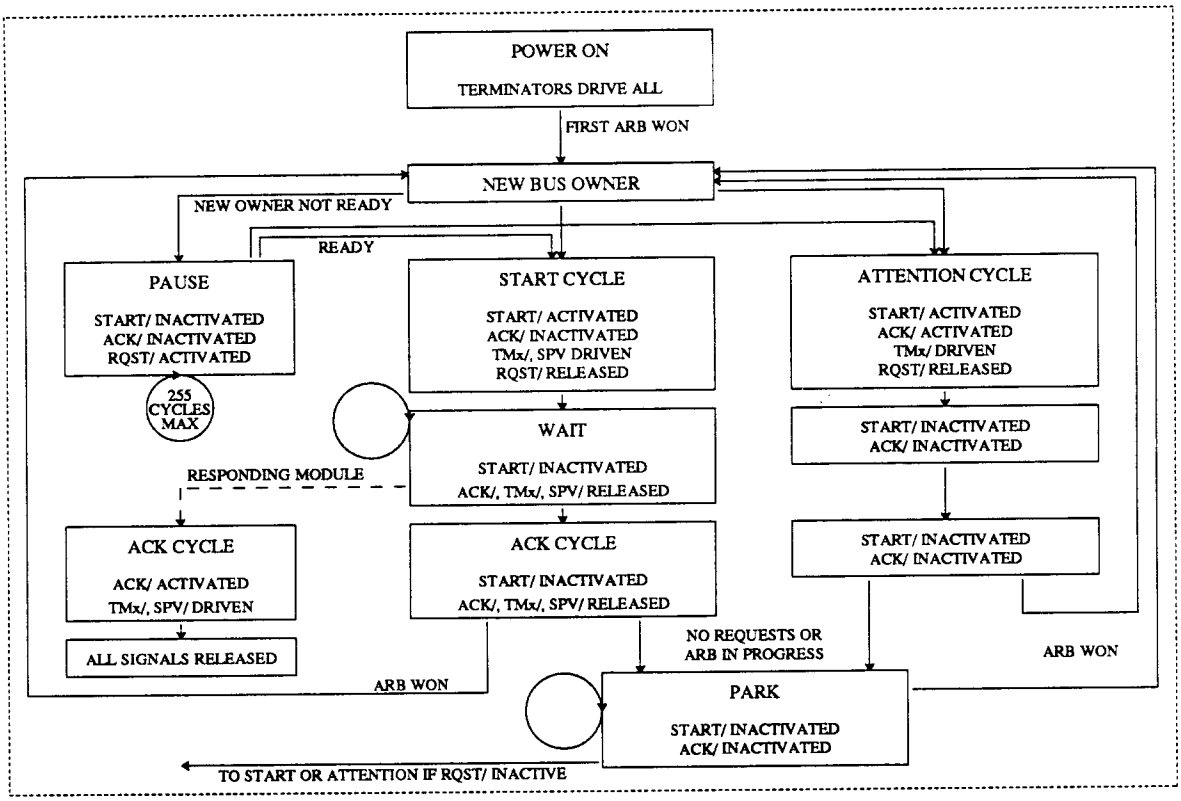

Figure 2.4: Signal Determinacy

Bus ownership implies responsibility to make sure that tri-state control signals have determinate values on every cycle. This figure shows the signal driving responsibilities of the bus owner.

is transferred. See Figure 2.4 for a flowchart of the bus owner's signal driving responsibilities.

2.6.2. Single Transfer Operations

A single transfer operation is composed of a START cycle specifying a byte, halfword or word transfer unit and its address followed by an ACK cycle specifying data and transfer status. Many cycles may pass between these two events, as long as the watchdog timeout does not occur. During the START cycle the transfer size is specified as shown in Table 2.2. If the operation type is read, TM1 is inactive; if it is write, TM1 is active. For byte transfers TM0 is active and AD<1..0> used for addressing. For halfword transfers TM0 is inactive, AD<0> active and AD<1> used for addressing. For word transfers TM0 is inactive and AD<1..0> used for addressing (that is, both inactive).

2.6.2.1. Byte, Halfword and Word Read Protocol

In Figure 2.5, the protocol for single transfer read operations is shown.

R(1) The bus master drives the address and control lines (ACK, TMx, SPV and START) to signal a START cycle. The TMx and AD<1..0> lines indicate the transfer unit size as shown in Table 2.2.

F(1) All potentially addressed slaves sample the control and address lines to determine if they are affected.

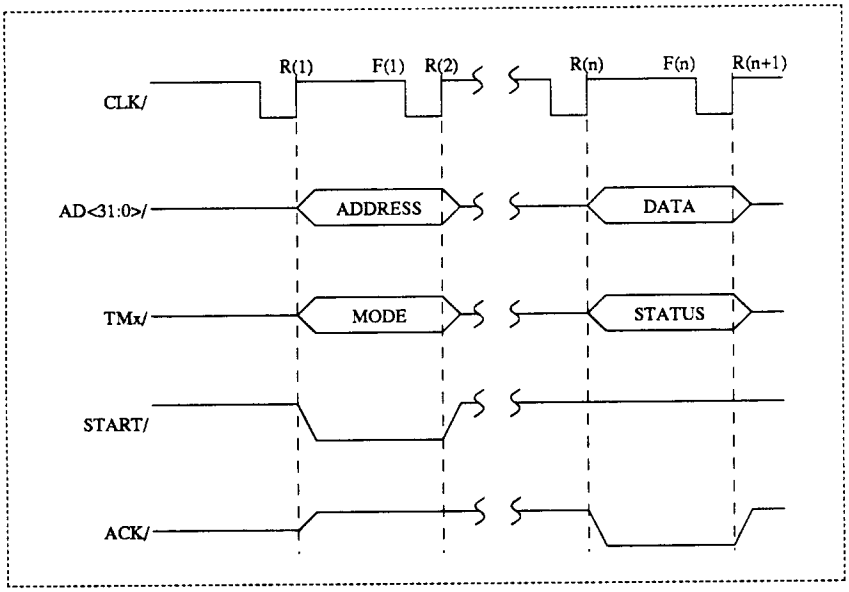

Figure 2.5: Word Read Protocol

Single transfer reads are composed of an address/mode cycle and a data/status cycle. The duration of a single transfer read can be many cycles as long as it is less than the system timeout period.

R(2) The master releases the address and control lines, drives START inactive, and waits for the ACK cycle.

R(n) The selected slave drives the data and SPV lines, drives TMx with status, and activates ACK to signal an ACK cycle.

F(n) The master samples the data, status and ACK lines on all falling edges until it sees the ACK cycle.

R(n+1)
The selected slave releases the data and control lines for the next master to use. The next master drives ACK determinate (inactive unless it is initiating an "attention" cycle, described in Section 2.6.5) and may initiate a START cycle.

(1) The parameter "n" will be less than the system timeout because a watchdog on the NuBus interface will generate an ACK cycle with a "timeout" status whenever the transfer takes too long.

2.6.2.2. Byte, Halfword and Word Write Protocol

The single transfer write operation protocol is shown is Figure 2.6.

R(1) The bus master drives the address and control lines (ACK, TMx, SPV and START) to signal a START cycle. The TMx and AD<1..0> lines indicate the transfer unit size as indicated in Table 2.2.

F(1) All potentially addressed slaves sample the control and address lines to determine if they are affected.

R(2) The master releases the TMx and ACK lines, drives the data and SPV lines, deactivates START, and waits for the ACK cycle.

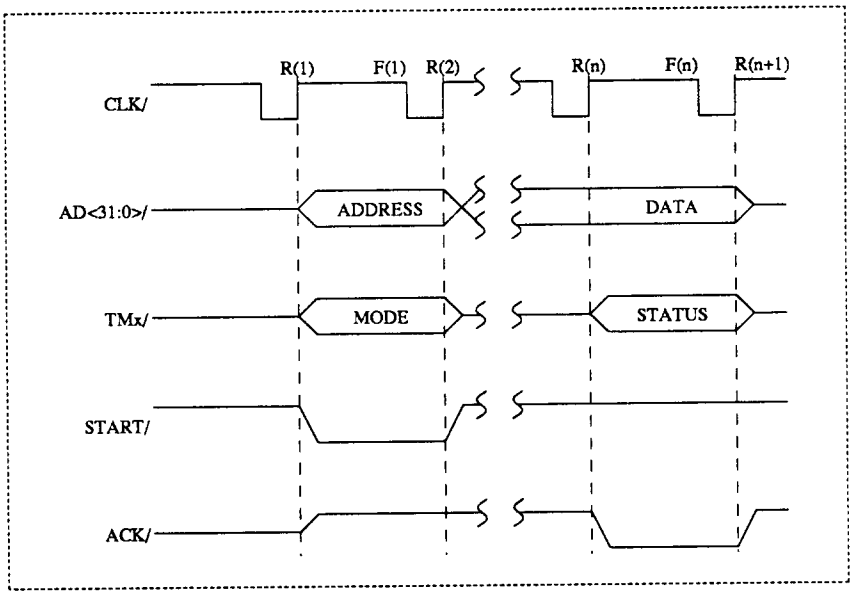

Figure 2.6: Word Write Protocol

A single transfer write operation appears very similar to a single transfer read. The important distinction is that the data must be valid on the first cycle after the address has been issued.

R(n) The selected slave drives the control lines (TMx and ACK) to signal an ACK cycle. The TMx lines indicate the transfer completion status as shown in Table 2.4.

F(n) The master samples the status and ACK lines on every cycle until is sees the ACK cycle. The slave may sample the data lines on or before the ACK cycle (F(2) through F(n)).

R(n+1)
The selected slave releases control lines and the master releases the data lines for the current owner to use. The current owner (which may differ from the master that initiated this transfer because of an overlapping arbitration) must ensure that ACK is determinate and may initiate the next START cycle.

(1) The parameter "n" will be less than the system timeout because a watchdog on the NuBus interface will generate an ACK cycle with a "timeout" status whenever the transfer takes too long.

2.6.3. Multi-Transfer Operations

A multi-transfer operation, commonly called a block transfer, is composed of a START cycle specifying a transfer unit size and its address followed by enough intermediate data strobes to transfer all but the last word of the transfer unit and finally an ACK cycle specifying the final word of data and the transfer status[3]. The data in a block transfer is contiguous in physical memory and aligned according to the unit size. Many cycles may pass during a block transfer, as long as the watchdog timeout does not occur.

---

[3] The ANSI/IEEE NuBus standard [IEEE88] defines an immediate ACK cycle (no intermediate ACK cycles) with successful status as the correct reaction to a block transfer by a slave that does not support block transfers. SpurBus nodes will not exhibit this behavior. In particular, SPUR's implementation does not distinguish this type of event from other forms of transaction error (parity, too few or too many transfers).

During the START cycle the operation type and transfer size are specified. If the operation type is read, TM1 is inactive; if it is write, TM1 is active. Because it is a block transfer TM0, AD<1> and AD<0> are, respectively, inactive, active and inactive. The encoding of TM0 as inactive in the START cycle is convenient because it forestalls a need for the slave to drive TM0 inactive immediately after the START cycle. As shown in Table 2.3, the transfer size is specified by one or more of the AD<5..2> lines: if AD<2> is inactive then transfer 2 words, else if AD<3> is inactive transfer 4 words, else if AD<4> is inactive transfer 8 words, else demand AD<5> to be inactive and transfer 16 words.

2.6.3.1. Block Transfer Read Protocol

The protocol of a block transfer read operation is shown in Figure 2.7.

R(1) The master drives an address with AD<5..0> encoding transfer size as shown in Table 2.3. It also drives SP and SPV as appropriate, TMx according to Table 2.2, START active and ACK inactive.

F(1) All potentially addressed slaves sample the address and control lines to determine whether they are affected.

R(2) The master releases the address, TMx and SPV lines, inactivates START, and begins waiting on TM0 and ACK.

R(n) The slave drives the data, SPV and TM0 lines. TM0 active and ACK inactive indicate that an intermediate word, not the last word, is valid on the data lines.

F(n) The master is continually sampling the data, TMx and ACK lines looking for TM0 active and ACK inactive. This is called an intermediate ACK cycle. The data lines contain a intermediate word on

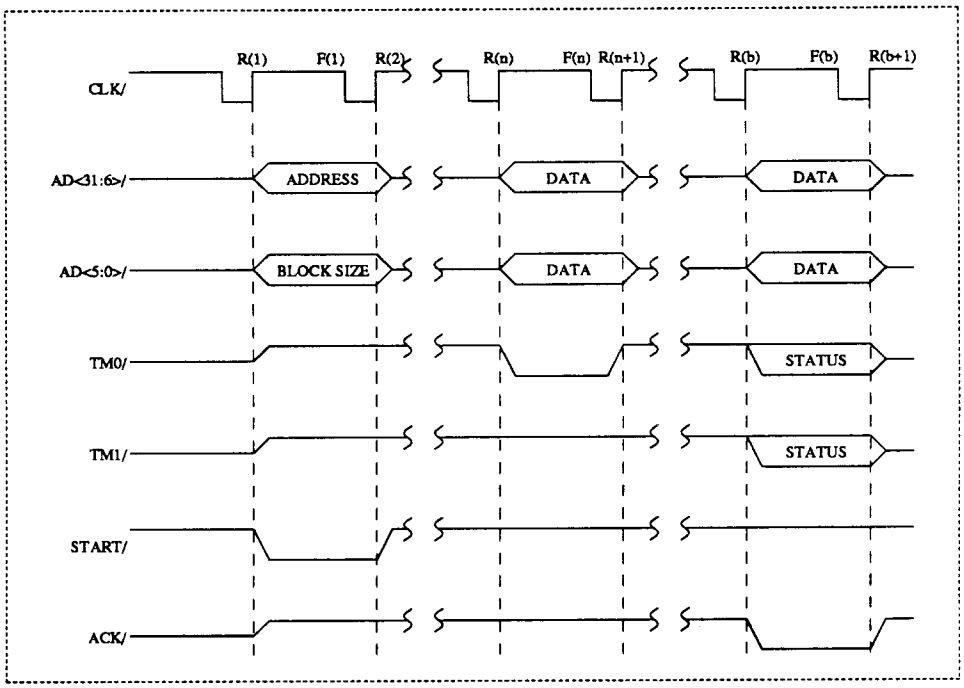

Figure 2.7: Block Read Protocol

Block transfers use the TM0 line to strobe intermediate words across the backplane. The slave controls the data and mode lines after the START cycle during a read.

the transfer unit.

R(n+1)
If the slave cannot provide the next word on the immediately following cycle it drives TM0 inactive.

NOTE:
These intermediate stages (n) are repeated B-1 times where the size in words of the transfer unit is B.

R(b) The slave drives the data, SPV, TMx (with a status code) and ACK lines to form the ACK cycle.

F(b) The master continually samples the data, TMx and ACK lines looking for intermediate or final ACK cycles. On the ACK cycle, the data lines contain the last word of the transfer unit and the TMx lines contain the transfer status. If this cycle occurs before the master believes that the last word is due then the master must detect this and realize that its transfer is over (in error).

R(b+1)
The slave releases the data and control lines for the next owner to use. The next owner must drive ACK to a determinate state, possibly beginning the next START.

(1) The parameter "b" will be less than the system timeout because a watchdog on the NuBus interface will generate an ACK cycle with a "timeout" status whenever the transfer takes too long.

2.6.3.2. Block Transfer Write Protocol

The protocol for a block transfer write operation is shown in Figure 2.8.

R(1) The master drives an address with AD<5..0> encoding transfer size as shown in Table 2.3. It also drives SP and SPV as appropriate, TMx according to Table 2.2, START active and ACK inactive.

F(1) All potentially addressed slaves sample the address and control lines to determine whether they are affected.

R(2) The master releases the TMx and SPV lines, inactivates START, drives the data lines with the first word of the transfer unit and begins waiting on TM0 and ACK.

R(n) The slave drives the TM0 line indicating that it has sampled the data lines on a previous sample edge or will sample the data lines on the next sample edge.

F(n) The master is continually sampling the TMx and ACK lines looking for an intermediate ACK cycle.

R(n+1)
The master must change the value on the data lines to the next word of the transfer unit. If the slave cannot accept data on consecutive cycles it drives TM0 inactive.

NOTE:
These intermediate stages (n) are repeated B-1 times where the size in words of the transfer unit is B.

R(b) When the last word has been sampled or will be sampled on the next sample edge the slave drives TMx with a status code and ACK to the active state.

F(b) The master continually samples TMx and ACK lines looking for intermediate or final ACK cycles. On the ACK cycle, the TMx lines contain the transfer status. If this cycle occurs before the master believes that the last word is due then the master must detect this and realize that its transfer is over (in error).

R(b+1)
The slave releases the control lines and the master releases the data lines for the current owner to use. The current owner must drive ACK to a determinate state and possibly initiate the next START cycle.

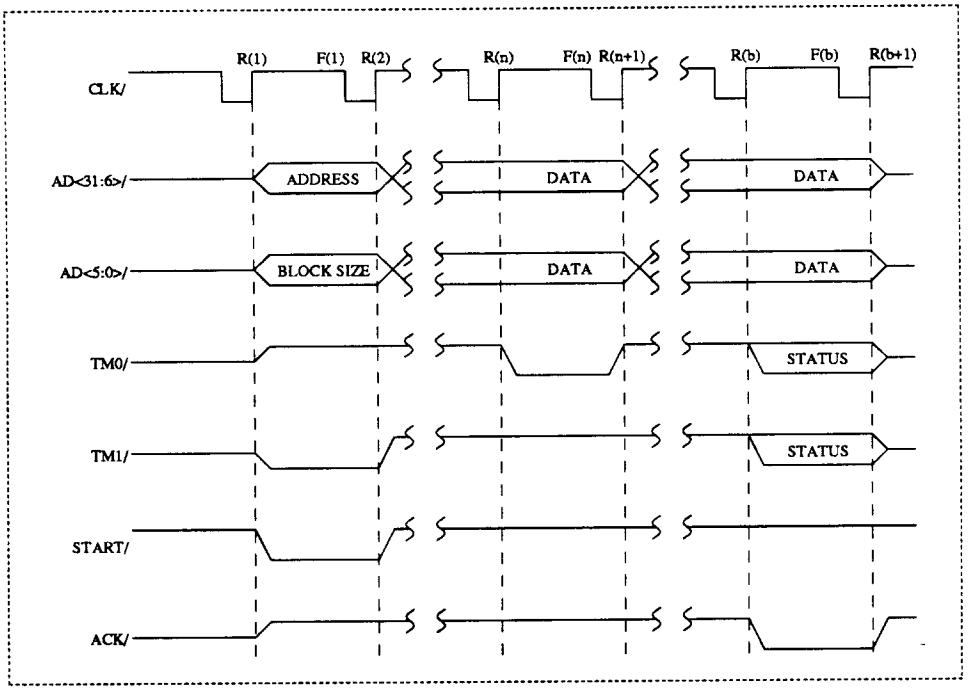

Figure 2.8: Block Write Protocol

A block write transfer is very similar to a block read transfer. The important difference, like the single word transfer case, is that the master retains control of the data lines and must be able to provide the next data word on the cycle after an intermediate acknowledge or address transmission.

---

(1) The parameter "b" will be less than the system timeout because a watchdog on the NuBus interface will generate an ACK cycle with a "timeout" status whenever the transfer takes too long.

2.6.4. Transfer Completion Status Codes

TM1/	TM0/	Status Message
L	L	Successful Completion
L	H	Unspecified Error
H	L	Bus Timeout Error
H	H	Try Again Later

Table 2.4: NuBus Status Codes

During the ACK cycle of any transfer the mode lines pass an overall status code. The "try again later" code is intended to be used to resolve internal slave resource conflicts (for example, deadlock in a bus-to-bus gateway).

The status of each transfer according to the slave is reported to the master using the mode lines (TMx) during the ACK cycle. Table 2.4 shows the encoding and status types.

Successful Completion
: This indicates that the slave believes the transfer is successfully complete.

Bus Timeout Error
: The NuBus specifies that a system timer (probably located with the system clock) will limit the duration between START and ACK cycles. The timer expires after 256 cycles and the watchdog logic will generate an ACK cycle with the "bus timeout error" status. The typical reason for a bus timeout is that the address transmitted during a START cycle is unimplemented by all available slaves. The timeout value of 256 cycles should be large enough that any valid operation will complete (ie., substantially longer than a reasonable transfer duration).

Try Again Later
: If the addressed slave is unable to respond to the transfer request for a transient reason, it can generate a "try again later" ACK cycle. The master's request is not in error and should be retried later.

Unspecified Error
: This is the catch-all error status. Typical reasons for an "unspecified error" are bus parity error in transit or memory parity/ECC error.

2.6.5. Attention Cycles

An attention cycle is any cycle during which START and ACK are both active. When this occurs no information is passed on the data lines and the TMx lines should both be active[4]. This type of cycle is intended to reinitiate bus arbitration. The primary situation where this is required occurs when the bus is requested and obtained, but the owner does not require a transfer.

2.6.6. Interrupts

Interrupts are not given special attention in a SpurBus system[5]. They are implemented as memory mapped single transfer write operations with data 1 or -1. The destination of an interrupt write transfer must be an address that a processor node will recognize as interrupt space. Typically part of a node's slot space will be used for this. Software can construct interrupt priorities and levels according the address of the interrupt write transfer.

2.6.7. Parity

Parity can be used to protect the integrity of the address/data lines during any transfer. A pair of lines are defined to implement the optional use of even word parity. One line, SP/, is inactive if an even number of the lines AD<31> to AD<0> are active; otherwise, it is active. The other line, SPV/, is active if SP/ is valid. Parity is optional on a cycle by cycle basis. Either master or slave may choose to ignore SP/, but must drive SPV/ inactive when they are driving the AD<31..0> lines.

The following policies should be used when parity errors are detected:

During a START cycle
: All slaves should ignore the transfer. The system timer will reinitiate arbitration if the master

---

[4] The ANSI/IEEE NuBus standard [IEEE88] defines three types of attention cycles according to the TMx lines. When both TMx lines are inactive, that specification declares the addressed slave to be internally locked; other internal users of the addressed resource are blocked until a following attention cycle of the type the SpurBus defines. The SpurBus attention cycle is used by the ANSI/IEEE NuBus as the SpurBus intends it and to clear resource locks. The other two combinations of TMx are reserved.

[5] In the ANSI/IEEE NuBus standard [IEEE88] there is a special low-cost, simple interrupt mechanism based on the NMRQ/ line, a RSVD/ line in SpurBus.

doesn't.

During transfers of a read operation
: The master should ignore the data it receives and treat the transfer as in error once the ACK cycle arrives.

During transfers of a write operation
: The slave should generate an ACK cycle with an unspecified error status code.

2.7. NuBus Arbitration

NuBus arbitration is distributed across nodes that act as bus masters. It uses five *"open-collector"* backplane lines[6] (RQST/ and ARB<3..0>/). Allocation of backplane bandwidth through NuBus arbitration prevents the "starvation" of any particular requester. In this sense this scheme is "fair". In the case of light loading, each requester will receive the same fraction of bandwidth (in the case of heavy load, the geographical contest resolution will bias bandwidth to nodes in higher numbered slots).

The NuBus arbitration mechanism resolves all contention between bus masters. When a bus master requires the bus it begins by declaring it requirement for bus ownership. All masters that declare at the same time form a "wave". One master at a time will be selected by an arbitration contest and gain bus ownership. Within a contest the geographical position of backplane nodes is used to select a winner. The winner uses, then releases the bus. The winner of the next contest within the original wave then gets the bus. This continues until all masters in the original wave have been given the bus and the last has released the bus. At this point a new wave is allowed to form. Because a new wave cannot begin until all members of the previous have been served, it is guaranteed that a high priority master cannot preempt a lower priority master.

2.7.1. Arbitration Specification

A bus master enters a wave by activating RQST/ (pulling it low). This is only allowed when RQST/ was not active on the immediately preceding sample sedge; that is, a wave is formed by those masters that activate RQST/ on the same clock cycle.

An example schematic for the arbitration logic on each bus master node is shown in Figure 2.9. In this diagram the line arb/ and GRANT are internal to a bus master. The arb/ line indicates the master's desire to use the bus and the GRANT line indicates that bus ownership has been attained. This logic corresponds to the following equations (where ARBx is the inverted value of ARBx/, ARBxo is the driven value and ARBxi is the read value):

ARB3o = arb * ID3
ARB2o = arb * ID2 * (ID3 + ARB3/i)
ARB1o = arb * ID1 * (ID3 + ARB3/i) * (ID2 + ARB2/i)
ARB0o = arb * ID0 * (ID3 + ARB3/i) * (ID2 + ARB2/i) * (ID1 + ARB1/i)

When a master enters arbitration by activating RQST/ it also activates the arbitration logic on its node. The slot ID of the node is driven onto the ARB<3..0>/ lines and the contesters determine the highest priority slot ID in parallel (slot zero has the lowest priority). After two clock cycles the arbitration logic on all nodes must have settled and the winner is the node whose slot ID is still on the ARBx lines.

The winner of arbitration will be the next master to initiate a transaction, but it may not be able to this immediately. Arbitration may be overlapped with the last winner's last (or only) transfer so the new winner may have to wait for this transfer to complete. At this point the new winner may begin a transfer. Once it has begun a transfer, it can release ownership of the bus; that is, re-initiate arbitration among the masters that lost to it.

---

[6] An open-collector line implements a "wired-AND" logic function. The line will take a low value if any master drives it low and it will take a high value if all masters drive it high.

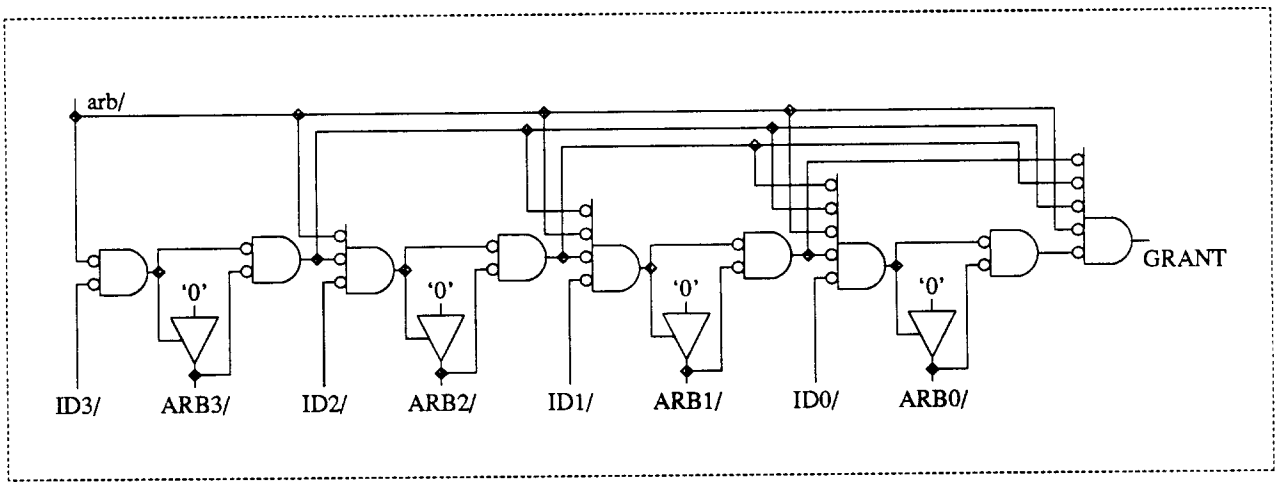

Figure 2.9: Possible Arbitration Schematic

An implementation of the distributed arbitration logic is shown here. The IDx lines are backplane supplied and unique to each bus port. They contain the binary coded slot number of that bus port. The open-collector ARBx lines are used to selected the slot number of the next bus master from those that have activated arb/. The winner will see GRANT active. Circuit design must guarantee that GRANT has settled within two clock cycles (200 nsec) of the activation of arb/.

After a winner has been selected in arbitration, the winner and losers alike continue to activate RQST/ to prevent new requesters from entering the wave. The losers will examine the ARBx lines again two cycles after the next START cycle they see on the bus. If the current winner stops driving its slot ID onto the ARBx lines by the end of the START cycle then a new winner will be selected. If the current winner is going to stop driving its slot ID at the end of a START cycle it must release the RQST/ line at the beginning of that START cycle. This ensures that nodes waiting to start a new contest may enter arbitration during the transfer of the last winner in the current contest.

2.7.2. Bus Locking

An important feature of arbitration mechanisms is the provision of locked multi-transfer transactions. A good example of this is the operation "test-and-set". In this operation a memory location is read and a integer value of 1 is written back into it without any intervening operation by any other processor. This is used to construct a variety of multiprocessor synchronization mechanisms. Notice that if an addressable slave contains an internal alternate requester for its memory (as in the case of a processor with local memory), bus locking may not lock out the internal requester[7].

The NuBus arbitration mechanism specifies that a new winner within a wave will be selected two cycles after each START cycle during that arbitration sequence. Usually the current winner will withdraw from arbitration on the START cycle it generates after winning. This corresponds to a single transfer per transaction (period of bus ownership). However, the current winner might not withdraw from the arbitration

[7] So the ANSI/IEEE standard [IEEE88] defines resource lock and unlock events that should begin and end bus locked sequences so that the addressed slave knows to lock out its internal requesters. The SpurBus does not define resource lock and unlock cycles; SPUR's implementation uses its cache coherency mechanism for 'test-and-set" so it does not need bus locking [Wood87].

on its START cycle. Because it is the current winner and because no new contender is allowed into the arbitration until all have withdrawn, the continued presence of the current winner ensures that it will win again. In this way the current master can complete many consecutive transfers. In general it is not good policy for a processor to hold the bus for too long because it increases the overall average time every processor is blocked waiting for the bus.

2.7.3. Bus Parking

When there are a small number of bus masters in the system or when requests are generally rare, but clustered, it is often the case that the last user of the bus will be the sole member of the next wave. When this is the case it could be held up two cycles arbitrating without competition. This would add unnecessary latency to the transfer duration.

In the NuBus specification "bus parking" is defined to allow this problem to be reduced. A bus master is parked on the bus if after withdrawing from arbitration there is no other contender (RQST/ becomes inactive). This parked master remains parked until a new wave is formed. While it is parked it may begin a transfer without arbitrating (note that it must still look at RQST/ to determine if it is still parked on the bus). If a wave begins after the parked master has begun a transfer the new winner will wait for the current transfer to end. The parked master will realize that it is no longer parked when it next wants to use the bus. Figure 2.4 in Section 2.6.1 shows the transition of ownership in conjunction with the responsibility to ensure determinacy of START and ACK.

2.7.4. Arbitration Timing

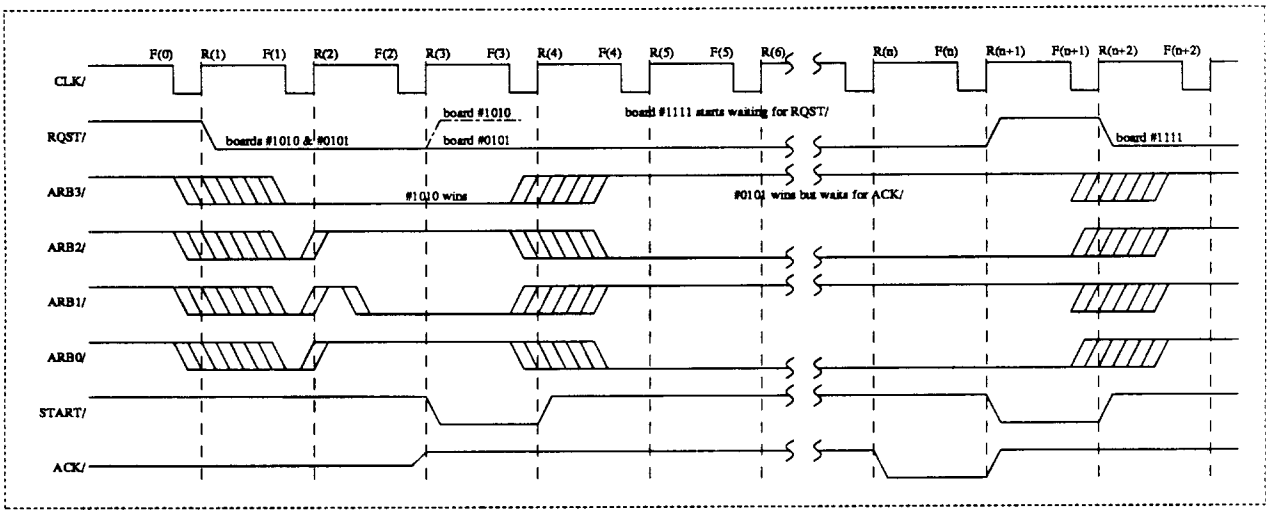

Figure 2.10: Arbitration Timing Example

The behavior of the arbitration logic is demonstrated with this sample wave containing nodes #10 and #5 followed by a wave containing only node #15. The bus is idle at the time that the first contest begins. Board #10 wins after two cycles, begins its transfer and releases bus ownership. This prompts #5 to arbitrate against itself, win and then wait for the end of the current transfer. When #10 is finished with the bus, #5 starts its transfer and releases bus ownership. This prompts #15 to enter a wave and begin to arbitrate against itself.

The timing of the arbitration logic is shown by example in Figure 2.10. The most important details are that the ARBx lines are driven beginning on the sample edge instead of the assertion edge and that the arbitration logic must settle within 200 nsec after starting.

In this example the bus masters on nodes in slot 5, 10 and 15 want to use the bus. Masters 5 and 10 go after the bus together and master 15 comes after the bus a few cycles later. At the time masters 5 and 10 become interested in the bus it is idle; that is, the last transfer is finished and the RQST/ line is inactivated.

F(0) Masters 5 (binary coded 0101) and 10 (binary coded 1010) notice that RQST/ is inactive and begin driving the ARBx lines with their respective IDx lines immediately (note that the ARBx lines are activated on the sample clock not on the assertion clock). Using the sample arbitration logic shown in Figure 2.9, driving the ARBx lines corresponds to activating the arb/ line on their respective nodes. The ARBx lines will be changing as the logic settles but must be stable early enough so that the internal signal GRANT correctly reports the winning master after two cycles.

R(1) Masters 5 and 10 both activate RQST/ (form a wave) to stop further masters from entering arbitration late (specifically, master 15 will be blocked until both 5 and 10 release RQST/).

F(1) Arbitration continues.

R(2) Arbitration continues; that is, both masters continue to activate their arb/ and RQST/ lines.

F(2) The arbitration period is over. Masters 5 and 10 look at their respective GRANT lines to see who is the winner. Master 10 sees an active GRANT and master 5 sees an inactive GRANT. Master 10 is the winner. Both masters continue to activate RQST/ and their own arb/ lines.

R(3) Master 10 begins the START cycle of its transfer. Since it does not want to lock the bus it releases its activation of RQST/, but it continues to activate its arb/ line. Master 5 continues to activate RQST/ and its arb/ line.

F(3) Master 10 releases its local arb/ line which releases its activation of the ARBx lines. Master 5 detects the START cycle and begins waiting for this arbitration to settle.

R(4) Master 5 continues to activate RQST/ and its arb/ line.

F(4) Arbitration continues. Master 15 wants to use the bus and looks at RQST/. Since RQST/ is active, master 15 is not allowed to arbitrate. It will poll RQST/ each cycle until it is inactive.

R(5) Arbitration continues; that is, master 5 continues to activate RQST/ and its arb/ line.

F(5) This arbitration is over. Master 5 sees its GRANT active and is the winner. Master 15 sees RQST/ still active.

R(6) Master 5 would like to begin its transfer, but master 10's transfer is not finished. Master 5 will poll the ACK/ line each sample edge waiting for the current transfer to end. It will continue to activate RQST/ and its arb/ line so that it holds bus ownership until it can begin a transfer.

R(n) Master 10 finishes its transfer by activating ACK/. Master 5 is still activating RQST/ and its arb/.

F(n) Master 5 sees the ACK cycle. Master 15 still sees RQST/ active.

R(n+1)
Master 5 releases RQST/. This allows RQST/ to become inactive. Master 5 also begins the START cycle of its transfer.

F(n+1)
Master 5 releases its arb/ line. Master 15 sees RQST/ inactive so it activates (immediately) its arb/ line. It also sees the START so it will monitor for the ACK to track the current transfer.

R(n+2)
Master 15 activates RQST/ to form a contest with itself only. This contest will be won in two cycles and master 15 will proceed much as master 5 did after it won.

3. SpurBus Compliance with NuBus

The SpurBus design is based on early NuBus specifications [TI83, IEEE86] and it is not in full compliance with the ANSI/IEEE standard [IEEE88]. Because the SpurBus was designed specifically for SPUR's implementation some NuBus features are not used. The SpurBus subset applies to SpurBus nodes only; NuBus protocols not supported in SpurBus may still exist in a SPUR system.

3.1. SpurBus Non-Compliance with the ANSI/IEEE NuBus Standard

Early versions of Texas Instruments' NuBus [TI83] and the SpurBus define a single cycle RESET pulse as a Bus Reset. This feature is gone from the ANSI/IEEE standard. Because Bus Reset is used in SPUR's implementation, it is a feature of the SpurBus that must remain in non-compliance.

3.2. SPUR Implementation Non-Compliance with the ANSI/IEEE NuBus Standard

This section summarizes material presented throughout Section 2. It is clear that these could be corrected in future SpurBus implementations without rendering SPUR's implementation incompatible.

The pins for signals NMRQ/ and PFW/ from the ANSI/IEEE specification are not used (reserved) in SpurBus.

SpurBus only generates the resource unlock form of attention cycle as defined by the ANSI/IEEE specification.

Backplane pins A2 and C2 carry GND in SpurBus, but are reserved (RSVD) in the ANSI/IEEE NuBus.

SpurBus slaves that do not implement block transfers will respond to block transfers addressing their slot space with an error status instead of a successful status with no intermediate ACK cycles.

A configuration ROM in the high addresses of a node's slot space is not provided by SpurBus nodes.

3.3. NuBus Features Absent in SPUR [Wood87]

In SPUR's implementation of the NuBus, data transfers have two possible transfer unit sizes: 1 word and 8 words (respectively, 4 bytes and 32 bytes). The 8 word unit corresponds to a SPUR cache block size and most events will use this size unit. The 1 word size is provided for direct processor operations on physical memory or device registers (bypassing the cache).

SPUR nodes do not generate or detect parity on the bus. Refer to the SPUR Design Rationale document [Katz85b] for a discussion of this decision.

A SPUR node recognizes addresses in its slot space as interrupts if they fall within the low 1 MB; that is, if the address begins "F(id)0" (hex) where "id" is the Slot ID of the SpurBus node. This space folds around 16 unique interrupt addresses, all word aligned; that is, the four bits AD<5,2> are used to fix the interrupt type. The transfer unit size of a SpurBus interrupt is a word.

The SPUR processor cache controller design provides test-and-set operations on data obtained by the normal read for ownership cache operations. Because this operation is entirely carried out on data in the cache (or brought into the cache for this operation) the bus locked multi-transfer facilities are not used.

4. NuBus Extensions for SpurBus

The SPUR project will use NuBus memory and peripherals; however, its cache consistency and virtually addressed caches [Wood87,Wood86] make demands on the bus that the NuBus does not satisfy. An extended NuBus design, called the SpurBus, supports the needs of SPUR more fully.

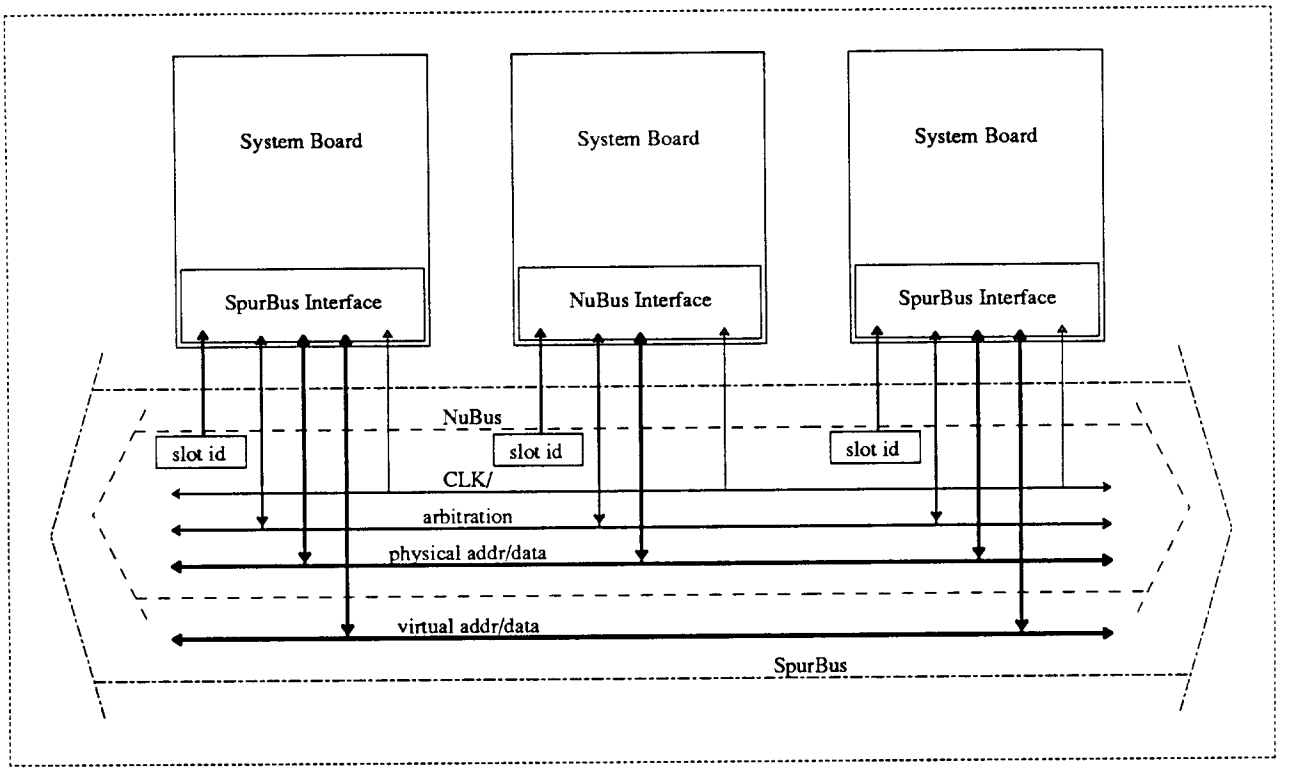

Figure 4.1: SpurBus Block Diagram

The SpurBus extends the NuBus design with a second data path. This second path is used by SpurBus snooping caches to pass shared data among themselves. It has its own control for transfers, but is synchronized to the NuBus data path for arbitration and START cycles.

4.1. Snooping Caches' Data Path

The SPUR system employs "snooping caches" [Katz85a,Wood87] to provide cache consistency across multiprocessor caches. In essence this means that each cache in the system must monitor all bus transfers and be prepared to provide data from its RAMs whenever it has the most up-to-date copy. This is implemented with state bits on each cache block that distinguish states "invalid", "owned private", "owned shared" and "unowned". The concept of ownership provides a cache the privilege of writing its copy of data and the responsibility for delivering its copy to other caches and main memory. There is at most one cache in the system that owns a cache block. When a block is not owned by a cache, it is implicitly owned by memory.

The implementation of a snooping cache requires bus bandwidth, the ability to monitor bus events, and the ability to cancel memory operations. The NuBus specification and the memory nodes that are currently available for NuBus racks (Texas Instruments products) do not make any allowances for cancelling operations. The addition of a second data path in SpurBus allows inter-cache operations to occur in parallel with memory operations without much loss of bandwidth. This way memory operations do not need to be cancelled; they can be allowed to finish without delaying the inter-cache operation.

This second path also allows the inter-cache address to be different from the memory address. This is important because it allows a SpurBus cache to be accessed by virtual addresses while the memory

system is accessed by physical addresses. During the START cycle of most SpurBus events the virtual address is sent on the inter-cache bus and the physical address is sent on the NuBus.

4.2. SpurBus Signals

The SpurBus extension to the NuBus design, shown in Figure 4.1, calls for an additional 48 backplane lines, shown in Table 4.1. These are the address/data lines, VAD<32..0>, and control lines, VACK, VTM0, VTM1, and VSACK for the inter-cache data path and 11 reserved lines[8].

The use of the inter-cache data path is synchronized with the NuBus data path START cycles. This means that only one set of arbitration logic and only one START line are needed. The master of the bus

CLASS	GROUP	SIGNAL	# OF PINS	PROTOCOL
Shared	Clock	CLK/	1	input-only
	Arbitration	ARB<3..0>/	4	open-collector
		RQST/	1	open-collector
	Utility	RESET/	1	open-collector
		ID<3..0>/	4	input-only
NuBus	Address/Data	AD<31..0>/	32	tri-state
	Parity	SP/	1	tri-state
		SPV/	1	tri-state
	Control	START/	1	tri-state
		ACK/	1	tri-state
		TM0/	1	tri-state
		TM1/	1	tri-state
SpurBus	Address/Data	VAD<32..0>/	33	tri-state
	Control	VACK/	1	tri-state
		VTM0/	1	tri-state
		VTM1/	1	tri-state
		VSACK/	1	open-collector
		Total Signals	86	
NuBus	Reserved	RSVD/	2	
SpurBus	Reserved	RSVD/	11	
	Power/Ground	+5	22	
		-5.2	14	
		+12	4	
		-12	4	
		GND/	43	
		Total Lines	186	

Table 4.1: SpurBus Backplane Lines

The SpurBus extends the NuBus backplane needs by 37 signal pins, 11 reserved pins and 42 power/ground pins. The resulting 186 pins define all but 6 pins on the two 96 pin connectors.

[8] In SPUR's implementation 7 of these 11 reserved lines are used for debugging. One line, WAIT, is used to block the issue of new START cycles; it is a backplane "single step." The other 6 lines are used to identify, with a 3 of 6 code, the responding third-party node in an inter-cache operation.

always performs a NuBus operation and can optionally perform a related inter-cache operation at the same time.

Some of the control for a NuBus operation applies to the inter-cache operation. Both data paths read or write together; that is, the read/write characteristic of a NuBus operation applies to both paths.

4.3. Inter-cache Data Path Control

The inter-cache data path is composed of address/data lines, VAD<32..0>, an acknowledgement line, VACK, a snooping flow control line, VSACK, and two multi-purpose control lines, VTM0 and VTM1.

VAD<32..0>

The inter-cache data path is VAD<31..0>. It has the same characteristics as the NuBus data path except that it is used to move cache lines according to the snooping cache protocols. The inter-cache data path is wider than the NuBus data path because of SPUR virtual addressing [Hill86]. Global virtual addresses in the SPUR system are 38 bits wide. Since cache lines are aligned on 32 byte boundaries the low order 5 bits are always zero and are not transferred. So inter-cache addresses are 33 bits.

VACK

The VACK line on the inter-cache data path corresponds to the NuBus ACK line. When transfers occur on the inter-cache data path (after an ownership ACK occurs on a snooping read) VACK terminates the operation. A cycle that has VACK active is called a VACK cycle[9].

VSACK

The VSACK line is an open-collector signal that all SpurBus snooping caches activate when they wish to delay an inter-cache operation. This allows the internal processing of each snooping cache to have differing timing. It defines the length of write for invalidation operations and defines the window of applicability of an ownership acknowledgement after a snooping read.

VTM0 and VTM1

These two control lines parallel the transaction mode lines on the NuBus data path. During a START cycle VTM0 is active if the inter-cache data path is being used. If a read is taking place then VTM1 indicates the type of snooping required as shown in Table 4.2. VTM1 activated indicates a "read for ownership" and VTM1 inactivated indicates a "read shared" operation.

After the START cycle of a snooping read all caches on nodes other than the current bus master have a window of time to determine whether they should provide the requested data. This window is controlled by VSACK. If a cache decides to provide the data it will activate VTM0 for a cycle before releasing VSACK. This is called an OACK (ownership ack) cycle.

After an OACK cycle VTM0 is used to strobe data across the inter-cache data path in the same way that TM0 is used on the NuBus data path. When the last word of the cache line is transferred VACK is activateed and VTMx carry operation status information as shown in Table 4.3. In the status code VTM0 indicates cache line clean/dirty state information. A cache line is "dirty" if the memory copy of this data is not up-to-date (ie. a store into this block has not yet propagated to the memory); otherwise, it is "clean."

During a "write for invalidation" operation (indicated by a NuBus write with VTM0 active) VTM0 and VTM1 are not used after the START cycle.

4.4. Snoop Operations

Three operations are needed to support SPUR snooping caches. Two of these are modifications on the NuBus block transfer read operation. The third is a non-NuBus operation for invalidating cache lines.

---

[9] The functions of VACK and VTM1 could be combined into one signal. They were originally split to provide inter-cache error status codes that did not materialize in SPUR.

VTM0/	TM1/	VTM1/	Inter-cache Operation
H			None
L	L		Write for Invalidation
L	H	H	Read for Shared Copy
L	H	L	Read for Private Copy

Table 4.2: Snooping Operation Encoding

During the START cycle of any NuBus operation the usage of the inter-cache data path is determined from the VTM0, VTM1 and TM1 lines.

VTM0/	Status Message and Cache Line Status
L	Successful Completion; Line Dirty
H	Successful Completion; Line Clean

Table 4.3: Snooping Operation Status Codes

During the VACK cycle of a snooping read on the inter-cache data path VTM0 indicates whether the data that came across the inter-cache bus matches its memory image (clean) or has been written since memory was last updated (dirty).

4.4.1. Snooping Reads

"Snooping reads" are variations on the NuBus block transfer read. In the START cycle of these operations the NuBus control describes a regular 32 byte read and the inter-cache control qualifies this as non-snooped, snooped for a read-only copy (called "read shared") or snooped for a writable copy (called "read for ownership"). The NuBus operation is unaffected; the inter-cache bus carries the modifiers and the potential inter-cache data.

Figure 4.2 shows the protocol for snooping reads.

R(1) The master drives AD<31..5> with a cache block physical address (in units of aligned 32 byte blocks), SP and SPV as appropriate[10], and VAD<32..0> with the virtual address of the same cache block (also in units of aligned 32 byte blocks). As shown in Tables 2.2 and 2.3, it also drives the control lines AD<4..0> to inactive, active, active, active and inactive, respectively, and TM0 and TM1 both to inactive. START, VSACK, and VTM0 are activated[11], and ACK and VACK are inactivated. If the operation is read for ownership, it also activates VTM1.

F(1) All potentially addressed slaves on the NuBus sample the AD<31..0> and TMx lines to determine whether they are affected. All SpurBus nodes sample the AD<4..0>, TMx, VAD<32..0> and VTMx

---

[10] The SpurBus inter-cache data path, VAD, should also have a parity and possibly a parity valid line. These are missing because the SpurBus was designed for SPUR and it was known that SPUR was not going to support parity.

[11] The code choice of activating VTM0 in the START cycle is unfortunate because VMT0 has meaning in the immediately following cycle (ownership acknowledgement). So some node must drive VTM0 determinate in the first cycle after START. The slaves are not a good choice because it is possible that no slave will respond or that a slave may respond immediately. SPUR's implementation avoided this because all slaves were known to require at least two cycles to determine cache block ownership, so the master drives VTM0 inactive on the cycle after the START. Inverting the sense of VTM0 in the START cycle encoding is the right solution because non-SpurBus reads could still be distinguished by inactivation of VSACK.

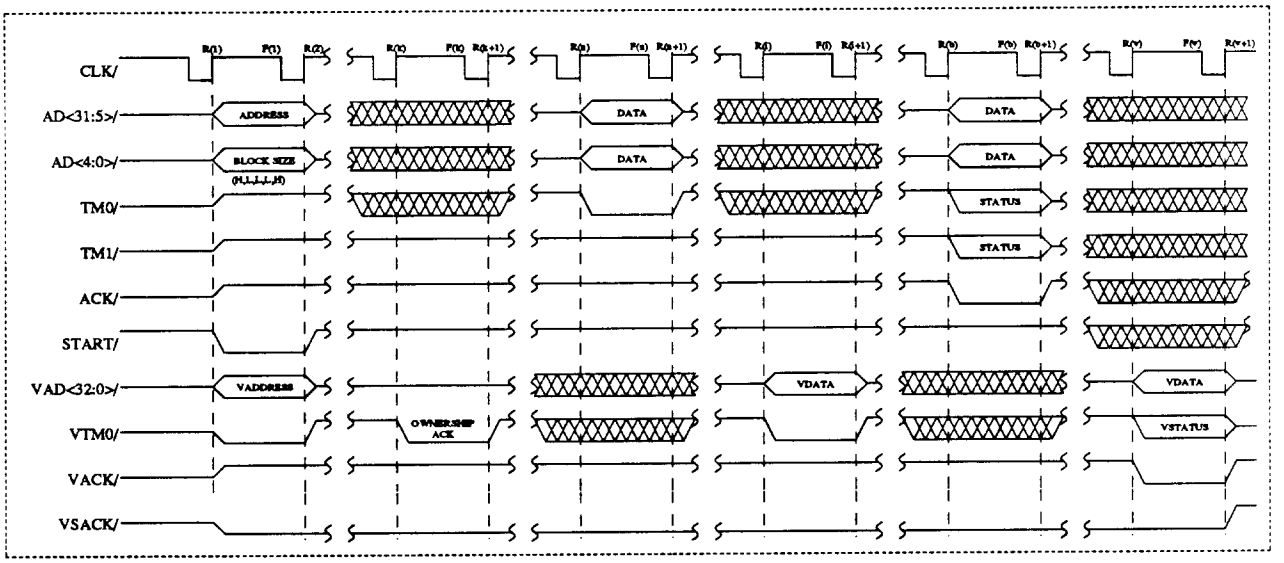

Figure 4.2: Snooping Read Protocol

A SpurBus snooping read may result in a cache-to-cache transfer of a shared data cache block on the inter-cache data path. There will be at most one cache responding and it will generate an ownership acknowledgement before VSACK is inactivated to indicate that it will reply. The address sent on the inter-cache bus is the virtual address of the block in the master's cache (less the block offset bits). The two transfers proceed with distinct control and may end at different times. In particular, the NuBus transfer may end before the inter-cache transfer; thereby, allowing the next NuBus owner, if it is not a SpurBus node, to begin an overlapping but independent NuBus-only event.

---

lines and check their cache block states to determine whether they are affected and should reply with a cache block of data.

R(2) All potentially affected slaves activate VSACK indicating that they are processing the inter-cache operation. When VSACK is eventually deactivated, the master knows that all slaves have completed inter-cache event processing and are ready to proceed with the next event. The master releases the ADx, TMx, SP, SPV, ACK, VADx, VTM1, VACK, and VSACK lines. It deactivates START and VTM0 (see footnote on R(1)). It will now wait on the TM0, ACK and VTMx lines.

NOTE:

The transfers on the NuBus data path and inter-cache data path are decoupled. In the following descriptions, events pictured sequentially must be sequential only if they apply to the same data path; otherwise, they may occur together or in either order. That is, R(i) must follow R(k), but R(n) could overlap either R(k) or R(i).

R(k) If a cache on a SpurBus slave decides to respond with a cache block on the inter-cache data path it activates VTM0 for a cycle. This is called an OACK cycle (ownership ack).

F(k) The master samples the VTMx lines, sees the OACK, and determines that a inter-cache transfer will take place. For OACK to be meaningful, the master must also see VSACK active. This implies that data arriving on the NuBus data path is to be thrown away. The master then starts sampling the VADx, VTMx, VACK, and VSACK lines for data and completion status.

R(k+1)
If the responding inter-cache slave does not have the first word ready on immediately, it drives VTM0 inactive.

R(n) The addressed NuBus slave drives the ADx with data, TM0, SPV (and SP as appropriate) lines. This is one of the first B-1 words of the B word block.

F(n) The master sees the intermediate ACK on TM0 and captures the data from the ADx lines (if it has not decided to throw away this data in preference for inter-cache data).

R(n+1)
If the NuBus slave cannot transfer the next word immediately, it drives TM0 inactive.

NOTE:
These NuBus intermediate stages (n) are repeated B-1 times where the size of the transfer unit in words is B.

R(i) The responding inter-cache slave drives VADx with data and activates VTM0. This is one of the first B-1 words of the B word block.

F(i) The master sees VTM0 active and captures data from the VADx lines. It will keep this data in preference over data from the NuBus data path.

R(i+1)
If it cannot generate the next word immediately, the responding cache drives VTM0 inactive.

NOTE:
These inter-cache intermediate steps (i) are repeated B-1 times where the size of the transfer unit in words is B.

R(b) The addressed NuBus slave drives the ADx lines with the last word of the line, drives the TMx lines with a status code and activates the ACK line.

F(b) The master captures the final word of the NuBus data (if not overridden by inter-cache data) and the status code. Non-successful status codes on the NuBus data path should be forwarded to the master's processor even if an inter-cache transfer overrides NuBus data.

R(b+1)
The NuBus slave releases the ADx, SPV, TMx and ACK lines for the next NuBus owner to use. The next owner must drive ACK determinate. If the next owner is a SpurBus node, it will not begin a START cycle until VSACK has been deactivated. However, if the next SpurBus owner is a non-SpurBus NuBus node, it may begin an independent NuBus-only event before the current inter-cache event is completed. In this case SpurBus nodes must be prepared to be addressed on the NuBus while still handling an inter-cache transfer[12].

R(v) The responding cache drives the VADx lines with the last word of the block, drives the VTMx lines with a status code, activates the VACK line and activates the VSACK line one last time[13]. The status code[14] tells the master if the inter-cache block agrees with memory, VTM0 inactive, or

---

[12] SPUR's implementation of the SpurBus handles this case by stalling the overlapping NuBus-only event until the inter-cache transfer is over.

[13] Since a new SpurBus event cannot begin until VSACK has been seen inactive, activating VSACK on the VACK cycle ensures that the cycle after the VACK will not be used by a following SpurBus event. This is done so that the inter-cache slave knows that it can deactivate VTM0 and VACK on the next cycle. If VSACK was deactivated on this cycle then the next owner would be responsible for deactivating VTM0, and if that node was non-SpurBus, its NuBus-only event would not drive VTM0. In this way a false SpurBus event might be seen by SpurBus nodes.

[14] There should be an error status associated with inter-cache transfers. It is missing because the SpurBus follows the SPUR implementation and the SPUR implementation does not detect any errors on the responder's side of

disagrees with memory, VTM0 active. This is needed if the master will become the owner of the block; thereby, assuming responsibility for updating memory before replacing the block in its cache.

F(v) The master captures the final word of inter-cache data and associated status code.

R(v+1)
The responding inter-cache slave releases the VADx and deactivates the VTMx, VACK, and VSACK lines for the next owner to use. If the NuBus side is already idle and the next owner is a SpurBus node, it will not begin its START cycle until after it has seen VSACK inactive.

(1) The parameter "b" is constrained to be less than the system timeout on transfers. If the timeout fires the transfer will be aborted by a "timeout" ACK cycle generated by the system logic.

4.4.2. Write For Invalidate

The third operation needed to support snooping caches is called "write for invalidation." It is used by a cache with a non-private copy of a block that it wants to write. It must cause all other caches to

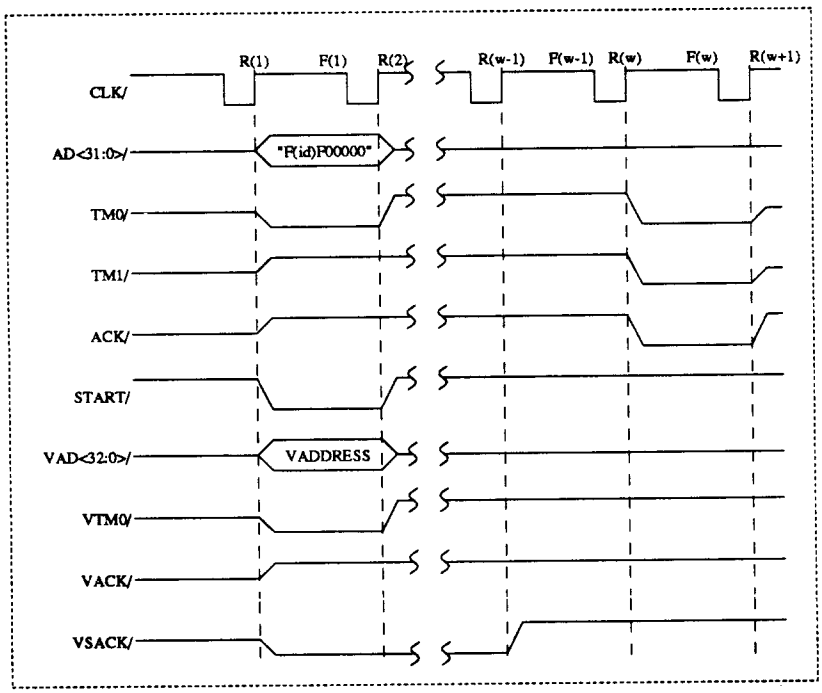

Figure 4.3: Write For Invalidation Protocol

A SpurBus write for invalidate operation is used to invalidate remote copies of a cache block while transferring ownership to the issuing cache's copy. It employs a NuBus word write whose address points back into the generating node's slot space (but not into interrupt locations) so that other NuBus slaves will not be involved. The invalidation information is the virtual address of the cache line to invalidate and is carried on the inter-cache data path. The duration of the operation is controlled by the VSACK line; after VSACK is deactivated, the originating master issues itself an ACK cycle to terminate the NuBus event.

---

an inter-cache transfer.

invalidate (throw away) their copies of that cache block. This requires a bus operation, but it doesn't correspond to a useful parallel NuBus operation. The SpurBus uses a NuBus single word write to an non-interrupting address in the owner's own slot space to implement the write for invalidation. The invalidation information is transmitted during the START cycle. The operation lasts long enough for all caches to process it, then the master generates an ACK cycle to itself.

The protocol for an write for invalidation operation is shown is Figure 4.3.

R(1) The master drives the ADx lines with "F(id)F00000" (hex) where "id" is the slot ID of the issuing master. This is a dummy physical address that falls into the master's own slot space. This guarantees that no other NuBus slave will respond to this operation. The master will complete this operation itself after VSACK is deactivated. Also driven by the master during the START cycle is TMx, VTMx, VACK, VSACK and the VADx lines as shown in Figure 4.3. This makes the operation a snooped word write. The VADx lines carry the virtual address of a cache block to be invalidated by other snooping caches.

F(1) All SpurBus snooping caches determine that a write for invalidation is active and look into their caches to see if they have the block to be invalidated.

R(2) All affected SpurBus slaves activate VSACK to indicate that they are in the process of handling the write for invalidation. The master releases the ADx, VADx, VTMx, TMx and VSACK lines. It will delay as long as VSACK remains active.

R(w-1)
The last SpurBus node to complete the write for invalidation releases VSACK causing it to become inactive.

F(w-1)
The master polls VSACK each cycle until it observes VSACK inactive. This should occur in less than the NuBus watchdog timeout period.

R(w) The master completes its own operation by driving a successful status code on the TMx lines and activating ACK to form an ACK cycle.

R(w+1)
The master releases the TMx and ACK lines for the next owner. The next owner must drive ACK determinate, possibly initiating the next START cycle.

5. Mechanical Specifications

5.1. Boards

The mechanical structure of NuBus/SpurBus nodes follows the "Eurocard" design. They are triple height IEC boards (9U high where U = 1.74in) and have triple depth (11.024in).

5.2. Backplane Connectors

There are three 603-2-IEC-C096-M connectors mounted on each board. These are referred to as P1, P2 and P3 from top to bottom respectively. NuBus defines the pins on P1. SpurBus additionally defines some of the P2 pins. See Tables 5.1 and 5.2 for the pin assignments.

Pin/Row	Col. A	Col. B	Col. C	Pin/Row	Col. A	Col. B	Col. C
1	-12	-12	RESET/	17	AD23/	GND	AD22/
2	GND	GND	GND	18	AD25/	GND	AD24/
3	SPV/	GND	+5	19	AD27/	GND	AD26/
4	SP/	+5	+5	20	AD29/	GND	AD28/
5	TM1/	+5	TM0/	21	AD31/	GND	AD30/
6	AD1/	+5	AD0/	22	GND	GND	GND
7	AD3/	+5	AD2/	23	GND	GND	RSVD/
8	AD5/	-5.2	AD4/	24	ARB1/	-5.2	ARB0/
9	AD7/	-5.2	AD6/	25	ARB3/	-5.2	ARB2/
10	AD9/	-5.2	AD8/	26	ID1/	-5.2	ID0/
11	AD11/	-5.2	AD10/	27	ID3/	-5.2	ID2/
12	AD13/	GND	AD12/	28	ACK/	+5	START/
13	AD15/	GND	AD14/	29	+5	+5	+5
14	AD17/	GND	AD16/	30	RQST/	GND	+5
15	AD19/	GND	AD18/	31	RSVD/	GND	GND
16	AD21/	GND	AD20/	32	+12	+12	CLK/

Table 5.1: P1 Connector Assignments

The P1 connector on a NuBus node has these pin assignments. The connector is type 603-2-IEC-C096-F. Column B is optional, allowing a 64 pin minimum power subset. In the ANSI/IEEE NuBus standard [IEEE88], Pin A2 and C2 are RSVD/, pin C23 is PFW/ and pin A31 is NMRQ/.

Pin/Row	Col. A	Col. B	Col. C	Pin/Row	Col. A	Col. B	Col. C
1	-12	-12	VSACK/	17	VAD23/	GND	VAD22/
2	GND	GND	GND	18	VAD25/	GND	VAD24/
3	-	GND	+5	19	VAD27/	GND	VAD26/
4	-	+5	+5	20	VAD29/	GND	VAD28/
5	VTM1/	+5	VTM0/	21	VAD31/	GND	VAD30/
6	VAD1/	+5	VAD0/	22	GND	GND	GND
7	VAD3/	+5	VAD2/	23	GND	GND	VAD32/
8	VAD5/	-5.2	VAD4/	24	RSVD/	-5.2	RSVD/
9	VAD7/	-5.2	VAD6/	25	RSVD/	-5.2	RSVD/
10	VAD9/	RSVD/	VAD8/	26	-	-5.2	-
11	VAD11/	RSVD/	VAD10/	27	-	-5.2	-
12	VAD13/	GND	VAD12/	28	VACK/	+5	RSVD/
13	VAD15/	GND	VAD14/	29	+5	+5	+5
14	VAD17/	RSVD/	VAD16/	30	RSVD/	GND	+5
15	VAD19/	RSVD/	VAD18/	31	GND	GND	GND
16	VAD21/	GND	VAD20/	32	+12	+12	RSVD/

Table 5.2: P2 Connector Assignments

The P2 connector on a SpurBus node has these pin assignments. The connector is type 603-2-IEC-C096-F. Column B is optional, allowing a 64 pin minimum power subset. In the ANSI/IEEE NuBus standard [IEEE88], pin B10 is -5.2EN, B11 is -5.2OUT, B14 is +12EN and B15 is +12OUT.

Pin/Row	Col. A	Col. B	Col. C	Pin/Row	Col. A	Col. B	Col. C
1	-	-	-	17	-	-	-
2	-	GND	-	18	-	-	-
3	-	GND	-	19	-	GND	-
4	-	-	-	20	-	RSVD/	-
5	-	+5	-	21	-	RSVD/	-
6	-	+5	-	22	-	-	-
7	-	+5	-	23	-	GND	-
8	-	-	-	24	-	-	-
9	-	-	-	25	-	-	-
10	-	RSVD/	-	26	-	-	-
11	-	RSVD/	-	27	-	-	-
12	-	GND	-	28	-	+5	-
13	-	-	-	29	-	-	-
14	-	RSVD/	-	30	-	GND	-
15	-	RSVD/	-	31	-	GND	-
16	-	GND	-	32	-	-	-

Table 5.3: P3 Connector Assignments

The P3 connector on a NuBus node should have these pin assignments. The connector is type 603-2-IEC-C096-F. In the ANSI/IEEE NuBus standard [IEEE88], pin B10 is -5.2EN, B11 is -5.2OUT, B14 is +12EN, B15 is +12OUT, B20 is -12OUT and B21 is -12EN.

6. References

[Hill86] Hill, M.D., et al, "Design Decisions in SPUR," IEEE Computer, vol. 19, no. 11, November 1986.

[Hill87] Hill, M.D., "Aspects of Cache Memory and Instruction Buffer Performance," Computer Science Division Technical Report UCB/CSD 87/381, University of California, Berkeley, November 1987.

[IEEE86] IEEE, "NuBus - a Simple 32-Bit Backplane Bus, P1196 Specification Draft 2.0," The Institute of Electrical and Electronics Engineers, New York, NY, December 1986.

[IEEE88] IEEE, "IEEE Standard for a Simple 32-Bit Backplane Bus: NuBus," The Institute of Electrical and Electronics Engineers, New York, NY, August 1988.

[Katz85a] Katz, R.H., et al,"Implementing a Cache Consistency Protocol," Proc. Twelfth International Symposium on Computer Architecture, June 1985, pp 276-283.

[Katz85b] Katz, R.H. ed., Proc. of CS292i: Implementation of VLSI Systems, Computer Science Division Technical Report UCB/CSD 86/259, University of California, Berkeley, September 1985.

[Ouster88] Ousterhout, J.K., et al, "The Sprite Network Operating System," IEEE Computer, vol. 21, no. 2, February 1988.

[Taylor86] Taylor, G.S., et al, "Evaluation of the SPUR Lisp Architecture," Proc. Thirteenth International Symposium on Computer Architecture, June 1986.

[TI83] Texas Instruments, NuMachine NuBus Specification, Part number TI-2242825-0001, 1983.

[Wood86] Wood, D.A., et al, "An In-Cache Address Translation Mechanism," Proc. Thirteenth International Symposium on Computer Architecture, June 1986, pp 358-365.

[Wood87] Wood, D.A., S.J. Eggers, G.A. Gibson, "SPUR Memory System Architecture," Computer Science Division Technical Report UCB/CSD 87/394, University of California, Berkeley, December 1987.