System Specification for MBus SPARC Compatible

Rev 1.0

January, 1991
ι.012591

VIA Technologies, Inc. (408) 746-2200
860 East Arques Avenue
Sunnyvale, CA 94086

Overview of Processor Section

The processor section of the compatible is based around the Cypress CY7C600 Uni-Module Board. It is a complete SPARC chip set consisting of the CY7C601 (Integer Unit), the CY7C602 (Floating-Point Unit), the CY7C604 (Cache Controller and Memory Management Unit), and two CY7C157s (Cache RAMs). The processor section communicates with the rest of the system through the Mbus.

Memory Management Unit

The memory management unit resides in the CY7C604. It provides translation from a 32 bit virtual address range (4 gigabytes) to a 36 bit physical address (64 gigabytes), as provided in the SPARC reference MMU specification. Virtual address translation is further extented with the use of a context register, which is used to identify upto 4096 contexts or tasks. The cache tag entries and TLB entries contain context numbers to identify tasks or processes. This minimizes unnecessary cache tag and TLB entry replacement during task swithching.

The MMU features a 64 entry Translation Lookaside Buffer (TLB). The TLB acts as a cache for address mapping entries used by the MMU to map a virtual address to a physical address. These mapping entries, referred to as page table entries or PTE's, allow one of four levels of address mapping. A PTE can be defined as the address mapping for a single 4-Kbyte page, a 256 Kbyte region, a 16 Mbyte region, or a 4 Gbyte region. The TLB entries are lockable, allowing the user to exclude important TLB entries from replacement.

As specified by the SPARC reference MMU, the MMU provides translation for bits 31 through 12 of the virtual address to an expanded physical address mapping using bits 35 through 12. Bits 11 through 0 of the virtual address are not translated, and are defined as the page offset for the 4-Kbyte memory page.

Cache

The cache on the compatible is a 64 Kbyte direct mapped write through virtual cache. The cache controller and the cache tag RAMs reside in the CY7C604 and is designed to use two CY7C157 Cache RAMs for the cache memory. The cache is organized as 2048 cache lines of 32 bytes each. The CY7C604 has 2048 cache tag entries on-chip, one tag entry for each cache line. The virtual address field VA<15:5> selects one of the 2048 lines of the cache. Cache data replacement is always performed by replacing cache lines.

For the write through cache used in the compatible, write access cache hits cause both the cache and the main memory to updated simultaneously. A write access cache miss causes only the main memory to be updated (no write allocate). The selected cache line is invalidated for a write access cache miss.

During read access cache hits, the cached data is read out and supplied to the IU. In case of a read access cache miss, a cache line is fetched from main memory and loaded into the cache before the required data is supplied to the IU.

Each entry in the cache tag consists of the 16 bits of virtual address (VA<31:16>), a 12 bit context number (CXN<11:0>), one valid bit (V) and one modified bit (M). A supervisor bit (S) is included in the cache tag entry. For cache tag entries which are accesible by the supervisor only (access level 6 or 7), the S bit is set.

Main Memory Specifications

The main memory of the compatible consists of three sections:

1. Mbus Interface
2. Memory Array
3. DRAM Controller

The Main Memory supports a Level 2 Mbus Interface as per the SPARC Mbus Interface Specifications. Read and write transactions of byte, halfword, word, doubleword, and bursts of 16, 32, 64, and 128 bytes are supported. Coherent Invalidate transactions for the Level 2 Mbus Interface are also supported.

The Memory Array is organized as 2 banks of memory with each bank consisting of 8 1M x 9 DRAM modules (SIMMs) forming a data path 64 bits wide. Parity on a per byte basis is generated when memory is written and checked when it is read. The memory can be upgraded to use 4M x 9 DRAM modules without changing the design. The design of the memory allows for the following 6 configurations by setting up a configuration register in the Mbus Interface:

1.	8MB	1M x 9 SIMMs, non-interleaved, half populated (1bank)
2.	16MB	1M x 9 SIMMs, non-interleaved, fully populated
3.	16MB	1M x 9 SIMMs, interleaved, fully populated
4.	32MB	4M x 9 SIMMs, non-interleaved, half populated (1 bank)
5.	64MB	4M x 9 SIMMs, non-interleaved, fully populated
6.	64MB	4M x 9 SIMMs, interleaved, fully populated

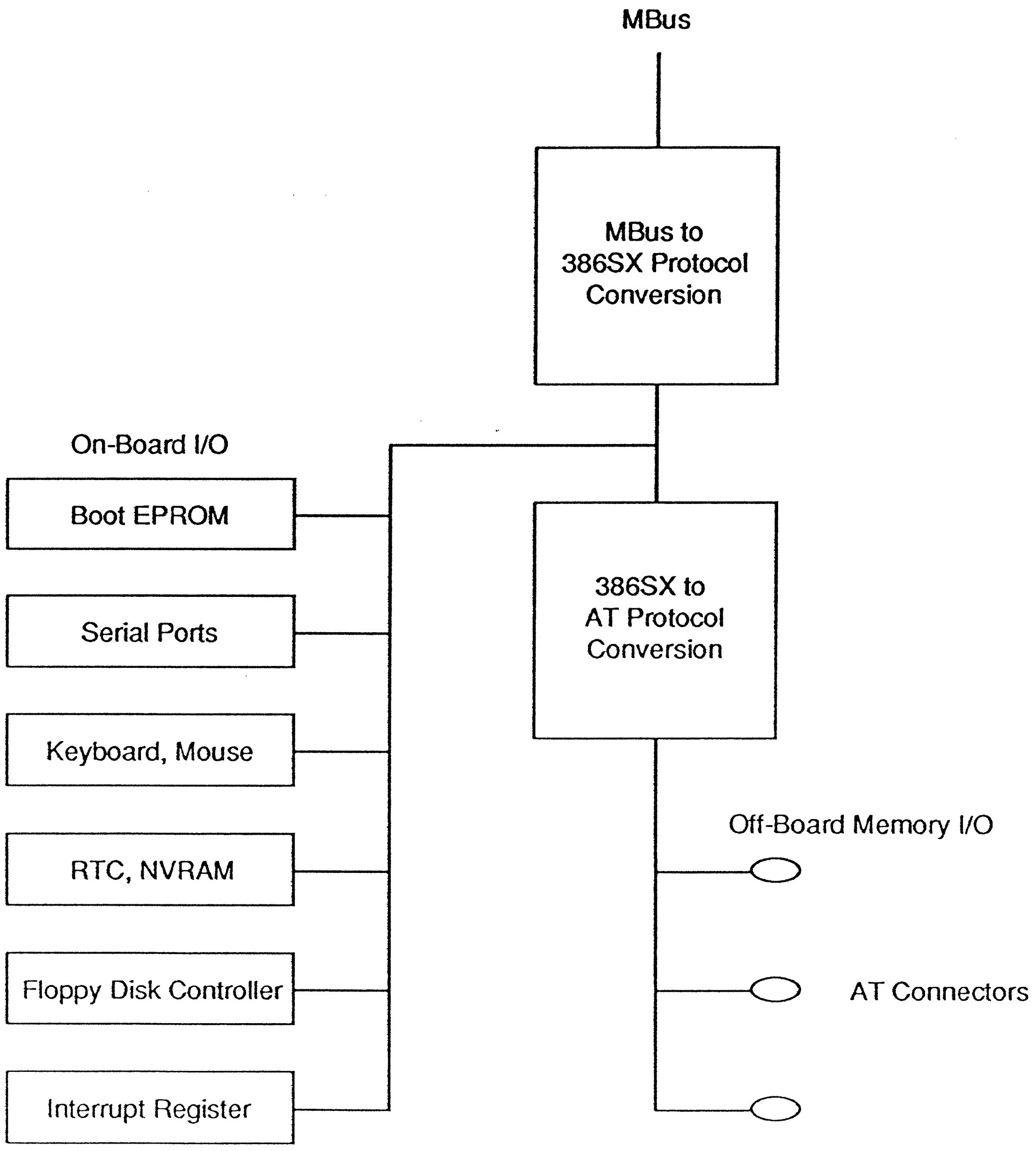
The DRAM Controller have two sets of DRAM address and control logic so that it can control each bank of memory independently of the other. Interleaving the two banks of memory is done on a doubleword basis.

Transactions to Main Memory are initiated by an Mbus master. The transaction is sent across MAD<63:0>, which is a multiplexed address and data bus, and received by the Mbus Interface. The address, transaction type and size are latched on the other side of the Mbus Interface. For reads, the data is parallel loaded from each of the memory banks into the Mbus Interface and transmitted back to the Mbus master who initiated the transaction. For writes, the Mbus master holds the data on MAD<63 :0> until the DRAM Controller acknowledges that it has written the data to the Memory Array. Burst transactions cause the DRAM Controller to do multiple DRAM page mode cycles. The DRAM Controller has programmable wait states based on DRAM speed and system clock frequency to provide maximum memory bandwidth given various memory configurations.

Reference: SPARC Mbus Interface Specification, Revision 1.1, March 29, 1990

On-Board and AT Bus I/O

Overall Architecture



General Description

Devices can be interfaced in one of two ways. The first is for standard on-board devices,which can be connected to a specialized byte-wide bus. The second is an AT interface.

AT Interface

There are two levels of bus conversion in the AT interface. The first translates Mbus cycles into the the bus protocol of the Intel 386sx.There is no actual 386 processor involved, just logic which executes the bus cycle sequences of the 386sx, with a data path to adapt the 64- bit wide MAD data bus to a 16 bit width.The 386sx protocol is used as an intermediate format which can easily interface to other system busses, in this case the AT.

In the second level of bus conversion, 386sx bus cycles are translated into AT cycles.Data transfer sizes supported are: 64, 32, 16 and 8 bits.The AT bus has 24 bits of address and 16 bits of data. Devices on the AT can be connected to either the full 16-bit data bus, or just to the lower 8 bits. In the case of a 16-bit access to an 8-bit AT data path device, the AT logic executes a bus sizing sequence, performing two 8-bit transfers on the AT bus.In the case of a byte access to either an 8-bit or 16-bit data path AT device, the AT logic copies data from/to the low 8 bits of the bus to/from the high 8 bits, when the byte address that it sees is odd (A[0] = 1).

The AT divides its address space into memory and I/O spaces,and accesses to both spaces are supported. Memory and I/O spaces are distinguished by Mbus address bit 24. To address AT memory space, accesses are made within AT address space with bit 24= 1. To address AT I/O space, accesses are made with bit 24= 0.

In addition to the set of I/O devices on board, there is the capability of expanding, by means of plugging standard AT daughtercards into the connectors provided.

On-Board Devices

Any transfer size permitted by the MBUS is allowed(byte,halfword,word,double,16,32,64, and 128 byte burst).

EPROM

The size of the boot prom is 256K bytes. It consists physically of a 256K by 8, CMOS EPROM, part number AM27C020, speed 100ns.

The boot prom is always selected when Mbus bit MBL (MAD<45>) = 1.Data accesses in Bypass Mode (ASI = 20-2f) will access the boot prom using PA<17:0>.The physical base address is 0xFF0000000. Alternatively, the PROM can be accessed at address F00000000. The only difference is that a write operation to 0xFF0000000 will result in a null cycle, while a write to 0xF00000000 will proceed as a write operation.

Serial Ports

There are two identical serial ports, referred to as A and B. They are contained in one Z85C30 SCC chip from Zilog or AMD. The external connection is RS-232. Both ports interrupt on Mbus level 12. Port A has higher priority than Port B. The registers are located as follows:

Address			
0xf01000000	Port B control	byte	read/write
0xf01000001	Port B data	byte	read/write
0xf01000002	Port A control	byte	read/write
0xf01000003	Port A data	byte	read/write

The clock oscillator input frequency is 4.9152 Mhz.A device recovery time of 800 ns must be observed between accesses.

Keyboard/Mouse

This is also a Z85C30 SCC chip, with Port A corresponding to the keyboard, and Port B the mouse. The interrupt is on level 12, with lower priority than the 2 serial ports. The register address locations are as follows:

Byte Address			
0xf02000000	mouse control	byte	read/write
0xf02000001	mouse data	byte	read/write
0xf02000002	keyboard control	byte	read/write
0xf02000003	keyboard data	byte	read/write

The clock oscillator input frequency is 4.9152 Mhz.A device recovery time of 800 ns must be observed between accesses.

Real-Time Clock/ NVRAM

This is a Mostek MK48T02B, size 2K bytes. The base address is 0xF04000000.

Floppy Disk Controller

This is an Intel 82077 floppy disk controller. Register locations are as follows:

0xF03000000	Status Register A	byte	read
0xF03000001	Status Register B	byte	read
0xF03000002	Digital Output Register	byte	read/write
0xF03000004	Main Status Register	byte	read
0xF03000004	Data Rate Select Register	byte	write
0xF03000005	Data (FIFO)	byte	read/wr
0xF03000007	Digital Input Reg/Configuration	byte	read/wr

ISDN

This is an AMD AM79C30A (Digital Subscriber Controller). Its Base address is 0xF05000000.

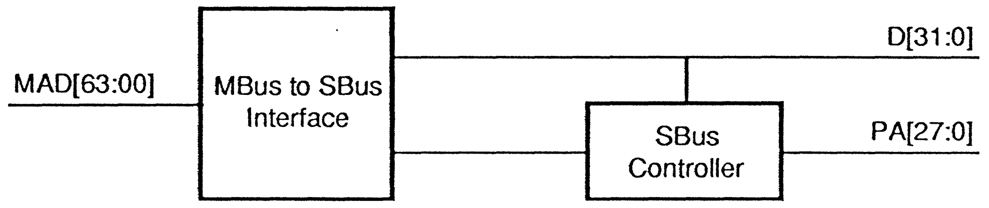
SPARC Mbus to SBus Interface

The compatible has a SBus interface to attach the frame buffer and other SBus peripherals. The SBus interface is attached to Mbus, the main system bus through which the CPU communicates with the Memory subsystem as well as all other sections of the machine. The Mbus to SBus (M2S) interface can be a master or a slave on either bus. The Mbus interface is designed to operate at clock frequencies up to 40 Mhz while the SBus interface is designed to operate within the range of 16.67 to 25 MHz. For Mbus clock frequencies above 33.33 Mhz, the SBus interface will operate at half of the Mbus clock frequency. In addition to the bus interface logic, this section of the machine also contains an SBus Controller.

Mbus to SBus Interface

This section contains the logic for connecting the 64-bit Mbus to the 32-bit SBus. This interface can behave as both a master and a slave on either the Mbus or the SBus. For transactions going from Mbus to SBus, the M2S interface is an Mbus slave for an Mbus master like the CPU. After receiving the transaction, the M2S interface then becomes an SBus master and initiates a transfer to the targeted SBus slave. For transfers going from SBus to Mbus, the M2S interface is an SBus slave for an SBus master like a DVMA master. After receiving the transfer, the M2S interface then becomes an Mbus master and initiates a transaction to the targeted Mbus slave.

There is single set of control logic to control the interfaces on both busses. Only one Mbus to SBus transaction or one SBus to Mbus transfer can be processed by the M2S interface at any one time.



The M2S Interface supports Mbus Read and Write transactions of the following sizes: byte, halfword, word, doubleword. A doubleword Mbus transaction becomes two single word transfers on the SBus.

The M2S Interface supports SBus Read and Write transfers of the following sizes: byte, halfword, word.

The M2S interface supports bus sizing as a master on the SBus. Bus sizing allows a master to initiate a word or halfword transfer to a slave device without regard to whether or not the slave supports a transfer that large. The intent is to allow a master to treat the slave as though it were a word or halfword device, even though the slave may implement only halfword or byte transfers. Bus sizing can occur only during word or halfword transfers. Bus sizing reduces software complexity. For example, an 8-bit frame buffer that is otherwise functionally identical to 32-bit frame buffer can use the 32-bit software unmodified.

Mbus as a Master

If the M2S interface is not busy, then the Mbus transaction is accepted. A protocol conversion is performed and a bus request is made to the SBus Controller. The transfer proceeds on SBus after ownership is acquired. The M2S interface then waits for an acknowledgment from the targeted SBus slave. When the acknowledgment is received, it is converted to an Mbus acknowledgment and the transaction is completed. For a read transaction the read data is driven onto Mbus during the cycle in which the acknowledgment is given. If the M2S interface is busy, then a Relinquish and Retry (R&R) acknowledgment is given, indicating that the Mbus master should relinquish ownership of Mbus and retry the transaction after the Mbus is rearbitrated.

Also, if the Mbus transaction is a doubleword transaction or if the acknowledgment from the SBus slave results in bus sizing taking place on SBus, then a R&R acknowledgment is given back to the Mbus master. The R&R acknowledgment frees up the Mbus for other transactions while the multiple SBus cycles are taking place. When the SBus transfers have completed, the M2S interface waits for the Mbus master to reissue the transaction before giving the Mbus acknowledgment.

SBus as a Master

If the M2S interface is not busy, then the SBus transfer is accepted. A protocol conversion is performed and a bus request is made to the Mbus Arbiter. The transaction proceeds on Mbus after ownership is acquired. The M2S interface then waits for an acknowledgment from the targeted Mbus slave. When the acknowledgment is received, it is converted to an SBus acknowledgment and the transfer is completed. For a read transfer the read data is driven onto SBus on the cycle following the one in which the acknowledgment was given. If the M2S interface is busy, then a Rerun acknowledgment is given, indicating that the SBus master should retry the transfer.

SBus Controller

The SBus Controller contains the logic for controlling the SBus. It supports transfers of the following sizes:

Byte	
Halfword	(2 bytes)
Word	(4 bytes)
Two Word Burst	(8 bytes)
Four Word Burst	(16 bytes)
Eight Word Burst	(32 bytes)
Sixteen Word Burst	(64 bytes)

The SBus Controller supports up to 4 SBus masters of which one is the M2S interface. It employs a fair arbitration scheme so that all pending bus requests are granted before a master is allowed to acquire the bus again. It supports geographically selecting six SBus slaves, one being the M2S interface. Virtual to physical address translation is done through an eight-entry fully associative TLB with LRU as replacement policy. The TLBs provide translation for a 32 Mbyte address space for each SBus slot. Translation can also be disabled on a per SBus slot basis.

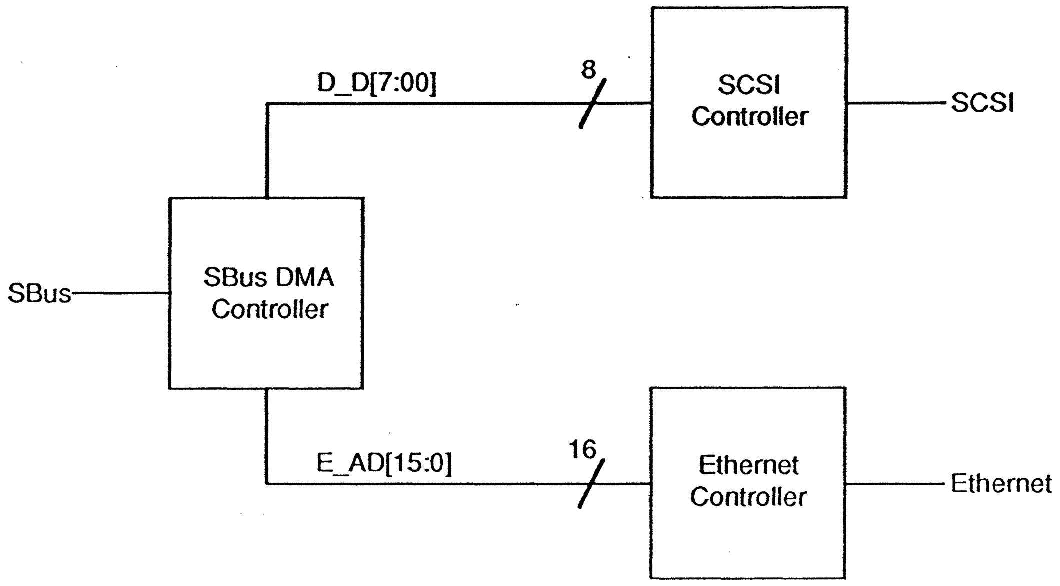
The SBus Controller also has an 8-bit counter to generate a bus timeout acknowledgment if an SBus slave does not give any acknowledgment by the 256th slave cycle.

DMA I/O Structure

The DMA section of the Sparc processor consists of 3 main sections:

1. Sbus DMA controller
2. SCSI controller
3. Ethernet controller

Numbers 2 and 3 consist of the following ICs: NCR 53C90, and AMD7990, respectively. The Sbus DMA controller is made from discrete logic which will be integrated into a gate array.



SCSI Controller (NCR 53C90)

Data is transferred over the 8 - bit SCSI cable at 3 Mbytes/sec worst case.The chip is controlled via a set of 13 byte - wide registers, memory mapped into M bus physical addresses. SCSI commands are given using one of these registers. For data transfers, there is a 16 deep byte-wide FIFO. Data in the FIFO appears to the CPU as another of the 13 registers. Block transfers of data can be done either by the CPU using this register for programmed I/O, or by means of the DMA controller. In the latter case, the SCSI controller communicates its need to send or receive another 8 bits by means of an signal called DREQ, which is connected to one of the DMA controller's channels. That is, the SCSI chip is "implicitly addressed" during DMA.

In DMA mode, an interrupt will be generated when the transfer counter decrements to zero. In non-DMA mode, an interrupt can be generated after each byte transferred, if interrupts are enabled. The interrupt bit appears in a register visible to the CPU.

Ethernet Controller (AMD7990)

The Ethernet controller (LANCE) functions primarily as a bus master. Its master bus cycles are converted into Sbus master cycles by the DMA controller. The CPU initializes the LANCE with a pointer to a control block in main memory, from which the LANCE reads its initialization information. The CPU and the LANCE communicate by means of shared main memory and semaphores, using circular buffer descriptor data structures for the transmit and recieve buffers.

Graphics

The graphics on the SUN compatible would be supported by a Frame Buffer on the Sbus. All offset addresses need to be added to the slot base address to get the effective address.

Frame Buffer

The data organization for Frame buffer is as follows. Each byte of video data corresponds to a display pixel. A color map described translates the byte of Video data into the display pixel. The high order byte in Color Video RAM maps to the first visible pixel in the upper left corner of the display monitor. Consecutive bytes are displayed as consecutive pixels along the horizontal scan line, left to right. After 1152 pixels are displayed on one scan line the next pixel is displayed on the next scan line. A total of 900 such lines are displayed per frame at 66 such frames per second.

Color Video RAM

The Color Video RAM is located in a dedicated area of system memory space and is dual ported; one for video refresh, and the second for processor access. The size of the Color Video RAM is 1 MByte, and is organized as an array of 256K x 32. The Color Video RAM is updated by reading and writing it directly, like memory.

Address Mapping:

Device offset	Device	Size	Type	Sbus Transaction Size
0 0030 0000	Color Video RAM	32-bit	Read-Write	Byte, halfword,word

Color Map

The Color Map maps bytes in Color Video RAM into pixels. It contains three 256 Byte sections :one each, for red, green and blue. Besides the Color Map, several registers exist in this section. The sub-space address of the internal registers and RAM space has to be loaded into the address register before any meaningful transaction can be done in this section.

To read color data, the SPARC loads the address register with the address of the color palette RAM location to be read. The SPARC then performs three successive read cycles (red, green, and blue) to access the data from the color palette. Following the blue read cycle the address register increments to the next location which the SPARC may read by simply reading another sequence of red, green and blue data from the Color palette. A write to the color palette is done exactly the way a read is done, except that the data bytes of red, green, and blue are written. When accessing the color palette RAM, the address register resets to $00 after the blue, read or write cycle to location $FF.

Device Offset	Device	Sub-Space	Size	Type
0 0010 0000	Address reg	$xx	byte	Read-Write
0 0010 0001	Color palette	$00-$FF	byte	Read_Write
0 0010 0002	Read Mask reg	$04	byte	Read-Write
0 0010 0002	Blink Mask reg	$05	byte	Read-Write
0 0010 0002	Command reg	$06	byte	Read-Write
0 0010 0002	Cntrl/Tst reg	$07	byte	Read-Write

Refer to Brooktree part Bt458 data sheet for details on the internal registers and also about programming the Color Map. It is required that bit CR6 in command register be set to a 1. Bit CR7 should be set to 0 to reflect a 4:1 multiplexing of pixels in hardware.Set bits 0 and 1 ,viz CR0 and CR1 to 0. The rest of the command register bits is left to the discretion of the software programmer.

The Color Map is generally written during vertical retrace. Following the assertion of Level 5 interrupt for the Color Map, the map update must complete in 600 microSeconds (the vertical retrace time) to avoid being visible in the display. Longer updates will complete but the display appearance may be affected.

References: Brooktree Bt458 data sheet.

Interrupt Register

At the beginning of a vertical retrace the video interrupt is set. This is a level 5 interrupt on theSbus. The interrupt is kept pending until the software clears it.The only relevant bits are as shown below. At the time of power-up interrupt is disabled and need to be explicitly enabled by software by writing a '1' to the ENB bit. Note that the INT pending bit is writable by software and this can aid in diagnostics as well.

Device Offset	Device	Size	Type
0 0020 0000	IntReg	Byte	Read-Write

1	1	6
INT	ENB	not relevent
7	6	5 0

D Prom

As per the SBus specification an IDPROM is located at the base address of this slot. It is a 8Kx8 device and carries the relevant FCode.

Address	Device	Size	Type
0 0000 0000	IDProm	Byte	Read

Reserved

0 0040 0000 to 0 00F0 0000 reserved

Mbus Arbitration for SPARC Clone

The Mbus arbiter is a separate unit from both the Mbus slaves and masters. It can arbitrate among 4 Mbus masters and incorportates both a linear and a rotating priority algorithm. One bus request will have the highest priority. This can be used to minimize latency for an Ethernet Controller device. The other three bus requests have a rotating priority. Arbitration is overlapped with the current bus cycle. Bus parking is employed which means that the current master retains ownership of the bus until it is taken away by a request from another master. Back-to-back transactions by different masters are not allowed. There must be at least one dead cycle in between each transaction during which the bus busy signal is deasserted.

Each Mbus master has dedicated bus request and bus grant signals. The master requests the Mbus by asserting its bus request signal. Upon receiving a bus grant from the bus arbiter, the requesting Master can start using the bus by asserting the bus busy signal as soon as the previous masters deasserts it.

The Mbus arbiter looks at the bus requests from the 4 Mbus masters and the bus busy signal and generates the bus grants for each master. Only one bus grant is asserted at any time.

Physical Address Map

The physical address map is allocated to minimize hardware decoding for the different devices in the compatible system.

Physical Address Assignments

Mbus

Physical Base Address	Device	Physical Address Space
0x 0 0000 0000	Main Memory	A[p:0]
0x C 0000 0000	reserved	
0x D 0000 0000	reserved	
0x E 0000 0000	SBus	
0x F 0000 0000	AT	
	Mbus Configuration Address Map	
0x F F000 0000	(Mbus ID =0x0)	Boot PROM
0x F F100 0000	(Mbus ID =0x1)	
0x F F200 0000	(Mbus ID =0x2)	
0x F F300 0000	(Mbus ID =0x3)	
0x F F400 0000	(Mbus ID =0x4)	
0x F F500 0000	(Mbus ID =0x5)	
0x F F600 0000	(Mbus ID =0x6)	
0x F F700 0000	(Mbus ID =0x7)	
0x F F800 0000	(Mbus ID =0x8)	
0x F F900 0000	(Mbus ID =0x9)	
0x F FA00 0000	(Mbus ID =0xA)	
0x F FB00 0000	(Mbus ID =0xB)	
0x F FC00 0000	(Mbus ID =0xC)	
0x F FD00 0000	(Mbus ID =0xD)	
0x F FE00 0000	(Mbus ID =0xE)	
0x F FF00 0000	(Mbus ID =0xF)	

Memory - upto 64 Mbytes

0x 0 0000 0000 to 0x 0 03FF FFFF

SBus

Physical Base Address	Device
0x E 0000 0000	M2S SBus Slave Interface
0x E 1000 0000	SBus Slot1 - SBus Connector 1
0x E 2000 0000	SBus Slot2 - SBus Connector 2
0x E 3000 0000	SBus DMA Controller
0x E 4000 0000 to 0x E F000 0000 reserved	

BOOT_PROM

Physical Base Address	Physical Address Space
0x F F000 0000	A[17:0]

ON_- BOARD IO AND AT

Physical Base Address	Device	Physical Address Space
0x F 0000 0000	Boot Prom	A[17:0]
0x F 0100 0000	Serial Ports	A[1:0]
0x F 0200 0000	Keyboard, Mouse	A[1:0]
0x F 0300 0000	Floppy Disk Controller	A[2:0]
0x F 0400 0000	Real Time Clock/EEPROM	A[10:0]
0x F 0500 0000	ISDN	A[2:0]
0x F 0600 0000	Interrupt Register,Timers	A[4:0]
0x F 0700 0000	reserved	
0x F 110A 0000	AT Memory	A[23:0]
0x F 1000 0000	AT IO	A[9:0]
0x F 2000 0000 to 0x F F000 0000 reserved		

Interrupt Assignments

The Mbus allows for 16 levels of interrupts. The interrupt sources are the following, with 15=highest priority, 0= lowest.

15 - Non-Maskable Interrupts (NMI)
14 - Clock (Monitor, Profiler)
13 - Audio, AT Bus level 3
12 - Keyboard, mouse, serial ports, AT Bus level 4
11 - Floppy Disk, AT Bus level 6
10 - Clock (Timer), AT Bus level 7
9 - SBus level 7, AT Bus level 9
8 - SBus level 6, AT Bus level 10
7 - Video, SBus level 5
6 - Software Interrupt Level 6
5 - Ethernet, SBus level 4, AT Bus level 11
4 - Software Interrupt Level 4
3 - SCSI, SBus level 3, AT Bus level 12
2 - SBus level 2, AT Bus level 14
1 - Software Interrupt Level 1~~, AT Bus level 15~~
0 - No Interrupt

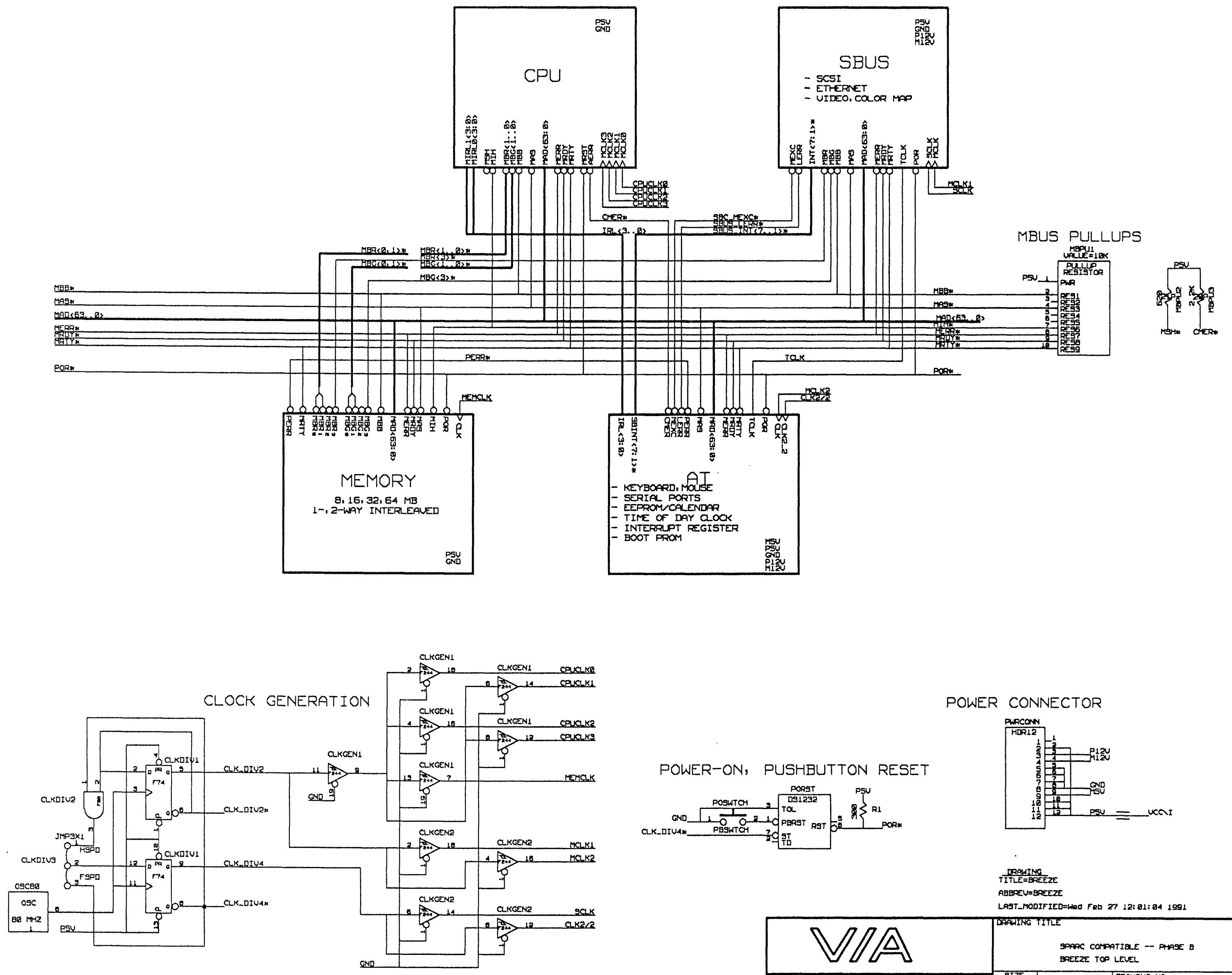
Non-Maskable Interrupts include:

1. Push-Button Switch
2. Memory Parity Error
3. CPU Memory Error (Asynchronous Error on Memory Access)
4. SBus Controller Memory Exception
5. SBus Late Error

Interrupts are auto-vectored. That is, the address of the interrupt service routine is formed from a trap base register plus an index formed from the level of the interrupt.

REVISIONS

ZONE	LTR	DESCRIPTION	DATE	APPR.



DRAWING
TITLE=BREEZE
ABBREV=BREEZE
LAST_MODIFIED=Wed Feb 27 12:01:04 1991

VIA

DRAWING TITLE
SPARC COMPATIBLE -- PHASE B
BREEZE TOP LEVEL

SIZE C | REV. | DRAWING NO. BREEZE
SCALE | SHEET 1 OF 1

status of m2m chip:

DESIGN: Partially done.
MIH signal is not being handled correctly.
Parity error checking on memory reads is not being done.

SIMULATION: Functionally about 90% simulated.
The arbiter circuitry not simulated.
Diagnostics : Not Done.

BUGS: Several relating to IO. For example, the perr signal should be open-collector, the mrty pin should be tristate etc.
Initial values of registers.

SYNOPSYS: Mapped to Fujitsu library (0.8um).
The arbiter circuitry not compiled yet.

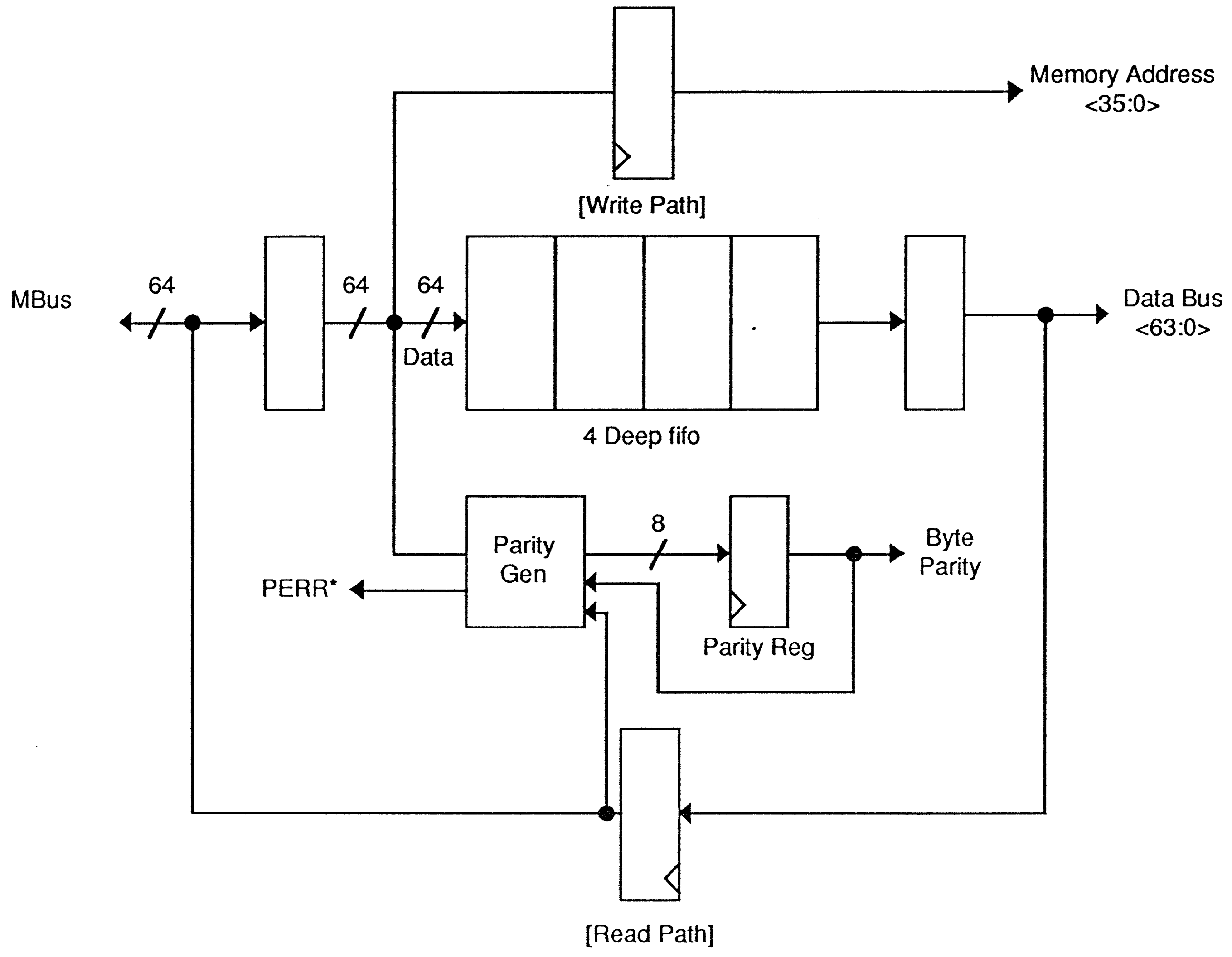
LOCATION OF FILES:

```
      verilog hdl desc. : ~biyani/pc/m2m/v10.0/vlg
      verilog library   : /tools/lib/model/hdl, Fujitsu libraries
      synopsys environment : ~biyani/pc/m2m/v3.0
      simulation environment : /s/users/sparc/breeze/memory
```

General Description

This ASIC connects the MBus to the memory controller. Ideally, the interface logic and the memory controller should be in the same ASIC, but it results in a very high pin count ASIC. Also, the combined ASIC would need external drivers to drive the address, RAS and CAS lines. Thus, the need to have two ASIC's, the memory interface ASIC and the memory controller ASIC.

The function of the Memory Interface logic is fairly straightforward and is shown below.



VIA Technologies, Inc. (408) 746-2200
860 East Arques Avenue
Sunnyvale, CA 94086

What makes the chip complicated is that the Memory Interface Logic has to be sliced into two chips. The slicing becomes a little tricky as the address is available on only the lower 36 bits. Another complication is the design of the dual-ported FIFO , simultaneous Read and Write should be done on the 4-deep fifo. This would significantly improve the performance of the memory system.

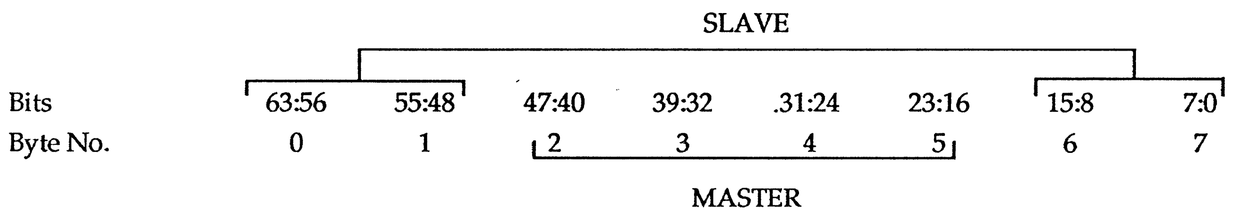
Objectives for ASIC

This ASIC designed for memory interface must achieve the following objectives:

- Only one load on MBus
- Fit in a 128 PQFP
- Run up 40MHz
- The filling and emptying of the on board buffers can take place simultaneously.
- The ASIC should support all burst modes.

Slicing Methodology

Byte Defintion



Even though the two slices are identical, I define them as the MASTER and SLAVE. This would become clear later in the SPEC. The slicing is done as follows:

MASTER Slice would receive bytes 2,3,4,5 from MBus

SLAVE Slice would receive bytes 0,1,6,7 from MBus.

The advantage of this scheme is that the 36 bits of physical MBus address is split 16 bit in SLAVE and 20 bits in MASTER. Further, the Type (MBus <39:36>) and Size (MBus <42:40>) will originate for the memory controller from the 'MASTER SLICE'.

Internal Registers

Configuration/Parity Error Register

Address 0XFFnXXXXX8

(This is a Read/Write Register)

31 28	27 24	23	22	21	20	19	18	17 9	8 0
PERR	PI	PE	I	MS	CS	S	R	Upper_Bound	Lower_Bound

This register would be read from the 'MASTER Slice', using double word read.

Upper-bound = highest address supported by DRAM controller.

Lower-bound = lowest address supported by DRAM controller.

The Memory Interface would respond if Lower-bound < Target Address (MBus < 31:23>) < Upper-bound and MBus <35:32> is OX) in the address phase of MBus operation.

R	RAM Size
S	Spare
CS	Clock Speed
MS	Memory Speed
I	Interleave
PE	Parity Enable (4 bytes)
PI	Parity Invert <bytes 2,3,4,5> (per byte)
PERR	Parity Error <bytes 2,3,4,5> (per byte)

MBus Port Register

Address OXFFnXXXXXC

This is a Read/Write register. This would be read from the SLAVE slice using the double word READ. The double word Read/Write at address OXFFnXXXXX8 would access this register from the SLAVE. This register is accessed from SLAVE.

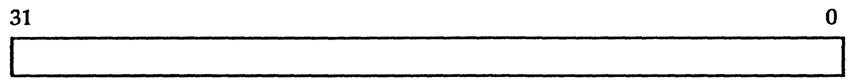
31 28	27 24	23	22 16	15 8	7 4	3 0
PERR	PI	PE	SPACE	MDEV	MREV	MVEND

MVEND	Vendor ID
MREV	Rev # of device
MDEV	Device #
PE	Parity Enable
PI	Parity Invert <bytes 0,1,6,7> (per byte)
PERR	Parity Error <bytes 0,1,6,7> (per byte)

Bus Error Register I

Address OXFFnXXXXX0

This again is a 32 Read/~~Write register.~~ This can be accessed by double word Read or Write. This register contains a copy of the address from MAS cycle for operation. This is accessed from MASTER.



Bus Error Register II

Address OXFFnXXXXX4

This register is identical to Bus Error Register I except its address. This is accessed from the SLAVE. The Read/Write access to this register is done by doing a double word access on address OXFFnXXXXX0.

Pinouts of Memory Interface

Count	Pin Name	Type	Description
32	MAD <31:0>	I/O	MBus multiplexed 32 bit address data
32	MD <31:0>	I/O	Memory Data Bus
16	MA <15:0>	O	Memory Address Bus
4	PARITY <3:0>	I/O	Byte Parity
1	MASTER/SLAVE	I	Defines physical location of chip
1	C-Mrdy*		I Handshare signal from memory controller. Tells interface controller done.
1	TYPE	I/O	This is a Multifunction pin. In the MASTER it is an output pin describing the type of operation as Read/Write. If SLAVE it has the same meaning as above, but is an input.
3	SIZE <2:0>	I/O	this is a multifunction pin. In the MASTER it drives the size information to the memory controller. In SLAVE it is an input pin and receives the size information.
1	MAS*	I	Memory Address Strobe on MBUS
4	ID <3:0>	I	For MASTER these 4 pins define the MBus slot ID through some internal jumper setting. These pins have no meaning for SLAVE.
1	REGIN/REGOUT	I/O	This is a multifunction pin. For the MASTER slice this pin is an output indicating a register operation was decoded. For SLAVE this is an input pin indicating a register operation.
1	REGADDRO /REGADDRI	I/O	This is a multifunction pin. For the MASTER slice this is an input pin with the dw address of the register set.
1	MRDY*	O	In the MASTER slice it drives the MBus MRDY*

			signal. In slave slice is unconnected.
1	MERR*	O	This is a multifunction pin. For the MASTER slice this is the MERR* signal for the MBus*. For the slave this is nc.
1	AERR*	O	This is an open collector signal and is the PERR* from each slice.
1	MREQ*/MOP*	I/O	This is a multifunction bit. In MASTER it drives 'MREQ' signal to memory controller. In SLAVE it is an input indicating a memory operation.
1	MEMBSY*	I	This is an input indicating that the memory is busy. This signal comes from the memory controller.
1	CLK	I	Clock Line
1	RESET*	I	Reset input
4	CONFIG <3:0>	O	In the MASTER this feeds the configuration information to the memory controller. In SLAVE it is NC.
109			

Software Format of Configuration Registers

To have a thorough idea of qhat a double word register access would give to the processor it is better to see it pictorially. this is because the bytes sent to each of the slices and not contiguous.

DW access on location address OXFFnXXXXX0

63 48	47 16	15 0
BUS ERR REG II <31:16>	BUS ERR REG I <31:0>	BUS ERR II <15:0>

Byte No. 0 1 2 3 4 5 6 7

DW access on location on register address OXFFnXXXXX8

63 60	59 56	56	54 48	47 44	43 40	39	38	37	36	35	34	33 25	24 16	15 8	7 4	3 0
PERR <0,1,6,7>	PI <0,1,6,7>	PE <0,1,6,7>	SPARE	PERR [2,3,4,5]	PI [2,3,4,5]	PE [2,3,4,5]	I	MS	CS	SPARE	R	UPPER BND	LOWER BND	MDEV	MREV	MVEND

PERR <0,1,6,7>	Indicates Parity error for bytes 0,1,6,7
PERR <2,3,4,5>	Indicates Parity error for bytes 2,3,4,5
PI <0,1,6,7>	Indicates Parity Invert for bytes 0,1,6,7
PI <2,3,4,5>	Indicates Parity Invert for bytes 2,3,4,5
PE <0,1,6,7>	For enabling parity for bytes 0,1,6,7
PE <2,3,4,5>	For enabling parity for bytes 2,3,4,5

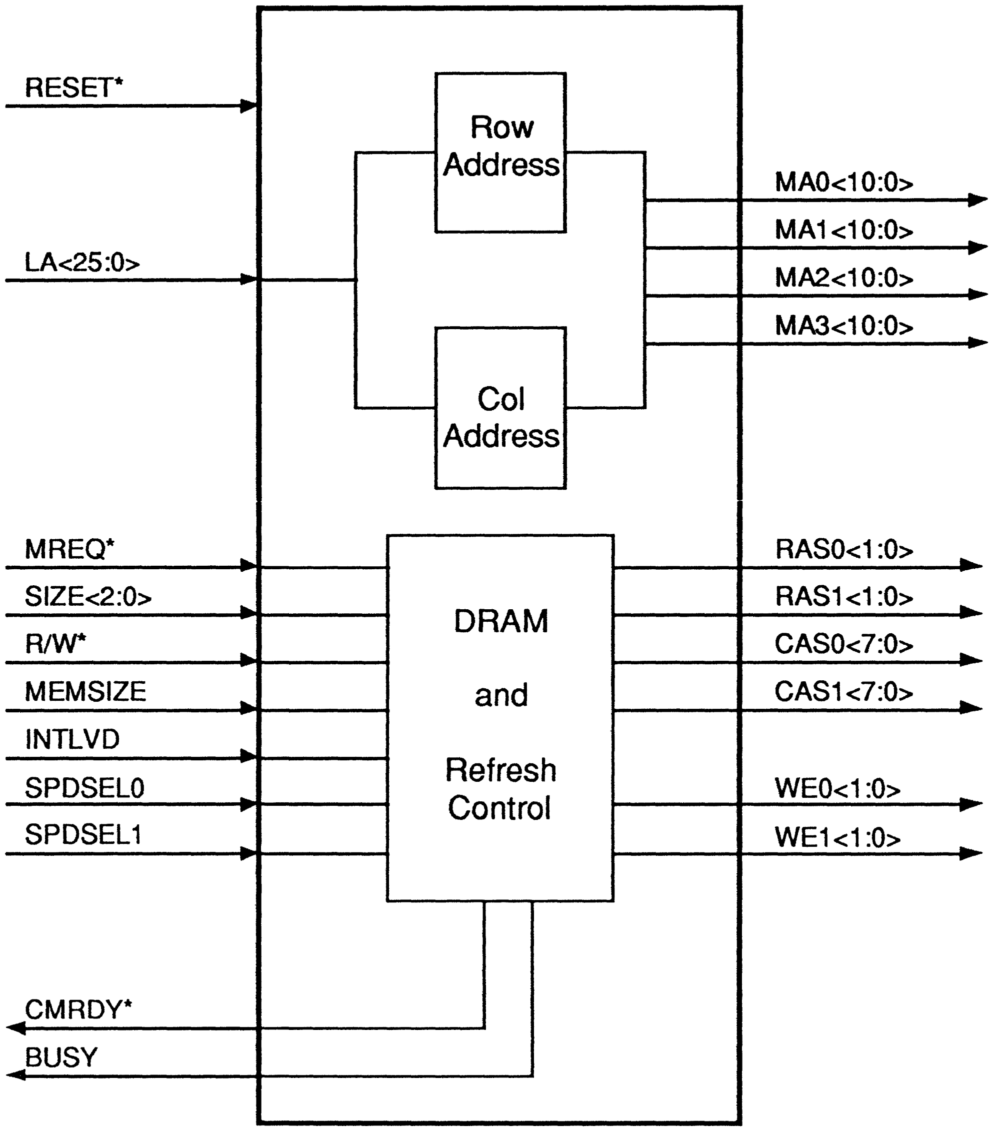
```
/s/users/sparc/breeze/sim/v

status of DRMC chip:

  DESIGN : complete

  SIMULATION :test suite in /s/users/sparc/breeze/sim/v/{debug.v,mem.diag6,rgrs}
  testing of rgrs is not complete and could yield unknown problems.

  BUGS: none outstanding.

  SYNOPSYS: mapped to att.

  LOCATION of files:
    verilog hdl description of chip:
            /s/users/sparc/breeze/memory/v/tst/{memctrl.v,rascas.v,rowcol.v,
             coladr.v,maddr.v,maxadr.v,refcntr.v,enable.v}
    simualtion environment: /s/users/sparc/breeze/sim/v
```

MBus DRAM Controller Specification

'ecember, 1990
.011691

VIA Technologies, Inc. (408) 746-2200
860 East Arques Avenue
Sunnyvale, CA 94086

Block Diagram



DRAM Controller Block Diagram

Overview

The DRAM controller is a high-performance CMOS integrated circuit that provides all the necessary control signals between the DRAM array and a pair of MBUS interface ICs in a SPARC based workstation. The rest of the document describes the functions supported in this IC.

Features:

- 2-way Interleaving for High Performance.
- 1 to 128 byte DRAM Read, Write Transaction Using Fast Page Mode.
- Byte Wide Parity.
- CAS Before RAS Refresh Scheme.
- Supports 1 Megabyte and 4 Megabyte RAM Modules.
- Memory Configurations Supported:
  - 8 Mbytes - 1M x 9 SIMM DRAM's even bank non-interleved.
  - 16 Mbytes - 1M x 9 SIMM DRAM's both banks and interleaved.
  - 32 Mbytes - 4M x 9 SIMM DRAM's even bank non-interleaved.
  - 64 Mbytes - 4M x 9 SIMM DRAM's both banks and interleaved.
  - Clock Speed - 25 Mhz, 33 MHz and 40 MHz.
- No External Buffers Needed for RAS*, CAS*, WE* and Memory Address For 64 Mbytes of DRAM.
- Needs a Pair of VIA MBus Interface IC's or Equivalent For Optimum Performance on MBus.
- Built in Scan Chain for 100% Fault Grading.

PINOUT

Signal Name	No. of Pins	Input/Output	Signal Description
LA<25:0>	26	Input	Latched 26 bit memory address
SIZE<2:0>	3	Input	Transaction size
RD/WR*	1	Input	Rd if =1 else wr
MREQ*	1	Input	Request for memory operation
MEMSIZE	1	Input	1=4M Byte ,0=1M Byte sel
SPDSEL0	1	Input	Wait state generator bit 0
SPDSEL1	1	Input	Wait state generator bit 1
INTLVD	1	Input	Turns on 2-way interleaving
CMRDY*	1	Output	Operation done
BUSY	1	Output	Controller Busy
MA0<10:0>	11	Output (12 ma)	DRAM address copy 0
MA1<10:0>	11	Output (12 ma)	DRAM address copy 1
MA2<10:0>	11	Output (12 ma)	DRAM address copy 2
MA3<10:0>	11	Output (12 ma)	DRAM address copy 3
RAS0<1:0>*	2	Output (12 ma)	Even DRAM RAS
RAS1<1:0>*	2	Output (12 ma)	Odd DRAM RAS
CAS0<7:0>*	8	Output (12 ma)	Even DRAM CAS
CAS1<7:0>*	8	Output (12 ma)	Odd DRAM CAS
WE0<1:0>*	2	Output (12 ma)	Even DRAM WE
WE1<1:0>*	2	Output (12 ma)	Odd DRAM WE
RESET*	1	Input	System Reset
VDD	4		Power
VSS	4		Ground
SCAN_OUT	1	Output(4ma)	Scan Data Out
SCAN_IN	1	Input	Scan Data In
TM_OE*	1	Input	Test Mode/Output Enable

Pinout Summary:	107	Signal I/O
	4	VDD
	4	VSS

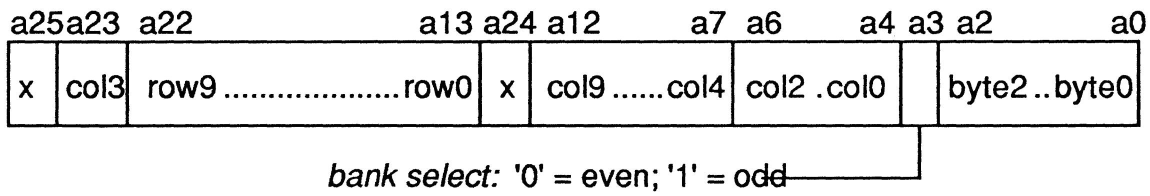
	115	Total Pins

Functional Description

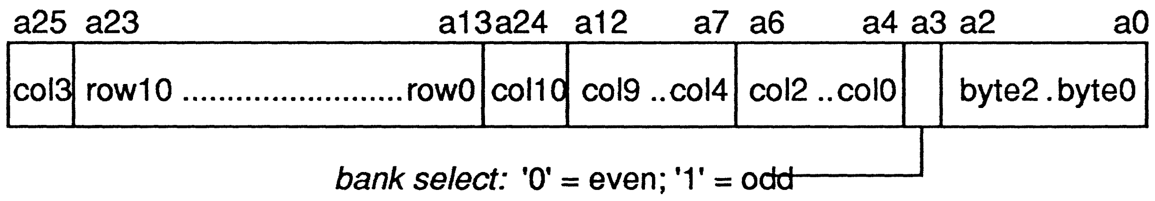
The DRAMC is initiated by MREQ* signal being active for a cycle. At which time if the DRAMC is not busy doing either a memory operation or refreshing the DRAMs a new memory operation is started on the DRAMs. Depending on the information on the MEM_SIZE and INTLVD pins, the address on the LA<25:0> are muxed onto MAn<10:0> as row and column addresses at the appropriate time intervals. Again one address bit is used to select the bank. The address muxing is as shown below

Interleaved memory mapping:

1 MegaByte SIMM

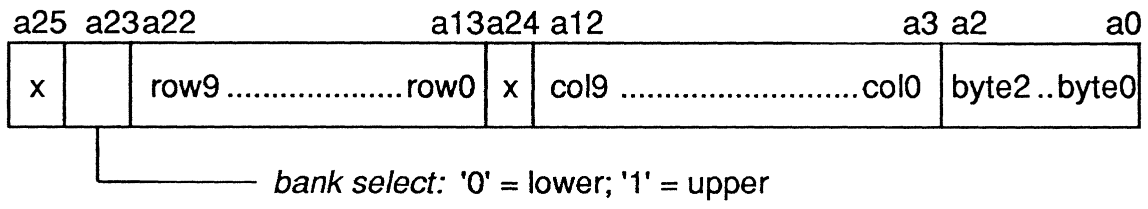

a25 a23 a22 a13 a24 a12 a7 a6 a4 a3 a2 a0
x
col3
row9 .... row0
x
col9 ...... col4
col2 . col0
byte2 .. byte0
bank select: '0' = even; '1' = odd

4 MegaByte SIMM

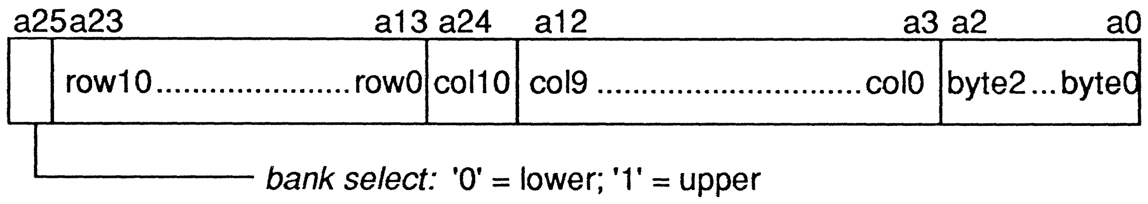

a25 a23 a13 a24 a12 a7 a6 a4 a3 a2 a0
col3
row10 .... row0
col10
col9 .. col4
col2 .. col0
byte2 . byte0
bank select: '0' = even; '1' = odd

Non-Interleaved memory mapping:

1 MegaByte SIMM


a25 a23 a22 a13 a24 a12 a3 a2 a0
x
row9 .... row0
x
col9 .... col0
byte2 .. byte0
bank select: '0' = lower; '1' = upper

4 MegaByte SIMM


a25 a23 a13 a24 a12 a3 a2 a0
row10 .... row0
col10
col9 .... col0
byte2 ... byte0
bank select: '0' = lower; '1' = upper

CAS WIDTH SELECT:

The DRAMC is optimized for system clock speeds of 25Mhz to 40 MHz and DRAM access times of 100 ns or better for the fast page mode operation. The deafult count for RAS_to_CAS delay,*viz* RCD is 2 cycles and the default CAS* cycle time is 1 cycle. However additional cycles can be added by setting the SPDSEL0 and SPDSEL1 pins to appropriate levels as shown below.

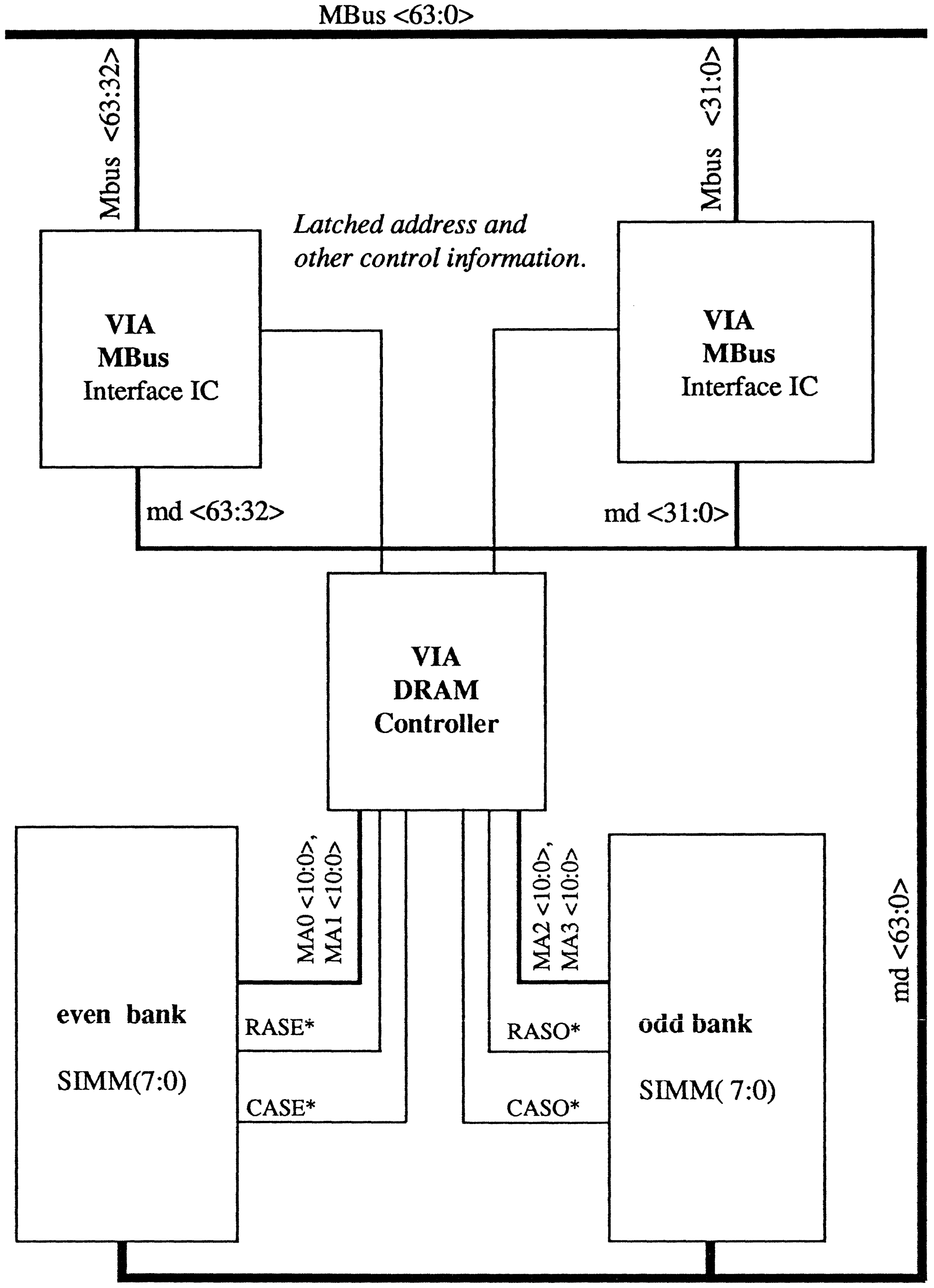
spdsel1	spdsel0	CAS*	RCD
0	0	----	----
0	1	----	+1
1	0	----	+1
1	1	+1	+1

TRANSACTION SIZE:

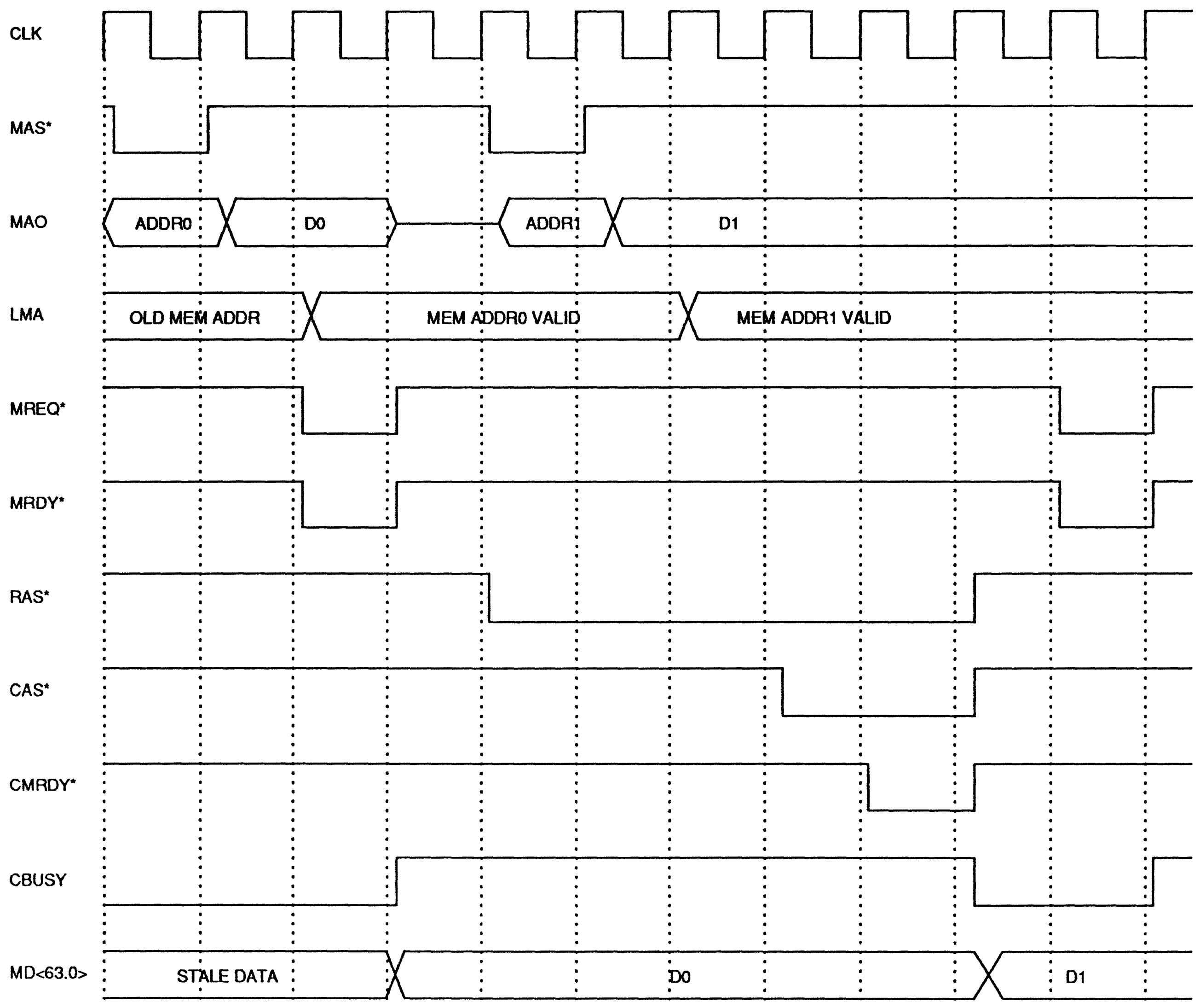
The SIZE<2:0> lines carry the encoded version of the transaction size as shown in the table below.

SIZE<2:0>	Transaction
000	Byte
001	2 Bytes
010	4 Bytes
011	8 Bytes
100	16 Bytes
101	32 Bytes
110	64 Bytes
111	128 Bytes

Typical Memory Organization



Typical M2M and DRAMC Interface Timing



```
/s/users/sparc/breeze/at/STATUS

status of M2SX chip:


    DESIGN: complete

    SIMULATION: test suite in /s/users/sparc/breeze/at/{main.v,test.v}
    additional testing which could be done: half speed mode

    BUGS: none outstanding

    SYNOPSYS: mapped to att (13571 cells, minimum cycle time 16.48 ns),vti

    LOCATIONS of files:

        verilog hdl description of chip: /s/users/sparc/breeze/at/synopsys/pure_hdl
        verilog library:           /tools/lib/model/hdl
        synopsys environment : /s/users/sparc/breeze/at/synopsys
        simulation environment : /s/users/sparc/breeze/at


status of AT subsection:


    DESIGN: complete

    SIMULATION: test suite in /s/users/sparc/breeze/at/test.v
    additional testing which could be done: refresh,dma, off-board master

    BUGS: none outstanding

    LOCATIONS of files:

        drawings:/s/users/sparc/breeze/at/at
        simulation environment:/s/users/sparc/breeze/at
        verilog libraries:           /tools/lib/model
                                       /tools/lib/lai_vlog
                                       /tools/lib/v
                                       /tools/lib/oki
                                       /s/users/sparc/breeze/v/breezeCustom.v
                                       /s/users/sparc/breeze/v/breezeLaiInt.v
                                       /s/users/sparc/breeze/at/synopsys/pure_hdl
```

M2SX Chip Specification

Rev 1.0

ebruary, 1991
1.020591

VIA Technologies, Inc. (408) 746-2200
860 East Arques Avenue
Sunnyvale, CA 94086

Introduction

The M2SX chip provides a means by which Mbus slave accesses are transformed into accesses in 386SX protocol. That is, the Mbus interface of the chip acts as an Mbus slave, while the 386SX side acts as a master. Then, other logic can translate the 386 master cycles into bus cycles of a standard system bus, such as the AT.

Another function of the M2SX chip is to handle accesses to basic on-board devices, such as the boot prom and serial ports. These do not proceed as 386SX cycles, but do use the 386 address and data busses. No additional "glue" logic is necessary to connect these to the M2SX.

Mbus Slave Interface

Mbus is a multiplexed address and data bus. During the address phase the Mbus master drives the physical address (PA), transaction type (TYPE) and size (SIZE), and other control signals. The M2SX chip compares PA<35:32> with its Mbus Slot Address, MSLOT, to determine if whether it is the target of the transaction. MSLOT is a field in its internal Status/Control Register. The internal registers themselves are addressed when PA<35:28>=FF hex and PA<27:24> equals the value on the ID<3:0> inputs. Finally, physical addresses of FF0xxxxxx will result in boot PROM accesses.This address comparison is summarized below:

if PA<35:0> = FFnxxxx00 , where n = ID<3:0>	-> Internal Registers
else if PA<35:0> = FF0xxxxxx	-> Boot Prom space
else if PA<35:32> = MSLOT<3:0>	-> on-board I/O and AT space

Internal Registers

offset(bytes)	description	access type	size(bits)
0	Bus Error Register	Read	64
8	Status/Control Register	read/write	32
C	Mbus Port Register	read	32
10	Latency Register	read/write	64
18	Recovery Register	read/write	64

AT/ On Board IO Address Mapping

Mbus address bit pattern								
31	30	29	28	27	26	25	24	
0	0	0	1	X	X	X	1	-> AT memory space
0	0	0	1	X	X	X	0	-> AT I/O space
0	0	0	0	X	0	0	0	-> on-board I/O device #0 (boot prom)
0	0	0	0	X	0	0	1	-> on-board I/O device #1
0	0	0	0	X	0	1	0	-> on-board I/O device #2
0	0	0	0	X	0	1	1	-> on-board I/O device #3
0	0	0	0	X	1	0	0	-> on-board I/O device #4
0	0	0	0	X	1	0	1	-> on-board I/O device #5
0	0	0	0	X	1	1	0	-> on-board I/O device #6
0	0	0	0	X	1	1	1	-> on-board I/O device #7

The M2SX chip will accept two transaction types: Read (TYPE<3:0>=0001) and Write (TYPE<3:0>=0000). All other transaction types are not supported.

TYPE<3:0>	Transaction Type
0000	Write
0001	Read
001x	Illegal
01xx	Illegal
1xxx	Illegal

Transaction Sizes Supported

For on-board I/O and AT accesses, all Mbus size transactions are supported, that is byte, halfword, word, doubleword, and bursts of 16,32,64 and 128 bytes.

For the internal registers, byte, halfword, word and doubleword reads are supported. For writes, only word and doubleword sizes are allowed. Unsupported size transactions will result in a bus error acknowledgement being given.This is summarized in the following table:

Allowed internal register accesses

size	type	
	read	write
byte	OK	illegal
short	OK	illegal
word	OK	OK
double	OK	OK
burst	illegal	illegal

There are three transaction status bits, MERR*, MRDY*, and MRTY*, that the M2SX chip uses to send an acknowledgment back to the Mbus master for a transaction it receives. The transaction status is shown as follows:

MERR*	MRDY*	MRTY*	Acknowledgment
1	1	1	Idle Cycle
1	1	0	Relinquish and Retry
1	0	1	Valid Data Transfer
0	1	1	Error1 => Bus Error

Valid Data Transfer. On Mbus Write transactions, the M2SX chip will assert MRDY* when it senses the 386 READY* signal becoming active, which signals that the 386 peripheral has accepted the write data. For Mbus Read transactions, MRDY* is asserted when the read data is driven by the M2SX onto Mbus.

Bus Error Acknowledgment. The M2SX will assert MERR* when it detects an unsupported operation for an Mbus transaction which has targetted it. An unsupported operation is defined as one for which the TYPE and/or SIZE is not supported, as defined above.

Relinquish and Retry Acknowledgment. The M2SX chip will give a Relinquish and Retry acknowledgment (R&R) if it is the target of an Mbus transaction, but the 386 bus is not immediately available because another master is controlling from the 386 side.

Mbus Port Register

The Mbus Port Register (MPR) is a read only register which contains the Implementation Number and Version Number of the M2SX Chip. It is addressed when PA<35:0>=FFnxxxxxC where n=ID<3:0>. The format of the MPR is shown below:

16	8	4	4
Reserved	MDEV	MREV	MVEND
31 16	15 8	7 4	3 0

MDEV: Mbus Device Number. This field contains a unique number which identifies the device as a VIA M2SX Chip.

MREV: Mbus Revision Number. This field contains the revision number for the VIA M2SX Chip.

MVEND: Mbus Vendor Number. This field contains the vendor number for VIA.

Status/Control Register

The Status/Control Register (SCR) is addressed when PA<35:0>=FFnxxxxx8 where n=ID<3:0>. The format of the SCR is shown below:

27	1	4
Reserved	ALT	MSLOT
31 5	4	3 0

MSLOT: Mbus Slot Address (read/write, bits 3:0). This field contains a unique number among all Mbus modules which reside on Mbus. MSLOT is compared with PA<35:32> to determine whether this chip is the target of the transaction. This register should be configured by the Operating System after it has read all of the MPRs for the modules which reside on Mbus. The Mslot value is set to value 0 upon reset.

ALT_MASTER (read only, bit 4). This status bit being high indicates that another bus master has control of the 386SX bus.

Bus Error Register

The Bus Error Register is a read only register which contains information about the circumstances that caused the M2SX Chip to assert MERR*. It is addressed when PA<35:0>= FFnxxxxx0 where n=MID<3:0>. MERR* is asserted when the Mbus Slave interface face receives a transaction which it does not support. The physical address, size, type, and acknowledgment for the transaction is saved in the Bus Error Register. The format of the Bus Error Register is shown below:

MID: Module Identifier. This field contains a copy of the Mbus Module Identifier of the Mbus master which was the source of the transaction which led to the bus error.

OCC: Error Occurred. This bit = 1 if a bus error has occurred since the Bus Error register was last read. It is cleared upon reading this register. OCC is initially = 0 upon system reset. If a bus error occurs, the bit is set and the register's contents are updated. The Bus Error Register's contents are then held until it is read by the CPU, at which point it is re-armed again.

4	20	1	3	4	36
MID	Reserved	OCC	SIZE	TYPE	PA
63 60	59 44	43	42 40	39 36	35 0

SIZE: Mbus Transaction Size. This field contains a copy of the Mbus transaction size.

TYPE: Mbus Transaction Type. This field contains a copy of the Mbus transaction type.

PA: Physical Address. This field contains a copy of the physical address for the Mbus transaction.

Latency Register

This 64-bit register is used with on-board I/O (OBIO) accesses. There is an eight-bit value for each of theeight possible OBIO devices. This value is programmable by the CPU and has a default value of 0xA. The ordering is such that the value for OBIO device 0 (boot prom) comes from register bits [63:56], while the value for device 7 comes from register bits [7:0]. The latency value set for a device determines the number of additional clock cycles that the read or write strobe is active for an I/O access to that device. The total number can vary between 1 and 256 clock cycles. For example, consider a system with a 25ns Mbus cycle time, and a read of the boot prom in which the latency register bits[63:56] = 0x8.The read strobe will be active for (1+ 8)x25ns=225ns.

LAT0	LAT1	LAT2	LAT3	LAT4	LAT5	LAT6	LAT7
63							0

Recovery Register

This 64-bit register is also used with on-board I/O (OBIO) accesses. There is an eight-bit value for each of the eight possible OBIO devices. This value is programmable by the CPU and has a default value of 0x1. The ordering is such that the value for OBIO device 0 (boot prom) comes from register bits [63:56], while the value for device 7 comes from register bits [7:0]. The recovery value set for a device determines the number of additional clock cycles between the time that the chip select becomes active for that device and the time that the read or write strobe becomes active. This serves two functions: 1) to ensure enough address set-up time before the read or write strobe becomes active and 2) to provide enough recovery time (time between successive read or write strobes to the same device.) The total number can vary between 1 and 256 clock cycles. For example, consider a system with a 25ns Mbus cycle time, and a write of the serial port in which the latency register bits = 0x1f.The recovery register will provide (1+ 31)x25ns=800ns between when the chip select for the serial port becomes active and when the write strobe becomes active.

REC0	REC1	REC2	REC3	REC4	REC5	REC6	REC7
63							0

386 memory and I/O spaces

The 386 processor has a concept of memory and I/O address spaces. The cycles are the same except for the state of the MEMIO bit, which is high for memory cycles and low for I/O cycles. This function is simulated by the M2SX chip by making the state of PA<24> during each 386 access determine whether a memory or I/O cycle is performed. Setting PA<24> = 1 causes a memory access, PA<24> = 0 causes an I/ O access.

"BIG-ENDIAN" VS. "LITTLE-ENDIAN" CONVERSION

The Mbus and the 386 are opposites with respect to the "big-endian" vs. "little-endian" convention. Because of this, for byte accesses, the least significant bit of the Mbus address is complemented before being put out as a 386 address.

Low-order addresses formed by 386sx sequencer		
Mbus cycle size	initial byte address from Mbus	byte address sequence
doubleword	0	0,2,4,6
word	0	0,2
word	4	4,6
halfword	0	0
halfword	2	2
halfword	4	4
halfword	6	6
byte	0	1
byte	1	0
byte	2	3
byte	3	2
byte	4	5
byte	5	4
byte	6	7
byte	7	6

M2SX Pinout

Signal Name	# Pins	I/O		Signal Description
MAD<63:0>	64	I/O	(4ma)	Mbus Address/Control/Data
MAS*	1	Input		Mbus Address Strobe
MERR*	1	Output	(4ma)	Mbus Error Indicator
MRDY*	1	Output	(4ma)	Mbus Data Ready Indicator
MRTY*	1	Output	(4ma)	Mbus Retry Indicator
ID<3:0>	4	Input		Mbus Module Identifier
MCLK	1	Input		Mbus Clock
RSTIN*	1	Input		Mbus Reset
A<23:0>	24	Output	(4ma)	386SX address
D<15:0>	16	I/O	(4ma)	386SX Data
ADS*	1	Output	(4ma)	386SX Address Strobe
READY*	1	Input		386SX Bus Ready
WR	1	Output	(4ma)	386SX Write
MIO	1	Output	(4ma)	386SX Memory Cycle
BHE*	1	Output	(4ma)	386SX Byte High Enable
CPUCLK	1	Input		386SX Processor Clock
CLK2/2	1	Input		Slow mode clock
HOLD	1	Input		386SX Hold Request
HLDA*	1	Output	(4ma)	386SX Hold Acknowledge
IOR*	1	Input		AT I/O Read Strobe
MEMR*	1	Input		AT Memory Read Strobe
SCAN_OUT	1	Output	(4ma)	Scan Data Out
SCAN_IN	1	Input		Scan Data In
TM_OE*	1	Input		Test Mode/Output Enable
OBIO_WR*	1	Output	(4ma)	On-board I/O write strobe
OBIO_RD*	1	Output	(4ma)	On-board I/O read strobe
OBIO_CS<7:0>*	8	Output	(4ma)	On-board I/O chip selects

Total 138 Signal pins.

Clocking Considerations

The Mbus may operate at clock rates of as high as 40Mhz, while most 386 system logic, such as an AT chipset, has a maximum of from 25 to 33 Mhz. If the Mbus is running at a clock rate higher than a particular set of 386 system logic can accomodate, there is provision for running the 386 system logic at half the Mbus clock rate, and having the M2SX chip automatically synchronize to the slower speed by means of the CLK2/2 synchronizing input.

Alternate Bus Masters

The M2SX has provision for allowing alternate bus masters on the 386sx bus. If the 386SX bus request signal (HOLD) is asserted, the M2SX will assert the bus acknowledge (HLDA) and go into a state in which it tristates its address data and control lines, until HOLD is negated. During that time, internal accesses to the M2SX can proceed, but onboard I/O accesses by the CPU will generally result in a retry acknowledgement on the Mbus. The exception is the case of a boot prom access to space 0xFF0XXXXXX, in which the M2SX will wait until the alternate bus master cycle is over, then proceed with the cycle normally.

Signal Descriptions

MAD<63:0>: Mbus Multiplexed Address and Data.

MAS*: Mbus Address Strobe. This signal goes active low to indicate the portion of the address cycle when the MAD bus contains address information.

MERR*. This signal is asserted when the M2SX detects an error condition, such as an unsupported operation.

MRDY*. This signal is asserted by the M2SX to indicate the successful completion of a transfer.

MRTY*. This signal is asserted by the M2SX to indicate that the requested operation cannot be completed now, and should be retried later.

ID<3:0>. These pins define a unique ID for each Mbus module in the system. In the case of the M2SX they determine the Mbus space in which its configuration registers lie.

MCLK: Mbus System Clock.

RSTIN*: Mbus System Reset.

A[23:0]: Address Bus for the 386SX.

D[15:0]: Data Bus for the 386SX.

ADS*. This signal from the M2SX, when active low, is used by 386SX system logic to store the 386 address A[23:0], and signals 386 system logic that a bus cycle is beginning.

READY*. This signal from 386 system logic,when active low, indicates to the M2SX chip that a 386 cycle is complete.

WR. 386 bus control signal which when high indicates a write transfer.

MIO. 386 bus control signal which when high indicates a memory cycle.

BHE*. 386 bus control signal which when active low, indicates that valid data is on bits 15:8 of D(15:0).

CPUCLK: 386 Processor Clock. Normally = MCLK/2, except if chip is configured for slow mode, in which case it = MCLK/4.

CLK2/2: Slow Mode Clock. Normally =MCLK. It could be the case MCLK is running at a sufficiently high clock rate such that the 386 system support logic external to theM2SX cannot use MCLK directly as its clock input, and must divide MCLK down by 2. In that case, connecting this pin to MCLK/2 will make the M2SX automatically synchronize the signals it sends to the 386 support logic, running at the slower speed.

HOLD. This signal is asserted by the 386 system support logic when it desires to take control of the 386SX bus.

HLDA. This signal is asserted by the M2SX to indicate to the 386 system support logic that it has granted the 386 bus to it.

IOR*. This signal 's positive-going edge clocks data into the M2SX on AT I/O reads.

MEMR*. This signal 's positive-going edge clocks data into the M2SX on AT Memory reads.

OBIO_WR*. Write strobe for on-board I/O accesses.

OBIO_RD*. Read strobe for on-board I/O accesses.

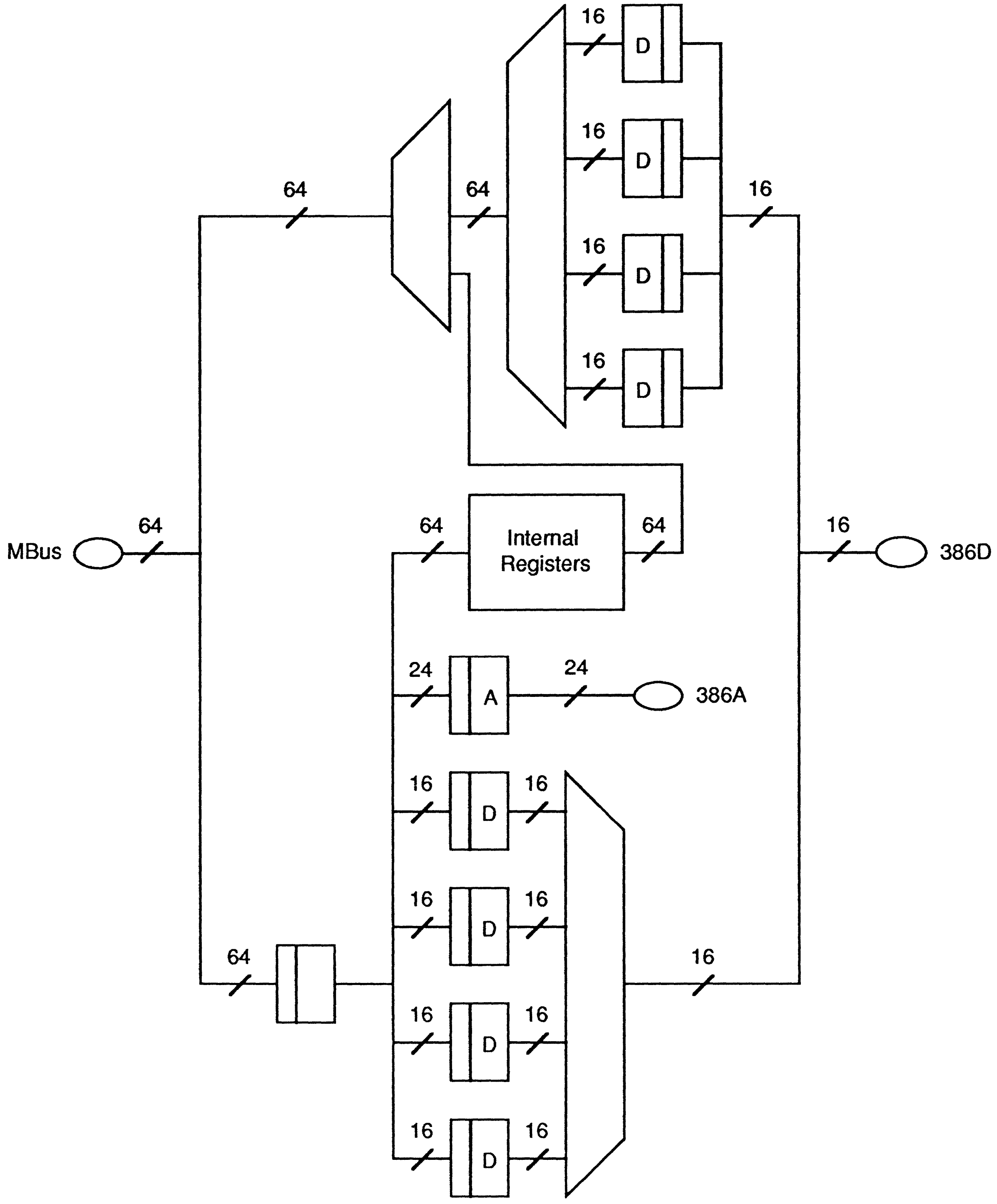
OBIO_CS<7:0>*. Chip selects for on-board I/O accesses.

SCAN_OUT

SCAN_IN

SCAN_MODE

M2SX Data Path



```
/s/users/sparc/breeze/sbus/STATUS

status of SBus subsection:

        DESIGN: complete

        SIMULATION: test suite in /s/users/sparc/breeze/v
                    m2s.diag contains a concatenation of m2s.diagX individual
                    diagnostics for the Mbus to SBus interface logic.
                    tlb.diag contains a concatenation of tlb.diagX individual
                    diagnostics for the SBus Controller and its I/O MMU.
                    all.diag contains a concatenation of m2s.diag and tlb.diag.
                    Simulation environment includes the DMA subsection.
                    See /s/users/sparc/breeze/v/verilog.cmd for verilog
                    simulation files.

        BUGS: none outstanding

        LOCATIONS of files:
                drawings: /s/users/sparc/breeze/sbus/sbus
                simulation environment : see above
                verilog libraries: see /s/users/sparc/breeze/v/verilog.cmd



status of M2SCHIP chip:

        DESIGN: complete

        SIMULATION: test suite in /s/users/sparc/breeze/v
                    m2s.diag contains a concatenation of m2s.diagX individual
                    diagnostics for the Mbus to SBus interface logic.
                    tlb.diag contains a concatenation of tlb.diagX individual
                    diagnostics for the SBus Controller and its I/O MMU.
                    all.diag contains a concatenation of m2s.diag and tlb.diag.
                    Simulation environment includes the SBus subsection
                    with the DMA subsection mapped to an SBus Driver.
                    See /s/users/sparc/breeze/v/verilog.cmd for verilog
                    simulation files.

        BUGS: none outstanding

        SYNOPSYS: not mapped at all

        LOCATIONS of files:
                verilog hdl description of chip:
                    /s/users/sparc/breeze/sbus/v/m2sChip.v
                    /s/users/sparc/breeze/sbus/v/m2s.v
                    /s/users/sparc/breeze/sbus/v/sbusController.v
                    /s/users/sparc/breeze/sbus/v/tlbs.v
                verilog library: /tools/lib/model/hdl
                synopsys environment : none
                simulation environment : /s/users/sparc/breeze/v
```

M2S Chip Specification

Rev 1.0

nuary, 1991
3.020891

VIA Technologies, Inc. (408) 746-2200
860 East Arques Avenue
Sunnyvale, CA 94086

M2S Pinout

Signal Name	# Pins	Input/Output	Signal Description
MAD<31:0>	32	Input/Output (4ma)	Mbus Address/Control/Data
MAS*	1	Input/Output (8ma)	Mbus Address Strobe
MERR*	1	Input/Output (8ma)	Mbus Error Indicator
MRDY*	1	Input/Output (8ma)	Mbus Data Ready Indicator
MRTY*	1	Input/Output (8ma)	Mbus Retry Indicator
MBR*	1	Output (4ma)	Mbus Request
MBG*	1	Input	Mbus Grant
MBB*	1	Input/Output (8ma)	Mbus Busy Indicator
MID<3:0>	4	Input	Mbus Module Identifier
MCLK	1	Input	Mbus Clock
SBR<3:1>*	3	Input	SBus Request
SBG<3:1>*	3	Output (4ma)	SBus Grant
D<31:0>	32	Input/Output (4ma)	SBus Data
RD	1	Input/Output (4ma)	SBus Transfer Direction
SIZ<2:0>	3	Input/Output (4ma)	SBus Transfer Size
PA<27:0>	28	Output (4ma)	SBus Physical Address
AS*	1	Output (4ma)	SBus Address Strobe
SEL<5:1>*	5	Output (4ma)	SBus Slave Selects
ACK<2:0>	3	Input/Output (4ma)	SBus Transfer Acknowledgment
CLK	1	Input	SBus Clock
RESET*	1	Output (4ma)	SBus Reset
POR*	1	Input	Power-On Reset
MEXC*	1	Output (4ma)	SBC Memory Exception
TLB_MISS	1	Output (4ma)	SBC TLB Miss
SCAN_OUT	1	Output (4ma)	Scan Data Out
SCAN_IN	1	Input	Scan Data In
SCAN_MODE	1	Input	Scan Mode
INT_OP_OUT	1	Output (4ma)	Internal Operation to Other Chip
LEGAL_OP	1	Input/Output (4ma)	Legal Mbus Operation
MDOUBLE	1	Input/Output (4ma)	Mbus Double Word Transaction
MID_MATCH	1	Input/Output (4ma)	Mbus ID Compare for R&R
MPA <3:07>	4	Input/Output (4ma)	Mbus Physical Address <3:0>
MTYPE	1	Input/Output (4ma)	Mbus Transaction Type (R/W*)
SLOT_DET*	1	Input/Output (4ma)	Mbus Transaction for This Mbus Slot
INT_OP_IN*	1	Input	Internal Operation from Other Chip
ODD	1	Input	Odd Chip Position

Pinout Summary:		
	143	Signal I/O
	10	VDD
	7	VSS
	160	Total Pins

INTRODUCTION

The M2S chip contains the logic for connecting the 64-bit Mbus to the 32-bit SBus. This interface can behave as both a master or slave on either Mbus or SBus. For transactions going from Mbus to SBus, the M2S chip is an Mbus slave for an Mbus master like the CPU. After receiving the transaction, the M2S chip then becomes an SBus master and initiates a transfer to the targeted SBus slave. For transfers going from SBus to Mbus, the M2S chip is an SBus slave for an SBus master like a DVMA master. After receiving the transfer, the M2S chip then becomes an Mbus master and initiates a transaction to the targeted Mbus slave.

Due to limitations, the Mbus data path has been word sliced. This means that two M2S chips are required to interface to Mbus, one to MAD <63:32> and the other to MAD <31:0>. Both chips are required to be connected to SBus. The chips' position (even/odd) and/or the address of the transfer determine which chip drives the bussed output signals. There is a set of pins for passing information between the two chips.

Since Mbus and SBus have different bus data widths, data buffers are needed to provide temporary storage while data is being packed or unpacked. There are two sets of 4-byte buffers, one for data transfers from Mbus to SBus and the other for data transfers from SBus to Mbus. This allows a pair of M2S chips to handle byte, halfword, word and doubleword transfers on Mbus and byte, halfword and word transfers on SBus.

Mbus and SBus may be running at different clock frequencies. Mbus will be typically be running at 33 or 40 MHz while SBus has to run between 16.67 and 25 MHz. In order to keep both busses synchronized the SBus clock will be at the same frequency as the Mbus clock for clock frequencies of 25 MHz or less and at half of the Mbus clock frequency for frequencies greater than 25 MHz. Note that Mbus clock frequencies less than 16.67 MHz, between 25 and 33 MHz, and greater than 50 MHz are all illegal.

The M2S chip also contains the logic for an SBus Controller. The SBus Controller can arbitrate between 4 SBus masters, 1 being the M2S Logic and the other three being external SBus masters. It supports geographically selecting six SBus Slaves, one being the M2S Logic, the other five being external SBus slots. Virtual to physical address translation is done through an eight-entry fully associative TLB with LRU as a replacement policy. The TLBs provide translation for a 32 MByte address space for each SBus slot. Translations can also be disabled on a per SBus slot basis.

Mbus Slave Interface

Mbus is a multiplexed address and data bus. During the address phase the Mbus master drives the physical address (PA), transaction type (TYPE) and size(SIZE), and other control signals. The M2S chip compares PA<35:32> with its Mbus Slot Address, MSLOT, to determine if whether it is the target of the transaction. MSLOT is a field in its internal Status/Control Register. The internal registers themselves are addressed when PA<35:28>=FF hex and PA<27:24> equals the value on the MID<3:0> inputs. This address comparison is summarized below:

PA<35:32>	=	MSLOT<3:0>
	or	
PA<35:0>		
FFnxxxxxC		Mbus Port Register
FFnxxxxx8		Status/Control Register
FFnxxxxx0		Bus Error Register

where n = MID<3:0>

The M2S chip will only accept two transaction types: Read (TYPE<3:0>=0001) and Write (TYPE<3:0>=0000). All other transactions types are illegal.

TYPE<3:0>	Transaction Type
0000	Write
0001	Read
001x	Illegal
01xx	Illegal
1xxx	Illegal

The M2S chip will accept Read and Write transaction sizes of byte, halfword, word and doubleword. All other transaction sizes are illegal.

SIZE<2:0>	Transaction Size	
000	Byte	
001	Halfword	(2 bytes)
010	Word	(4 bytes)
011	Doubleword	(8 bytes)
1xx	Illegal	

Doubleword transactions are transformed into two separate SBus word transfers. This is done because many SBus slave devices may not support two-word burst transfers. The first two bus cycles are made atomic so that double word width registers on an SBus slave device can be written on two consecutive bus cycles. If bus sizing occurs, then any bus cycle after the first two is not atomic.

There are three transaction status bits, MERR*, MRDY*, and MRTY*, that the M2S chip uses to send an acknowledgment back to the Mbus master for a transaction it receives. The transaction status is shown as follows:

MERR*	MRDY*	MRTY*	Acknowledgment
1	1	1	Idle Cycle
1	1	0	Relinquish and Retry
1	0	1	Valid Data Transfer
0	1	1	Error1 => Bus Error

Relinquish and Retry Acknowledgment: The M2S chip will give a Relinquish and Retry acknowledgment (R&R) if it is the target of an Mbus transaction, but is currently busy processing an SBus transfer. If the SBus transfer receives a Rerun Acknowledgment on the first SBus cycle, then an R&R will be given back to the Mbus master. Also, for Mbus Read transactions, if the transaction size is a doubleword or if the transaction results in dynamic bus sizing on SBus, then a R&R is also given. When the read data from SBus has been buffered in the M2S chip, it will wait for the Mbus master to retry the transaction before sending it the data. This allows other Mbus masters to use the bus while the M2S chip is doing a long transfer on SBus.

Valid Data Transfer: For Mbus Read transactions MRDY* is asserted when the read data is driven by the M2S chip onto Mbus. For Mbus Write transactions MRDY* is asserted after the write transfer has completed on SBus. Mbus doubleword transactions are transformed into two separate word transfers on SBus. MRDY* is not asserted until after the second word transfer has completed.

Bus Error Acknowledgment: The M2S chip will assert MERR* when it detects an illegal operation for an Mbus transaction which has targeted it. Also, if the SBus transfer receives an Error Acknowledgment, then MERR* will be asserted.

Mbus Master Interface

If an SBus master wants to perform an operation on an Mbus slave, then it must do it through the M2S chip. After the M2S chip receives an SBus transfer as an SBus slave, it does a protocol conversion and then becomes an Mbus master. It must request Mbus if it is not already being granted the bus. After receiving the grant it can start the transaction as soon as the bus becomes free.

The M2S chip supports protocol conversions on SBus transfers of byte, halfword or word. During the address phase of the Mbus transaction the M2S chip assembles and drives out a doubleword with the following format:

4	16	1	3	4	36
MID	0xFFFC	0	SIZE	TYPE	PA
63 60	59 44	43	42 40	39 36	35 0

MID: Module Identifier. This field is sourced by all Mbus modules and reflects the value input into the module on the MID<3:0> input pins.

SIZE: Transaction Size. This field encodes the size of the transaction as log2 of the number of data bytes being transferred. The encodings are the same as that shown previously for the Mbus slave interface, except that doubleword transactions are not generated.

TYPE: Transaction Type. This field encodes the type of transaction. Only two types of transactions are supported, Read (TYPE<3:0>=0001) and Write (TYPE<3:0>=0000). TYPE<3:1> are always driven with zeroes.

PA: Physical Address. This field contains the physical address from the SBus Controller.

The address is driven out for one cycle followed by any write data on the next cycle. The Mbus master interface asserts MAS* during the address cycle to indicate to the Mbus slave that this is the first cycle of the transaction and that the address is on the bus. For a Write transaction the data is held on the bus until the slave gives an acknowledgment.

The Mbus master interface will accept all acknowledgments that are defined by Mbus. The transaction status is decoded as follows:

MERR*	MRDY*	MRTY*	Acknowledgment
1	1	1	Idle Cycle
1	1	0	Relinquish and Retry
1	0	1	Valid Data Transfer
1	0	0	Undefined
0	1	1	Error1 => Bus Error
0	1	0	Error2 => Timeout
0	0	1	Error3 => Uncorrectable
0	0	0	Retry

Relinquish and Retry Acknowledgment: For a Relinquish and Retry acknowledgment, a Rerun acknowledgment will be given by the SBus slave interface back to the SBus master.

Undefined, Error Acknowledgment: For the Undefined or any of the Error acknowledgments, an Error acknowledgment will be given by the SBus slave interface back to the SBus master. The type of error and the physical address, size, and other information for the transaction that generated the error will be saved in the Bus Error Register.

SBus Slave Interface

To determine if an SBus transfer is targeting the M2S chip, the SBus slave interface checks to see if AS* and SEL* are both asserted. If they are, then the M2S chip will proceed to do a protocol conversion on the SBus transfer into an Mbus transaction.

The M2S chip decodes the RD input to determine whether the transfer is a Read (RD=1) or a Write (RD=0).

The M2S chip will accept Read and Write transfers of byte, halfword and word. All other transfer sizes are illegal.

SIZ<2:0>	Function	
000	Word transfer	(4 bytes)
001	Byte transfer	
010	Halfword transfer	(2 bytes)
011	Illegal	
1xx	Illegal	

There are three transfer acknowledgment signals, ACK<2:0>, that the M2S chip uses to send an acknowledgment back to the SBus master for a transfer it receives. The acknowledgment encoding is shown as follows:

ACK<2:0>	Acknowledgment
111	Idle/Wait
110	Error acknowledgment
100	Rerun acknowledgment
011	Word (data) acknowledgment

Error Acknowledgment: If the transfer size is illegal, then the SBus slave interface will give an Error Acknowledgment. An Error acknowledgment will also be given if an Mbus transaction receives an Undefined or Error acknowledgment.

Rerun Acknowledgment: The M2S chip will give a Rerun acknowledgment if it is selected for an SBus transfer, but is currently busy processing an Mbus transaction. For this situation a flag also set which will cause the Mbus slave interface to give Relinquish and Retry acknowledgments until the SBus slave interface has accepted a transfer and completed the transaction on Mbus. This avoids a deadlock situation when the SBus master is in a mode that prevents it from servicing a slave request due to its master mode requirements. A Rerun acknowledgment will also be given if an Mbus transaction receives a Relinquish and Retry acknowledgment.

Word Acknowledgment: For SBus Read transfers the Word acknowledgment is given the cycle before the read data is driven by the M2S chip onto SBus. For SBus Write transfers the word acknowledgment is given after the write transaction has completed on Mbus.

SBus Master Interface

If an Mbus master wants to perform an operation on an SBus slave, then it must do it through the M2S chip. After the M2S chip receives an Mbus transaction as an Mbus slave, it does a protocol conversion and becomes an SBus master. It must acquire ownership of SBus by first making a request to the SBus Controller. After receiving the grant it must drive the low order 32 bits of the Mbus physical address onto D<31:0> during the address cycle. The SBus Controller, knowing that the M2S chip has been granted the bus, passes the low order 28 bits of this data through to PA<27:0>. The high order nibble, PA<31:28>, is decoded to generate the SBus Slave Selects, SEL*. No virtual address translation is necessary since the M2S chip is already providing a physical address. The M2S chip must also drive RD and SIZ<2:0>. If the transfer is a Write, then the write data is driven onto the bus the cycle following the address cycle. The M2S chip then waits for an acknowledgment from the targeted slave or from the SBus Controller in the case of a timeout.

The SBus master interface will recognize all of the SBus Acknowledgments. A table of ACK<2:0> encodings is shown below:

ACK<2:0>	Acknowledgment
111	Idle/Wait
110	Error acknowledgment
101	Byte (data) acknowledgment
100	Rerun acknowledgment
011	Word (data) acknowledgment
010	Reserved
001	Halfword (data) acknowledgment
000	Reserved

Rerun Acknowledgment: For a Rerun acknowledgment, a Relinquish and Retry acknowledgment will be given by the Mbus slave interface back to the Mbus master.

Reserved, Error Acknowledgment: For the Reserved and Error acknowledgments, a Bus Error acknowledgment will be given by the Mbus slave interface back to the Mbus master. The type of error and the physical address, size, and other information for the transfer that generated the error will be saved in the Bus Error Register.

Data Acknowledgment: For Byte or Halfword Data acknowledgments the SBus master interface supports dynamic bus sizing as defined in the SBus Specification. This features allows the SBus master interface to initiate a word transfer to a halfword or byte wide SBus slave device, or a halfword transfer to a byte wide SBus slave device. Dynamic bus sizing never occurs for a byte transfer. During dynamic bus sizing, multiple SBus cycles are used to transfer each byte or halfword. For a word transfer two SBus cycles are required for an SBus slave which responds with Halfword Data acknowledgments. Likewise, four SBus cycles are required for an SBus slave which responds with Byte Data acknowledgments. The SBus master interface must generate the correct address for the datum being transferred during each SBus cycle of the transfer. The SBus slave must respond with the same data acknowledgment for each SBus cycle of a bus sizing operation. If the SBus slave gives a Rerun acknowledgment, then the SBus master interface will rerun the current SBus cycle. It will not restart the transfer at the original bus cycle.

In addition to supporting dynamic bus sizing the SBus master interface will also support port locations within a data word as shown below:

31 24	23 16	15 8	7 0
BYTE 0	BYTE 1	BYTE 2	BYTE 3
HALFWORD 0		HALFWORD 1	
WORD			

When the SBus master interface performs a byte write, it will place a copy of the write data on byte 0 in addition to the byte's natural location within the word. For example, if the byte's address ends in 10, then the write data will appear at both bytes 0 and 2. Similarly, for halfword writes a copy of the data will appear on halfword 0 in addition to the halfword's natural location within the word. If the halfword address ends in 10, then the write data will appear at both halfword 0 and 1.

When reading data from an SBus slave, the location of the data depends on the slave's data acknowledgment. For a Byte Data acknowledgment the data is transferred on byte 0, the most significant byte. For a Halfword Data acknowledgment the data is transferred on halfword 0, the most significant halfword. Byte addressing within the halfword is determined by PA<0>. Similarly, for Word Data acknowledgments the data is transferred on the entire word. Halfword addressing within the word is determined by PA<1> and byte addressing within the word is determined by PA<1:0>.

Mbus Port Register

The Mbus Port Register (MPR) is a read only register which contains the Implementation Number and Version Number of the M2S Chip. It is addressed when PA<35:0>=FFnxxxxxC where n=MID<3:0>. The format of the MPR is shown below:

16	8	4	4
Reserved	MDEV	MREV	MVEND
31 16	15 8	7 4	3 0

MDEV: Mbus Device Number. This field contains a unique number identifying the device as a VIA M2S Chip.

MREV: Mbus Revision Number. This field contains the revision number for the VIA M2S Chip.

MVEND: Mbus Vendor Number. This field contains the vendor number for VIA.

Status/Control Register

The Status/Control Register (SCR) is a read/write register which contains the Mbus Slot Address, MSLOT. It is addressed when PA<35:0>=FFnxxxxx8 where n=MID<3:0>. Writes must be done with doubleword transactions with the MSLOT write data replicated for the odd word. The format of the SCR is shown below:

28	4
Reserved	MSLOT
31 4	3 0

MSLOT: Mbus Slot Address. This field contains a unique number among all Mbus modules which reside on Mbus. MSLOT is compared with PA<35:32> to determine whether this chip is the target of the transaction. This register should be configured by the Operating System after it has read all of the MPRs for the modules which reside on Mbus.

Bus Error Register

The Bus Error Register is a read only register which contains information about an error, reserved or undefined acknowledgment that the M2S chip received as a master on Mbus or SBus. It is addressed when PA<35:0>= FFnxxxxx0 where n=MID<3:0>. The physical address, size, type, and acknowledgment for the transaction is saved in the Bus Error Register. Once an error has been detected, the Bus Error Register cannot be overwritten by another error until it is read or until POR* is asserted. The format of the Bus Error Register is shown below:

4	8	4	4	1	3	4	36
MID	Rsvd	MCODE	SCODE	0	SIZE	TYPE	PA
63 60	59 52	51 48	47 44	43	42 40	39 36	35 0

MID: Module Identifier. This field contains a copy of the Mbus Module Identifier. If the M2S Chip was an Mbus slave for the Mbus transaction, then the MID will be the Module Identifier for the Mbus master. If the M2S Chip was the Mbus master for the Mbus transaction, then the MID is the Module Identifier for the M2S Chip sourced from the MID<3:0> input pins.

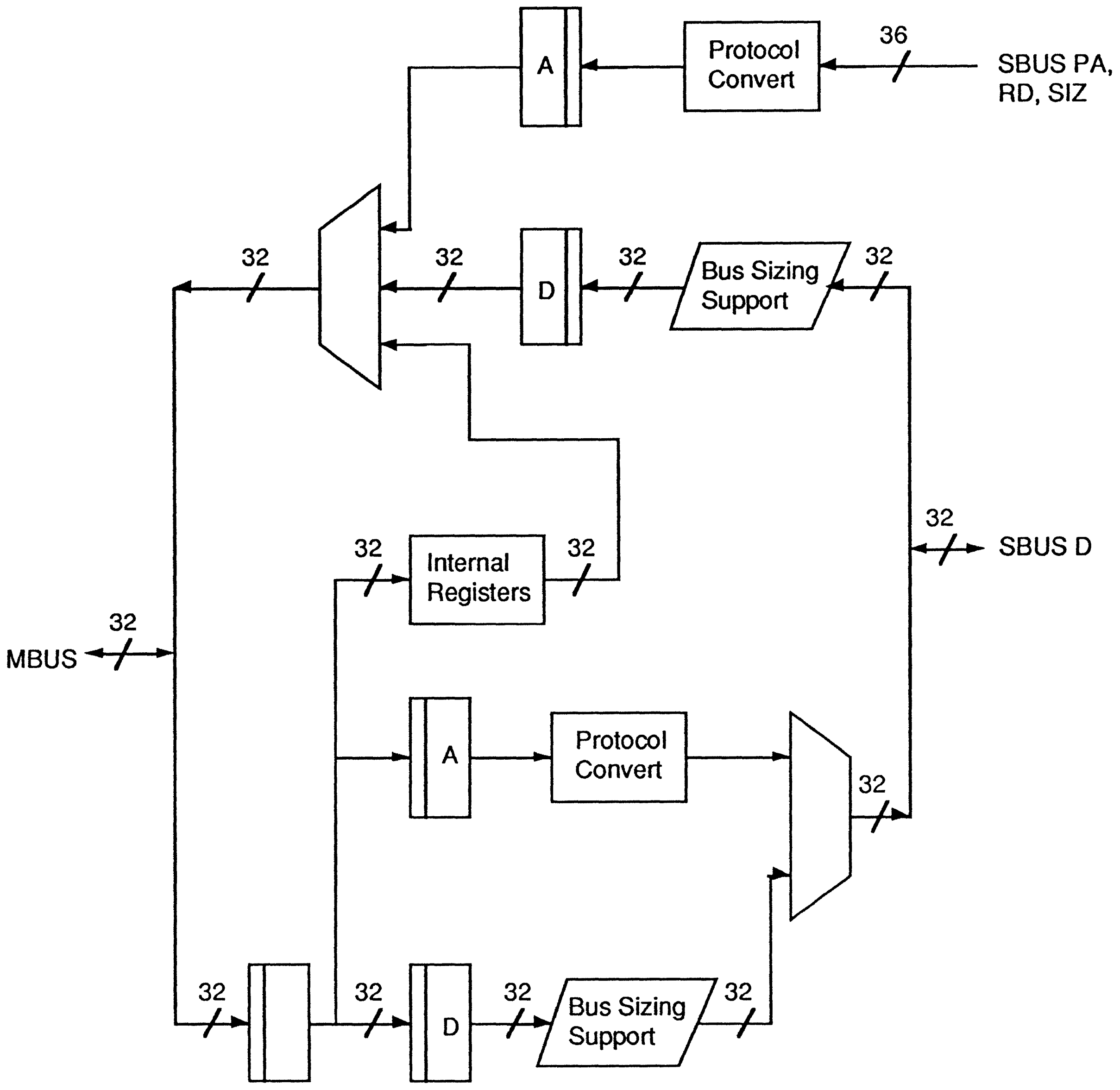
MCODE: Mbus Error Code. This field contains the error code information for an Mbus Bus Error. Bit 51 is a 1 if an Mbus Bus Error occurred. Bits 50-48 are a copy of the Mbus transaction status bits; MERR*, MRDY*, and MRTY*, respectively. Bit 51 is cleared after reading the Bus Error Register and whenever POR* is asserted.

SCODE: SBus Error Code. This field contains the error code information for an SBus Bus Error. Bit 47 is a 1 if an SBus Bus Error occurred. Bits 46-44 are a copy of the SBus transfer acknowledgment bits; ACK<2:0>. Bit 47 is cleared after reading the Bus Error register and whenever POR* is asserted.

SIZE: Mbus Transaction Size or SBus Transfer Size. This field contains a copy of either the Mbus transaction size or the SBus transfer size depending on which bus the error occurred.

TYPE: Mbus Transaction Type or SBus Transfer Direction. This field contains a copy of the Mbus transaction type or the SBus transfer direction depending on which bus the error occurred. For an SBus Bus Error, the SBus transfer direction is recorded in bit 36. Bits 39-37 will be zeroes.

PA: Physical Address. This field contains a copy of the physical address for either the Mbus transaction or the SBus transfer.


A
Protocol Convert
36
SBUS PA, RD, SIZ
32
32
D
32
Bus Sizing Support
32
32
SBUS D
32
Internal Registers
32
32
MBUS
A
Protocol Convert
32
32
32
D
32
Bus Sizing Support
32

Bus Arbitration

The SBus Controller can arbitrate between 4 masters. It employs a fair arbitration scheme which is similar to round-robin prioritization. While the bus is idle, the master that is given the highest priority for the bus changes every cycle in a rotating fashion. Except for an atomic transfer, once a master has been granted use of the bus, he is not allowed to use the bus again until all of the other masters that had requested the bus while he was using it have been granted use of the bus. The other masters may not necessarily be granted the bus in chronological order. Each master has its own request line, BR*. The SBus Controller arbitrates these requests and provides each master with its own grant line, BG*. The assignment of request and grant lines is as follows:

BR<0>*, BG<0>*	-	SBus slot 0. (M2S master, internal)
BR<1>*, BG<1>*	-	SBus slot 1. (on-board or plug-in master)
BR<2>*, BG<2>*	-	SBus slot 2. (on-board or plug-in master)
BR<3>*, BG<3>*	-	SBus slot 3. (on-board or plug-in master)

BR<0>* and BG<0>* are internal signals that are used by the M2S SBus master interface.

Translating Virtual to Physical Addresses

Translation of virtual addresses to physical addresses is done through an I/O MMU. The I/O MMU stores the eight most recently used Page Table Entries (PTEs) in a fully associative TLB and uses one of those to translate the virtual page address to the physical page address. The virtual address bits for within a page are passed through unmodified to the same bits of the physical address. The SBus Controller provides 36 bits of physical address from a translation of the 32-bit virtual address. Virtual address translation for up to 32 MBytes of physical address space is provided for each SBus slot. Translation can be enabled or disabled for each of these slots by writing the appropriate value into the SBus Controller Control Register. For the M2S SBus master, translation is permanently disabled since it already has a physical address from Mbus. More details on the I/O MMU are described elsewhere in this specification.

Slave Selects Decoding

Each SBus slave is geographically addressed, which means that they each receive a unique unary encoded address signal, called SEL*. The SBus Controller translates a 32-bit virtual address driven by a master and generates a 36-bit physical address, PA<35:0>, of which the low order 28 bits, PA<27:0>, are driven onto SBus. PA<35:28> are decoded to generate SBus slave selects, SEL<5:0>*. PA<35:32> are compared with MSLOT<3:0> to determine whether the target is on SBus or Mbus. MSLOT<3:0> is the Mbus slot address for the M2S chip. PA<35:12> comes from a field in the TLB while PA<11:0> are passed through from VA<11:0>. If translation is disabled, then PA<31:0> comes directly from VA<31:0> and PA<35:32> = 0 hex.

The SBus slave selects are decoded as follows:

```
if (PA<35:32> < > MSLOT<3:0>) then
  SEL<0>*=0 // SBus slot 0 (M2S SBus Slave)

if (PA<35:32> = = MSLOT<3:0>) then
  PA<31:28>
      0      -  if (M2S owns SBus) then
                SBus Controller Internal Registers
      1      -  SEL<1>*, SBus slot 1
      2      -  SEL<2>*, SBus slot 2
      3      -  SEL<3>*, SBus slot 3
      4      -  SEL<4>*, SBus slot 4 (Slave only)
      5      -  SEL<5>*, SBus slot 5 (Slave only)
   6 thru F  -  reserved
```

SEL<0>* is an internal signal that is used to select the M2S SBus Slave.

Note that if some SBus master other than M2S tries to select the SBus Controller Internal Registers (PA<35:32> == MSLOT<3:0> and PA<31:28> = 0 hex), then no SBus slot will be selected and a timeout will be given by the SBus Controller.

Address Strobe Generation

The SBus Controller asserts an address strobe, AS*, to indicate to slaves that the slave selects, SEL<5:0>*, and the physical address, PA, are valid on the bus. AS* is kept asserted until the clock cycle following the final acknowledgment for the transfer, whether it be a data, rerun, or error acknowledgment.

Slave Acknowledgment Monitoring

The SBus Controller monitors the SBus lines RD, SIZ<2:0>, and ACK<2:0> to determine when to de-assert the BG<3:0>* and AS* lines. If a master is performing a burst transfer, then the SBus Controller must count the number of word acknowledgments given by the slave to determine when the transfer has completed. A 32-byte burst transfer would be completed after the fourth word acknowledgment. A slave must give word acknowledgments for a burst transfer. For non-burst transfers or for byte or halfword acknowledgments the transfer is complete after one acknowledgment cycle.

If the SBus Controller sees an error acknowledgement on a non-CPU DVMA cycle, then the error will be saved in the ID/Control Register. If the MEXC_EN bit is a 1, then MEXC* will be asserted to interrupt the CPU.

Bus Timeout Generation

The SBus Controller will drive an error acknowledgment on ACK<2:0> if a slave does not generate its own acknowledgment no later than the 255th clock cycle following the assertion of AS*. The SBus Controller will drive ACK<2:0> on the 256th clock cycle with the error acknowledgment, followed by an idle acknowledgment on the next cycle. On the cycle after that AS* is de-asserted.

SBus Controller Slave Interface

If PA<35:28>=n0 hex , where n=MSLOT<3:0> and the M2S SBus master owns SBus, then an SBus transfer is targeting the SBus Controller Internal Registers. The SBus Controller will assert an internal slave select signal and AS* just like it would for any other slave cycle. The SBus Controller slave interface will look to see if both of these signals are asserted. If they are, then the SBus Controller slave interface will proceed to execute the transfer.

The SBus Controller slave interface decodes the RD input to determine whether the transfer is a Read (RD=1) or a Write (RD=0).

The SBus Controller slave interface will only accept word transfers. All other transfer sizes are illegal.

SIZE<2:0>	Function
000	Word transfer (4 bytes)

There are three transfer acknowledgment signals, ACK<2:0>, that the SBus Controller slave interface chip uses to send an acknowledgment back to the SBus master for a transfer it receives. The acknowledgment encoding is shown as follows:

ACK<2:0>	Acknowledgment
111	Idle/Wait
110	Error acknowledgment
100	Rerun acknowledgment
011	Word (data) acknowledgment

Error Acknowledgment: If the transfer size is not a word, then the SBus Controller slave interface will give an Error acknowledgment.

Rerun Acknowledgment: If the M2S SBus master tries to access an internal register on the SBus Controller while it is processing a TLB Miss, then the SBus Controller will give it a Rerun acknowledgment.

Word Acknowledgment: The SBus Controller slave interface will give Word acknowledgments for word transfers to the SBus Controller Internal Registers. For Read transfers the Word acknowledgment is given the cycle before the read data is driven onto SBus.

SBus Controller Master Interface

When the SBus Controller detects a TLB Miss, it will give the SBus master a rerun acknowledgment and proceed to fetch a PTE from main memory. All other bus requests are held off while the SBus Controller master interface assembles the physical address of the PTE from the Page Table Pointer, slot address of the SBus Master, and VA<24:12>. The SBus Controller master interface does not require a translation cycle. RD is driven high and SIZ<2:0> is driven with zeroes for the size of a word. The SBus Controller drives out PA<35:0>, AS*, and SEL* as it normally does. The SBus Controller master interface then waits for an acknowledgment from the targeted slave. Upon receiving the PTE, the SBus Controller master interface will build the TLB and update the appropriate entry.

The SBus Controller master interface will recognize all of the SBus Acknowledgements. A table of ACK<2:0> encodings is shown below:

ACK<2:0>	Acknowledgment
111	Idle/Wait
110	Error acknowledgment
101	Byte (data) acknowledgment
100	Rerun acknowledgment
011	Word (data) acknowledgment
010	Reserved
001	Halfword (data) acknowledgment
000	Reserved

Reserved, Error, Byte, Halfword Acknowledgment: For the Reserved, Error, Byte and Halfword acknowledgments, the SBus Controller master interface will assert MEXC* to cause an interrupt to the CPU. A copy of the virtual address is saved in the Fault Address Register. The ACK_ERROR bit is set in the ID/Control Register and ACK<2:0> is copied into SCODE<2:0>.

Rerun Acknowledgment: For a Rerun acknowledgment the SBus Controller master interface will retry the SBus transfer.

ID/Control Register

The SBus Controller has an ID/Control Register which contains status and control information. There is also a unique hardwired ID number. The format of the register is shown below:

4	5	1	1	1	4	7	1	1	1	2	1	1	1	1
DEV_ID	▨	L	T	A	SCODE	IOBVA	MEXC_EN	RESET	DM	▨	ME3	ME2	ME1	0
31 28	27 23	22	21	20	19 16	15 9	8	7	6	5 4	3	2	1	0

DEV_ID: Device ID Number (Read Only). These bits are hardwired to a value that is used to identify the device. For the current implementation, the field is set to 0000.

L: LRU Error (Read Only). This bit indicates that a TLB entry has been updated for a TLB Miss when all the TLB entries were locked. MEXC* is asserted when this bit is a 1. This bit is cleared after reading the ID/Control Register and whenever POR* is asserted.

T: TLB Error (Read Only). This bit indicates that an invalid PTE was used to update a TLB entry for a TLB Miss. MEXC* is asserted when this bit is a 1. This bit is cleared after reading the ID/Control Register and whenever POR* is asserted.

A: Ack Error (Read Only). This bit indicates that the SBus Controller master interface received a Reserved, Error, Byte, or Halfword Acknowledgment while reading a PTE from main memory. MEXC* is asserted when this bit is a 1. This bit is cleared after reading the ID/Control Register and whenever POR* is asserted.

SCODE: SBus Error Code (Read Only). This field contains the error code information for an SBus Bus Error. Bit 19 is a 1 if an SBus Bus Error occurred. Bits 18-16 are a copy of the SBus transfer acknowledgment bits, ACK<2:0>. MEXC* is asserted if bit 19 is a 1 and MEXC_EN is also a 1. Bit 19 is cleared after reading the ID/Control Register and whenever POR* is asserted.

IOBVA: I/O Base Virtual Address (R/W). These bits are used to compare with VA<31:25> to determine if the address is within the 32 MByte virtual address space for SBus devices.

MEXC_EN: Memory Exception Enable (R/W). This bit is used to enable the MEXC* output when SCODE<3> is a 1. This bit is cleared when POR* is asserted.

RESET: Reset (R/W). This bit is used to assert RESET*, the SBus reset signal. Software can assert an SBus Reset by writing a 1 to this bit. RESET* will remain asserted until the bit is cleared. POR* will assert the RESET* signal.

DM: Diagnostic Mode (R/W). This bit is used to inhibit the LRU from being decremented when writing a TLB entry. When TLB entries are invalidated, the LRUs for those TLBs that have a larger value than the one being written are decremented so that no "holes" are left in the LRU counts. Setting this bit will allow writing a TLB without affecting the LRU fields of other TLBs. This may be useful when testing the TLBs while running some sort of diagnostic.

ME3-ME1: MMU Enable for SBus Slot 3-1 (R/W). These bits are used to enable virtual address translation for the masters in SBus slots 3, 2, and 1. Virtual address translation is enabled for a particular SBus slot by writing its corresponding MMU Enable bit to a 1. SBus slot 0 is used by an M2S Interface which already has the physical address, so no translation is needed. Translation for this SBus slot is always disabled. These bits are cleared whenever POR* is asserted.

The ID/Control Register is selected when PA<35:0>=n00xxxxx0 where n=MSLOT<3:0>. The ID/Control Register is read to or written from SBus D<31:0>.

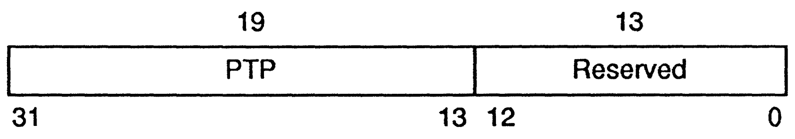
Fault Address Register

The SBus Controller has an Fault Address Register (FAR) which contains a copy of the 32-bit virtual address of a reference that resulted in a memory exception. A memory exception can occur if LRU Error, TLB Error, or Ack Error is set, or if SCODE<3> and MEXC_EN are both 1. Once an error occurs, the FAR is not reclocked for any subsequent error until the first error is cleared by reading the ID/Control Register or by asserting POR*.

The FAR is a read only register. It is selected when PA<35:0>=n00xxxxx4 where n=MSLOT<3:0>. The FAR is read from SBus D<31:0>.

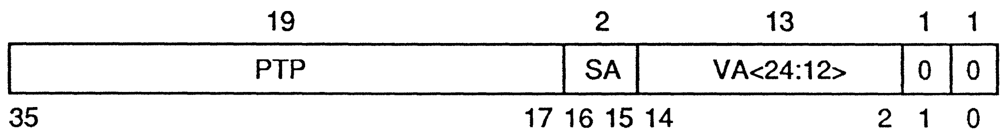
Page Table Pointer

The SBus Controller has a page table pointer (PTP) which contains the high order bits of the physical address for the start of the I/O Page Table. When the SBus Controller detects a TLB Miss, it tries to resolve it by doing a 'table walk' to fetch a PTE. VA<24:12> are used in conjunction with the PTP to index into the page table. The format of the PTP is defined as follows:



The PTP is a read/write register. It is selected when PA<35:0>=n00xxxxx8 where n=MSLOT<3:0>. The PTP is read to or written from SBus D<31:0>.

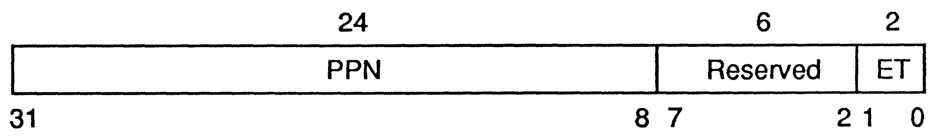
The format of the physical address for fetching a PTE as it appears on Mbus PA<35:0> is shown as follows:



The 2 LSBs must be zeroes since all PTEs in main memory have the size of a word (4 bytes). The SA field is the SBus slot address which has the same value as the SBus slot number. SBus slot 0 does not use any TLBs, so S<1:0>=0 is illegal.

Page Table Entry

The page table entry (PTE) has a format similar to that found in the SPARC Reference MMU. Memory management bits for Referenced, Modified, and Cacheable are not needed, so they are not defined. The access permission code is also not needed, so those bits are also omitted. The format of the PTE is defined as follows:



PPN: Physical Page Number. These are the high order 24 bits of the physical address which appear on PA<35:12> after the virtual address is translated.

ET: Entry Type. These bits are 10 for a valid PTE.

TLB Entries

The TLBs are read/write registers. Since they are 64 bits they must be read and written in two single word transfers. The first word of the transfer reads/writes bits <63:32> of the TLB while the second word read/writes bits <31:0> onto/from SBus D<31:0>. The TLBs are selected when PA<35:0>=n02xxxxxx, where n=MSLOT<3:0>. They can only be accessed from the M2S interface. The CPU should use Mbus doubleword accesses which will be transformed into two SBus word atomic transfers by the M2S interface.

The TLBs are organized as an 8-entry associative memory. The encoded SBus slot address and VA<24:12> are used to compare with the correspondingly field in all eight TLB entries simultaneously. If there is a match, the physical page address from the entry that matched is used for the virtual address translation.

When TLB is full and an entry needs to be replaced, a Least Recently Used (LRU) replacement policy is used to decide which entry will be replaced. A 3-bit field in each TLB entry is used to keep track of when they were last accessed. If there is a TLB Hit, the LRU for the TLB that hit is set to 0. The LRU for all other valid TLB entries whose value was originally lower than the TLB hit are incremented, while the rest remain unchanged. If there is a TLB Miss and the TLB is not full, then the LRU for the new entry is set to 0 and the LRU for all other valid entries are incremented. If there is a TLB Miss and the TLB is full, then the TLB entry whose LRU has the highest value is replaced. The LRU for the new entry is set to 0 and the LRU for all the other TLB entries are incremented.

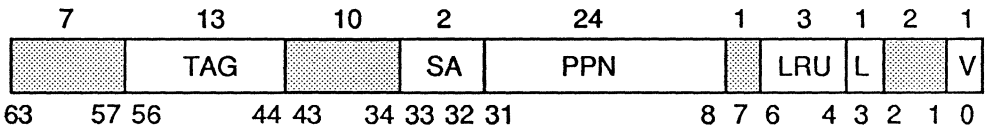
TLB entries can also be individually locked by setting their Lock bit. A TLB entry must be valid in order to be locked. A locked TLB entry does not participate in the LRU replacement policy.

The physical address to access the TLBs is defined as follows:

PA<35:0>	TLB
n01xxxx00	TLB0
n01xxxx08	TLB1
n01xxxx10	TLB2
n01xxxx18	TLB3
n01xxxx20	TLB4
n01xxxx28	TLB5
n01xxxx30	TLB6
n01xxxx38	TLB7

where n=MSLOT<3:0>

The format of the TLB is defined as follows:



TAG: Address TAG. These bits are compared with VA<24:12> to determine if this is the TLB to be used for the translation.

SA: Slot Address. These bits are the SBus slot address of the SBus master which owns the TLB entry.

PPN: Physical Page Number. These bits are a copy of the PPN field in the PTE. They are the high order 24 bits of the physical address which appear on PA<35:12> when the translation is completed. PA<11:0> come directly from VA<11:0>.

LRU: Least Recently Used. These bits are used to implement the LRU replacement policy.

L: Lock. This bit indicates whether the TLB entry is locked or not. A locked TLB entry does not participate in the LRU replacement policy.

V: Valid Bit. This bit indicates whether the TLB entry is valid or not.

TLB Hit Comparison

The following four conditions must occur in order to have a TLB Hit:

1) Incoming VA<31:25> must be equal to the IOBVA.
2) The valid bit in the referenced TLB must be on.
3) Incoming VA<24:12> must match the TAG field in the referenced TLB.
4) Incoming SBus slot address must match the SA field in the referenced TLB.

If any of the above conditions is not met, then a TLB Miss occurs. If the incoming VA<31:25> does not equal the IOBVA, then the SBus Controller will give an error acknowledgment back to the master. When a TLB Miss occurs, the SBus Controller gives a rerun acknowledgment to the master and then proceeds to read a PTE from main memory using the PTP, the SBus slot address, and VA<24:12> to construct the address. All other bus requests are held off until the corresponding TLB entry has been validated. When the master retries its transfer, it should get a TLB Hit and the SBus Controller will be able to translate the virtual address.

I/O MMU Flush Operations

The SBus Controller supports flush operations that allow software invalidations of selected entries in the TLB. Several different types of TLB flushing operations are supported. The types are: page, segment, region, slot, and entire flush; in which each type can include or exclude locked TLB entries. The format of the virtual address for TLB flushing is shown below:

7	1	6	6	4	6	2
0000010	REGION	SEGMENT	PAGE	TYPE	▨	SA
31 25	24	23 18	17 12	11 8	7 2	1 0

Bits 31-28 must be zero so that the SBus Controller Slave Interface will be selected. Bits 27-25 must be 010 to be decoded as an I/O MMU Flush Operation.

REGION: Virtual Region Address. This bit selects a 16 MByte region to be flushed.

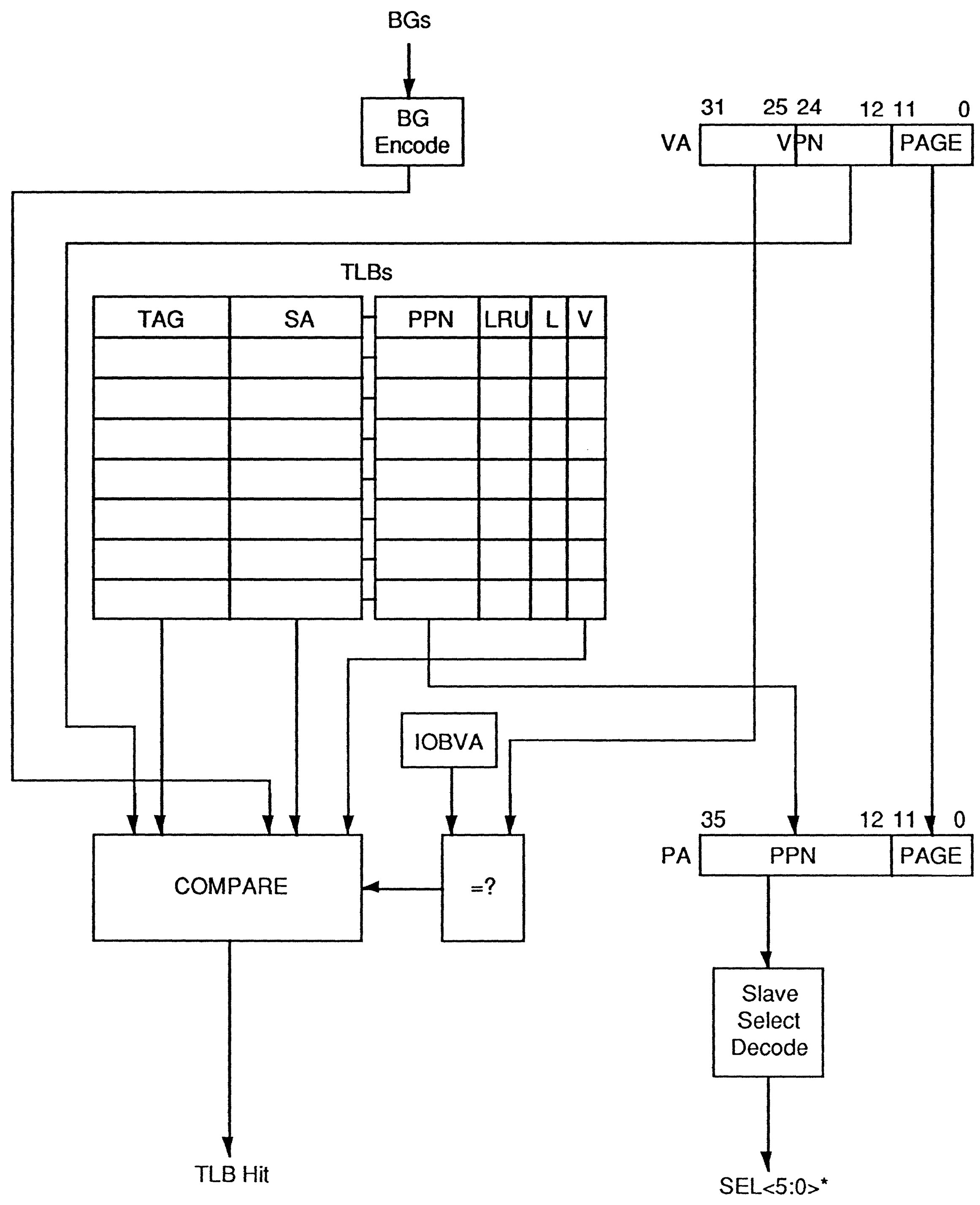
SEGMENT: Virtual Segment Address. These bits select a 256 KByte segment to be flushed.

PAGE: Virtual Address Page. These bits select a 4 KByte page to be flushed.

TYPE: Flush Operation Type. These bits define the address comparison required to match a TLB entry for flushing. The different types are listed in the table below:

Type	Flush		Comparison
0000	Page	(including locked entries)	Slot, region, segment, page
0001	Segment	(including locked entries)	Slot, region, segment
0010	Region	(including locked entries)	Slot, region
0011	Slot	(including locked entries)	Slot
0100	Entire	(including locked entries)	None
0101-0111	Reserved		
1000	Page	(excluding locked entries)	Slot, region, segment, page
1001	Segment	(excluding locked entries)	Slot, region, segment
1010	Region	(excluding locked entries)	Slot, region
1011	Slot	(excluding locked entries)	Slot
1101	Entire	(excluding locked entries)	Unlocked
1110 - 1111	Reserved		

An I/O MMU flush operation is selected when PA<35:24> = nnnn010x and PA<23:0> = xxxxxx where nnnn = MSLOT<3:0>.


BGs
BG Encode
31 25 24 12 11 0
VA
VPN
PAGE
TLBs
TAG
SA
PPN
LRU
L
V
IOBVA
=?
COMPARE
35 12 11 0
PA
PPN
PAGE
Slave Select Decode
TLB Hit
SEL<5:0>*

```
/s/users/sparc/breeze/dma/STATUS

status of SBUSDMA chip:


    DESIGN: complete

    SIMULATION: test suite in /s/users/sparc/breeze/dma/{debug.v,do_test.v,do_test_e.v}

    BUGS: none outstanding

    SYNOPSYS: mapped to att ,vti,
    fujitsu (area 6034, cycle time 30.58),lsi.

    LOCATIONS of files:

        verilog hdl description of chip:
                    /s/users/sparc/breeze/dma/synopsys
                    /s/users/sparc/breeze/dma/synopsys/pure_hdl
        verilog library:            /tools/lib/model/hdl
        synopsys environment : /s/users/sparc/breeze/dma/synopsys
        simulation environment : /s/users/sparc/breeze/dma

    Notes

        There is a problem with Synopsys  generating spurious buffers near I/Os in the ver
        It is a known problem (STAR 4095) and is expected to be fixed in the production re
        Synopsys 2.0, scheduled for late March 1991. There is a similar problem with I/Os
        and Test Compiler.

status of DMA subsection:


    DESIGN: complete

    SIMULATION: test suite in /s/users/sparc/breeze/dma/{debug.v,do_test.v,do_test_e.v}

    BUGS: none outstanding

    LOCATIONS of files:

        drawings:/s/users/sparc/breeze/dma/dma
        simulation environment:/s/users/sparc/breeze/dma
        important directories for verilog:
                    /s/users/sparc/breeze/dma
                    /s/users/sparc/breeze/dma/synopsys
                    /s/users/sparc/breeze/dma/synopsys/pure_hdl
                    /tools/lib/model
                    /tools/lib/lai_vlog
                    /s/users/sparc/breeze/v
```

SBus DMA Controller

Rev 1.0

January, 1991
1.012991

VIA Technologies, Inc. (408) 746-2200
860 East Arques Avenue
Sunnyvale, CA 94086

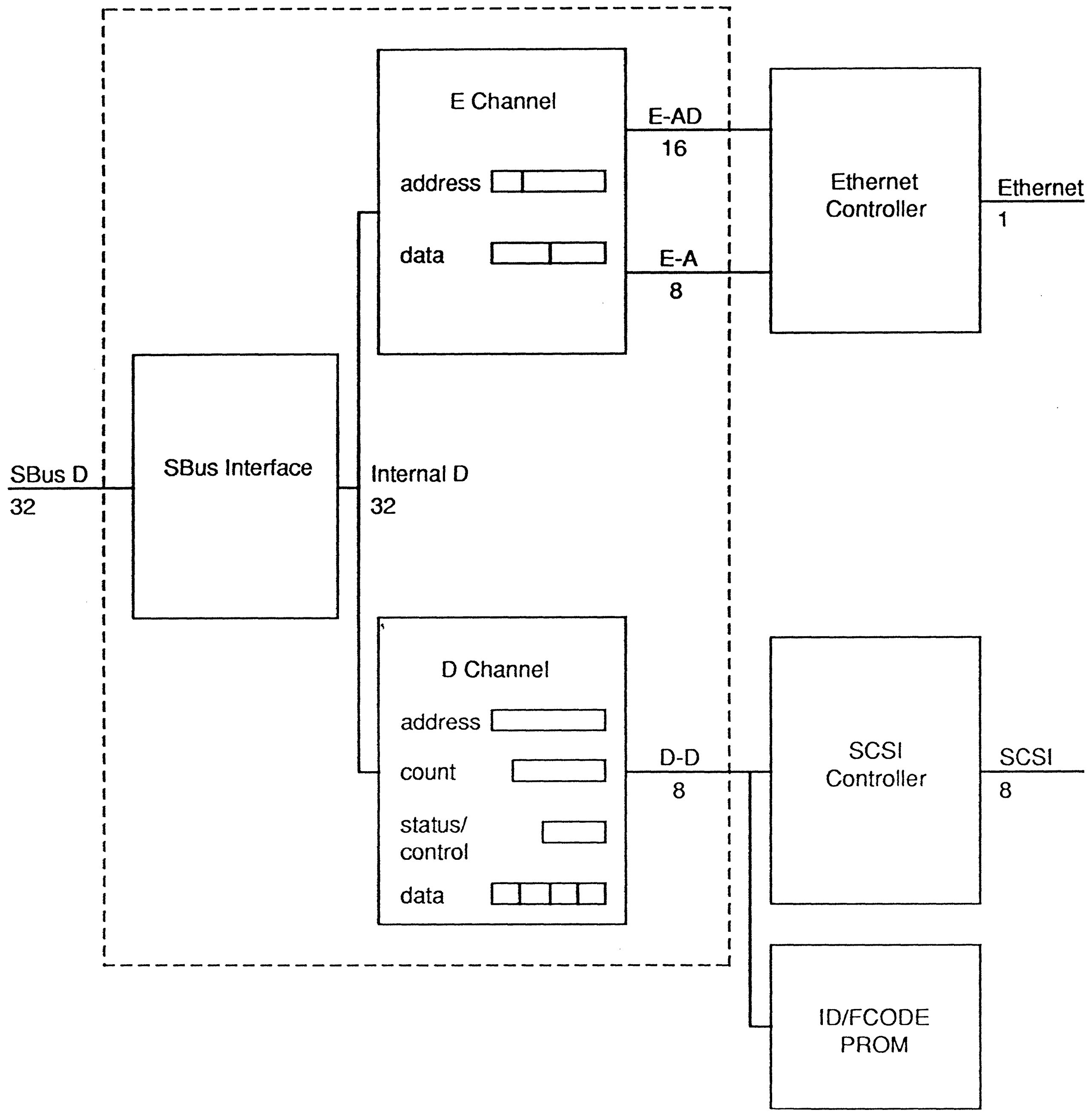


Figure 1. SBus DMA Data Path

WRITE: Read/Write (Bit 8, Read/Write). The state of this bit determines the direction of D channel DMA transfers. When set, the direction of transfer is from I/O peripherals towards S bus memory, (Sbus memory write); if clear, the direction is from Sbus memory to peripheral (Sbus memory read).

EN_DMA: Enable DMA (Bit 9, Read/Write). When set, the D channel state machine will respond to DMA request signals from the D channel controller. When clear it will not respond,and the transfer is in effect suspended.

REQ_PEND: Request Pending (Bit 10, Read Only). This bit becomes set at the time a D channel controller first makes a DMA request, and remains active until the block transfer completes by reaching its terminal count. This bit can be used to poll the state of the transfer.

BYTE_ADDR: Byte Address (Bits 12–11, Read Only . This two-bit field contains the lowest two bits of the address of the next byte to be accessed by the D channel controller.

EN_CNT: Enable Counter (Bit 13 , Read/Write). The set state enables operation of the Byte Counter Register. Since TC is a function of the byte count, this bit also implicitly enables TC.

TC: Terminal Count (Bit 14, Read Only). This bit is set when the byte counter makes a transition from 000001 to 000000. TC is cleared by RESET or FLUSH.

Reserved: Bits 31-15. This register is accessed as a 32 bit value, via an Sbus word access, even though not all 32 bits are currently used.

D channel Address Counter (read/write)

31 – 0
address count

This counter is written with the starting address of a D channel DMA transfer, to or from SBus memory. The counter increments by 1 or 4 depending on the size of the transfer. The contents of this register are not defined by RESET.

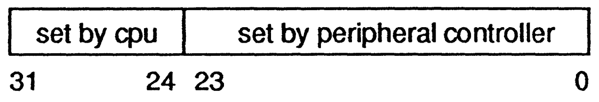
D Channel Byte Counter (read/write)

31 – 24	23 – 0
00000000	byte count

The 24-bit byte counter contains the number of bytes left to move in the current D channel DMA transfer. When the counter decrements to zero it will generate an interrupt , if INT_EN is set. Its action is enabled by setting EN_CNT in the SCR. The contents of this register after RESET are not guaranteed to be any particular value.

For the D channel and E channel external registers, additional address lines are connected directly to the peripheral controllers, as needed.

E Channel Address Register



The E Channel Address Register contains the Sbus virtual address sent out during E channel S bus accesses. The lower 24 bits come from latching the address sourced from the E channel peripheral controller in its bus master mode.The upper 8 bits can be set by a CPU register access to the address location specified in the memory map. On powerup or Sbus reset, the value is initialized to value hex ff.

Slave Accesses (Reads and Writes to Internal and External Registers)

1) A CPU LOAD or STORE causes the Sbus Select (SEL*) connected to the DMA controller to be asserted. An Sbus physical address intended for the DMA controller and address strobe are also active.

2) The DMA controller looks at two bits of the physical address to determine whether the access is for 1) Sbus ID, 2)DMA controller internal registers, 3) D channel external registers, or 4) E channel external registers.

3) The sequence depends on the type of register accessed:

   3.1 If the access is to the internal ID, that 32-bit value is enabled for read back. If the access is to an external ID/Fcode PROM, a sequence takes place using the D controller's 8 bit data bus, by which a byte is read back.

   3.2 If the access is to a 32-bit internal value, the data on Sbus D[31:0] is read from or written to an internal register, specified by Sbus address lines PA[3:2].

   3.3 If the access is to the D channel external registers, the data transfer over the Sbus must be of size byte.The data is passed between the Sbus and D channel data D_D[7:0], through the DMA controller without any buffering in the Pack/Unpack register. The DMA Controller asserts the appropriate slave access signals to the peripheral controller, such as D_CS* and D_RD*.

   If the access is to the E channel external registers, the data transfer over the Sbus must be of size halfword.The data is passed between the Sbus and E channel data E_AD[15:0], through the DMA controller without any buffering in the Pack/Unpack register. The DMA Controller asserts the appropriate slave access signals to the peripheral controller, such as E_CS* and E_READ.

4) Data is transferred from the appropriate Unpack register until the register is empty.

5) The DMA controller generates an Sbus acknowledge corresponding to the size of the transfer.

6) If a slave access is attempted while either the D or E channels is actively accessing the Pack/ Unpack registers, the DMA controller will assert Rerun Acknowledge.

Signal Descriptions

SBus Interface Signals

$\overline{\text{ACK}}$[2:0]: SBus Acknowledge (bidirectional, active low, TTL input levels, internal pullup, 4ma output drive, Pins 15, 77 and 14 where $\overline{\text{ACK}}$[2] is Pin 15). The SBus Acknowledge is asserted by the DMA Controller to indicate its response to an SBus selection when in slave mode.

$\overline{\text{AS}}$: SBus Address Strobe (input, active low, TTL level, internal pullup, Pin 82). The SBus Address Strobe being active signals that a valid address is on the PA lines.

$\overline{\text{BR}}$: SBus Request (3-state output, active low, CMOS, 4ma output drive, Pin 13). The DMA Controller drives SBus Request low to request control of the SBus.

$\overline{\text{BG}}$: SBus Grant (input, active low, TTL level, internal pullup, Pin 12). The SBus Grant being active signals the DMA Controller that it has been given control of the SBus by the SBus arbiter.

CLK: SBus Clock (input, Pin 44). SBus Clock is derived from the main SBus system clock.

D[31:00]: SBus Data (bidirectional, TTL input levels, internal pullup, 4ma output drive, Pins 120-115, 113-106, 102-93, 90-83 respectively where D[31] is Pin 120). SBus Data are the data lines of the SBus.

$\overline{\text{INTREQ}}$: SBus Interrupt Request (open drain output, active low, 4ma output drive, Pin 14). SBus Interrupt Request is used to interrupt the CPU at the completion of a transfer or the occurrence of an error.

$\overline{\text{LERR}}$: SBus Late Error (input, active low, TTL level, internal pullup, Pin 11). The SBus Late Error signal can be used to indicate an error in an SBus cycle. It does not become active until after the cycle has already finished.

PA[3:1]: SBus Physical Address, Low Order (input, TTL level, internal pullup, Pins 4-6 where PA[3] is Pin 4). The DMA controller decodes the pattern on these lines when responding as a slave to determine which register in a group of registers is to be accessed.

PAX, PAY: SBus Physical Address, High Order (input, TTL level, internal pullup, Pins 9-10 where PAX is Pin 9). The DMA Controller decodes the value on these lines to determine whether a slave access is to an Internal Register, D Channel Register, E Channel Register, or Fcode Prom.

RD: SBus Read/Write (bidirectional, internal pullup, 4ma output drive, TTL input level, Pin 79). SBus Read/Write signifies the direction of data transfers. High signifies Read, low indicates Write.

$\overline{\text{RESET}}$: SBus Reset (input, active low, TTL level, internal pullup, Pin 78). SBus Reset is used to initialize the state of the DMA Controller.

$\overline{\text{SEL}}$: SBus Select (input, active low, TTL level, internal pullup, Pin 81). SBus Select being active signifies that the DMA Controller is being selected as a slave during an SBus cycle.

SIZ[2:0]: SBus Transfer Size (bidirectional, active high, TTL input levels, 4ma output drive, internal pullup, Pins 16-18 with SIZ[2] being Pin 16). The SBus Transfer Size lines are used by an SBus master to specify the number of bytes to be transferred on that bus cycle.

$\overline{\text{D\_IRQ}}$: DMA Interrupt Request (input, active low, TTL level, internal pullup, Pin 40). DMA Interrupt Request is asserted by the D Channel Controller to signal data transfer completion or other event.

$\overline{\text{D\_RD}}$: DMA Read Strobe (CMOS output, active low, 4ma output drive, Pin 45). DMA Read Strobe becomes active in two cases: in conjunction with $\overline{\text{D\_CS}}$, to perform a read access of the D Channel Controller's internal registers, or in conjunction with $\overline{\text{D\_ACK}}$ during an actual DMA SBus write transfer.

D_RESET: DMA Reset (CMOS output, 4ma output drive, Pin 39). DMA Reset can be used to reset the D Channel Controller.

D_REQ: DMA Request (input, TTL level, Pin 47). DMA Request is asserted by the D Channel Controller to request the transfer of a byte during a DMA transfer.

$\overline{\text{D\_WR}}$: DMA Write Strobe (CMOS output, active low, 4ma output drive, Pin 46). DMA Write Strobe becomes active in two cases: in conjunction with $\overline{\text{D\_CS}}$ to perform a write access of the D Channel Controller's internal registers, or in conjunction with $\overline{\text{D\_ACK}}$ during a DMA SBus read transfer.

$\overline{\text{ID\_CS}}$: Secondary Device Select (bidirectional, active low, TTL input level, 4ma output drive, Pin 21). Pulling Secondary Device Select high signifies the existence of an external PROM to the DMA Controller. It is driven by the DMA Controller to select an external PROM.

$\overline{\text{SLOW}}$: Fast/Slow DMA Acknowledge (input, active low, TTL level, internal pullup, Pin 1). Fast/Slow DMA Acknowledge being pulled low adds a two-clock delay to D Channel cycles.

VDD: Power (Pins 7, 19, 43, 61, 72, 104).

VSS: Ground (Pins 8, 30, 42, 52, 71, 80, 91, 105, 114).