VME-Eclipse CPU (VIP10) Specification

PANKAJ PATEL

Silicon Graphics, Inc.

VME Eclipse CPU (VIP10) Specification

VME-Eclipse CPU (VIP10) Specification

PANKAJ PATEL

Silicon Graphics, Inc.

1. INTRODUCTION

This document specifies the architecture and programming interface of the VME Eclipse CPU including the local bus peripherals (serial I/O, timers, clock, memory) and the graphics DMA channel. A substantial portion of the control logic for these circuits is embedded in two 144 -pin PPGA gate arrays which are also described here.

The VIP10 is a 20 Mhz processor. Also the instruction cache is 64K bytes and the data cache is 32k bytes.

2. COMPATIBILITY WITH IP10

With the exception of the VME Arbitration scheme, VME bus timer, and VME to local memory address mapping, the VIP10 is identical to the IP10 Eclipse CPU. The VIP10 has a hardware option of VME interrupt level 7 or Location Monitor. The Reset Switch and Ethernet Non-volatile RAM that were located on the Console PC Board in the Eclipse have been moved onto the main VIP10 board. The VIP10 can also be reset from the VME reset line on P1 connector while the IP10 only drives the reset line.

The SCSI bus connector has been replaced by low profile high density SCSI II connector on VIP10. Also the two genral purpose serial ports connectors are replaced by high density 3-row 15-pin Sub-D connectors on VIP10 while the IP10 uses regular 9-pin Sub-D connectors for all three serial ports.

3. COMPATIBILITY WITH IP4

With the exception of the interrupt structure, the VIP10 is designed to be compatible with the IP4 to the extent of allowing the same kernel to boot the machine.

The interrupt registers of the VIP10 differ from their IP4 counterparts. The local interrupt register is expanded to accomodate additional sources of interrupt and provided with a mask (whereas the local interrupts on the IP4 cannot be masked). Also, the mask register has no effect on the status register (whereas on the IP4 only unmasked bits are readable). The whole interrupt circuit is implemented in an 44-pin PLCC.

However, the VIP10 has additional functionality beyond the IP4 (the ethernet and printer interfaces, and the graphics DMA channel). (All the address maps in this document have been copied directly from the IP4 documentation, with all exceptions / additions being explicitly called out).

Another minor departure from IP4 is the real-time clock. A different chip is used from the IP4 SmartWatch package for reasons of cost and user-serviceability of the battery.

4. CPU ARCHITECTURE

4.1 BIT AND BYTE NUMBERING CONVENTIONS

The MIPs processor as configured for use in VIP10 is big-endian for byte-numbering within the word. Bit-numbering, however, is little-endian by convention. Thus the least-significant bit is bit 0, whereas the least-significant byte is byte 3. All DMA transfers proceed in increasing order of byte address: therefore the most significant byte is the first byte written in the 32-bit word.

5. PHYSICAL ADDRESS SPACE

Address Range	Size	Usage
0xffffffff 0x40000000	3 Gb	unused (will cause read bus error or write interrupt)
0x3fffffff 0x30000000	256 Mb	VME A32, D8:16:32, Extended Supervisory Data Access (On VME bus, address 0x0 forced as highest nibble)
0x2fffffff 0x20000000	256 Mb	VME A32, D8:16:32, Extended Non-priv. Data Access (On VME bus, address 0x0 forced as highest nibble)
0x1fffffff 0x1fc00000	4 Mb	Boot PROM
0x1fbfffff 0x1f000000	12 Mb	Local I/O (Duarts, Timers, etc.)
0x1effffff 0x1e000000	16 Mb	VME A24, D8:16:32, Standard Non-priv. Data Access
0x1dffffff 0x1df00000	8 Mb	VME IH(1-7), D8:16, Interrupt Acknowledge cycles
0x1deffffff 0x1d200000	6 Mb	unused (but will map into either VME A16 or VME IACK)
0x1d1fffff 0x1d100000	1 Mb	VME A16, D8:16:32, Short Non-priv. Data Access
0x1d0fffff 0x1d000000	1 Mb	VME A16, D8:16:32, Short Supervisory Data Access
0x1cffffff 0x1c000000	16 Mb	VME A24, D8:16:32, Standard Supervisory Data Access
0x1bffffff 0x10000000	192 Mb	VME A32, D8:16:32, Extended Non-priv. Data Access (On VME bus, address 0x1 forced as highest nibble)
0x0fffffff 0x00000000	256 Mb	Local Memory

Accesses to physical addresses above `0x3fffffff` (labelled as unused above) and accesses to Local Memory addresses beyond the maximum size of 16/32Mbytes are illegal addresses and are detected and reported via bus errors on reads and interrupts on writes. Note that if a write occurs in the address ranges `0x01000000` to `0x0fffffff` or `0xJ0000000` to `0xJfffffff` (where J is within 0x4 to 0xf) which are illegal addresses, the write will cause an interrupt and the write will overwrite a Local Memory address computed by AND'ing the address with `0x00ffffff`.

Note that the (cheaper) VME implementation on the VIP10 does not allow the CPU to access local memory by using addresses which alias to local memory through VME as on the IP4. Such accesses time out.

6. VIP10 BUSES

The block level architecture of the VME Eclipse CPU is shown in figure 1. The CPU has two independent buses. The MIPs CPU drives the local bus which interfaces to the caches, floating point co-processor, local memory, and local peripherals. The interface control logic for all these devices resides in the CTL1 gate array. The CPU local bus is a 20 Mhz bus optimized for high-bandwidth (160 Mbytes/sec) interactions between the MIPs CPU and the caches. It is capable of performing two pipelined cache accesses per clock cycle. Accesses to slower devices (local memory and peripherals) result in stall states produced by the control logic in the CTL1 array. A one-deep write buffer accepts isolated writes at CPU speeds, but any other memory or peripheral accesses that are attempted before the buffered write completes are stalled until the write completes. This implementation preserves the order of actual execution of reads and writes (unlike the MIPS write buffer).

Another bus, namely the I_O bus, connects the I/O devices and the VME interface to memory through a second port. This port also gives the CPU access to I_O bus-resident devices. The I_O bus has a burst mode of operation providing a peak bandwidth of 40 Mbytes/sec, especially targeted to supporting fast pixel moves to and from the frame buffer.

I/O channels for SCSI, ethernet, and the printer are implemented in a gate array called the IOC1. This is a 144-pin PPGA device that contains the necessary interface logic and DMA channels to allow the connection of the SCSI and ethernet controllers to the I_O bus with a minimum of additional glue logic. The IOC1 also contains a DMA channel to support the interface to a printer/audio input-output circuit.

6.1 I_O BUS

The I_O bus is a 10 Mhz multiplexed synchronous bus connected to a port into memory supporting burst mode of operation at upto 40 Mbyte/sec. (This is also the peak sustained bandwidth of the memory operating in burst mode).

6.1.1 I_O BUS TRANSFERS

I_O bus transfers are controlled with the following set of signals:

1. **ioads** : address strobe (Master drives)
2. **iodir** : direction (Master drives)
3. **iowait** : slave wait (Slave drives)
4. **iodly** : master delay (Master drives)
5. **ioack** : acknowledge (Slave drives)
6. **iobe(0:3)**: byte enables (Master drives)

Each access on the I_O bus consists of an address phase and a data phase. The address phase is one clock in duration. In this phase, the bus master drives the address, byte select, address strobe, and direction lines. The byte enable and direction lines remain valid through the data phase as well.

The data phase follows the address phase and is at least one clock in duration. The actual duration of valid data on the bus depends on a pair of handshake lines. **iowait** is driven by the slave , and **iodly** is driven by the master. During a read, the slave drives **iowait** in the clock after the address phase if data is not valid during that cycle. The slave can extend the data phase indefinitely by asserting **iowait**. The first clock of valid data during a read is indicated by **iowait** being deasserted by the slave during that clock. However, the data phase can still be extended at this stage by the master asserting **iodly**. The last clock of the data

phase has both **iowait** and **iodly** deasserted.

During writes, the handshake lines are used in the opposite way, with the master asserting **iodly** to delay the start of valid data on the bus and the slave asserting **iowait** to delay the end of valid data on the bus.

6.1.2 I_O BUS BURST MODE

The burst mode is an efficient way of transferring sequential address locations without having to send the address across for each word transferred. It is supported with a page-mode memory implementation. Its only use on the bus is for dma transfers to the graphics subsystem.

In this mode, the CPU memory controller uses a built-in DMA channel to transfer a burst of data to the graphics subsystem. The start of the DMA burst is indicated by the assertion of **ioads** as in normal cycles along with the **grxreq** line. Data is handshaked with the usual lines except that the **grxreq** can turn off to suspend the transfer. During graphics subsystem accesses, the **iodly** signal is pipelined a clock in advance of its normal timing to allow the graphics subsystem to synchronize it before use.

6.1.3 I_O BUS ADDRESSING ERRORS

The default state of the **iowait** signal when it is not driven is unasserted. This is accomplished by the use of a pull-up on the signal. Therefore there is no time-out mechanism required, and the result of an addressing error is a one-clock data phase. (The VME channel provides a time-out mechanism for the case of an addressing error over the VME bus.) However, the I_O bus does require a mechanism to signal addressing errors. This function is performed by the **ioack** line. During I_O bus accesses, one of three interfaces must be selected: the IOC1, the graphics subsystem, or the CTL1 chip in the CPU. The selected interface drives the **ioack** line in the clock after the address strobe. If no interface is selected, the CTL1 array senses that the **ioack** line is unasserted and causes an interrupt on writes and a bus error on reads.

6.1.4 I_O BUS ARBITRATION

The arbiter resides in the CTL1 chip on the CPU and provides a request-grant pair for the IOC1.

The arbiter has a two-level structure to parallel the dual-clock architecture of the Eclipse CPU. The I/O request, internal DMA request, and refresh request are passed through an arbiter running off the I_O bus clock which generates a 'unified' request to the CPU's arbiter. The CPU arbiter runs synchronously to the CPU to minimize cpu latency, and generates CPU grants or I/O grants allowing sufficient time between turning off the I/O grant and turning on the CPU grant to allow synchronization of the I/O grant deassertion to the I/O bus clock.

Since the refresh counter is also used to generate VME bus timeouts, the arbitration scheme allows refresh to occur during VME access from the cpu. This has the benefit of ensuring the integrity of memory during "lboot" which rapidly times out one location after another. Finally, as a safeguard, the refresh counter is allowed to run even when refresh is disabled (so that shutting off refresh does not kill the bus timeout function).

During burst mode DMA transfers, a bus monitor mechanism is used to guarantee a minimum bus bandwidth allocation to the cpu in the presence of burst transfers. This is described below in the section on graphics DMA.

6.1.5 I_O BUS CLOCK

The I_O bus is clocked synchronously with the ethernet controller. A two-phase clock is used to allow reliable operation in the presence of clock skew between devices on the VME Eclipse CPU and also between boards.

6.2 VME BUS

6.2.1 CPU VME MASTER INTERFACE

The VIP10 acts as a master on the VME Bus by performing read or write cycles to the VME address ranges specified in the Physical Address Space table. The VIP10 is capable of generating VME A32:24:16, D32:16:8 and IACK cycles. The address range specifies the VME address size and the data type referenced specifies the VME data size.

Interrupt Acknowledge (IACK) cycles are used to service a pending VME interrupt. An IACK cycle is generated by performing a read within the IACK address range with the address bits A3, A2, and A1 set to a value ranging from 1 to 7 indicating what level VME interrupt is being handled. The data returned contains information identifying the interrupting board. Note that the VIP10 cannot generate a D32 cycle for an IACK cycle because when A1 is one, it would attempt to address a 32 bit word on an odd halfword address and this is not allowed by the cpu.

The VIP10 differs from the IP6 in cacheability of VME-resident memory. On the IP6 any memory on the VME is capable of being cached if accessed through the space for cached access. On VIP10, due to the cache parity implementation which copies local memory parity on cache loads, incorrect cache parity is generated on cached VME references. This causes subsequent references to the address to cause a cache miss. In effect, this precludes the use of the cache for VME references.

6.2.2 VME ARBITER

The VIP10 VME bus arbiter is a Round Robin (RRS) arbiter, where as the IP10 VME bus arbiter is a single-level arbiter at level 3. The VIP10 is a Release When Done master (RWD) where as IP10 is Release On Request (ROR) master.

The VIP10 VME bus arbiter is implementated in the pals as compared to IP10 implementation which is done in the CTL1 gate array. The VME BUS REQUEST3 and the VME BUS BUSY lines going into the CTL1 gate array are used as control lines Internal Request (INTREQ) and Internal Bus Busy (INTBBSY). A pal monitors the status of the CPU to decide if it needs the VME bus. If the CPU needs the VME bus, then it drives the VME REQUEST LEVEL 3 on the VME back plane. When it gets a grant on VME Grant Level 3 from the VME ARBITER, then it drives the INTREQ and INTBBSY inactive and monitors the status of the VME Bus Busy from the VME back plane. When it sees the Bus Busy go active it then activates the INTREQ and still continues to monitor the back plane Bus Beasy. When it sees the Bus Busy go away it activates the INTBBSY. This effectively makes the CPU Release When Done (RWD) master.

Another pal monitors the VME Request(3:0) lines and arbitrates the VME Bus using Round Robin algorithm (RRS) and drives the VME Grant(3:0) lines accordingly. When the CPU is reset, the arbiter resets and starts arbitrating at level 3.

The VIP10 can reside in slot one and arbitrate the VME bus or reside in any other slot and just become a requester on the VME bus. This option is switch selectable. When the VME Bus Arbiter is enabled the four request lines from the VME back plane are multiplexed to the arbiter pal. When the Arbiter is disabled the four grant lines from the VME back plane are multiplexed to the arbiter pal and the arbiter pal passes the grants thru.

6.2.3 VME READ-MODIFY-WRITE REGISTER

Read-modify-write cycles to VME are supported by VIP10 for compatibility with IP4. The VME RMW flag can be set as follows:

Name	Address	R/W	Comments
VMERMW	1fa60000	R/W	Set VME RMW flag

A VME Read-Modify-Write cycle is generated by 1) Reading the VMERMW address to enable a RMW cycle, 2) Performing a VME Read, and 3) Performing a VME Write. Other non-VME bus cycles may be performed between steps 1 and 3. Step 3 also clears the VME RMW flag so that subsequent VME cycles will not be Read-Modify-Write.

The VME RMW flag is also cleared by init (power-up and reset).

The following addresses have been added to the VIP10 address space for strobes used to turn on and off control bits. The state of these bits can be inspected by reading the status register of the VIP10.

The one strobe in common with IP4 is at the same address (set VME RMW flag). Note that on VIP10, however, these addresses respond to a write as well as a read in the same way.

Name	Address	R/W	Comments
VMERMW	1fa60000	R/W	set VME RMW flag
ACTPUP	1fa60004	R/W	turn on active bus pullup
VMEFBON	1fa60018	R/W	turn on vme fast bustimout
VMEFBOF	1fa6001c	R/W	turn off vme fast bustimout
ENRASO	1fa60024	R/W	enable CTL ras decoder

The VMEFBON / VMEFBOF strobes turn on / off the fast timer option. This feature is designed to facilitate vlsi testing of the CTL1 gate array and has no use in normal operation. The ACTPUP strobe is provided to eliminate 64 termination resistors on the pcb. When this strobe is read, the active pullup circuit inside CTL1 is enabled. This circuit detects when the I_O and dram data buses are idle (floating) and enables the VME bus data onto them. The effect of this circuit is to always have some buffer enabled on these buses, preventing extra power dissipation in the CMOS inputs connected to the bus.

6.2.4 VME TO LOCAL MEMORY ACCESS

The VIP10 supports access from VME bus masters to local memory in either single or block transfer mode with either A24 addressing or A32 addressing.

In A24 mode, only the low 4Mbytes of local memory are accessible to VME masters in non-priviledged mode. There is no priviledged access to local memory in A24 mode. In A24 mode, the entire 16Mbytes of VME memory can be switch mapped at any 4Mbyte boundary.

In A32 mode, the low 16Mbytes of local memory is accessible to VME masters in either priviledged or non-priviledged mode. In A32 mode, the entire 4Gbytes of VME memory can be switch mapped at any 16Mbyte boundary.

The memory cycle time for VME master accesses is as follows:

1. Single cycles: 800 ns - 1 us. The smaller number is for the case when the IO bus is already owned by an IO-resident device other than the CPU when the VME cycle begins (this happens when the system is lightly loaded, for instance). For 32-bit transfers, this is a bandwidth of 4 Mbytes/sec.

2. Back-to-back cycles: 600 ns, once the IO bus is acquired. This cycle time is achieved by not releasing the IO bus for each access. However, to qualify as back-to-back cycles, AS* for the new cycle must be reasserted within 200 ns of Dtack. A bus timer mechanism (the same one used for graphics DMA) preempts the transfer at a (programmable) maximum time to allow latency intolerant devices like ethernet to get through.

   For 32-bit transfers, this is a bandwidth of 8 Mbytes/sec.

3. Burst cycles: 400 ns, once the burst gets under way. this cycle time is achieved by the use of page mode memory cycles. Once again, this number is based on DS* for the new cycle being asserted within 200 ns of Dtack. Also, the same bus timer mechanism prevents the VME master from hogging the bus. Note that the VME spec restricts burst transfers to 256-byte boundaries. The VIP10 implementation actually allows bursts of upto 4K bytes aligned on page boundaries.

   For 32-bit transfers, this is a bandwidth of 10 Mbytes/sec.

In the case of VME burst requests simultaneous with graphics DMA, VME gets priority. This means that as long as there is any pending transfers on VME, the graphics dma is held off the bus. The bus timer mechanism applies globally, that is, it ensures that the totality of VME and graphics DMA channels get

only a certain proportion of the bus bandwidth. This implies that the higher priority channels slow down the lower priority channels when they become active.

6.2.5 LOCATION MONITOR

The VIP10 can be interrupted by another VME bus master using Location Monitor (LM) feature in A24 addressing mode only. The Location Monitor is new to VIP10. The top 256 bytes of the 4Mbytes in A24 address space are reserved for location monitor. An access to any of these locations will set the Location Monitor interrupt. The Location Monitor interrupt can be cleared by reading the following adrress.

Name	Address	R/W	Comments
CLRVLM	1f8c0000	R	Clear VME Loction Monitor Interrupt

The VIP10 supports Location Monitor at the expense of regular VME Interrupt level 7. That is VIP10 can be interrupted by maskable 1 thru 7 regular VME interrupts like IP10 or by maskable 1 thru 6 regular VME interrupts and maskable location monitor as described above. The above options are hardware options (i.e. soldering zero ohm resistor on the corresponding line on the back of the board.)

6.2.6 PROGRAMMABLE VME BUS TIMER

The VIP10 supports a switch programmable VME Bus Timeout Timer. The programmable timer is new to VIP10. It monitors the VME Bus **DS0,DS1, DTACK** and **BERROR** signals. It can be programmed at 32, 64, 128, or 256 micro-seconds timeout periods. The CTL1 also has a fixed (order of milli-second) VME Bus timeout timer. The programmable VME Bus Timer effectively overrides the CTL1 fixed timer. The programmable bus timer is disabled if the VIP10 is not a VME Arbiter.

6.2.7 VME SYSTEM CLOCK

The VIP10 provides a high drive 16Mhz System Clock on the VME back plane. The System Clock is switched off if VIP10 is not a VME Arbiter.

7. LOCAL PERIPHERALS

7.1 CPU CONTROL REGISTER

The CPU Control Register is accessed as follows:

Name	Address	R/W	Comments
CPUCTRL	1f880002	R/W	CPU Control (8-bit)

All 8 bits are cleared to zero by init (power-up and reset). Here's the control register bit assignment:

CPU CONTROL REGISTER

15	14	13	12	11	10	9	8	7 6 5 4 3 2 1 0
FPER	DOG	BAD	ARB	SLA	RPAR	SIN	SERDATA	RES7-0

RES0-7 reserved: this field must be zero when writing

SERDATA serial memory data out.

SIN SYSINIT - Force the vmebus SYSRESET line to be asserted. Note that vmebus SYSRESET line causes the INIT line to be asserted which resets the entire IP6 including the MIPs processor. The effect of setting this bit is similar to powering the system down and up.

RPAR Enable parity checking and error reporting on reads from the local memory. When cleared parity errors are ignored.

SLA SLAVE - Allow VME bus masters to access the IP6 memory as a slave. Otherwise all VME bus access attempts are ignored.

ARB Clear to enable the arbiter on CTL for a single external request level. Set to disable CTL vme arbiter (for cpu with external arbiter)

BAD GENBAD - Force bad parity to be written into local memory.

DOG ENABWDOG - Set to high to enable watchdog timer. If enabled for the full timeout period, the timer activates the WDOG signal which forces initialization of the VME bus and IP6. Set ENABWDOG to low to reset the watchdog timer. While ENABWDOG is low, timeout is disabled.

FPER fast peripheral cycle: set to 1 for 12.5 Mhz operation

7.2 CPU AUX CONTROL REGISTER

The CPU Aux Control Register is accessed as follows:

Name	Address	R/W	Comments
CPUAUXCTRL	1f8e0000	R/W	CPU Control (8-bit)

The aux control register (different from IP4) controls the on-board led nibble, the console led, the console memory chip select, and resets the graphics subsystem. The register powers up cleared. In this state the graphics subsystem reset is asserted, which means of course that it must be deasserted before any attempt is made to access the graphics subsystem.

The led bits in their power-on state turn on the leds. The console led must be turned off at the end of successful completion of power-on diagnostics.

CPU AUXILLARY CONTROL REGISTER

31	30	29	28	27 26 25 24
GRXRST\	SERCLK	CONS.CS	CONS.LED\	CPU.LED\

GRXRST\ clear to assert reset to the graphics subsystem

SERCLK serial memory / XILINX clock

CONS.CS console memory chip select (active high)

CONS.LED\ clear to turn on console LED

CPU.LED\ four-bit CPU LED; clear a bit to turn on the corresponding LED

7.3 CPU STATUS REGISTER

The IP4 SYSID register has two bits: the LSB reads the serial SYSID prom data, and the next lsb shows the co-processor present status. A write to the SYSID address also clears the serial PROM address counter, which increments as a side effect of each read of the SYSID address.

On VIP10, the serial novram output replaces SYSID prom output on the lsb, and the next lsb still reads the co-processor present status. However, the mechanism for reading the serial memory is different, involving the serial clock and chip select as described above. Also the other bits of the byte read other useful status which is either invisible on IP4 (like the VME RMW flag) or simply non-existent (like the DMA and serial memory controls); and byte 0 of the SYSID address is the memory configuration register.

Name	Address	R/W	Comments
SYSID	1f800001	R	Coprocessor Present status.

With that, here's the register:

CPU STATUS/SYSID REGISTER:

23	22	21	20	19	18	17	16
VMERMW	DMARST	DMARDY	DMAEN	DMAERR	VMEFBT	FPPRES	SERDATA

VMERMW asserted during a vme read-modify-write cycle.

DMARST asserted during a (software) reset of the graphics DMA channel

DMARDY asserted at the end of DMA transfer

DMAERR asserted if the DMA terminated in an error condition

VMEFBT asserted when vme fast bus timeout / accelerated refresh is active

FPPRES floating point co-processor present indicator

SERDATA serial memory data output state

7.4 LOCAL MEMORY

The CTL1 array supports upto 256 Mbyte of physical memory. Other processor implementations can add memory by replacing the RAS decoder with a PAL that decodes RAS lines generating multiple RAS signals to refresh banks in parallel; however, the first VIP10 rev has provision for two memory banks for a total capacity of 32Mbytes using 4Mb Drams.

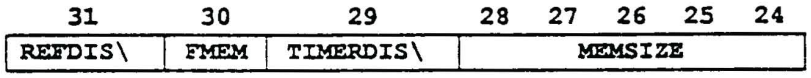
7.4.1 MEMORY CONFIGURATION REGISTER

The VME Eclipse CPU has a memory configuration register used for programming the number of stall states for memory accesses and memory size. The register defaults to 0 on power-up and must be set to the approprate value based on processor speed and memory configuration. This register is located at the following address:

Name	Address	R/W	Comments
MEMCFG	1f800000	R/W	memory configuration register (8-bit)

The format of this register is shown below:

MEMORY CONFIGURATION REGISTER



REFDIS set to zero this bit disables memory refresh

FMEM set to one this bit reduces the CAS pulse width on reads by one clock for 12.5 Mhz operation

TIMERDIS set to zero this bit stops the timer used for refresh, pulse stretching on SERCLK access, and the 60 sec watchdog

MEMSIZE this field sets the size of physical memory. The msb must be turned on for 4M DRAMs, off for 1M DRAMs. The ls nibble gives the fraction of memory that is stuffed. For IP6 this fraction cannot exceed 1/4. Other CPU's could be implemented using the CTL array with upto 64Mbyte of 1M DRAMs or 256 Mbyte of 4M DRAMs.
The setting of the low nibble is as follows:
0000 - 1/16 populated
0001 - 2/16 "
0010 - 3/16 "
0011 - 4/16 "
0100 - 5/16 "
n - n+1/16 "
This means that the register can never specify an empty memory.

The refresh counter and timer generate a variety of slow timing signals within CTL1. The timer produces a carry every 64usec if the TIMERDIS\ bit is set to 1. This generates a refresh request. If refresh is enabled (the REFDIS\ bit is set to 1) then a refresh cycle follows (after a bus acquisition delay). The refresh cycle is implemented as a burst during which dram is accessed sequentially four times with the row address being incremented after each access. The entire burst lasts 1.6us. The refresh counter actually has more bits than are required for row address. In fact, the refresh counter generates two carries: one which occurs once every millisecond, used by the vme bustimer, and another which occurs once every 16 seconds, used by the watchdog timer. (These timers further divide by four so that the vme bus timeout period is 4 msec and the watchdog timer timeout period is 64 sec, very approximately).

The entire counter is accessible at the REFADR address. Note that once the REFDIS\ and TIMERDIS\ bits are set, the timer is running continously and provides a precise way to measure time.

Name	Address	R/W	Comments
REFADR	1fa40004	R/W	Refresh counter/timer (32-bit)

For CTL1 testing, the following addresses allow the timer and refresh counter to be incremented (for obvious reasons, the REFDIS\ and TIMERDIS\ bits must be 0).

Name	Address	R/W	Comments
INCREF	1fa6002c	R	increment refresh
INCTIM	1fa60030	R	increment timer

7.4.2 RAS DECODER OPTION

This is a CTL1 configuration option which allows the use of an internal decoder for generating RAS(0:3). This option is not recommended to be used on VIP10. The VIP10 has an external PAL to decode RAS to eight banks, in which case the mode bit is left in its power-on state. (The four pins on CTL1 used for driving the RAS out to memory bank are encoded select lines allowing CTL1 to select one of the sixteen possible banks.)

7.4.3 ERROR ADDRESS REGISTER

The Error Address Register is accessed as follows:

Name	Address	R/W	Comments
ERRADR	1fa40000	R	Error Address Register (32-bit)

The error address register gets loaded when there is a memory parity error, regardless of the bus master that caused the error. If there are multiple sequential errors the register holds the address of the first error. In the case of errors during access from the I/O channel multiplexer, the top nibble of the error address indicates which device was active when the error occurred (audio, printer, ethernet, scsi respectively in bits 31,30,29,28).

This register differs from the IP4 error register in a couple of respects. It is loaded only on parity errors, and always holds the error address no matter which bus master was accessing memory.

7.4.4 PARITY ERRORS

The Parity Error Register is accessed as follows:

Name	Address	R/W	Comments
PARERR	1faa0005	R	Parity Error Register (8-bit)
CLRERR	1faa000n	R	Clear bit "n" of Parity Error Register

7.4.4.1 PARITY ERROR REGISTER

VIP10 checks parity during reads from its local memory. When a parity error occurs the parity error register is loaded with info on which byte(s) had a parity error. The register also has sticky bits that identify for what kind of accesses parity errors have occured. The Parity Error Register (8-bit) may be read at address PARERR. Each of the access type bits that caused a parity error may be individually cleared by reading the CLRERR address substituting "n" as follows:

n (hex)	Read Result
0	Clear GDMA access bit and byte parity bits
1	Clear DMA access bit and byte parity bits
2	Clear CPU access bit and byte parity bits
3	Clear VME access bit and byte parity bits

Bit assignments in the parity error register are:

23	22	21	20	19	18	17	16
0	1	2	3	VME	CPU	DMA	GDMA
Parity bits				Access types			

1. The four parity bits (high means error) indicate which byte(s) within the longword generated the most recent parity error.
2. The VME, CPU, DMA and GDMA bits (high means error) indicate which of the four types of local memory access have generated a parity error since the bit was last cleared. Here DMA stands for any of the devices controlled by the I/O channel multiplexer, and GDMA stands for graphics DMA.

Clearing any of the access bits also has the effect of clearing the parity error flags.

7.4.4.2 Detecting When Parity Errors Occur

Parity Errors may be detected on read accesses to local memory by the CPU, VME or DMA. For all parity errors, the error status is saved in the Parity Error Register.

For DMA accesses, parity errors do not notify the processor directly - the processor may poll the parity error register at the completion of a DMA block transfer from memory to determine whether any errors occurred. If one is detected, the processor can clear the appropriate bit of the register and then do one of the following:

1. Read the memory locations which were accessed during the block transfer to see if any of the locations still has bad parity (see CPU accesses below).
2. Retry the block transfer again noting whether the parity error occurs.

For CPU accesses, the processor receives a bus error during the read access for which a parity error occurs. The processor can use its internal state information to find the address at which the parity error occured.

For VME accesses to local memory, the VIP10 asserts the VME bus error signal during the read access for which a parity error occurs. The VME master which receives the bus error can abort the read and take appropriate action (probably notify the VIP10).

7.5 BOOT PROM

The Boot PROM is accessed as follows:

Name	Address	R/W	Comments
BOOT	1fcnnnnn	R	Boot PROM address "nnnnn" (32-bit)

The size of the Boot PROM is 64K 32-bit words or 256K bytes. The address "nnnnn" (hex) maps into "A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 0 0" (binary) where the 16 bits address one of 64K 32-bit words.

7.6 DUARTS

The VIP10 has two serial ports dedicated to the keyboard and mouse interface, and two general-purpose serial ports with full modem control (as against four on the IP4). Both duarts are driven by a 3.686Mhz clock.

Each of the duarts generates an interrupt. These interrupts are multiplexed with other local interrupts as described below in the section on local interrupts.

The VIP10 duarts are accessed as follows:

Name	Address	R/W	Comments
DUART0	1fb800n0	R/W	Duart 0 address "n" (8-bit)
DUART1	1fb800n4	R/W	Duart 1 address "n" (8-bit)

The address "n" (hex) maps into "A4 A3 A2 A1" (binary) which correspond to the address inputs to the 2681 duarts.

The Keyboard and Mouse ports are implemented with a 24-pin 2681 duart, DUART0. The 2 full modem control serial ports are implemented with a 40-pin 2681 duart, DUART1. The serial port assignments are shown below:

HARDWARE PORT NAMES	DUART#	Base Addr	Channel
Keyboard	DUART0	1fb80000	A
Mouse	DUART0	1fb80000	B
Port 1	DUART1	1fb80004	A
Port 2	DUART1	1fb80004	B

The Duart input and output pin name to signal assignments are shown below:

DUART	CHANNEL A		CHANNEL B		CLOCK OUTPUT	
DUART0	pin	signal	pin	signal		
Inputs	RXD0	RXDKBD	RXD1	RXDMOUSE		
Outputs	TXD0	TXDKBD	TXD1	TXDMOUSE		
DUART1	pin	signal	pin	signal	pin	signal
Inputs	RXD0	RXD1	RXD1	RXD2		
	IP0	CTS1	IP1	CTS2		
	IP3	DCD1	IP2	DCD2		
			IP5	Baud Clock		
			IP6	Baud Clock		
Outputs	TXD0	TXD1	TXD1	TXD2		
	OP0	RTS1	OP1	RTS2		
	OP4	DTR1	OP5	DTR2		
					OP3	Baud Clock

7.7 WATCHDOG TIMER

The watchdog timer is controlled by the ENABWDOG bit of the cpu control register. If the ENABWDOG bit is set (high), the system (VIP10 and VME bus) is reset (without loss of memory contents) when a timeout interval of approx. 67 seconds elapses after the bit is enabled. Note that the ENABWDOG bit is cleared upon system reset and therefore is disabled following powerup.

Note that the refresh timer and refresh counter inside CTL1 need to be enabled inorder for the watchdog to work. These counters are enabled by bits in the memory configuration register.

7.8 SYSTEM TIMER

The TIMER functions are accessed as follows:

Name	Address	R/W	Comments
TIMER	1fb4000n	R/W	TIMER Registers address "n" (8-bit)
CLRTIM0	1fa20000	R	Clear Timer 0 Interrupt
CLRTIM1	1fa00000	R	Clear Timer 1 Interrupt

The TIMER address accesses the internal registers of an 8254 counter chip.. The address "n" (hex) maps into "A1 A0 0 0" (binary) where "A1" and "A0" are the address inputs to the 8254.

Reading the CLRTIM0 address clears Timer 0 Interrupt, and reading the CLRTIM1 address clears Timer 1 Interrupt.

The 8254 programmable counter/timer includes 3 separate counters. Counter 2 is driven by a 3.6864 MHz clock and its output clocks the other two counters. The outputs of counters 0 and 1 generate level 2 and level 4 cpu interrupts respectively.

7.9 BATTERY BACKED UP REAL TIME CLOCK

The VIP10 watch is different from the IP4. The VIP10 watch is implemented using the National 8573 real-time clock chip.

The real-time clock is accessed as follows:

Name	Address	R/W	Comments
WATCH	1fbc00nn	R/W	watch address (8-bit)

The address "nn" (hex) maps into " 0 A4 A3 A2 A1 A0 0 0" (binary). All data written to the RAM is preserved by the battery backup when the system power is turned off.

7.10 SYSTEM ID PROM, ID PROM, SYSTEM CONSOLE

The implementation of ID memory on VIP10 differs from IP4 / Clover.

The device used in this application is the 93CS56 serial EEPROM, which will be mounted on the VIP10 CPU Board. The CPU Board also provides a power-on indicator LED, a status indicator LED, and reset switch in addition to the serial memory. On VIP10 all this circuitry appears on the CPU board. On the Eciplse IP10 these functions were on the System Console board.

The serial memory is operated by sending a "command" and address in serial form. The command selects the operation to be performed, such as "read", "enable write", "write", "disable write", etc. The hardware provided for the serial memory interface therefore is completely general, allowing different device sizes / command codes to be accomodated by software.

The serial memory data is accessed through the CTL1 gate array. The data out to the serial memory is driven from the SERDATA bit in the control register in CTL1. The data output of the serial memory appears in the SYSID register. The chip select is driven from the SERCS bit in the auxilliary control register, and the serial clock is driven from the SERCLK bit in the auxilliary control register.

To access the serial memory is somewhat tedious but fortunately the serial memory is seldom accessed.

The procedure involves turning on the serial chip memory chip select (SERCS) and then writing the address/command word, msb first, in bit serial fashion to bit 8 of the cpu control register while toggling the serial clock. (Note that the 93CS56 requires a leading '0' at the start of the bit stream for each command). The inner loop looks like this:

```
{ read-modify-write the control register to get the next bit in;
  delay for 4 usec;
  SERCLK <- 1;
  delay for 4 usec;
  SERCLK <- 0 ;
}
```

In the case of writes, this is followed by a similar sequence to get the data out to the serial memory. In the case of reads, this is followed by the sequence:

```
{ SERCLK <- 1;
  delay for 4 usec;
  read the sysid register to get the next bit of the word
  SERCLK <- 0
  delay for 4 usec;
}
```

Finally, the SERCS bit is turned off. Note that the serial memory is organized as 16-bit words, so that the read loop must shift 16 times.

Note that writes must be preceded by the ENABLWR comand and followed by the DISABLWR command.

This last step is necessary because writes must be disabled before a power down to maintain the integrity of data in the EEPROM.

The PRE input on the serial memory is tied to the console CONS_LED bit. This means that to read / write the serial memory the console led must be on (a 0 in the corresponding bit), whereas to set the protected register the led must be off.

Also note that the 93CS56 chip requires the 'PRE' pin to be low for the entire duration of a write (including the programming time).

7.11 IP6 INTERRUPTS

7.11.1 MIPS INTERRUPT INPUTS

The CPU chip has six, level-triggered interrupt inputs. On the VIP10, these interrupts are wired to the following functions:

- Level 0 interrupt is connected to the seven levels of VME Bus interrupts (IRQ1 through IRQ7).
- Level 1 interrupt is connected to interrupts from local devices on the IP4.
- Level 2 interrupt is generated by interrupts from timer0.
- Level 3 interrupt is generated by the floating point coprocessor.
- Level 4 interrupt is generated by interrupts from timer1.
- Level 5 interrupt is used to report write errors. This interrupt occurs when a write operation attempts to access an illegal physical address. Refer to the sections on Physical Address Space and Memory Configuration for more information. This interrupt also occurs when a VME Bus Error occurs while the IP4 is a VME master writing to the VME Bus.

Note that the six interrupt signals have a different function when the VIP10 reset signal is active; they provide information to the cpu about various configuration options.

7.11.2 LOCAL I/O INTERRUPT STATUS REGISTER

The local I/O interrupt circuit on IP6 differs from IP4 in three respects: first, a mask register has been provided to selectively mask individual interrupt inputs; second, interrupts have been added for ethernet, audio, and the graphics subsystem, expanding the status register to 10 bits; and finally, the interrupts have been rearranged so that their position in the word reflects their relative priority so as to simplify the prioritization in software .

The Local I/O Interrupt Status Register is accessed as follows:

Name	Address	R/W	Comments
LIOSTAT	1f980002	R	Local I/O Interrupt Status (10-bit)
LIOMASK	1f98000b	R/W	Local I/O Interrupt Mask (8-bit)

The Local I/O Interrupt Status Register has a bit equal to zero (active low) for each pending interrupt listed below:

```
Bit 0   DUART0                                  Mask bit 0
Bit 1   DUART1                                  Mask bit 1
Bit 2   Retrace                                 Mask bit 2
Bit 3   Centronics                              Mask bit 3
Bit 4   SCSI chip                               Mask bit 4
```

```
Bit 5   ethernet                                          Mask bit 5
Bit 6   GE interrupt                                      Mask bit 6
Bit 7   GE fifo full                                      Mask bit 7
Bit 8   VME ACFAIL                                        Mask bit 'ACF'
Bit 9   Vertical status (no interrupt: status only)
```

The mask register has 8 bits for interrupts 0 - 7. The ACFAIL interrupt is masked through a separate bit in the VME interrupt mask register. VERT STAT has no mask bit and cannot generate an interrupt. It is provided for the CPU to poll.

A '0' in a given mask position disables the corresponding interrupt level.

The fifo full interrupt has some special properties to allow its use in polled mode and as an interrupt source. When this bit is enabled as an interrupt (unmasked) then the corresponding status bit stays on once it is turned on by a fifo full condition. This allows the interrupt routine to determine the cause of the interrupt even if the fifo is no longer full when the interrupt routine is entered. Once the interrupt routine recognizes the source of the interrupt as the fifo full signal then it can mask the interrupt and poll the status bit. Once masked, the status bit loses its latching property and simply follows the state of the fifo full signal (after synchronizing delays, though).

7.11.3 VME BUS INTERRUPT STATUS & VME BUS INTERRUPT MASK REGISTER

The VME interrupt status register differs from its IP4 counterpart in the following way: the mask bit only masks the interrupt-generating capability of the interrupt input; regardless of the state of the mask, the interrupt can still be polled by reading the VME interrupt status register.

The VME bus Interrupt Status and Mask Registers are accessed as follows:

Name	Address	R/W	Comments
VMESTAT	1f840003	R	VME Interrupt Status (8-bit)
VMEMASK	1f84000b	R/W	VME Interrupt Mask (8-bit)

Writing a one into any of bits 7 to 1 of the Interrupt Mask enables a VME bus interrupt level to interrupt the CPU chip at CPU interrupt level 0. When one or more unmasked VME interrupts is received, the Interrupt Status Register has a bit set corresponding to each pending VME interrupt that is also enabled by the mask. The VME interrupt levels are assigned to bits of the Mask and Status Registers as follows:

7	6	5	4	3	2	1	0
IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	ACF

Here the 'ACF' bit is used to mask the acfail interrupt going to the local interrupt multiplexer.

7.11.4 INTERRUPT CONTROLLER

The vme interrupt and mask register, and the lio interrupt and mask register, along with the interrupt multiplexers are implemented in the INT1 gate array.

7.12 SWITCHES AND JUMPERS

There are four switches and two zero ohm jumpers on VIP10.

(1) Switch S2 - positions (8-1) are used for selecting the A32 slave VME address. Position (8-1) corresponds to address bits (A31-A24) respectively (off=1/on=0).

(2) Switch S3 - positions (2-1) are used for selecting the A24 slave VME address. Position (2-1) corresponds to address bits (A23-A22) respectively (off=1/on=0).

(3) Switch S3 - position (4) is used for enabling(off) / disabling(on) the VME Arbiter.

(4) Switch S4 - position (1-4) are used for selecting the Programmable VME Bus Timeout time periods. Position (1-2-3-4) corresponds to timeout period of (32-64-128-256) micro-seconds respectively. Maximum of only one position switch should be on at any time. All off represents timer off condition.

(5) Resister R303 selects VME IRQ-7 Interrupt while resistor R304 selects VME Location Monitor. Either one of them have to be stuffed on the board.

(6) Resist0r R33 selects IO-BUS clock (10 MHz) for the SCSI bus controller while R34 selects 16 MHz Clock. Again either one of the two have to be stuffed on the board.

Configuration information is stored in the serial non-volatile memory.

7.13 GRAPHICS CHANNEL

The graphics interface provides a DMA channel for transfers between raster memory and the CPU's local memory.

The graphics subsystem is connected to the I_O bus through a set of transceivers controlled by the CTL1 array.

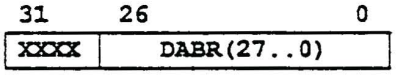
DMA transfers are initiated by the CPU first setting up the graphics subsystem by directly writing control registers in the graphics subsystem which enable the DMA transfer (this is described in detail in the HQ / GE5 specification) and then setting up the CTL1 array with the memory address of the table of memory descriptors.The transfer is started by the CPU writing to a control port. The CTL1 array then requests the I_O bus, and handshakes each word across the interface until the terminal count is reached for each of the descriptors in the list.

7.13.1 PROGRAMMING INTERFACE

The DMA is programmed through a descriptor array in memory pointed to by a hardware channel register. The registers and memory structures are described below.

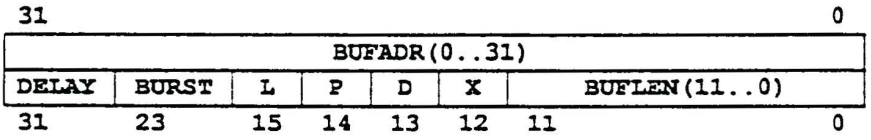
[1] Eclipse CPU Registers: the following registers are provided to work the dma interface:

1. Descriptor Array Base Register(DABR):



Note that the DABR holds a full physical address of the base of the table.

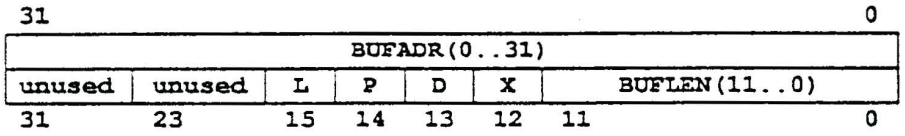
2. Channel Registers: the buffer descriptor gets copied into the channel registers shown below by the DMA channel as it chains through the descriptor array (except for the BURST field which is loaded statically by the CPU). DMA bursts are regulated by the burst and delay length register, shown below:



L	1 indicates end of list
P	1 selects pause after channel load (diagnostics)
D	DMA direction (1 moves to graphics)
BUFLEN	buffer length register
BURST	length of dma burst
DELAY	length of dma delay

All these registers are read/write registers but under normal operation the CPU only programs the DABR and the BURST registers, with the DMA hardware inside CTL1 loading the remaining fields from memory.

[2] Descriptor Array: each entry in the descriptor array has the following structure:



L	1 indicates end of list
P	1 selects pause after channel load (diagnostics)
D	DMA direction (1 moves to graphics)
BUFLEN	buffer length

Note that it is possible to have a descriptor table that spans any number of pages in memory. These pages will always be contiguously allocated, so that the dma hardware can address the descriptor table entries by simply incrementing physical addresses without regard to mapping.

The BUFADR register cannot count past a page boundary.

The BUFLEN field of the descriptor specifies the number of bytes in the buffer as follows: BUFLEN = (number of lwords - 1) * 4. This means that the two lsbs of the buffer length are always zero and are not stored in the hardware.

It is possible to set up a sequence of descriptors that change direction (interleaved reads and writes). Note that at the end of a DMA the descriptor array base register inside CTL1 points to the next (physically adjacent) descriptor. This means that if descriptor lists are allocated sequentially in physical memory the cpu does not need to reload the descriptor array base pointer after every dma completes.

The graphics channel is programmed through the following addresses:

Name	Address	R/W	Comments
DABR	1fa40008	R/W	Descriptor Array base (32-bit)
BUFADR	1fa4000c	R/W	Buffer Address Register(32-bit)
BUFLEN	1fa40012	R/W	Buffer Length Register (16-bit)
BURST	1fa40010	R/W	Burst / delay register (16-bit)
STDMA	1fa60008	R	start dma
STOPDMA	1fa6000c	R	stop dma
RSTGRX	1fa60010	R	reset graphics
UNRGRX	1fa60014	R	unreset graphics
RLDBRST	1fa60020	R	reload burst

DMA is initiated by turning on the DMAEN bit using the 'start DMA' control strobe after setting up the table in memory and loading the DABR with a pointer to the table. The DMA hardware threads through the list of buffers until it reaches a buffer descriptor with the end-of-list bit set, transferring each buffer through the graphics interface. At the conclusion of the transfer, the DMA hardware sets the DMARDY bit, turns off the DMAEN bit, and idles. For normal operation, the cpu never need load the buffer address or length registers.

The pause bit is provided for diagnostic use only. Setting this bit in the descriptor causes the dma to pause after loading the channel. At this point, a diagnostic program can read the dma registers to check that they have been loaded with the correct values. The dma channel can be restarted again using the "start dma"

control strobe.

At any point in the DMA the cpu can pause the dma by reading the "stop dma" address. DMA will of course stop at burst boundaries, and can be resumed by reading the "start dma" address. With the channel stopped in this fashion, reading the channel registers will yield their current values.

7.13.2 BUS MONITOR

The burst counter (8-bit) sets a maximum burst length in 100 ns increments. At the end of this interval dma / vme bursts are terminated.

The delay counter (8-bit) sets a minimum bus grant period to the cpu in 100 ns increments. Following a dma burst, and after any pending i/o requests are completed (disk, ethernet), the cpu is guaranteed to own the bus for this period of time whether it needs the bus or not. During this time it cannot be preempted by dma or i/o.

This hardware is designed to allow dynamic switching of the bus loading depending on whether or not the cpu needs quick real-time response. The nominal setting will be 25.6 us bursts and 2us delay. During an interrupt service routine, the first task is to reload the bus monitor to increase the time allocated to the cpu and decrease the burst length. Thereafter, the routine can determine whether the interrupt to be serviced can tolerate less bus bandwidth or not and either leave the bus monitor alone or set it back accordingly.

Note that the burst and delay registers both count up towards 255, and therefore must be loaded with the 1's complement of the desired burst/delay count.

7.13.3 CHANNEL DIAGNOSTICS

In addition to the pause feature described above, the channel registers can be tested for correct incrementer operation using the following addresses:

Name	Address	R/W	Comments
INCCH	1fa60028	R/W	increment channel
INCDAB	1fa60034	R/W	increment chtabadr
INCBUR	1fa60038	R/W	increment burst
RLDBRST	1fa60020	R/W	reload burst

The INCCH address, when read, causes the buffer address to be incremented and the buffer length to be decremented. A read of the INCDAB address increments the DABR (descriptor array base register), and a read of the INCBUR address increments both the burst and delay registers.

The burst and delay registers are actually implemented as a register followed by a counter. For this reason, during diagnostics, the value loaded into the burst and delay registers cannot be directly read back without first transferring the value from the input register to the counter. This is done by reading the RLDBRST address following a write to the burst and delay registers.

8. I_O CHANNEL PERIPHERALS

8.1 ETHERNET INTERFACE

8.1.1 DESIGN GOAL

The goal of the ethernet implementation is to support the ethernet transfer rate of 10Mbit/sec with no data under- or over- runs.

8.1.2 ARCHITECTURE

The AMD7990 is used in 16-bit mode. The total addressible memory range is constrained by the AMD7990 architecture to no more than 16M bytes. However, the Eclipse CPU supports upto 256 Mbytes of memory. To allow the ethernet direct access to all of physical memory (thereby accomodating driver implementations that save an extra "mbuf" copy), addresses generated by the AMD7990 are mapped. The high nibble of ethernet address is ignored, and the next 8 bits of address are used as an index into a mapping table with upto 256 entries. This number is derived from hardware limitations in the AMD7990 which allow at most 128 buffers to be chained together for transmit and 128 for receive. Each map entry directs data to a region of memory no larger than a (4 Kbyte) page.

The AMD7990 has an on-board fifo that is 48 bytes deep. The AMD7990 bus interface is programmed to transfer data in bursts of 16 bytes each per bus acquisition / release. Each burst lasts for approximately 5 usec, with about 8usec between bursts. However, the maximum latency tolerance of the AMD7990 is the ethernet transfer time of 32 bytes, or about 26 usec.

The ethernet controller can be soft-reset separately by reading from a port address.

The ethernet controller's on-chip registers may be accessed by the CPU at any time without requiring synchronization to the state of DMA transfers. The IOC1 prevents deadlock by retaining ownership of the I_O bus throughout a DMA burst. The CPU can only acquire the I_O bus when the ethernet controller has released it (or not been granted it), and therefore must be in an idle state, able to respond to slave reads and writes.

8.1.3 ETHERNET PROGRAMMING INTERFACE

8.1.3.1 ETHERNET CONTROLLER REGISTERS

The internal registers of the AMD7990 ethernet controller are mapped as follows:

Name	Address	R/W	Comments
ETHERREGS	1f950n00	R/W	Ethernet chip address "n" (8-bit)
ETHERRST	1f960004	R	Ethernet chip reset on
ETHERRDY	1f960000	R	Ethernet chip reset off
ETHERWAIT	1f960008	R/W	Ethernet chip wait state control (4-bit)

The ETHERREGS address accesses the internal registers of the AMD7990 chip. The address "n" (hex) maps into "0 0 0 A0" where A0 is the address input to the AMD7990. The ethernet interrupt goes to the Local I/O Interrupt circuitry.

8.1.3.2 ETHERNET ADDRESS MAP

The ethernet address map is accessed as follows:

Name	Address	R/W	Comments
EMAP	1f920nnn	R/W	Ethernet address map address "nnn" (16-bit)

The address "nnn" (hex) specified in the EMAP address corresponds to "1 0 A7 A6 A5 A4 A3 A2 A1 A0 1 0" where "A7" to "A0" are an index into the page map which has 256 entries. The memory address for ethernet transfers is generated by concatenating the low-order 12 bits of the 16-bit address with 16 bits from the map entry indexed by the high-order 8 bits of the ethernet address. The ETHERWAIT register is a 4-bit register used to program the behaviour of the ethernet-to-IO bus interface inside the IOC1 during ethernet dma accesses. This register comes up cleared. The format of this register is shown below:

ETHERNET WAIT STATE CONFIGURATION REGISTER

27	26	25	24
RDWAITSL	RDY1	RDY0	WRWAITSL

RDWAITSL set to one this bit causes IO waits generated by CTL during ethernet memory reads to be passed on to the ethernet controller

RDY(1:0) this bit pair programs a fixed delay into the IOC for ethernet to memory accesses:
00 selects 1 wait state (default)
01 selects 0 wait states
10 selects 2 wait states
11 selects no wait states

WRWAITSL set to one this bit causes IO waits generated by CTL during ethernet memory writes to be passed on to the ethernet controller

The ethernet-to-IObus interface inside the IOC1 can work in one of two modes: in the wait mode, the interface is stalled by waits on the IO bus (RDWAITSL and WRWAITSL turned on above). In the programmed delay mode, the interface assumes a (programmable) fixed memory speed for wait state generation to the ethernet controller. This results in faster throughput since there are no synchronizing delays as in the wait mode. The RDY(1:0) register should be set to 11 (no wait state).

8.2 SCSI INTERFACE

8.2.1 DESIGN GOAL

The goal of this implementation is to maintain full software compatibility with the IP4 implementation while maintaining performance at the same level.

8.2.2 ARCHITECTURE

The WD33C93 SCSI controller provides the complete SCSI physical interface (with no external components) for the single-ended option. (Another version of the chip supports the differential option and synchronous protocol).

The WD33C93 is equipped with a simple byte-wide DMA interface. The IOC1 provides an address pointer and page map, and buffers bytes upto a dword before initiating a memory access. The address map has been extended in width to address a maximum of 256 Mbytes of address space; however, the additional bits added at the most-significant end of the map should be set to zero in existing software (making it compatible without need for any change).

For testability, the address pointer and map have been made read/write registers (where their IP4 counterparts are write only).

Buffers are required to be aligned on dword boundaries in memory. The DMA transfer may, however, terminate before a dword boundary is reached. The programming interface to the SCSI circuitry includes a way to flush partial dwords to memory for SCSI writes to memory.

Finally, CPU accesses to the SCSI controller's on-chip registers may be safely performed while DMA is in progress (as against the IP4 implementation, which prevents such accesses). The possibility of deadlock is ruled out by designing the IOC1 such that **dack** to the SCSI controller is never prolonged based on I_O bus arbitration. In other words, **dack** is not asserted unless the following read / write can take place regardless of the state of the I_O bus. This means that once the CPU starts a read / write to the SCSI controller's registers, the CPU is guaranteed to get in after the present **dack** is deasserted (if it is asserted) and before the next assertion of **dack**.

8.2.3 SCSI PROGRAMMING INTERFACE

8.2.3.1 SCSI CONTROLLER REGISTERS

The SCSI Control functions on the SCSI WD33C93 chip are accessed as follows:

Name	Address	R/W	Comments
SCSI	1fb00n01	R/W	SCSI chip Registers address "n" (8-bit)

The SCSI address accesses the internal registers of the WD33C93 chip. The address "n" (hex) maps into "0 0 0 A0" (binary) where "A0" is the address input to the WD33C93. The WD33C93 is hooked up to be accessed in the Indirect Addressing Mode.

The SCSI interrupt signal goes to the Local I/O Interrupt circuitry.

8.2.3.2 SCSI ADDRESS REGISTER AND ADDRESS MAP

The DMA Address Register is accessed as follows:

Name	Address	R/W	Comments
DMALO	1f900002	R/W	DMA Low Addr Reg (16-bit)
DMAHI	1f920nnn	R/W	DMA High Addr Reg at address "nnn" (16-bit)
DMAFLUSH	1f940000	R	DMA Flush bytes to memory (no data)
SCSIRST	1fa80004	R	SCSI controller reset
SCSIRDY	1fa80000	R	SCSI controller reset
SCSIBSTAT	1fa80009	R	SCSI byte status register (4-bit)
MAPINDEX	1f910003	R/W	SCSI address map index register (6-bit)

The DMA Address Register consists of an address counter which generates a longword offset address into a page and a page map that generates the page number for each DMA access.

Writing to the DMAHI address loads a value into the page map. The address "nnn" (hex) specified in the DMAHI address corresponds to "0 0 A7 A6 A5 A4 A3 A2 A1 A0 1 0" where "A7" to "A0" are an index into the page map which has 256 entries. DMA operations use the page number at index 0 first and then go up the indices sequentially each time the address counter reaches a page boundary. The page map should be set up before writing the DMALO address. The current index into the page map is reset to zero whenever the DMALO address is written.

In addition, the page map index register is accessible to the CPU. This is useful for diagnostics. The map index register is shown below:

7	6	5	4	3	2	1	0
7	6	5	4	3	2	1	0

The data written to DMAHI is formatted as follows. The low 16 bits represent the high address (bits 12 to 27) of a DMA address as shown below.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12

Writing to the DMALO address loads a value into the address counter and resets the byte-in-word counter to zero to align the transfer. Specifically, the lower 12 bits written to DMALO are written to the lower 12 bits of the address counter (bits 0 and 1 are ignored since DMA accesses are automatically aligned on a 4-byte boundary) as shown below (note: "x" marks ignored bits). Bit number 15 is the DMADIR flag which controls the direction of SCSI transfers, and is cleared by the soft resetfunction.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMADIR	x	x	x	11	10	9	8	7	6	5	4	3	2	x	x

DMADIR SCSI/DMA data direction flag - When low transfer data from memory to the SCSI chip (DMA WRITES SCSI). When high transfer data from the SCSI chip to memory (DMA READS SCSI).

In the case of setting up DMA WRITES SCSI, writing to the DMALO address also transfers the first four

bytes from memory to 4 byte gathering registers. Therefore the page map must be set up before the write to DMALO occurs.

Writing to the DMAFLUSH address forces the DMA to transfer 4 bytes from the byte gathering registers to memory if less than four bytes of data are held in the byte gathering registers. This should be performed after a DMA READS SCSI operation has received all its data from the SCSI bus.

The SCSIBSTAT register is the byte status register, displaying for each byte in the data register either a "0" (indicating valid data) or "1" (data flushed to memory / disk). This register is made visible for diagnostic reasons only.

8.3 CENTRONICS INTERFACE

8.3.1 DESIGN GOAL

The goal of this implementation is to provide an interface comparable in functionality and speed to the IKON parallel interface card currently in use in the Clover product line (without the Versatec option).

8.3.2 ARCHITECTURE

The IOC1 provides direct access and DMA to the printer port. The IOC1 contains a 32-bit data register that is accessible to the CPU. The CPU performs 'direct' writes to the printer interface by writing to this register, and the IOC1 sequences the actual handshake of one or more of the bytes in the data register (depending on bits in the control register) out to the printer interface.

In DMA mode, the CPU sets up an address pointer and a byte count and starts the printer channel by writing to the DMASTART address. The channel then reads memory, and handshakes the bytes across the interface. Completion of the transfer sets a bit in the status register and also generates an interrupt to the CPU.

The printer interface carries a set of printer status lines (namely, END-OF-PAPER, END-OF-INK, ON-LINE, and FAULT). These lines are received by the IOC1 and appear in the status register.

The CPU initializes the printer by reading the PRST address, waiting the required duration (~ 5ms), and then reading the PRDY address to turn it off. The reset signal is driven out to the printer on the INPUT-PRIME line.

The control register also has a bit to disable the printer interface. In this mode, the DMA channel is available to drive the audio circuitry. Additional control register bits specify the direction of transfer, and the sampling/playback rate in this mode.

A 256-entry map is available to the printer channel. The structure of the map and the map index are identical to the corresponding registers for the SCSI channel. The map index is automatically cleared when the DMALO address is written, and increments whenever the DMA channel crosses a page boundary. However, the map index is read/write, allowing it to be set to a different value after the write to DMALO. This feature is useful in conjunction with the double-rank register structure for double-buffered audio pattern playback or recording.

8.3.3 PROGRAMMING INTERFACE

The printer registers are accessed at the following location:

Name	Address	R/W	Comments
PRDMALO	1f9d0006	R/W	DMA Low Addr Reg (16-bit)
PRDMAHI	1f920nnn	R/W	DMA High Addr Reg at address "nnn" (16-bit)
PRDMADR	1f9f000c	R/W	DMA data Reg (32-bit)
PRDMACT	1f9c0002	R/W	DMA byte count (16-bit)
PRDMACN	1f9f000a	R/W	DMA control(16-bit)
PRDMAST	1f9c0205	R	DMA status (8-bit)
PRST	1f9f0004	R	turns on reset to channel
PRDY	1f9f0000	R	turns off reset to channel
DMASTART	1f9e000c	W	starts the channel
DMASTOP	1f9e0004	W	stops the channel
PRSWACK	1f9e0008	W	generates a "soft" acknowledge
PCHRLD	1f9d0000	W	reloads the printer channel registers
MAPINDEX	1f9e0003	R/W	Printer map index register (5-bit)
PRBSTAT	1f970001	R	Printer byte status register (4-bit)
A/DREG	1f9c0305	R/W	A/D I/O register (8 bit)
AOGNDAC	1f9c0005	W	Audio output gain control

Writing to the PRDMAHI address loads a value into the page map. The address "nnn" (hex) specified in the PRDMAHI address corresponds to "0 1 A7 A6 A5 A4 A3 A2 A1 A0 1 0"

Writing to the DMALO address loads a value into the address counter and resets the byte-in-word counter to zero to align the transfer. Specifically, the lower 12 bits written to DMALO are written to the lower 12 bits of the address counter (bits 0 and 1 are ignored since DMA accesses are automatically aligned on a 4-byte boundary) as shown below (note: "x" marks ignored bits). Bit number 15 is the DMADIR flag which controls the direction of AUDIO transfers:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMADIR	x	x	x	11	10	9	8	7	6	5	4	3	2	x	x

DMADIR AUDIO/DMA data direction flag - When low transfer data from memory to the AUDIO circuit. When high transfer data from the AUDIO circuit to memory.

The data register is read/write and byte addressible. The control register is read/write, and the status register read-only. The control register is accessed as a half-word (16-bit). The format of the control register is shown below:

15	14	13	12	11	10	9	8	7-2	1	0
ASL1	ASL0	DMABSY	LPEN	AUDEN	PINTEN	PINT	LPBUSY	x	LPFSL	ATEST

```
ASL(1:0)    audio clock speed select
            0: no clock; 1: 32Khz; 2: 16Khz; 3: 8Khz;
DMABSY      indicates dma operation in progress
LPEN        enables loop mode
AUDEN       audio enable. Enabling audio shuts off the printer interface
PINTEN      interrupt enable
PINT        pending interrupt indication on read;
            the interrupt is reset by writing a '1' to PINT.
LPBUSY      line printer busy status (for polling)
x           unused
LPFSL       clear to select IOC-generated LPF clock; set to select IOC
            LPF clock input
ATEST       test bit to facilitate vlsi testing. Set to speed up audio
            clocks
```

The format of the status register is shown below:

7	6	5	4	3	2	1	0
RES	RES	RES	RES	FAULT\	EOL	EOP	ONLINE

```
ONLINE    : printer on-line

EOP       : printer out of paper indication

EOI       : printer out of ink indication

FAULT\    : printer in abnormal state indication (active low)
```

The printer DMA low address (DMALO), byte count (PRDMACT), and mapindex (MAPINDEX) registers are based on a dual-rank register with an input register feeding a counter. The input register is always writeable by the CPU. However, the counter is loaded from the input register only when the CPU issues a START DMA write. Alternatively, (for diagnostics / testing) the CPU can write to the channel reload address to cause the counter to be loaded from the input register.

The input register feature is most useful to implement double buffering for audio input/output. In conjunction with the LPEN bit, it also allows the implementation of tone generation for audio output.

When the LPEN bit is turned on and DMA is started, the channel automatically reloads at the completion of the DMA. If the input registers have changed in the interim, the LPEN bit provides a means to smoothly switch from one buffer to another without CPU intervention on completion of one buffer. If the input registers are unchanged, the LPEN bit provides a means to sustain an output pattern for tone generation.

The PRBSTAT register is the byte status register, displaying for each byte in the data register either a "0" (indicating valid data) or "1" (data flushed to memory / printer). This register is made visible for diagnostic reasons only.

The printer interface cannot properly handle byte counts that are not a multiple of four. DMA must be set up with an integral number of words. Any additional bytes (< 4) can be individually written to the printer port using the data register (see below).

All printer / audio DMAs need to be explicitly stopped on completion of the DMA. Otherwise the DMA circuitry will start operating as soon as the count is loaded with a non-zero value.

A special form of DMA is supported for sequences of four or less bytes. In this case, a memory buffer is not needed, and the data bytes can be written directly to the data register. Only 1, 2, or 4 bytes can be output in this way, with the sequence always being lsbyte aligned. The processor must use the appropriate data width to write the data register because the byte enables on the bus are used to load the PRBSTAT register. The DMA address registers are not used in this form of DMA. The byte count must be set to the correct value before starting DMA. Also, the channel does not need to be explicitly stopped on completion as for 'regular' DMA.

8.3.3.1 AUDIO CLOCKS

The audio circuitry is driven by two clocks: the sampling/playback clock, and the LPF clock. These clocks are generated by the IOC1 chip. Three separate sampling rates may be programmed by setting control bits in the CPU aux control register: 32, 16 and 8 Khz, based on LPF cut-off at 13, 6.5, and 3.25 Khz. Alternatively, the LPF clock can be driven externally (allowing different filter cutoff rates to be tested in the lab) by flipping the LPFSL bit in the DMA control register.

For correct operation, the audio sampling clock must be turned off before DMA is started. Then DMA is started, and actual sampling (or playback) is begun only when the sampling rate bits in the control register (ASL(1:0)) are written with the required sampling rate. On completion of DMA, the sampling rate must be set to 0 to turn off the clock.

9. IMPLEMENTATION DETAILS

9.1 CTL1 ARRAY

The CTL1 array contains the control logic for the CPU write buffers, the local peripherals, the memory, and the I_O bus. In this sense the CTL1 is the hub of the Eclipse CPU, tying together all the different devices and interfaces. In addition, the CTL1 array contains some of the CPU's control registers.

The memory controller in the CTL1 array supports CPU accesses with 3 stall states (plus 1 fixup) for writes and 4 stall states (plus 1 fixup) for reads. For I/O bus accesses the memory controller supports page mode accesses with a 100ns cycle time for read / write.

The memory controller avoids resynchronization delays for I/O bus accesses by running the DRAM clocks (RAS, CAS) off the I_O bus clock for I_O bus accesses and off the CPU clock for CPU accesses. There is a resynchronization penalty when the memory changes hands from CPU to an I_O master, but once the arbitration is complete the memory accesses are synchronous, making high bandwidth transfers such as burst mode possible.

CTL1 controls two external write buffers. The write buffers themselves are implemented with octal TTL registers because the width of the buses makes it prohibitively expensive to integrate the write buffers into the CTL1 chip. Logically, the two write buffers are identical (in fact the two write buffers share the same address/tag register). One write buffer is used for local peripherals and the other is used for memory.

The CTL1 array has a bi-directional 32-bit port used to input address during CPU or I/O memory accesses, to output the memory address during DMA transfers, and to input or output data during CPU access to CTL1 registers.

9.2 CTL1 BUGS

The first production rev of CTL1 has a problem with the parity error checking circuitry which has been partially solved on the board. A full solution requires a new revision of CTL1.

This problem has been solved by restricting parity error checking to cpu accesses only. All writes to memory update the parity bits correctly, but only cpu reads check parity. In other words, IO reads of

memory do not check parity. This means that the access type bits in the parity status register for bus masters other than the cpu will never get set.

9.3 CTL1 EXTENSIONS

The CTL1 array supports upto 256 Mbyte of physical memory. Other processor implementations can add memory by expanding the (external) RAS decoder to decode additional RAS lines for memory banks beyond the 8 banks provided on VIP10.

The CTL1 array will support processor operation upto 20 Mhz. Provisions have been made to stretch memory and peripheral timings to handle the higher clock rate.

9.4 VIP10 PHYSICAL LAYOUT

The VME Eclipse (VIP10) CPU is implemented on three 6U (double height) VME, maximum 8 layer printed circuit board. The three boards are called 1) the CPU board 2) the MEMORY board and 3) the I/O board. Figure 2 shows the CPU board block diagram and the inter-connection between three boards, Figure 3 shows the MEMORY board and Figure 4 shows the I/O board block diagram. Figure 5 shows the front plate and the inter-connection cable arrangements of the three boards together. Functionally the boards are as follows:

The CPU Board:- 1) CPU (R3000) and FPU (R3010) PGAs 2) Instruction and Data caches 3) Boot Proms 4) Interrupt Controller 5) CTL gate array 6) Serial I/O DUARTS (2681) and Ports 7) Real time clock (NS8275) 8) Interval Timer (8254) 9) Reset switch and logic 10) VME Bus Arbiter and System clock.

The MEMORY Board:- 1) Memory Parity Generator Logic 2) 16/32 MBytes of Main Memory Banks (using 4Mx1 DRAM Chips) 3) RAS Decoder and Buffers 4) CAS Buffers 5) Address Buffers 6) Centronics Port

The I/O Board:- 1) IOC2 gate array 2) SCSI Controller (WD33c93) and Port 3) Ethernet Controller (7990) and Port 4) Centronics Interface 4) VME Interface 5) Graphics Private Bus Interface 6) Real Time Clock Battery. 7) Audio circuitry and jack plug 8) All DIP Switches.

The VME Eclipse CPU interfaces with external cables through a row of connectors that form the VME Eclipse Front panel. This is true for the SCSI Port, Ethernet Port, Parallel Port, the keyboard/mouse port and the two serial ports. The graphics bus has been moved to the unused P2 connector on the VME backplane to communicate with multiple copies of the PGR2 board set. This allows for multiple headed graphics systems.

A socketed Lithium battery powers the real-time-clock. The battery is expected to have a life of five years. The 5V Standby supply available on the VME backplane is dropped to 3V and ORed with battery supply line. This gives the flexibility to use the available 5V standby supply and/or the battery back up.

The cache memory is laid out for 64K instruction and 32K data caches using 16K X 4's, and 8K X 8's respectively.

All the DIP switches (S2, S3 and S4) are located on the I/O board. The SCSI controller clock selector jumper resister (R33 & R34) are located on the back of I/O board. And the VME IRQ7 and VME Location Monitor selector jumper resistors (R303 & R304) are located on the back of CPU board.

There are three fuses on the board set. One on CPU board and two on I/O board. On CPU board F1, 1amp fuse is connected between 12V supply line and the 12V output to keyboard and mouse port pin 7. On I/O board F1, 2amp fuse is connected between 12V supply line and the 12V output to Ethernet port pin 13 and F4, 2amp fuse is connected between the 5V line and the 5V output to SCSI port pin 50.

9.5 CTL1 GATE COUNT

The CTL1 array is implemented using the LMA9239C channel-free array from LSI. It has about 8400 gates and is packaged in a 144 plastic PGA.

9.6 IO CIRCUITRY

This section describes the external connections of the IOC1 to the ethernet, SCSI and printer interfaces.

The IOC1 interfaces the channel bus to the I_O bus. The channel bus has a 16-bit data path hooked to the ethernet controller, the SCSI controller, and the printer and audio data latches. Chip-select signals, DMA handshake signals, and slave address lines are driven by the IOC1 with no additional glue logic required.

The printer interrupt is generated by the IOC1 whereas the SCSI and ethernet interrupts are generated directly by the corresponding controllers.

A single 1K X 16 static RAM array provides the map for all channels. The ethernet, SCSI, and audio channels are assigned 256 locations each. A point of interest concerns the connection of the data pins of the static RAM. For ease of address generation, the data I/O pins of the RAM are connected to the bits (12:25) of the data bus. However, for compatibility with IP4, the CPU must see the RAM on the low bits of the data bus. This sticky situation is solved by a shifter inside the IOC1 which is used to shift the RAM data bits before presenting them to the CPU, so that it appears that the RAM is hooked to the lsb of the bus.

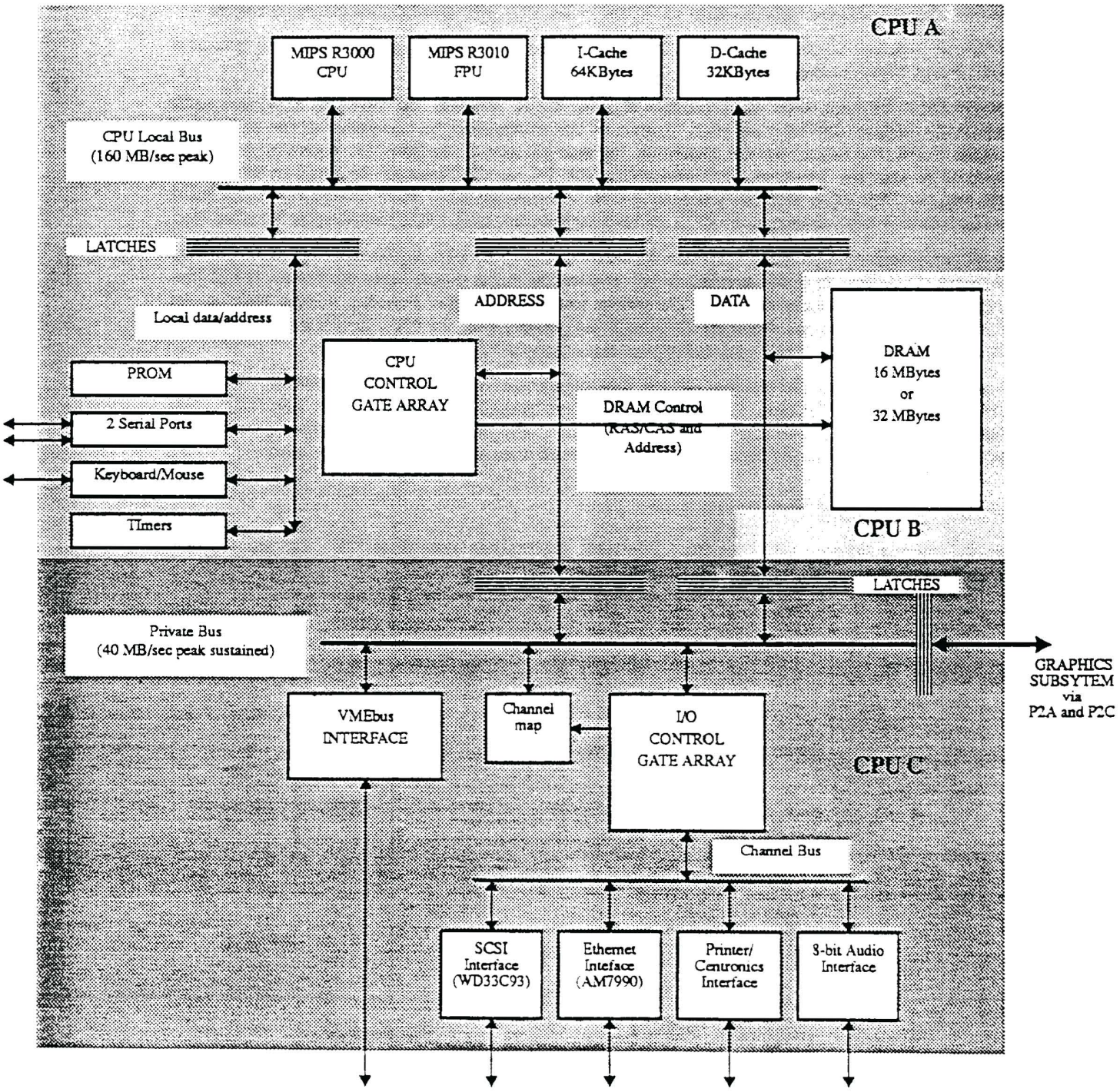
The physical interface to the printer is realized externally through an 8-bit register and high-current driver. The external interface logic also terminates and receives the status signals from the printer, and buffers other control signals driven by the IOC1. The printer status lines are input through a tri-state buffer on the data lines of the channel bus. When the CPU reads the printer status, the IOC1 supplies some of the bits from an internal register and also reads the printer status buffer to fill in the remaining bits.

The audio circuitry is based on an 8-bit A/D and D/A circuit. The input to the A/D is formed by mixing the microphone input, the line input, and the D/A output and passing them through a low-pass filter (MF6 switched capacitor filter). The filter output also drives the power amplifier. The D/A input to the mixer is conditioned by a programmable attenuator allowing software to set the output level for playback.

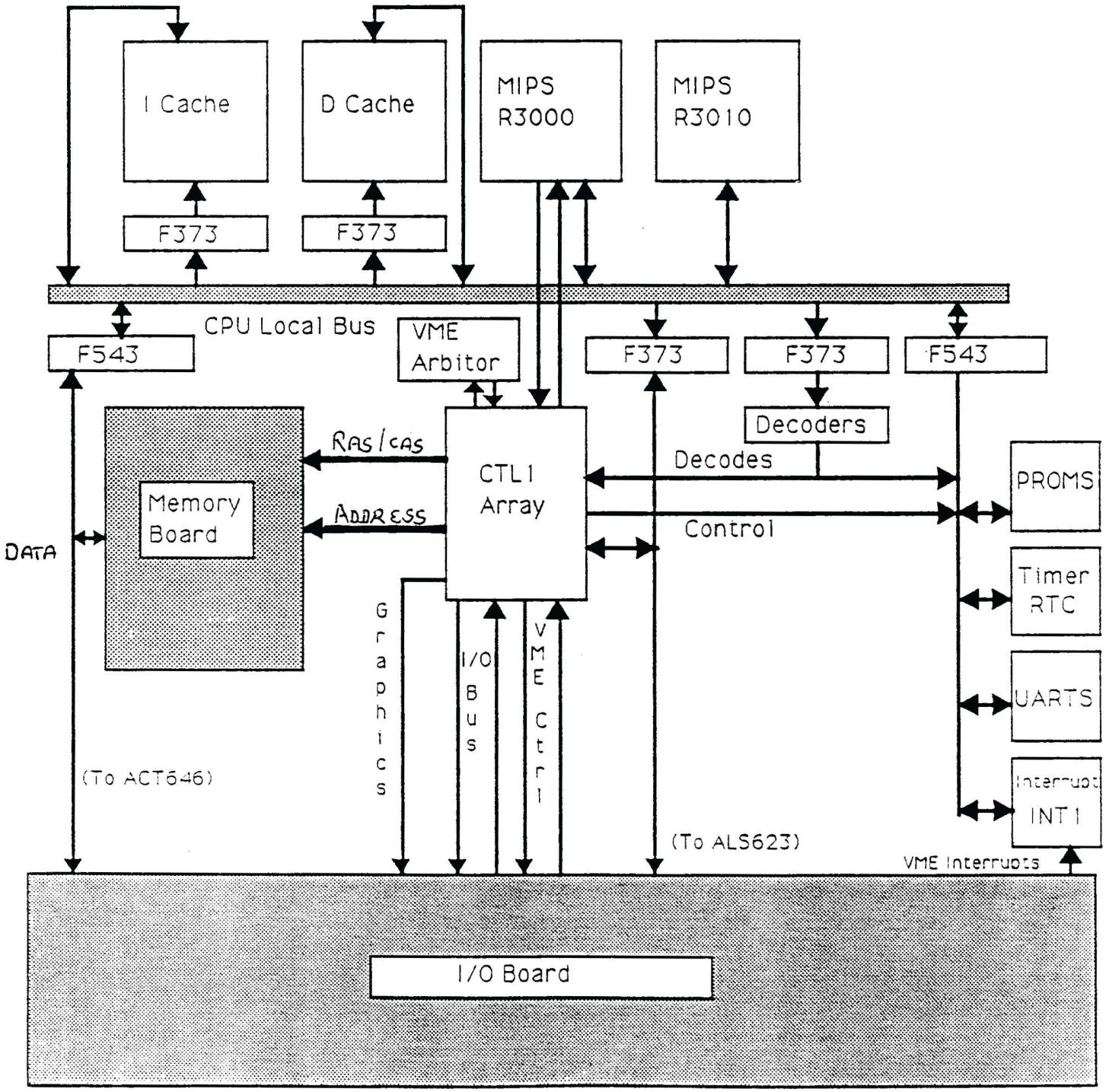
9.7 IOC GATE COUNT

The IOC array is implemented using the LMA9190C channel-free array from LSI. The gate count is approximately 6200, and the package is a 144-pin plastic PGA.

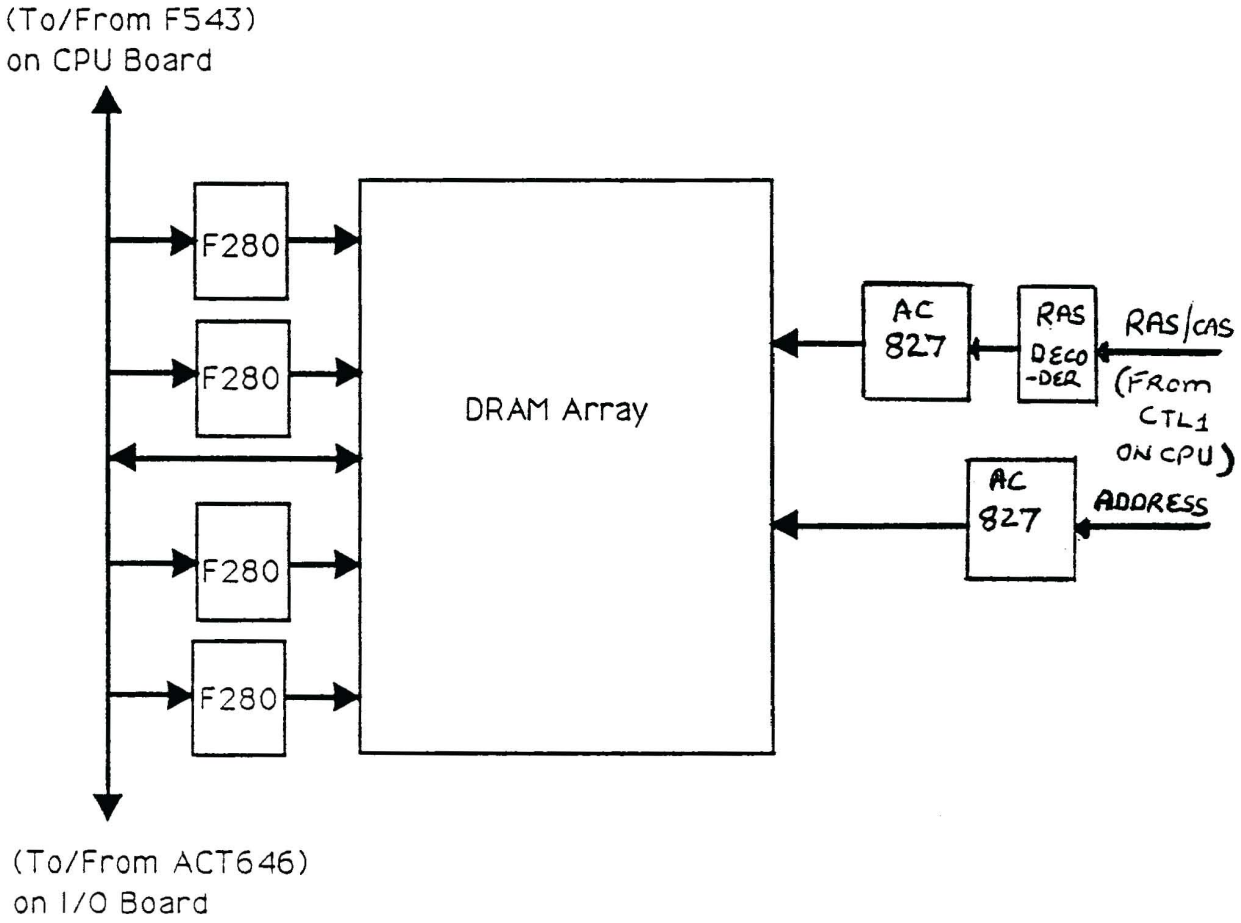
CPU SUBSYSTEM ARCHITECTURE
FIGURE - 1



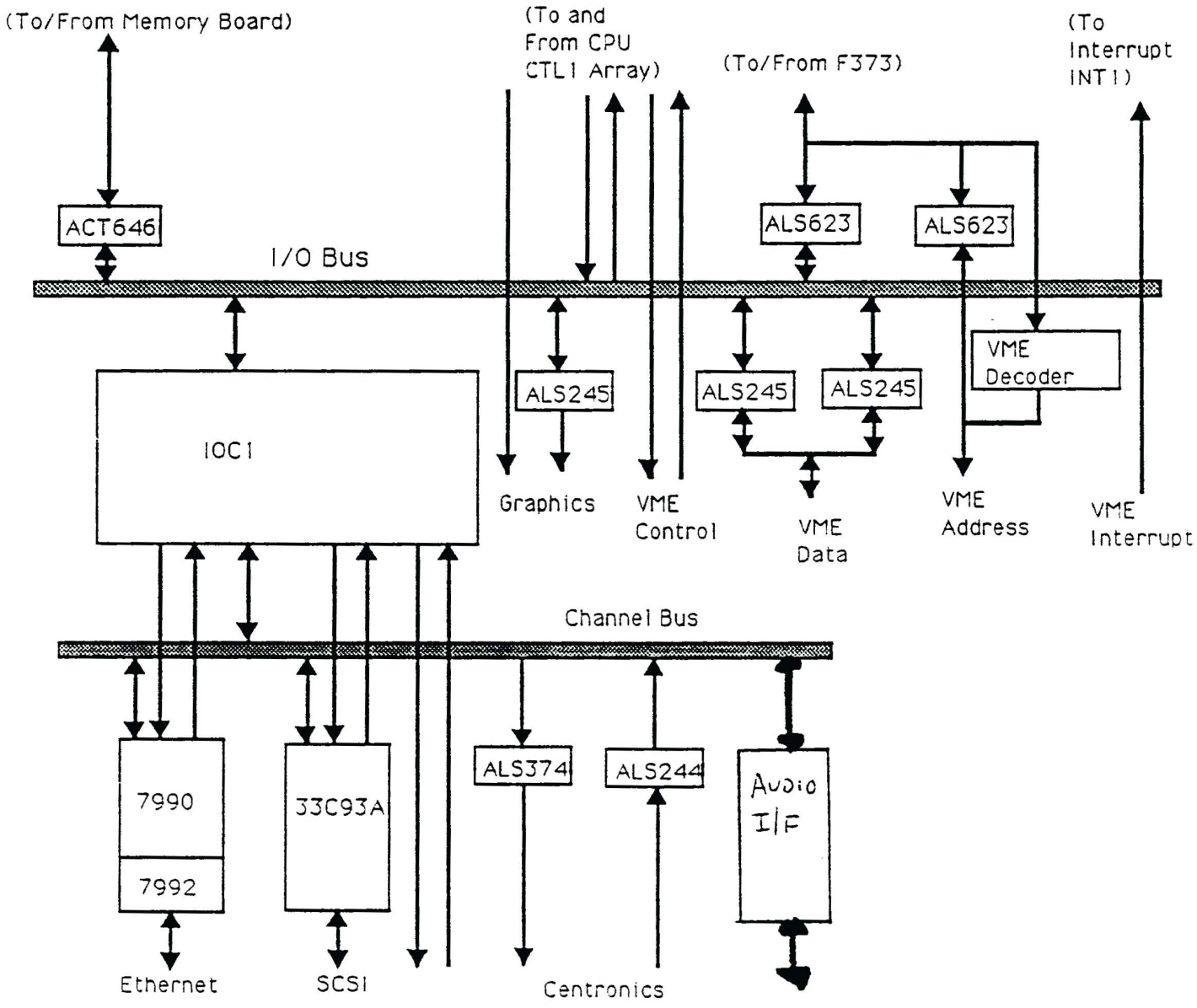
CPU BOARD BLOCK DIAGRAM
FIGURE - 2



MEMORY BOARD BLOCK DIAGRAM
FIGURE - 3



I/O BOARD BLOCK DIAGRAM
FIGURE - 4



FRONT PLATE AND INTER-CONNECTION DIAGRAM
FIGURE - 5