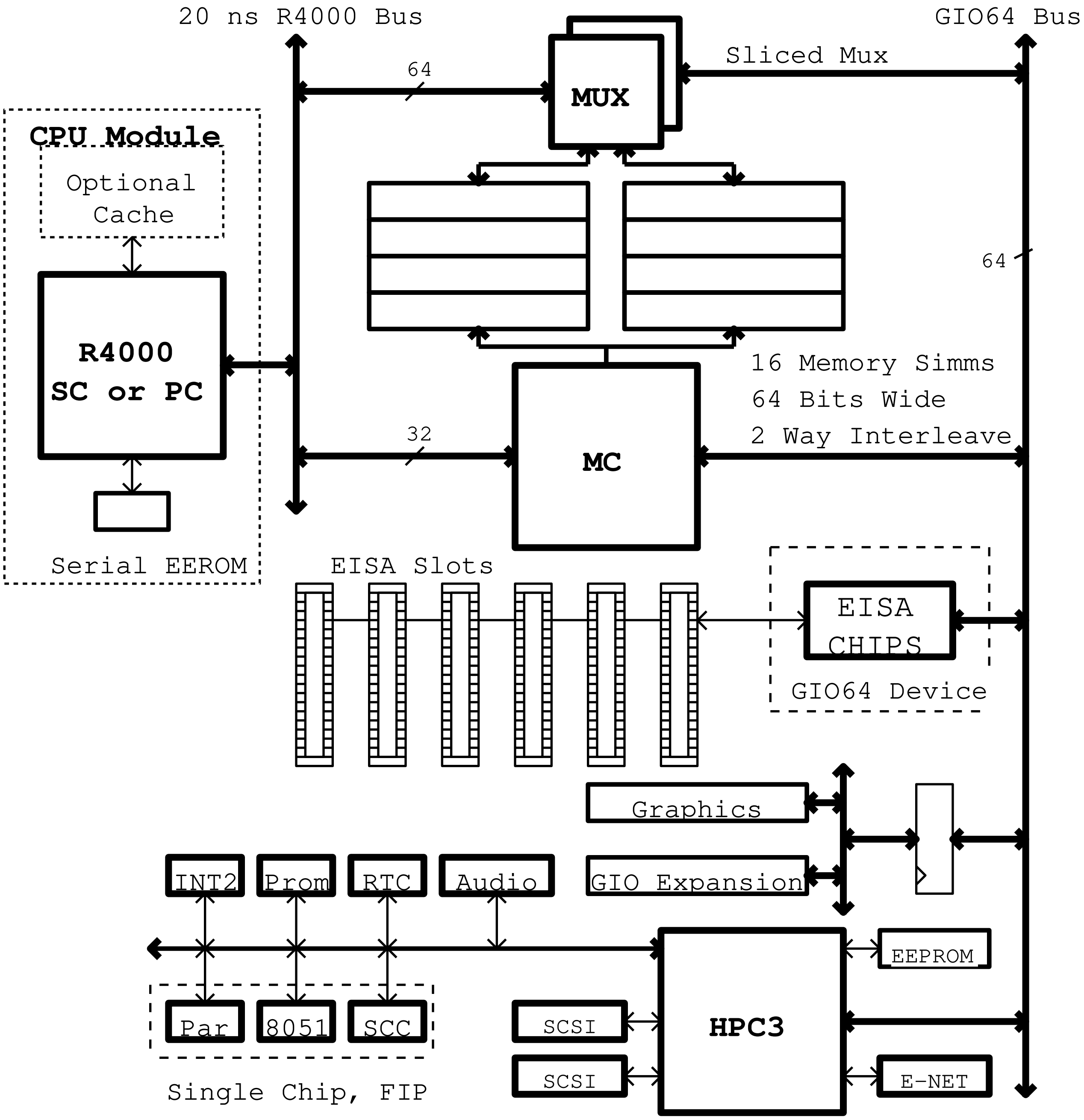


1. Introduction

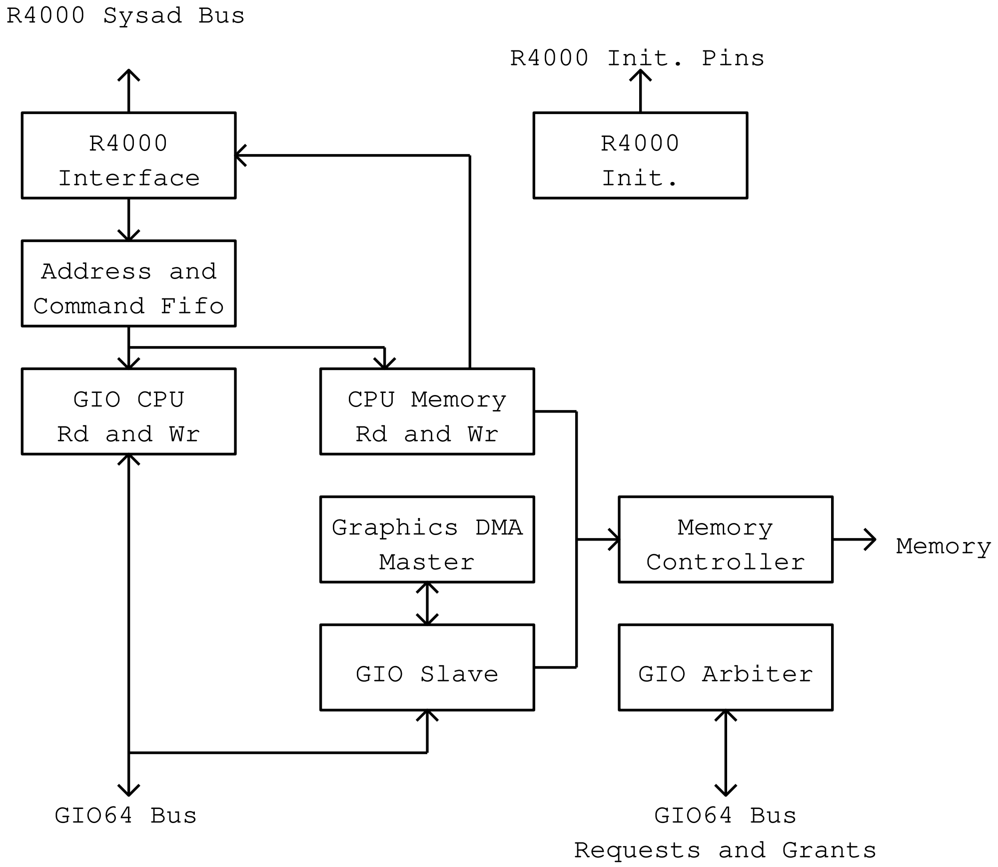
This document specifies the architecture of the MC, Memory Controller, gate array for MIPS R4000 based Fast Forward machines. This array is the interface between the R4000, main memory, GIO64 bus, and the EISA bus. This chip will handle all the bus traffic from the R4000 as well as requests to main memory from the GIO64 or EISA bus. A block diagram of a complete R4000 based machine is shown below:



1.1 MC Features

The memory controller gate array supports the R4000, GIO64 bus, and EISA interface to main memory as well as the interface from the R4000 to GIO64 and EISA devices. These two interfaces run at different speeds. The R4000 bus is currently a 50 MHz bus. The GIO64 bus will run at speeds up to 33 MHz. The MC chip is made up of eleven major blocks: GIO64 arbiter, CPU memory controller, GIO64 memory controller, GIO64 graphics DMA master, GIO64 DMA slave, GIO64 single reads and writes, CPU request state machine, memory refresh, CPU interrupts, R4000 initialization, and parity checking logic. The R4000 can be either in the small package, PC, or the large package, SC, that supports the second level cache. A block diagram of MC is shown below.

MC Block Diagram



The GIO64 arbiter determines what device has control of the GIO64 bus as well as main memory. This is really more of a memory arbiter then it is a GIO64 arbiter since it also handles requests from the CPU and memory refresh which are not GIO64 devices. The arbiter is programmable so that it can handle both long burst devices and real time devices. The seven devices that can request the GIO64 bus and memory are: the CPU, memory refresh, the HPC I/O controller, the two GIO64 expansion slot, the GIO64 graphics DMA master, graphics, and the EISA bus.

The memory controller state machine is a programmable interface to the main memory system. Main memory is constructed with standard 36 bit wide simms. To allow a wide variety of memory configurations, many different size simms will be supported. The memory controller maps physical addresses to a bank of memory. Memory and processor speeds will change over time and the interface should be flexible enough that it can be reprogrammed to work with various speed DRAMs and processors, while still maximizing performance. This is done by programming when the control signals should be active and the number of cycles a control signal will be active. The control signals are changed on cycle and half cycle boundaries. There are two copies of the memory controller state machine, one for the CPU accesses which is synchronous to the CPU clock and the other is for GIO64 accesses, synchronous to the GIO64 clock.

The MC chip contains one DMA master for graphics DMA. This DMA master is programmable by the CPU. The CPU sets up a DMA descriptor in MC and then tells the DMA engine to start. When the DMA operation is complete a status register bit is set and an interrupt is generated if it is enabled. The CPU can also poll a register in MC to determine when the DMA is complete. The DMA master supports virtual DMA so that user processes can program the DMA master. The memory address is a virtual address instead of being a physical address. The DMA master uses a TLB in MC and the UNIX page tables to translate the virtual addresses into physical addresses.

The MC chip contains one DMA slave that is used by all of the GIO64 bus masters to access main memory. The DMA slaves uses the GIO64 memory controller to interface to memory. The DMA slave does not keep any state about DMA operations that get preempted, so it is the responsibility of the DMA master to keep enough information about the transfer to restart the transfer if necessary. This includes the next memory address to access.

The CPU can issue reads and writes to GIO64 devices directly and the GIO64 single reads and writes state machine in the MC chip is responsible for performing the transfer over the GIO64 bus. This block is not used for accessing memory however. This block is called by the CPU request state machine to orchestrate the GIO64 transfer. This block also handles CPU reads and writes to EISA devices. EISA devices for the most part look like GIO64 devices, but there are some dedicated control signals for CPU reads and writes to EISA devices.

The MC chip handles all CPU read and write requests. These may be requests to main memory, a MC register, a GIO64 device, or an EISA device. The CPU read state machine is responsible for issuing the read and returning the data to the CPU. The CPU write state machine handles all of the CPU writes. There is a address and command fifo inside the MC chip that is fifteen entries deep so that it can hold many graphics writes. The number of entries in the address fifo is programmable. The depth of the fifo can be adjusted to maximize the writes performance to graphics while at the same time minimizing the number of entries the graphics has to accept when the graphics fifo is "full". All write data, except MC register write data, from the CPU will be buffered inside the MUX chip. The MUX fifo can store at least one 32 word block from the CPU. If the block size of the second level cache is less then 32 words then it is possible that the write buffer can hold a block write and some graphics writes.

Memory refresh is handled by the memory controller. Refresh will be done at a fixed interval in small bursts. The control of the RAS and CAS lines are parameterized to allow flexibility in DRAM speeds and system clock rates. The number of lines to refresh in each burst and the number of cycles between bursts is also programmable.

On the large package R4000 there is only one maskable interrupt pin, instead of one for each level like the R3000 and small package R4000. On the R4000 an interrupt to any level can be generated by a write to the CPU. The INT2 array will be used to collect all of the interrupts and send six interrupt signals (one for each interrupt level) to the MC chip. This will cause an interrupt write to the R4000 by MC. This provides a solution for interrupts that will work with both the small and large package R4000.

The R4000 needs to be initialized by the boot time mode control interface at power up. This involves controlling the three reset signals on the R4000 and setting up the serial EEROM so that the R4000 can read out the 256 configuration bits stored inside the EEROM. The MC chip also needs to coordinate the reset sequence of the rest of the machine. To make it easy to change between running the machine in big and little endian mode the CPU can read and write the R4000 configuration EEROM.

The MC chip will check parity on the sysad, syscmd, and GIO64 bus. The MUX chip will check parity on the sysad bus, memory data, and the GIO64 bus. The MUX chip will generate 8 parity error signals that will be connected to MC to indicate that a parity error has occurred. The MC chip will log the the byte(s) that had the parity error(s) and the memory address.

1.2 MC Gate Count

The MC array will be implemented using LSI's one micron LCA100K technology. The die will be a 100182, which has about 80K usable gates, of which about 64K are used. This chip will be packaged in a 299 ceramic pga for prototyping and hopefully change to a 304 mquad for production if the package gets qualified.

1.3 Bit and Byte Numbering Conventions

The MC chip can operate in a big or little endian machine. The endianess of the CPU is set by the CPU serial initialization EEROM. The endianess can be changed by writing the CPU control register.

GIO64 bus operations can be either big or little endian. There is a bit in the byte count cycle that indicates the endianess of the transfer.

Big-endian means that byte 0 is bits (31:24), byte 1 is bits (23:16), byte 2 is bits (15:8), and byte 3 is bits (7:0). Little-endian is just the opposite so, byte 0 is bits (7:0), byte 1 is bits (15:8), byte 2 is bits (23:16), and byte 3 is bits (31:24).

The bit numbering scheme is always little-endian, so that bit zero is always the least significant bit and bit 31 is the most significant bit.

The following figures show byte addressing for the big and little endian modes.

Big Endian

Bit Numbers

31 24	23 16	15 8	7 0	Word Address
8	9	10	11	8
4	5	6	7	4
0	1	2	3	0

Addresses of Bytes Within Words

Little Endian

Bit Numbers

31 24	23 16	15 8	7 0	Word Address
11	10	9	8	8
7	6	5	4	4
3	2	1	0	0

Addresses of Bytes Within Words

1.4 Other Documents

Other related documents are the MIPS R4000 Processor Interface, the GIO64 Bus Specification, the MUX Chip Specification, the Virtual DMA programmer's Guide, and the HPC3 Chip Specification.

1.5 Signal Naming Conventions

Signal names that end with a trailing "_n" represent active low signals. This is used instead of the trailing backwack or trailing underscore because VHDL does not allow the backwack character or trailing underscores in signal names.

2.0 MC Chip Functional Blocks

The MC chip is made up of eleven major blocks: GIO64 arbiter, CPU memory controller, GIO64 memory controller, GIO64 graphics DMA master, GIO64 DMA slave, GIO64 single reads and writes, CPU request state machine, memory refresh, CPU interrupts, R4000 initialization, and parity checking logic. Each of these functional blocks are described in detail in the following sections.

2.1 GIO64 Arbiter

There is one arbiter that is used to get access to either the main memory or the GIO64 bus. Even though the CPU does not really need the GIO64 bus to access main memory, other device on the GIO64 bus would have to be stalled while the CPU accessed main memory since almost all GIO64 device will access main memory. Therefore it makes sense to only have one arbiter for both main memory and the GIO64 bus. The GIO64 arbitrator determines whether the CPU, memory refresh, HPC, GIO64 expansion slot 0 or 1, the EISA bus, GIO64 graphics DMA master, or the graphics slot has control of the memory system. The CPU will own the memory system if it is not being requested by any other device so that the CPU memory access time will be minimized.

There are two kinds of GIO64 devices: real time and long burst. The real time devices need to use the bus within a fixed amount of time after they request the bus or data could be lost. Once a real time device gets the bus it only uses it for a short amount of time. Other devices that only use the bus for a short amount of time have also been placed in the real time devices category.

A long burst device wants to use the bus for a long period of time, but can tolerate waiting a long time to get the bus. They have no real time constraints on the data they are transferring. Long burst devices are preemptable by real time devices.

There are three devices that are always treated as real time devices: EISA, HPC and refresh. Graphics and the two GIO64 expansion slots can be configured as either real time or long burst devices. Long burst devices that are using the bus when a real time device requests the bus will be preempted so that the real time device can use the bus. After each real time device is serviced the arbiter starts at the top of the real time device list looking for a real time device that wants the bus. The first real time device in the list that wants the bus is granted it. If no real time device wants the bus then it is returned to the long burst device that was preempted. The real time device arbiter is not a round robin arbiter, but rather a prioritize arbiter, so it is possible for the bus grant order to be: EISA, HPC, EISA, HPC, EISA, graphics, etc. The real time devices are serviced in the order that is given below:

1. EISA Bus
2. HPC
3. Refresh
4. Graphics, (if configured as real time)
5. GIO64 Expansion Slot 0, (if configured as real time)
6. GIO64 Expansion Slot 1, (if configured as real time)

A real time device can only hold the GIO64 bus for 5 µs before it must give up the GIO64 bus. This restriction is not enforced by the GIO64 arbiter so it is up to the real time devices to give up the bus when its 5 µs is up. A real time device is not allowed to request the bus more often then every 20 µs if it is not preempted. It is the responsibility of the device to make sure that this specification is also met. Refresh will request the bus every 62 µs and hold the bus for about 800 nanoseconds.

HPC is a special real time in that it may request the bus many times within 30 µs and can be configured for the maximum time it is allowed to be on the bus. HPC will request the bus multiple times in 30 µs because it only requests the bus when one of the fifo's crosses the high water mark. Since HPC has many fifo's it is possible that they will all need service at different times. If HPC, EISA and refresh are the only real time devices then there is no reason to limit the amount of time HPC is on the bus, which in the worst case it 10 µs every 30 µs. If there are other real time devices then HPC will have to limit the amount of time it is on the bus to 5 µs to guarantee that the other real time devices will get the bus in worst case conditions in some reasonable amount of time.

The worst case real time device bus acquisition time can be calculated as follows:

1. Worst case CPU block read, 1 µs for 32 word block.
2. EISA service time of 16 µs.
3. HPC service time, 5 µs.
4. Refresh, 800 ns.
5. Next real time device.

This means that a real time device should get the bus within 23 µs of requesting the GIO64 bus.

Unlike the real time devices, long burst devices are issued the bus in round robin order, except the CPU is granted the bus between every long burst device. The CPU and GIO64 graphics DMA master are always long burst devices. The graphics slot, and two GIO64 expansion slots can be configured as long burst or real time devices. If the GIO64 graphics DMA master and GIO64 Expansion slot 0 both want the bus and no real time devices wanted the bus, the bus would be granted as follows: CPU, GIO64 graphics DMA, CPU, Graphics, CPU, GIO64 graphics DMA etc. There is two counters that determine the ratio of time that the bus is issued to the CPU and long burst devices. CPU_TIME determines how long the CPU owns the bus once it is granted the bus. When a real time device preempts a long burst device the counter stops and is resumed once the bus is given back to the device that was preempted. The CPU is given the bus for the entire CPU_TIME period even if it is not using the bus. This is necessary since the CPU is not going to run for very long before it will need the bus and if the bus is given to a different device the CPU could be stalled for a very long time. There is also a LB_TIME counter that is for all of the long burst devices, except the CPU, and works just like the CPU_TIME counter. Unlike the CPU when a long burst device is finished using the bus the bus is granted to CPU even if the LB_TIME counter has not expired. Once either the CPU_TIME or LB_TIME counter expires the device is preempted and given to the next device.

The EISA bus will use the real time device or one of the expansion slot request/grant pairs. The EISA bus master and dma requests from all EISA devices will be intercepted and collected and not passed directly to the EISA chip set to get control over the EISA arbitration. One EISA request at a time will be given to the EISA chip set at a time and the bus will be given to other devices between each EISA device. A EISA device will only be granted the bus when the EISA chip set also owns the GIO bus so that the 2.5 µS rule can be obeyed.

2.2 Memory Controller

The memory system state machine controls the accesses to the DRAMs. Since we know in the future that the processor chips will get faster it is a good idea to have a interface to memory that is parameterized so that the interface does not have to be changed, just the parameters have to be changed. This state machine is parameterized so that it is easy to change the way main memory is accessed. The parameters are:

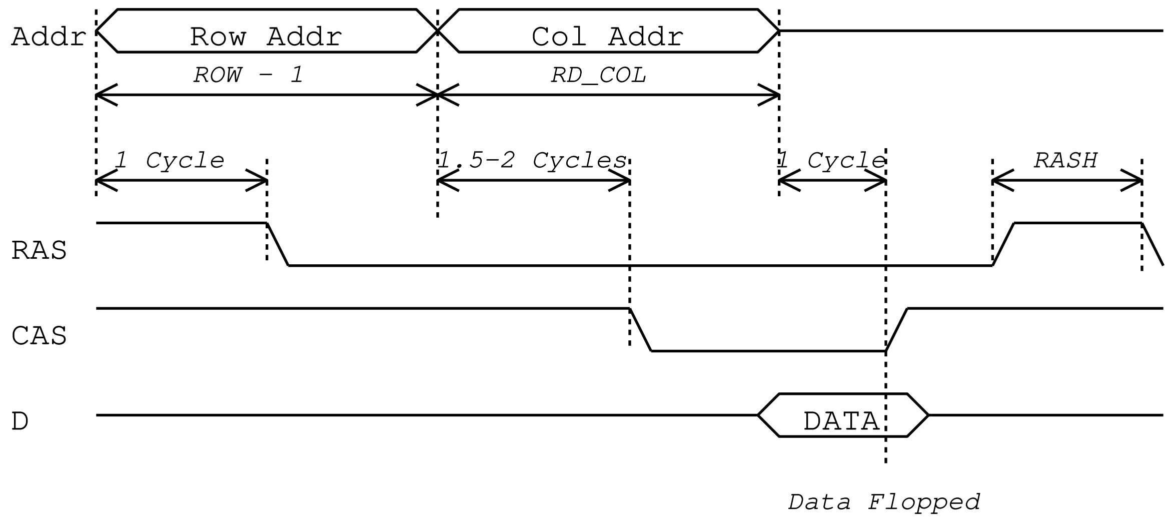
RASH	number of cycles RAS must be high before it can be dropped again to satisfy the RAM precharge time.
ROW	number of cycles minus one that the row address is driven before switching to the column address.
RD_COL	number of cycles column address is driven before the next column can be driven for a page mode access for reads. This number can be adjusted if the processor can not accept data as fast as the memory system can return data.
WR_COL	number of cycles that the column address is driven before switching to the next column address for writes.
CBR	number of cycles CAS is low before RAS is taken low for CAS before RAS refresh.
RCASL	number cycles after dropping RAS that CAS is low during refresh.
RASL	number of cycles after raising RAS before dropping CAS during refresh.
CAS_HALF	drive CAS low on a half cycle boundary for memory reads. CAS will be high for only one half of a cycle during page mode reads.
ADDR_HALF	change the column address on a half cycle boundary. This can only be set for GIO accesses for memory reads.

For normal memory reads and write RAS is always dropped one cycle after the row address is driven and CAS is dropped two cycles after the column address is driven. For writes data is driven from the MUX chips the cycle before CAS is dropped and on reads the data is flopped in the MUX chips on the same edge as CAS is driven high.

Some diagrams of the memory accesses follow on the next few pages. When a cycle range is given in the diagrams the range is controlled by the CAS_HALF and ADDR_HALF parameters. These parameters are for memory reads only. The ADDR_HALF is only used for GIO accesses.

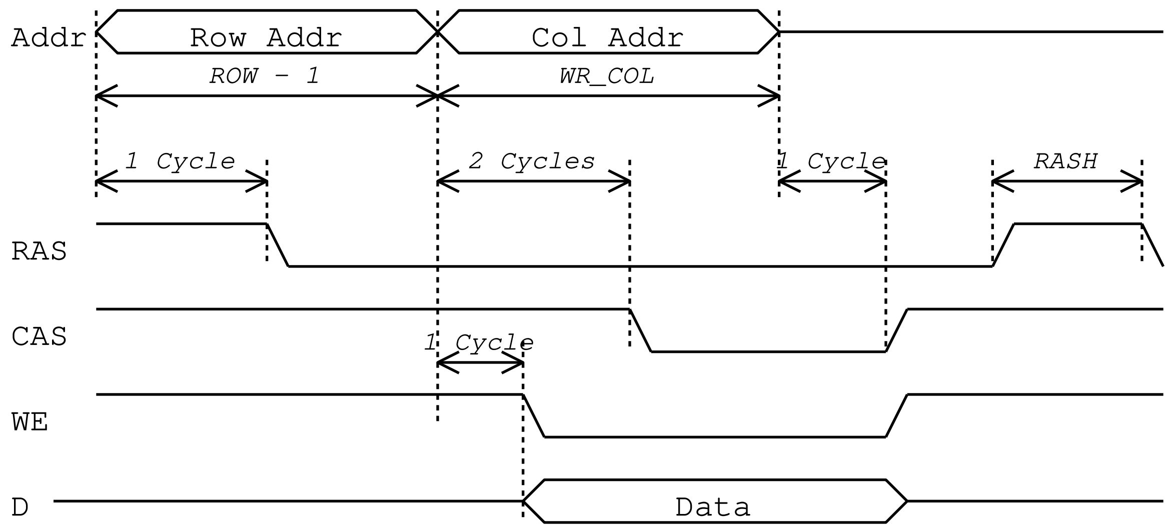
2.2.1 Memory Reads

A memory read is parametrized as follows:



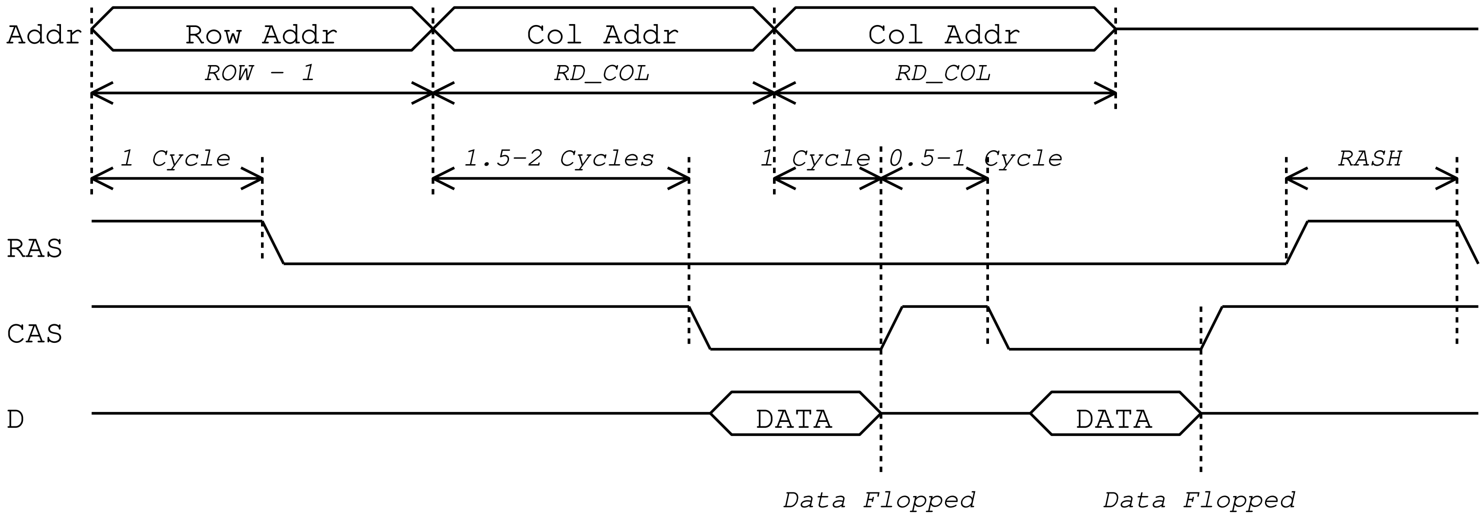
2.2.2 Memory Writes

Memory writes are parameterized as follows:



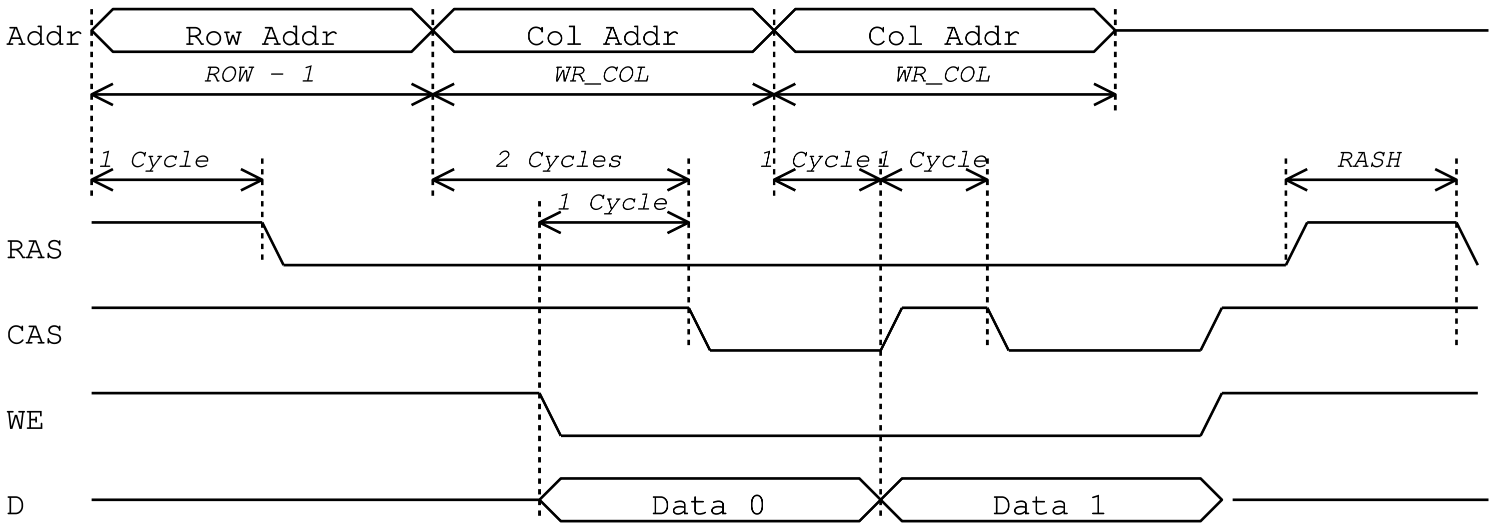
2.2.3 Memory Reads, Page Mode

Page mode memory reads are parameterized as follows:



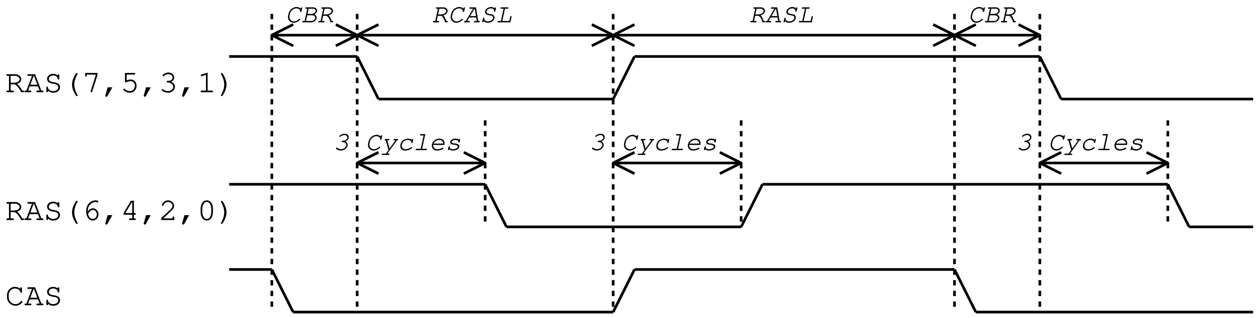
2.2.4 Memory Writes, Page Mode

Page mode writes are parameterized as follows:



2.2.5 Memory Refresh

Memory refresh is accomplished by using the CAS before RAS refresh cycle. When using this type of refresh cycle an internal row address counter is used so the MC chip does not need to provide a refresh address. Memory refresh is parameterized as follows:



2.2.6 Memory Address Signals

The correlation between the memory address signal from the MC chip and the memory address is set up so that both the symmetrical and nonsymmetrical address 16M density DRAMs can be used in the Fast Forward machines. The nonsymmetrical address DRAMs use less power than the symmetrical address parts. The mapping of a memory address to the memory address pins is as follows, the memory address bits are in the boxes:

Memory Address Signals

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

The bit numbers of the CAS signals are the same as the little endian byte number that they should control. Interleave memory A is connected to CAS(15:8) and memory B is connected to CAS(7:0). There is two RAS signals for each bank of DRAMS. The simms with two subbanks, (512Kx36, 2Mx36, and 8Mx36) uses both RAS signals and the simms with one subbank only use one RAS signal from MC. RAS(1:0) is used for bank 0, RAS(3:2) for bank 1, RAS(5:4) for bank 2, and RAS(7:6) for bank 3. The odd RAS signals are used for simms with two subbank only. RAS numbering is tricky since on a one subbank simm the RAS signals are labeled RAS0 and RAS2. These should both be connected logically to a even RAS signal. Two subbank simms have four RAS signals. RAS1 and RAS3 on a two subbank simm should logically be connected to an odd RAS bit.

2.3 Graphics DMA Master

There is a separate document that describes the graphics DMA master. This

DMA master is capable of doing virtual DMA. Since the DMA is virtual a user process can set up a DMA transfer. This will be a much cleaner way to implement the functionality of 3 way transfers.

2.4 GIO64 DMA Slave

The GIO64 DMA slave is used by GIO64 bus masters to read and write memory on their behalf. This DMA slave is used by HPC and the EISA chips to read and write memory. The DMA slave uses the GIO memory controller interface to access the memory system. It is capable of handling subblock order requests. The DMA slave retains no information about any transfer that gets preempted. It is up to the master to keep all information that is necessary to complete the transfer. This DMA slave is used by both pipelined and nonpipelined GIO64 devices. The peak memory bandwidth is 266 Mbytes/second when the GIO64 bus is running at 33 MHz.

2.5 GIO64 Single Reads and Writes

The CPU can issue reads and writes to GIO64 devices. These transfers do not involve using the GIO64 DMA master or slave. This functional block is responsible for executing the GIO64 read or write on behalf of the CPU request state machine. If the processor issues a write then the data to be written will be in the CPU write buffer inside the MUX chip. Once the GIO64 bus has been granted to the GIO64 single read and write state machine the CPU request state machine will transfer the data from the CPU write buffer to the GIO64 bus. If the processor issues a read the GIO64 single read and write state machine will request the GIO64 bus, issue the read and then tell the CPU request state machine that the data is on the GIO64 bus. The CPU request state machine will transfer the data to the the CPU from the GIO64 bus through the MUX chip. Once the data has been transferred to the processor the CPU request state machine will acknowledge to the GIO64 slave that the data have been transferred by dropping masdly.

The expected bandwidth of programmed I/O over the GIO64 bus is as follows:

GIO64 PIO CPU read bandwidth: 13 Mbytes/second at 33 MHz.
GIO64 PIO CPU write bandwidth: 66 Mbytes/second at 33 MHz.

Cache block reads and writes to a GIO64 devices will result in one GIO64 bus operation for each double word to be transferred. Data to/from a 32 bit GIO64 device can not be cached.

To increase the bandwidth of CPU writes to GIO64 devices there is a bit in the CPU control register that will allow back to back writes to GIO64 device occur back to back in a minimum of 4 GIO clock cycles. This potentially could have tristate overlap problems since the MC chip drives the address and byte count fields and the MUX chip drives the data. If the tristate overlap does not work back to back GIO64 writes can be turned off so that there is a dead cycle between MC and MUX driving the GIO64 bus.

2.6 CPU Request State Machine

The CPU request state machine is responsible for executing any requests the CPU issues. Inside the MC chip there is a command and address fifo. This fifo has fifteen entries so that writes to the graphics system can be buffered up before being sent over the GIO64 bus. The operations in the fifo will always be issued in order.

The state machine looks at the oldest request in the fifo and uses the other blocks within the MC chip to execute the request. When each request is complete it deletes the entry from the fifo to get the next request. There are six different types of requests: memory reads and writes, GIO64 reads and writes, and MC chip reads and writes. All of these different kinds of requests are handled by the CPU request state machine.

The CPU request state machine is also used to issue interrupt writes to the processor.

The expected peak memory bandwidth for CPU requests is 266 Mbytes/second for reads and writes at 50 MHz.

2.6.1 Semaphores

There is 16 user single bit semaphores in the MC chip. They are each on a 4K page so that they can selectively be mapped into the users address space. A write to a semaphore register just writes the value of bit 0 into the register. A read from a semaphore register will return the value of the semaphore register and then change the semaphore value to a one.

There is also one system semaphore that is in the same page as the rest of the MC privileged registers.

2.6.2 RPSS Counter

The RPSS counter is a 32 bit counter that increments every 100 nanoseconds. Since the clock rate of the processor may not always be 50 MHz there is a RPSS divider registers that determines both how much the processor clock needs to be divided by and also how much to increment the RPSS by each time the clock divider rolls over. For a 50 MHz processor clock the divider should be four, (count from zero to four, so it is dividing by five), and the increment should be one. This will increment the rpss counter every five processor clocks.

2.6.3 EISA Lock

The EISA bus can lock the CPU out of main memory by asserting eglock_n to the MC chip. Once the eglock_n signal is asserted the CPU will not be granted the bus until it is deasserted.

2.6.4 CPU Lock

The CPU can lock the EISA bus out of main memory by writing the EISA_LOCK register in the MC chip. This will assert the gelock_n signal to the EISA chips. It is the responsibility of the EISA chips to lock the EISA bus since the GIO64 arbiter will still grant the bus to the EISA chips even if the EISA bus is locked. The the CPU is finished with its locked cycle sequence it then must clear the EISA_LOCK register. The software should change the long burst time register to a small value so that if the CPU does get preempted during a locked cycle it does not have to wait a long time for a long burst device before the CPU can unlock the EISA bus.

2.7 Memory Refresh

Refresh will be done in bursts just like the current machines. The number of lines to do in each burst is programmable as well as how often the bursts occur. This flexibility will allow for future DRAMs which may have different refresh requirements. Refreshing four rows every 62 µs will work with the 1Mx36, 2Mx36, 4Mx36, and 8Mx36 DRAMs. For the 256Kx36 and 512Kx36 simms, 8 rows need to be refreshed every 62 µs. The CAS before RAS refresh method will be used so that the row counter inside the DRAMs can be used. The eight different RAS lines will be staggered to reduce the refresh current surges. Four RAS lines are dropped in one cycle and the other four, three cycles later.

The refresh counter is loaded with the counter preload value. The counter counts down to zero and then reloads the counter with the counter preload value. At the same time it sends a refresh request to the GIO64 arbiter. The arbiter returns the refresh grant signal when it is ready to hand control of the memory system over to the refresh logic. The refresh logic sends the start refresh signal to the memory system controller which then refreshes the number of rows indicated in the CPU control register. The refresh counter roll over is also used by the watch dog timer as a counter increment.

2.8 CPU Interrupts

Interrupts are handled differently on the R4000 then they are on the R3000. On the R3000 there are interrupt pins on the package, which is the same way that the R4000 small package handles interrupts. On the R4000 large package there is one maskable interrupt pin. Interrupts on the R4000 can also be generated by writing to an internal register in the processor. Since the writing method supports all six levels of interrupts it will be used for both large and small package R4000's. The INT2 chip will collect all of the interrupts and the six interrupt lines from INT2 will be connected to the MC chip instead of the processor. The MC chip will generate writes to the processor when the interrupt lines change. This write will use the sysad bus to send the data to the processor.

2.9 R4000 Initialization

The R4000 needs to be initialized by the boot time mode control interface at power on. There are three reset like signals, cpu_vccok, cpu_cold_rst_n and cpu_reset_n. These signals have to be sequenced correctly and with the correct timing. The MC chip will control the reset signals. Part of the power on reset sequence includes reading in 256 configuration bits that are stored in a serial EEROM. The MC chip will set up the EEROM for reading and watch over the complete reset process. The reset process is quite lengthy due to the time that is spent waiting for PLL's to lock. The reset sequence is given below:

1. Reset to MC is deasserted, all MC reset outputs are asserted.
2. MC chip reads three bits from EEROM.
3. Cpu_vccok to R4000 is asserted, R4000 reads 256 EEROM bits.
4. MC chip wait 100 milliseconds for R4000 PLL to lock.
5. Cpu_cold_rst_n to R4000 is deasserted.
6. MC chip waits 20 milliseconds for a stable tclock from R4K.

7. MC deasserts pll_reset_out_n and the reset to MC’s PLL’s.
8. MC waits about 20 milliseconds for PLL’s to lock.
9. MC chip reset and GIO reset are deasserted.
10. Cpu_reset_n is deasserted.

There is also an interface so that processor can read and write the contents of the EEROM through MC register reads and writes. This will allow the processor to change the configuration. This is necessary so that the endian mode can be changed. The processor can change the EEROM and then force a cold reset which will reload the CPU configuration bits from the EEROM. The processor will then be configured with the new values in the EEROM. The first three bits in the EEROM are used by the MC chip and not the processor. The first bit in the EEROM will be the endian mode, the second bit is the size of the HPC GIO64 interface (32 or 64 bits), and the third bit is reserved. The MC chip will read the endian mode bit and then drive a control signal to the rest of the machine that indicates the endian mode. The endian mode bit will be duplicated later in the EEROM for the processor. The software will have to update both bits to change the endian mode of the processor.

The R4000 interface block also contains a watch dog timer that will reset the machine if the watch dog location is not written to about once a minute. The timeout period changes with the refresh counter preload value.

2.10 Parity Checking Logic

Parity will be checked over the R4000 system bus and GIO64 bus by the MUX chip. The MUX chip will send byte parity error signals to the MC chip which will keep track of any parity errors and the memory address of the data that had the parity error. MUX will regenerate parity that is written into memory. Parity that is read from memory will be sent out on the GIO64 or sysad bus after being checked by the MUX chip. Parity over the GIO64 bus is optional so MC will keep track of which devices are sending parity and only record parity errors that occur when one of those devices is using the bus. Parity on data read from memory will always be checked because it should always be correct since it was regenerated when the data was written into main memory.

3. System Operations

The MC chip is the interface between the memory system, the R4000 processor, the GIO64 Bus and the EISA bus chips. In this section operations that involve the MC chip will be described.

3.1 Memory System

Most of the complexity in the MC chip involves the memory system since the GIO64 bus and the processor need access to the memory system in a timely fashion. The memory system supports a number of different operations which also add to its complexity. The memory system is made up of four to sixteen simms and a custom MUX ASIC which is used to mux the data from the dual interleave memory, CPU write data, and GIO64 write data. The MUX part also handles fanning out the data to both the GIO64 bus and the CPU. The MUX chips also do parity generation and checking.

3.1.1 Memory Simms and Configurations

The Fast Forward machines will use standard 36 bit wide simms so that the system will be as open as possible and to reduce cost. There are six different simms that will be supported: 256Kx36, 512Kx36, 1Mx36, 2Mx36, 4Mx36, and 8Mx36. These need to be 80 ns DRAMS, although the MC chip is flexible enough to handle different speed parts (see section on Memory System Controller). All of the simms in a system will need to be the same speed however.

There is room for four groups of four simms in the system. Each group of four simms is called a bank. Each bank is 128 bits wide, plus parity, so that the memory system can respond to cache block reads and writes from the R4000 in a timely fashion. The interface to the R4000 is 64 bits wide, so each bank allows for an interleave of two.

Simms must be added in groups of four, but there is no restriction on mixing simms of different depths as long as each group of four simms is the same size. This allows easy expansion. The minimum system is four 256Kx36 simms, (4 MBytes) and maximum memory capacity is sixteen 8Mx36 simms, (512MBytes). It is important to remember that the 8Mx36 simms will not be available until sometime in 1993. Using simms that are available today the maximum memory size is sixteen 2Mx36 simms, (128 MBytes).

The 512X36, 2Mx36 and 8Mx36 simms all are implemented using two subbanks of DRAMs. These are double sided simms, since each one contains 24 DRAMs. There is a configuration bit, BNK, for each bank of memory in the MEMCFGx registers that must be set if these simms are being used.

There are two registers inside the MC chip that are used to configure the simms that are installed. Each register holds the configuration information for two banks. For each bank there are four fields: size of the simm, the number of subbanks per simm, (1 or 2), the base address of the simm, and a valid bit that indicates that the bank has memory installed. For more information on the register format see the section on MEMCFG0 and MEMCFG1.

3.1.2 CPU Memory Reads

There are many different kinds of main memory reads the R4000 can issue, but only two different types that have to be handled differently. This first types is block reads which may or may not be coherent, it really does not matter since this is a single processor system. The second type of read is a double word, word or partial word request. These are handled in basically the same way except that the block reads of more than four words will require multiple accesses to memory using page mode.

Memory reads are a split transaction on the R4000 processor bus, which means the address and command will be sent in one bus transaction and the data will be returned with a different transaction. The R4000 sends the address and a command out onto the bus with the validOut_n signal active to initiate the read. This gets sent to the MC chip which flops the command and address. In the next cycle the MC chip determines what part of the physical address space is being accessed (memory, GIO64, EISA, PROM etc.). If the request is to main memory and the memory system is not busy at the time the bank of memory that the read targets will also be determined. The memory system controller state machine will be activated to execute the request. The state machine will access memory and control RAS, CAS and the memory address. If the request is for a double word or less it will only require one read from memory. The memory data will be flopped inside the MUX chips and then sent back over the processor bus to the R4000 in 64 bit pieces. If the request is for four words then the MUX chips will send the data back to the processor in two back to back cycles, (remember the memory system is 64 bits wide and has an interleave of two). If the request is for more then four words then the memory system controller will use page mode on the DRAMs to get the next 4 words.

If the memory system is not busy reads will take ten cycles from the time the processor puts the address and command on the bus until the memory system returns the first piece of data. Each double word requested will take an additional cycle to return to the processor. For requests of more then four words there is an additional delay of one cycle for each group of four words for the page mode access time.

The R4000 has a mode to increase performance called smiss restart or subblock ordering. In machine configurations with a second level cache and the block size of the second level cache is larger then the block size of the first level cache the processor can start to execute instructions once the data for the fist level cache miss is returned from memory. When the processor issues a block read and smiss restart is enabled the processor sends the address of the first level cache block that caused the cache miss instead of the second level cache block address. This data is returned first and followed by the rest of the second level cache block. Once the first level cache data is returned the processor can start execution again while the second level cache refill is still taking place.

Block read data can not be returned to the processor at a rate faster then in can write it into the second level cache since it has no way to fifo the data. Since there is no way for the processor to throttle read data that is being returned, it is up to the devices connected to the processor to throttle the data transfer. The rate that data can be returned depends on the second level cache write time. The RD_COL field in the CPU_MEMACC register is used to set the number of cycles between 128 bit transfers to the CPU. This parameter can be changed to throttle the data transfer rate back to the CPU. This parameter needs to be set to at least 3 for the memory timing to work. This means the CPU has to write the second level cache every 6 pclocks.

A breakdown of the memory read cycles is shown below:

Cycle	Function	Number of Cycles
1	Read on CPU Bus	1
2	MC Decodes Read	1
3	Row Address Sent to DRAMs	1
4	RAS Sent to DRAMs	1
5	Column Address Sent to DRAM's	2
7	CAS Sent to DRAMs	2
9	Next Column, Data Sent to MUX	1
10	Data on CPU Bus	1

3.1.3 CPU Memory Writes

The caches on the R4000 are writeback unlike the R3000 cache which is write through. Therefore every write is not being sent to the memory system so a deep write buffer is not necessary. There is a write buffer that will hold a small number of cache blocks in the MUX chip, which is needed to hold the write data until it can be written to main memory, a GIO64 device or an EISA device. The size of the buffer will depend on the second level cache block size we support (4, 8, 16 , or 32 words). The number of outstanding writes that will be allowed depends on the size of the write buffer in the MUX chip, the block size of the second level cache, and the depth of the address and command fifo in the MC chip. The address and command fifo is 15 entries deep. Just like processor reads there are two kinds of writes, block write of four to thirty-two words and writes of two words or less.

Unlike memory reads, memory writes are not a split transaction. The processor will send out the address and command in the first cycle and then in the following cycles send the data to be written. There may be dead cycles between the write data cycles due to the fact that the R4000 may have to read the data out of the second level cache which may not be as fast as the bus transfer rate. This data will be put into the MUX write buffer until it can be written into main memory. If the write buffer is full the WrRdy_n signal will be deasserted so that the R4000 will not issue another write, overwriting the data that is in the buffer. If the memory system is available the write will start as soon as the data is in the write buffer. The memory system controller will control the memory system for the CPU request state machine during memory writes. Main memory can be written in double word blocks. For writes of more than four words, page mode writes will be used to write the rest of the block.

For writes of less then a double word the DRAM CAS lines will be used as byte write enables.

3.1.4 CPU Triplet Requests

The R4000 may issue up to three requests in a group which should be handled together to maximize performance. The three requests occur when there is a first and second level cache miss and the data that is being replaced is dirty. The three requests are a block read, an invalidate or update, and a block write. The MC chip will need to handle all three requests pending at the same time and queue up the three requests.

The address and command fifo in MC is fifteen entries so it can hold a triplet. This is independent of the MUX write buffer that will be able to hold at least a complete second level cache block. The R4000 will have at most one triplet outstanding at a time. Also the order of the triplet is fixed as given above. It is also possible to have triplets without the write or without the update or invalidate.

3.1.5 EISA Memory and I/O Reads

EISA memory and I/O reads look the same except the eisa_memory signal will be assert for EISA memory accesses. EISA reads and writes look like a normal GIO64 bus transaction except that there is a decoded address strobe, gio_eisa_as_n, which is asserted instead of gio_as_n during the address cycle of a GIO bus transaction. Before the MC can issue a request to the EISA chips it must make sure that eisa_ecp_n is not asserted. The MC chip then starts the read from EISA by sending out the address and then byte count in what looks like a normal GIO64 read.

3.1.6 EISA Memory and I/O Writes

EISA writes look like normal GIO64 writes except that the MC chip must make sure the the eisa_ecp_n signal is deasserted before it can start another request. This signal is used to indicate the the EISA chips set can not currently accept another request because the address/data buffer is currently in use by the last CPU request.

3.1.7 GIO64 Memory Reads

Any GIO64 bus master can issue memory read requests. These can be 32 or 64 bit wide transfers. Once the GIO64 device has been granted the bus, the address is sent to the MC chip and the GIO64 DMA slave in the MC chip is used to access memory. The memory data is sent to the MUX chips and then driven onto the GIO64 bus. The GIO64 memory controller in the MC chip handles the actual memory access.

A preempted memory read will take five cycles from the time read is driven high to when the MUX will stop driving data onto the gio_ad bus. The GIO64 specification indicates that the bus should be tristated in two clocks.

3.1.8 GIO64 Memory Writes

GIO64 memory writes are a lot like GIO64 memory reads. The memory address is sent to the MC chip. The data is sent through the MUX chip and then written into memory. Again the GIO64 DMA slave and GIO64 memory controller are used to perform the writes into memory.

3.2 MC Register Reads/Writes

The processor can issue reads and writes to the MC registers. These reads and writes are handled differently then the processor reads and writes to main memory. The processor directly sends data to the MC chip for register writes. MC register reads are a split transaction like all processor reads. The processor sends out the address to the MC chip. The MC chip then returns data over the sysad bus. Because the MC only connects to sysad(31:0) the address of the registers in the MC chip changes when the processor when the processor endian mode is changed. The address map of the MC registers is described in section 5.

3.3 R4000 System Bus Interface

The R4000 system bus is a 64 bit multiplexed address/data bus with byte parity. There is a nine bit command field that is sent with all addresses and data that are sent over the bus. There is a validout_n signal which is active whenever the R4000 is driving data out and a validin_n signal which must be asserted when valid data is being returned to the R4000. The MC chip and MUX chip cannot drive the bus until the R4000 has released the bus to the external agent (the MC or MUX chips). There is an extrqst_n signal that the MC chip uses to get the R4000 to release the bus to the external agent.

Flow control for data being sent to the MC and MUX chips is accomplished through the rdrdy_n and wrrdy_n signals. The MC chip does not use the rdrdy_n signal since it always leaves one entry in the address/command fifo for a read. There can only be one read outstanding at a time so the MC chip can guarantee that it can always take one read. The R4000 does not have a mechanism for throttling the data being sent to it. However, the external agent cannot send data to the R4000 faster then it can write it into the second level cache.

The most significant bit of the command determines if this is an address cycle or a data cycle, the bit is a zero for address cycles. There are eight different commands that the bus supports.

SysCmd(7:5)	Command
0	Read Request
1	Read Request, Write Request forthcoming
2	Write Request
3	Null Request
4	Invalidate Request
5	Update Request
6	Intervention Request
7	Snoop Request

If the processor issues an invalidate or update request it will be acknowledged and no other action will be taken since there are no other caches to update or invalidate. The system will not issue any invalidate, update, or snoop requests to the processor so these three commands will not be used by the MC chip.

The format of the rest of the command depends on the command being issued. For reads and writes the size of the read or write is encoded in the remaining SysCmd bits.

The CPU request state machine in the MC chip handles all requests to and from the processor. For more information about the R4000 system interface see the R4000 System Interface Manual.

3.4 Timers

There are three timers inside the MC chip, the refresh counter, the watchdog timer, and the RPSS timer. The refresh counter is preloaded with the value in the refresh preload register when the counter counts down to zero. When the counter gets to zero a refresh burst is done and the counter is reloaded.

The watchdog timer counts the number of refresh bursts that take place (normally every 64 microseconds). The watchdog timer is a 20 bit counter which rolls over in about 67 seconds (if the refresh counter is programmed for 64 microseconds intervals). When the counter rolls over the machine will reset itself if the watchdog timer is enabled. Writing a register in the MC chip will reset the watchdog counter so that it will not roll over.

The RPSS timer is a thirty-two bit 100 ns timer. Since the clock speed of the interface may change the clock divider is programmable so that the units of this timer will still be 100’s of nanoseconds. The timer can be read by a user process. No interrupt is generated when this timer rolls over.

3.5 Three Way Transfers

Three way transfers can not be supported in this machine as there were in past machines because the R4000 has a write back cache. This causes two problems, the first being the data does not get written back to main memory until it is flushed or a miss occurs at the same cache block that is holding the write data, so the data in memory that is being transferred may not be consistent with the cache. The second problem is that there is no way to get the physical address since the actual write of the data may not occur for many cycles and other memory writes that were issued after the three way transfer writes may occur before the three way transfer writes.

To replace three way transfers virtual DMA support is being added. This will allow a user process to set up a DMA to graphics. The the DMA hardware does address translation it is safe for the user process to set up this transfer. This DMA engine can perform rectangular DMA as well as a number of other fancy DMA modes.

4. Physical Address Space

The physical address space of the R4000 is divided up as shown below. There are two different regions of local memory since the maximum memory the system will support is 512 MBytes and there is only one 512 MByte area in the physical address space that can be accessed in three different ways: user virtual (kuseg), cached kernel physical (kseg0), and uncached kernel physical (kseg1). The high local memory will only be accessible through the user virtual address space, kseg0.

CPU Memory Map

Range		Size	Usage
0xffffffff	0x80000000	2 GB	EISA Memory
0x7fffffff	0x30000000	1.25 GB	Reserved
0x2fffffff	0x20000000	256 MB	High System Memory
0x1fffffff	0x1fc00000	4 MB	Boot PROM
0x1fbfffff	0x1fb00000	1 MB	HPC and I/O devices
0x1fafffff	0x1fa00000	1 MB	MC registers
0x1f9fffff	0x1f600000	4 MB	GIO64 Expansion Slot 1
0x1f5fffff	0x1f400000	2 MB	GIO64 Expansion Slot 0
0x1f3fffff	0x1f000000	4 MB	Graphics System
0x1effffff	0x18000000	112 MB	Reserved (Future GIO Space)
0x17ffffff	0x08000000	256 MB	Low Local Memory
0x07ffffff	0x000a0000	~128 MB	EISA Memory
0x0009ffff	0x00090000	64 KB	EISA I/O Space Alias
0x0008ffff	0x00080000	64 KB	EISA I/O Space
0x0007ffff	0x00000000	512 KB	System Memory Alias

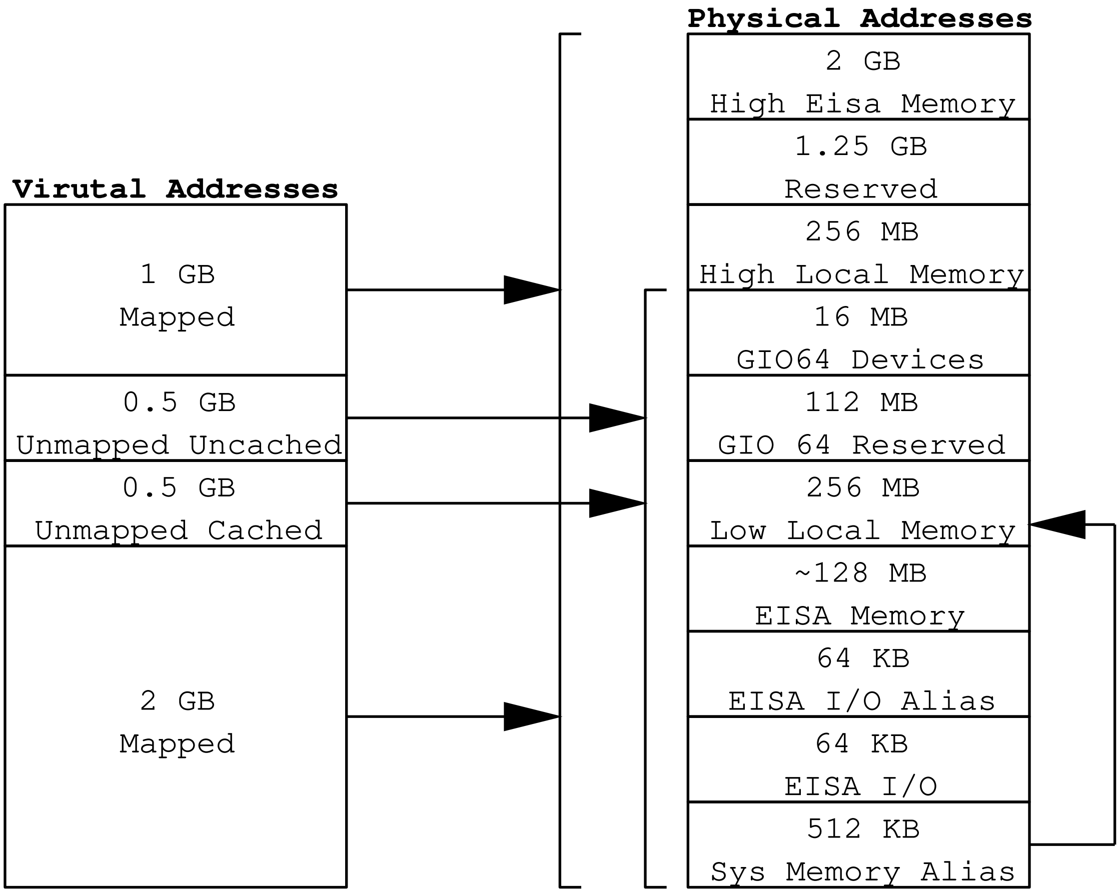
Accesses to the unused or reserved regions in the physical address space will cause a bus error response on reads and a bus error interrupt on writes. The bottom 512 KB of memory is just an alias for the memory located at address 0x08000000 to 0x0807ffff. This alias is necessary for the CPU exception vectors that are at physical addresses 0x00000000 and 0x00000080. This space is also used by ISA masters on the EISA bus that expect the system memory to be located in the first 640 KB of the address space and cannot address more than 16 MB of memory.

The EISA/ISA memory map is almost the same as the CPU memory map except that EISA devices can not interact with GIO64 devices. Also the EISA memory map does not need a region for the EISA I/O space since there is a distinction between I/O and memory cycles on the EISA bus. The EISA memory and I/O maps are as follows:

EISA Bus Memory Map

0xffffffff	0x80000000	2 GB	EISA Memory
0x7fffffff	0x30000000	1.24 GB	Reserved
0x2fffffff	0x20000000	256 MB	High System Memory
0x1fffffff	0x18000000	128 MB	Reserved, (GIO Space)
0x17ffffff	0x08000000	256 MB	Low System Memory
0x07ffffff	0x00100000	127 MB	EISA Memory
0x000fffff	0x000e0000	128 KB	BIOS ROM
0x000dffff	0x000c0000	128 KB	BIOS Exp. ROM
0x000bffff	0x000a0000	128 KB	Video ROM
0x0009ffff	0x00080000	128 KB	Reserved (System Memory)
0x0007ffff	0x00000000	512 KB	System Memory Alias

CPU Virtual To Physical Mapping



5. MC Internal Registers

Register Name	Address	R/W	Function
CPUCTRL 0	0x1fa00000/4	R/W	CPU control 0.
CPUCTRL 1	0x1fa00008/c	R/W	CPU control 1.
DOGC	0x1fa00010/4	R	Watchdog timer.
DOGR	0x1fa00010/4	W	Watchdog timer clear.
SYSID	0x1fa00018/c	R	System ID register.
RPSS_DIVIDER	0x1fa00028/c	R/W	RPSS divider register.
EEROM	0x1fa00030/4	R/W	R4000 EEROM interface.
CTRLD	0x1fa00040/4	R/W	Refresh counter preload value.
REF_CTR	0x1fa00048/c	R	Refresh counter.
GIO64_ARB	0x1fa00080/4	R/W	GIO64 arbitration parameters.
CPU_TIME	0x1fa00088/c	R/W	Arbiter CPU time period.
LB_TIME	0x1fa00098/c	R/W	Arbiter long burst time period.
MEMCFG0	0x1fa000c0/4	R/W	Memory size configuration register 0.
MEMCFG1	0x1fa000c8/c	R/W	Memory size configuration register 1.
CPU_MEMACC	0x1fa000d0/4	R/W	CPU main memory access Configuration parameters.
GIO_MEMACC	0x1fa000d8/c	R/W	GIO main memory access configuration parameters.
CPU_ERROR_ADDR	0x1fa000e0/4	R	CPU error address.
CPU_ERROR_STAT	0x1fa000e8/c	R	CPU error status.
CLR_ERROR_STAT	0x1fa000e8/c	W	Clears CPU error status register.
GIO_ERROR_ADDR	0x1fa000f0/4	R	GIO error address.
GIO_ERROR_STAT	0x1fa000f8/c	R	GIO error status.
CLR_ERROR_STAT	0x1fa000f8/c	W	Clears GIO error status register.
SYS_SEMAPHORE	0x1fa00100/4	R/W	System semaphore.
LOCK_MEMORY	0x1fa00108/c	R/W	Lock GIO out of memory.
EISA_LOCK	0x1fa00110/4	R/W	Lock EISA bus.
DMA_GIO_MASK	0x1fa00150/4	R/W	Mask to translate GIO64 address.
DMA_GIO_SUB	0x1fa00158/c	R/W	Substitution bits for translating GIO64 address.
DMA_CAUSE	0x1fa00160/4	R/W	DMA interrupt cause.
DMA_CTL	0x1fa00168/c	R/W	DMA control.
DMA_TLB_HI_0	0x1fa00180/4	R/W	DMA TLB entry 0 high.
DMA_TLB_LO_0	0x1fa00188/c	R/W	DMA TLB entry 0 low.
DMA_TLB_HI_1	0x1fa00190/4	R/W	DMA TLB entry 1 high.
DMA_TLB_LO_1	0x1fa00198/c	R/W	DMA TLB entry 1 low.
DMA_TLB_HI_2	0x1fa001a0/4	R/W	DMA TLB entry 2 high.
DMA_TLB_LO_2	0x1fa001a8/c	R/W	DMA TLB entry 2 low.
DMA_TLB_HI_3	0x1fa001b0/4	R/W	DMA TLB entry 3 high.
DMA_TLB_LO_3	0x1fa001b8/c	R/W	DMA TLB entry 3 low.
RPSS_CTR	0x1fa01000/4	R	RPSS 100 nanosecond counter.
DMA_MEMADR	0x1fa02000/4	R/W	DMA memory address.
DMA_MEMADRD	0x1fa02008/c	R/W	DMA memory address and set default parameters.
DMA_SIZE	0x1fa02010/4	R/W	DMA line count and width.
DMA_STRIDE	0x1fa02018/c	R/W	DMA line zoom and stride.
DMA_GIO_ADR	0x1fa02020/4	R/W	DMA GIO64 address, do not start DMA.
DMA_GIO_ADRS	0x1fa02028/c	R/W	DMA GIO64 address and start DMA.
DMA_MODE	0x1fa02030/4	R/W	DMA mode.
DMA_COUNT	0x1fa02038/c	R/W	DMA zoom count and byte count.
DMA_STDMA	0x1fa02040/4	R/W	Start virtual DMA.
DMA_RUN	0x1fa02048/c	R	Virtual DMA is running.DMA_MEM_ADRDS

DMA_MEMADRDS	0x1fa02070/4	R/W	DMA GIO64 address, set default and start DMA.
SEMAPHORE_0	0x1fa10000/4	R/W	Semaphore 0.
SEMAPHORE_1	0x1fa11000/4	R/W	Semaphore 1.
SEMAPHORE_2	0x1fa12000/4	R/W	Semaphore 2.
SEMAPHORE_3	0x1fa13000/4	R/W	Semaphore 3.
SEMAPHORE_4	0x1fa14000/4	R/W	Semaphore 4.
SEMAPHORE_5	0x1fa15000/4	R/W	Semaphore 5.
SEMAPHORE_6	0x1fa16000/4	R/W	Semaphore 6.
SEMAPHORE_7	0x1fa17000/4	R/W	Semaphore 7.
SEMAPHORE_8	0x1fa18000/4	R/W	Semaphore 8.
SEMAPHORE_9	0x1fa19000/4	R/W	Semaphore 9.
SEMAPHORE_10	0x1fa1a000/4	R/W	Semaphore 10.
SEMAPHORE_11	0x1fa1b000/4	R/W	Semaphore 11.
SEMAPHORE_12	0x1fa1c000/4	R/W	Semaphore 12.
SEMAPHORE_13	0x1fa1d000/4	R/W	Semaphore 13.
SEMAPHORE_14	0x1fa1e000/4	R/W	Semaphore 14.
SEMAPHORE_15	0x1fa1f000/4	R/W	Semaphore 15.

All of the MC registers will respond to two different addresses. It is up to the programmer to use the correct address depending on the endian mode of the processor. The MC is connected to the least significant 32 bits of the sysad bus. When a register is written the data must be driven on those bits. When register is read the data will be returned on those pins as well. If the processor is running in big endian mode the odd word addresses, (addresses that end in 4 and 0xc) are used. When the processor is running in little endian mode the even word addresses, (addresses that end in 0 and 8) are used.

5.1 CPU Control Register 0, CPUCTRL0

The CPU control register is a readable and writable register that controls some of the system functions described below.

Bit Name	Reset Value	Bit Number	Description
REFS	2	3:0	Number of lines to refresh in each burst divided by 2. This should be set for 4 lines unless there are 256Kx36 or 512Kx36 simms installed in the system, then this should be set for 8 lines. 1 – Refresh 2 lines. 2 – Refresh 4 lines. 4 – Refresh 8 lines. 15 – Refresh 30 lines.
RFE	1	4	Refresh enable. 0 – Refresh disabled. 1 – Refresh enabled.
GPR	0	5	Enable parity error reporting on GIO64 transactions. 0 – Disable parity error reporting. 1 – Enable parity error reporting.
MPR	0	6	Enable parity error reporting on main memory. 0 – Disable parity error reporting. 1 – Enable parity error reporting.
CPR	0	7	Enable parity error reporting on the CPU bus transactions. 0 – Disable parity error reporting. 1 – Enable parity error reporting.
DOG	0	8	Enable watchdog timer. If watchdog timer goes off it will reset the machine. 0 – Disable watch dog timer. 1 – Enable watch dog timer.
SIN	0	9	System initialization. Setting this bit will reset the entire system, which will have the same effect as cycling power on the machine. 0 – Do not reset machine. 1 – Reset machine.
GRR_	0	10	Graphics reset. Clearing this bit will assert reset to the graphics system. 0 – Assert graphics reset. 1 – Deassert graphics reset.
EN_LOCK	0	11	Enable EISA to lock memory from the CPU. This should be set to 0. Most likely EISA will not use eglock_n to run locked memory cycles. 0 – EISA cards cannot issue locks 1 – EISA cards can issues locks
CMD_PAR	0	12	Enable parity error reporting on syscmd bus. Version 1.2 of the R4000 will not support syscmd parity.
INT_EN	0	13	Enable interrupt writes from MC chip. This should be enabled or the R4000 will never get an interrupt.
SNOOP_EN	0	14	Enable snoop logic for graphics DMA’s.
PROM_WR_EN	0	15	Bus error interrupt enable for boot PROM

			writes. Deasserting this bit does not stop writes to the PROM, it only enables a bus error interrupt when PROM writes occur. HPC3 has a true PROM write enable register to block writes when a Flash PROM is used. 0 - Generate an interrupt on PROM writes. 1 - Do not interrupt on PROM writes.
WR_ST	0	16	Warm restart, starts a R4000 reset sequence without resetting any of the other chips and most of MC. Refresh continues during this reset sequence. 0 - Normal mode. 1 - Reset CPU.
UNDEF		17	Reserved.
LITTLE	0	18	Setting this bit will configure the MC chip to run in little endian mode. This bit is automatically set by the boot time initialization EEROM. The BIG/LITTLE pin on the MC chip will reflect the value of this register. If this bit does not match the endian mode of the R4000 the machine will not work. 0 - Big Endian. 1 - Little Endian.
WRST	0	19	Warm reset. Do a warm reset to R4000. This will generate a warm reset to the R4000 which does not reread the EEROM.
MUX_HWM	0x01	24:20	MUX chip CPU write fifo high water mark for de-asserting wrrdy_n. The lsb of this register is ignored. The high water mark is: 28 - (2nd level cache line size in words/2) This field is set to (32 - the high water mark). Therefore the smallest value that can be safely used for different lines sizes are: Line Size in Words / Water Mark / Value 4 / 26 / 6 8 / 24 / 8 16 / 20 / 12 32 / 12 / 20
BAD_PAR	0	25	Generate bad parity on data written by CPU to memory. This can be used to write a parity error diagnostic.
R4K_CHK_PAR_N		26	Send a syscmd to R4K that indicates that it should check parity on CPU reads from memory. If this is not asserted the R4K will not check parity on memory read data. 0 - Indicate to R4K to check parity. 1 - Indicate to R4K to not check parity.
BACK^2	0	27	Enable back to back GIO64 writes with no dead tristate cycles between MC and MUX. This should be enabled, but some testing is required to make sure the tristate overlap does not cause any problems. 0 - Disable add dead cycles. 1 - Enable back to back cycles.
BUS_RATE	0	31:28	Stall cycle between bus error data returned to

the CPU. This is required to throttle data returned to the R4000 to the rate the R4000 can write the second level cache. This should be set to the same value as the RD_COL value in the CPU_MEMACC register.

5.2 CPU Control Register 1, CPUCTRL1

The CPU control register one is a readable and writable register that controls some of the system functions described below.

Bit Name	Reset Value	Bit Number	Description
MC_HWM	0xC	3:0	MC chip CPU address/command fifo high water mark for de-asserting wrrdy_n. The value in this field is (17 - maximum number of entries desired in the fifo). The smallest value this field should be written with is 0x6 so that there is room in the fifo after all operations that are in flight and one read operations since reads are not stalled.
ABORT_EN	0	4	Enable GIO bus time outs. If this is disabled the system will hang on a bad GIO bus address.
UNDEF		11:5	Reserved.
HPC_FX	0	12	The endianess of HPC is fixed and is the HPC_LITTLE value below. This bit needs to be asserted for HPC1.5 and deasserted for HPC3.
HPC_LITTLE	0	13	Endian mode of HPC DMA if HPC_FX is asserted. This should be set to the endian mode of the CPU when a HPC1.5 is being used.
EXP0_FX	0	14	The endianess of EXP0 is fixed and is the EXP0_LITTLE value below.
EXP0_LITTLE	0	15	Endian mode of EXP0 DMA if EXP0_FX is asserted.
EXP1_FX	0	16	The endianess of EXP1 is fixed and is the EXP1_LITTLE value below.
EXP1_LITTLE	0	17	Endian mode of EXP1 DMA if EXP1_FX is asserted.
UNDEF		31:18	Reserved.

5.3 Watchdog Timer, DOGC and DOGR

The watchdog timer is a 20 bit counter that counts refresh bursts. If the watchdog timer is enabled and the counter rolls over to zero the machine will be reset as if power was cycled to the machine. If the refresh intervals are 64 microseconds apart, the counter will roll over in about 67 seconds. Writing to it with any data will clear the counter. If the timer is enabled the system needs to write to the dog reset, DOGR, location at least every 60 seconds or the timer will go off and reset the system. The watchdog timer enable is located in the CPUCTRL register. The format of the counter, DOGC, is shown below.

Bit Name	Reset Value	Bit Number	Description
DOG	0	19:0	Watchdog timer.

UNDEF		31:20	Reserved.

5.4 System ID, SYSID

The sysid register is a readable register that contains the revision of the chip and the EISA bus present bit.

Bit Name	Reset Value	Bit Number	Description
CHIP_REV	0	3:0	Revision of MC chip. 0 - Revision A 1 - Revision B
EISA		4	EISA bus is present. Determined by eisa_present_n pin.
UNDEF		31:5	Reserved.

5.5 RPSS Divider

The RPSS divider register determines how often and by how much the RPSS counter gets incremented. There is two fields. The first field is the amount to divide the CPU minus one. If this field is four the counter is incremented every five CPU clocks. The second field is the amount to add to the counter when it is incremented. For a 50 MHz processor the divider should be four, (divide by five), and the increment amount should be one. The RPSS counter will be incremented by one every 100 nanoseconds. For a 33 MHz processor the divider should be nine, (divide by 10), and the increment should be three. The RPSS counter in this case will be incremented every 300 nanoseconds by three.

Bit Name		Bit Number	Description
DIV	4	7:0	RPSS counter divider.
INC	1	15:8	RPSS counter increment.
UNDEF		31:16	Reserved.

5.6 R4000 Configuration EEROM Interface, EEROM

The R4000 reads a serial EEROM to set all of its configuration bits when it is powered up. One of these bits determines if the processor is configured in big or little endian mode. The MC chip also need to know if the processor is running in big or little mode. The first bit out of the EEROM will be used to determine which mode the processor is running in for the MC chip. These bits need to be changed so that the endian mode of the machine can be switched. In order to do this the processor needs to write the EEROM. This interface is provided so that the processor can write the EEROM and then force a cold reset which will force the processor to reload the bits from the EEROM. When the processor comes back up it will be using the new configuration values stored in the EEROM. The big/little endian bit is stored in the EEROM twice. The first bit is used by the MC chip to determine the endian mode. Another bit in the EEROM that is defined in the MIPS R4000 Microprocessor User's Guide will be used by the processor to determine the endian mode. The interface to the EEROM is the same as the interface to the system identification EEROM. It is up to the software to wiggle all of the control signals and the clock to the EEROM. The SI bit can not be written. The register is defined as follows:

Bit Name	Reset Value	Bit Number	Description
UNDEF	0	0	Reserved.
CS	0	1	EEROM chip select. Active high.
SCK	0	2	Serial EEROM clock.
SO	0	3	Data to serial EEROM.
SI		4	Data from serial EEROM.
UNDEF	0	31:5	Reserved.

5.7 Refresh Counter Preload, CTRLD

The refresh counter counts down and when it gets to zero it is reloaded with the value in this register. This counter operates at the frequency of the CPU, which will be 50 MHz (20 ns), for the first machine. This allows the interval for the refresh bursts to be completely programmable. This feature was added because when faster processors become available the counter preload value can be changed instead of changing the counter carry tap.

Bit Name		Bit Number	Description
REF	0x0C30	15:0	Refresh counter load value. The refresh counter gets reloaded with this value when it counts down to zero. This register should be set to the number of CPU cycles in 62.5 microseconds. When the sysad bus is running at less than 50 MHz this register needs to be changed.

5.8 Refresh Counter, REF_CTR

The refresh counter value can be read by reading this register. The format of the register is as follows:

Bit Name	Bit Number	Description
REFC	15:0	Refresh counter.

5.9 Arbitration Parameters, GIO64_ARB

The GIO64 arbiter has a number of parameters that are used to determine how it allocates time to the different devices. The GIO64 arbiter must know if each device is a real time device or a long burst device. The arbiter must also know the size of each device so that it can drive the GSIZE64 line and if each device can be a bus master. The HPC size bits is set at reset time from a bit in the R4000 initialization EEROM since MC needs this information to do the boot ROM fetches.

Bit Name	Reset Value	Bit Number	Description
HPC_SIZE		0	Width of data transfers to first HPC. The first HPC resides at 0x1fb80000 to 0x1fffffff. This should be set to 0 for HPC1 and HPC1.5. For HPC3 the size should be set to one. The R4000 serial EEROM should set this up when the machine is reset. 0 - 32 bit device. 1 - 64 bit device.
GRX_SIZE	0	1	Width of data transfers to graphics. Starter and Express graphics are 32 bit devices. Newport will be a 32 bit device in Indigo and a 64 bit device in other machines. 0 - 32 bit device. 1 - 64 bit device.
EXP0_SIZE	0	2	Width of data transfers to GIO64 Slot 0. For Indigo this should be 0. 0 - 32 bit device. 1 - 64 bit device.
EXP1_SIZE	0	3	Width of data transfers to GIO64 Slot 1. For Indigo this should be 0. 0 - 32 bit device. 1 - 64 bit device.
EISA_SIZE	0	4	Width of data transfers to the EISA bus. This should be set to 0. 0 - 32 bit device. 1 - 64 bit device.
HPC_EXP_SIZE		5	Width of data transfers to the second HPC that resides in the address space from 0x1fb00000 to 0x1fb7ffff. 0 - 32 bit device. 1 - 64 bit device.
GRX_RT	0	6	Graphics is a real time device. 0 - Long burst device. 1 - Real time device.
EXP0_RT	0	7	GIO64 expansion slot 0 is a real time device. 0 - Long burst device. 1 - Real time device.
EXP1_RT	0	8	GIO64 expansion slot 1 is a real time device. 0 - Long burst device. 1 - Real time device.
EISA_MST	0	9	EISA bus can be a GIO64 bus master. This should be zero in Indigo and one if Full House. 0 - Device is only a slave 1 - Device can be a master.
ONE_GIO	1	10	There is only one pipelined GIO64 bus. This should be set. 0 - System has two pipelined GIO64 buses. 1 - System has one pipelined GIO64 bus.
GRX_MST	0	11	Graphics can be a GIO64 bus master. This should be zero for all devices that exist. 0 - Device is only a slave 1 - Device can be a master.
EXP0_MST	0	12	GIO64 expansion slot 0 can be a GIO64 bus

Bit Name	Reset Value	Bit Number	Description
			master. This should only be set if a device that can become a bus master is installed. 0 – Device is only a slave 1 – Device can be a master.
EXP1_MST	0	13	GIO64 expansion slot 1 can be a GIO64 bus master. This should only be set if a device that can become a bus master is installed. 0 – Device is only a slave 1 – Device can be a master.
EXP0_PIPED	0	14	Expansion slot 0 is a pipelined device. This should be zero for Indigo and one for Full House. 0 – Device is nonpipelined. 1 – Device is pipelined.
EXP1_PIPED	0	15	Expansion slot 1 is a pipelined device. This should be 0. 0 – Device is nonpipelined. 1 – Device is pipelined.
UNDEF		31:16	Reserved.

5.10 GIO64 CPU Arbitration Time Period, CPU_TIME

The GIO64 arbiter has programmable time periods for the CPU and long burst devices. This register is the time period for the CPU. Once the CPU has been granted the bus it is allowed to use the bus for the time period. If once the time period is up another device wants to use the bus the CPU will be preempted. The CPU is give the bus for this time period even if it does not use it during this time period. The format of the CPU_TIME register is as follows:

Bit Name	Reset Value	Bit Number	Description
CPU_TIME	0x100	15:0	Number of GIO64 cycles in CPU time period.

5.11 GIO64 Burst Arbitration Time Period, LB_TIME

The LB_TIME register is just like the CPU_TIME register except it is for long burst devices. Unlike the CPU, when a long burst device is done using the bus the bus is given to the CPU. The format of the register is as follows:

Bit Name	Reset Value	Bit Number	Description
LB_TIME	0x200	15:0	Number of GIO64 cycles in long burst time period.

5.12 Memory Configuration Registers, MEMCFG0, MEMCFG1

The memory configuration registers indicate the size of the simms installed in the machine. There are four fields for each bank of simms. The first field indicates the base address of the simm. The second field is the size of the simm in megabytes. The third field indicates if the simm is valid, ie. installed. The last field indicates whether the simm contains one or two subbanks of DRAMs on it, (512Kx36, 2Mx36, and 8Mx36 simms contain two). The simms have to be installed in groups of four and all four simms must be the same size. The bank field should be zero for one subbank and one for two subbanks. The base address is an eight bit field. These eight bits will be compared with the address bits (29:22) along with the size and bank bit to determine which simm should be accessed. Memory needs to be configured in simm size order with the larges simms at the lowest base address. If the largest simms are not mapped first there will either be holes in the memory map or there will be overlap in the memory map. The base address of a simm must be aligned to the size of the bank boundary. For example a bank of 1Mx36 simms must be aligned to a 16 Mbyte boundary. This is because as the simms get larger more of the bottom base address bits are ignored. Address bits (31:30) must be zero when accessing main memory. The size encoding for the simms that will be supported and the number of subbanks (for setting the bank bit) are as follows:

00000 – 256K x 36 bits
00001 – 512K x 36 bits, 2 subbanks
00011 – 1M x 36 bits
00111 – 2M x 36 bits, 2 subbanks
01111 – 4M x 36 bits
11111 – 8M x 36 bits, 2 subbanks

Any other settings will have a defined, but very strange effect.

The valid bit indicates which simm slots contain simms. If more than one simm maps to an address a bus error interrupt will be generated and the memory operation will not complete.

The MEMCFG registers are defined as shown below. MEMCFG0 defines banks 0 and 1 and MEMCFG1 defines banks 2 and 3.

Bit Name	Reset Value	Bit Number	Description
BASE1		7:0	Base address for bank 1/3.
MSIZE1		12:8	Simm size for bank 1/3.
VLD1	0	13	Bank 1/3 is valid.
BNK1		14	Number of subbanks for bank 1/3.
UNDEF		15	Reserved.
BASE0		23:16	Base address for bank 0/2.
MSIZE0		28:24	Simm size for bank 0/2.
VLD0	0	29	Bank 0/2 is valid.
BNK0		30	Number of subbanks for bank 0/2.
UNDEF		31	Reserved.

5.13 Main Memory Access Configuration Parameters, CPU_MEMACC And GIO_MEMACC

The main memory access configuration parameter register holds the values that the main memory state machine uses when executing memory operations. This allows the timing critical parameters to be changed if it is necessary. The individual fields are described in the memory system controller section. The format of the CPU_MEMACC is show below: The number of cycles is in CPU clock cycles for the CPU register (20 ns clock) and GIO64 clock cycles for the GIO64 register (nonfixed clock rate).

Bit Name	Reset Value	Bit Number	Description
WR_COL	0x3	3:0	WR_COL equals the number of cycles the column address is driven before next column address can be drive for a page mode write access. When this register is set to 3 a page mode write will take place ever three cycles.
RD_COL	0x3	7:4	RD_COL equals the number of cycles the column address is driven before next column address can be drive for a page mode read access.
ROW	0x3	11:8	ROW equals the number of cycles minus one that the row address is driven before switching to the column address. This field needs to be set to 0x4.
RASH	0x4	15:12	RASH equals the number of cycles RAS must be high before it can be dropped again. This field can be set to 0x3.
RCASL	0x5	19:16	RCASL is the number of cycles RAS is low before CAS is driven high during refresh.
RASL	0x4	23:20	RASL is the number of cycles RAS is high before driving CAS low between lines during refresh.
CBR	0x1	27:24	CBR is the number of cycle CAS is low before RAS is taken low for refresh.
CAS_HALF	0	28	When asserted, CAS will be high for only one half of a clock cycle on page mode reads. This bit should be asserted so that three cycle page mode reads work.
UNDEF		31:29	Reserved.

The format of the GIO_MEMACC register is as follows:

Bit Name	Reset Value	Bit Number	Description
WR_COL	0x3	3:0	WR_COL equals the number of cycles the column address is driven before next column address can be drive for a page mode write access. The WR_COL should be set to two so that the memory system can keep up with the GIO64 bus.
RD_COL	0x3	7:4	RD_COL equals the number of cycles the column address is driven before next column address can be drive for a page mode read access. The RD_COL should be set to two so that the memory system can keep up with the GIO64 bus.
ROW	0x3	11:8	ROW equals the number of cycles to drive the row address before switching to the column address. This field should be set to 3 cycles.
RASH	0x4	15:12	RASH equals the number of cycles RAS must be high before it can be dropped again. This field should be set to 2 cycles.
CAS_HALF	0	16	Drive CAS high for only one half of a cycle during page mode reads. The Q_CAS bit should be set so that two cycle page mode reads will work.
ADDR_HALF	0	17	When asserted the column address is changed on

			the falling edge of the clock. The ADDR_HALF bit should be set so that two cycle page mode reads work.
UNDEF		31:18	Reserved.

5.14 CPU Error Address, CPU_ERROR_ADDR

The CPU error address register will contain the address of any CPU parity or bus errors. This register is only valid if one of the error bits is set in the CPU_ERROR_STATUS register. If the CPU_ERROR_STATUS register indicates a parity error bits (2:0) of should be ignored. The byte in error bits of the error status register can be used to determine the byte address of the parity error.

Bit Name	Reset Value	Bit Number	Description
ADDR	0	31:0	Address of error.

5.15 CPU Error Status, CPU_ERROR_STAT

The CPU error status register contains the cause of the bus error as well as the bytes that were in error for a parity error.

Bit Name	Reset Value	Bit Number	Description
BYTE	0	7:0	Byte(s) in error. Multiple bits are set if more than one parity error occurred on the same bus cycle. Bit 0 indicated a parity error occurred on byte lane 0 (bits 7:0 of the bus).
RD	0	8	Read parity error if PAR is asserted. If PAR = 1 and RD = 0 then a parity error occurred on CPU a memory write.
PAR	0	9	CPU parity error. Memory read if RD is asserted, otherwise the parity error occurred on a memory write. Memory parity errors can be disabled by deasserting the MPR (memory parity reporting enable) in the CPUCTRL0 register.
ADDR	0	10	Memory bus error. Address does not map to a valid bank of memory or the address for a MC register read or write was not correct for MC’s endian mode.
SYSAD_PAR	0	11	Sysad address or MC write data parity error. Only BYTE(3:0) in the error status register is valid. Parity checking can be disabled by deasserting the CPR bit in CPUCTRL0.
SYSCMD_PAR	0	12	Syscmd parity error. The BYTE field of the error status register is invalid. Parity error reporting can be disabled by deasserting CMD_PAR in CPUCTRL0. Some versions of the R4000 are know to not generate correct syscmd parity.
BAD_DATA	0	13	CPU sent a bad data identifier. The bad data bit was set in a data identifier from the

R4000. This error reporting can be disabled by deasserting the CPR bit in the CPUCTRL0 register.

5.16 GIO64 Error Address, GIO_ERROR_ADDR

The GIO error address register will contain the address of any GIO parity or bus errors. The GIO_ERROR_STATUS register will indicate if this register contains a valid address. Bits (2:0) should be ignored for 64 bit devices that generate parity errors and bits (1:0) for 32 bit devices.

Bit Name	Reset Value	Bit Number	Description
ADDR	0	31:0	Address of error.

5.17 GIO64 Error Status, GIO_ERROR_STAT

The GIO error status register contains the cause of the bus error interrupt as well as the bytes that were in error for a GIO bus parity error.

Bit Name	Reset Value	Bit Number	Description
BYTE	0	7:0	Byte(s) in error. Multiple bits can be set if more than one byte was in error. Bit 0 indicated a parity error occurred on byte lane 0 (bits 7:0 of the bus).
RD_PAR	0	8	GIO memory read parity error.
WR_PAR	0	9	GIO memory write parity error.
TIME	0	10	GIO transaction bus timed out. Timeouts are enabled with the ABORT_EN bit of the CPUCTRL1 register.
PROM	0	11	Write to PROM when PROM_WR_EN bit in CPUCTRL0 was not set.
ADDR	0	12	Parity error on GIO64 slave address cycle. This parity error checking can be disabled by deasserting the GPR bit in the CPUCTRL0 register.
BC	0	13	Parity error on GIO64 slave byte count cycle. This parity error checking can be disabled by deasserting the GPR bit in the CPUCTRL0 register
PIO_RD_PAR	0	14	Data parity error on GIO programmed I/O read. This parity error checking can be disabled by deasserting the GPR bit in the CPUCTRL0 register.
PIO_WR_PAR	0	15	Data parity error on GIO programmed I/O write. This parity error checking can be disabled by deasserting the GPR bit in the CPUCTRL0 register.

5.18 Semaphores, SYS_SEMAPHORE and SEMAPHORE_x

There are sixteen user semaphores and one system semaphore in the MC chip. When a read is issued to a semaphore the value of the one bit semaphore is returned and the semaphore is set to a one. The semaphore can be cleared or set by writing to the semaphore. The sixteen user semaphores are on different pages so that they can be selectively mapped in the user address space.

Bit Name	Reset Value	Bit Number	Description
SEM	0	0	Single bit semaphore.

5.19 Lock GIO64 Out of Memory, LOCK_MEMORY

The LOCK_MEMORY register when set to zero will lock all devices except the CPU out of main memory. This can be used by the CPU to do a locked sequence.

Bit Name	Reset Value	Bit Number	Description
LOCK_N	1	0	Lock device except CPU out of memory when set. 0 – Locked 1 – Unlocked

5.20 Lock EISA Out of Memory, EISA_LOCK

The EISA_LOCK register when written with a zero will assert gelock_n to the EISA chips. This will allow the CPU to issue a locked sequence over the EISA bus. When the locked sequence is over the EISA_LOCK register should be set to one. While this be is set to zero the EISA bus will not be granted to an EISA device. The actual use of this register may change since EISA is being implemented in a different way than was originally planned. The LOCK_MEMORY register can also be used to lock EISA out of main memory.

Bit Name	Reset Value	Bit Number	Description
LOCK	1	0	Lock EISA out of memory when set. 0 – Locked 1 – Unlocked

5.21 RPSS Counter, RPSS_CTR

The RPSS counter is a 100 nanosecond increment, 32 bit counter. It is readable, but not writable. When the maximum count it reached it just rolls over and no interrupt is generated. The RPSS_DIVIDER register determines when this register get updated.

Bit Name	Reset Value	Bit Number	Description
CNT	0	31:0	Counter Value

6. MC Pins

There are 220 signal pins on the MC array including the pins for the PLL. The 299 CPGA and 304 pin metal quad flat pack packages will be used for this part.

All inputs have TTL thresholds except for the clock inputs (cpu_clk, gio_clk and masterout_clk) which have CMOS thresholds. All outputs swing rail to rail except for the 3.3 V interface to the R4000: cpu_sysad, cpu_sysadc, cpu_syscmd, cpu_syscmdp, cpu_validin_n, cpu_extrqst_n, cpu_wrrdy_n, cpu_ivdack_n, cpu_cold_rst_n, cpu_reset_n, and cpu_vccok. All output buffers are 4 mA except the following buffers are 8 mA with moderate slew rate control: gio_adp, gio_ad, gio_read, gio_masdly, gio_slvdly, gio_vld_parity_n, gio_as_n, gio_gsize64, gio_bpre_n, mem_we, and mem_cas. The gio_adp, jtck, jtdi, and jtms pins have very weak pull ups on them.

6.1 R4000 Interface

The R4000 is a 64 bit multiplexed address and data bus.

cpu_sysad(31:0)	i/o	Address and data bus.
parity_error(7:0)	input	Parity error from MUX.
cpu_sysadc(3:0)	i/o	Parity over the cpu.sysad bus.
cpu_syscmd(8:0)	i/o	Command bus from R4000.
cpu_syscmdp	i/o	Parity on cpu.syscmd bus.
cpu_validin_n	output	System is driving cpu.sysad and cpu.syscmd with valid data.
cpu_validout_n	input	R4000 is driving cpu.sysad and cpu.syscmd with valid data.
cpu_extrqst_n	output	Request control of the system interface from the R4000.
cpu_release_n	input	R4000 released control of the system interface to the MC chip.
cpu_wrrdy_n	output	Signals that the R4000 is capable of accepting another write request.
cpu_ivdack_n	output	R4000 invalidate or update completed successfully.
cpu_modeclk	input	R4000 serial boot mode data clock.
cpu_eerom_dato	output	Serial EEROM data to set up EEROM read and for writing EEROM.
cpu_eerom_dati	input	Serial eerom data from EEROM.
cpu_eerom_cs	output	Chip select for serial EEROM.
cpu_eerom_sck	output	Serial EEROM clock.
cpu_vccok	output	Start reading serial eerom.
cpu_cold_rst_n	output	Release after EEROM is read.
cpu_reset_n	output	Release to start processor.

6.2 Main Memory Interface

mem_addr(11:0)	output	Address to memory, both row and column.
mem_ras(7:0)	output	Row address strobe, one per subbank.
mem_cas(15:0)	output	Column address strobe, one per byte width of memory.
mem_we	output	Memory write enable.
mux_gio_sel	output	GIO clock owns MUX.

mux_cpu_sel	output	CPU clock owns MUX.
mux_cpu_push	output	Push data onto CPU fifo.
mux_cpu_mem_oe	output	Output enable for MUX sysad and memory output buffers.
mux_data_sel(2:0)	output	Selects read/write data.
mux_dir	output	Data is going to memory if zero, otherwise data is from memory.
mux_graphics(1:0)	output	Determines which delay signal is use: 0 - slvdly 1 - grxdly(0) 2 - grxdly(1) 3 - grxdly(2)
mux_aen_mem	output	A register enable when cpu_sel = 1, else memory <-> GIO indicator.
mux_ben_ctrl	output	B register enable when cpu_sel = 1, otherwise fifo push/pop for GIO memory fifo.
mux_cen_fifo	output	C register enable when cpu_sel = 1, otherwise fifo push/pop for GIO memory fifo, part of GIO command.
mux_par_flush	output	Generate bad parity if cpu_sel = 1, otherwise flush GIO command fifo.
mux_giostb	output	GIO command is valid.
mux_mc_dly	output	Early masdly/slvdly for MUX chip used on GIO slave reads.

6.3 EISA Bus Interface

eisa_ecp_n	i	CPU command pending on the EISA bus.
eisa_eglock_n	i	The EISA bus wants to lock the CPU out of main memory.
eisa_gelock_n	o	CPU wants to lock the EISA bus out.
eisa_memory	o	This is a EISA memory operation, not an I/O operation.
eisa_present_n	i	EISA bus present.

6.4 GIO64 Interface Signals

gio_ad(31:0)	i/o	Least significant bytes of the GIO64 bus.
gio_adp(3:0)	i/o	Address and data parity bits.
gio_vld_parity_n	i/o	GIO64 has valid parity.
gio_as_n	i/o	Address strobe.
gio_grx_as_n	output	Graphics space address strobe. For graphics GIO bus slaves only.
gio_eisa_as_n	i/o	GIO64 address strobe for EISA.
gio_read	i/o	Read or write and valid bus cycle.
gio_masdly	i/o	Master delay.
gio_slvdly	i/o	Slave delay.
gio_dmasync_n	i	DMA synchronization signal from graphics.
gio_grxdly0	i	Graphics delay 0.
gio_grxdly1	i	Graphics delay 1.
gio_grxdly2	i	Graphics delay 2.
gio_grxrst_n	output	Graphics reset.

gio_bpre_n	output	Bus preempt.
gio_hpc_req_n	input	HPC bus request.
gio_hpc_gnt_n	output	HPC bus grant.
gio_exp0_req_n	input	GIO64 expansion slot 0 bus request
gio_exp0_gnt_n	output	GIO64 expansion slot 0 bus grant
gio_exp1_req_n	input	GIO64 expansion slot 1 bus request
gio_exp1_gnt_n	output	GIO64 expansion slot 1 bus grant
gio_grx_req_n	input	Graphics bus request.
gio_grx_gnt_n	output	Graphics bus grant.
gio_eisa_req_n	input	GIO64 bus request for EISA.
gio_eisa_gnt_n	output	GIO64 bus grant for EISA.
gio_gsize64	output	GIO64 bus master size. 0 - 32 bits wide. 1 - 64 bits wide.
gio_ctl(3:0)	output	Controls for flops that connect GIO64 bus to graphics. (1,0) - Active low OE from nonpiped to piped. (2) - Active high OE from piped bus 0 to nonpiped. (3) - Active high OE from piped bus 1 to nonpiped.

6.5 Misc Signals

cpu_clk	input	CPU clock. 50 MHz
masterout_clk	input	CPU clock, 50 or 75 MHz, masterout from processor.
gio_clk	input	GIO64 Clock. 33 MHz
int_bus_err	output	Bus error interrupt.
int_dma_done	output	DMA master operation complete.
int_cpu_n(5:0)	input	Interrupts from INT2.
big_endian	output	CPU is running in big endian mode.
reset_out_n	output	Kick one shot reset pulse generator. This is an open drain output. MC only drives this pin low to reset the machine. Therefore this pin needs a pullup.
gio_reset_out_n	output	Reset to GIO64 devices.
reset_in	input	Power on reset.
jtdi	input	JTAG data in.
jtdo	output	JTAG data out.
jtms	input	JTAG mode.
jtck	input	JTAG clock.
quick_boot	input	Shorten reset sequence.
pll_reset_in_in	input	On rev B MC only. Connect to pll_reset_out_n. This will reset the MC plls.
pll_reset_out_n	output	Reset PLL in MUX and HPC3.
cpu_pll_lp1	analog	CPU clock pll loop filter output.
cpu_pll_lp2	analog	CPU clock pll loop filter input.
cpu_pll_vss		CPU clock pll ground input.
cpu_pll_vdd		CPU clock pll power input.
cpu_pll_agnd		CPU clock pll ground output.
gio_pll_lp1	analog	GIO clock pll loop filter output.
gio_pll_lp2	analog	GIO clock pll loop filter input.

gio_pll_vss	GIO clock pll ground input.
gio_pll_vdd	GIO clock pll power input.
gio_pll_agnd	GIO clock pll ground output.

6.6 MC Pins Delays

The clock to output pin times are dependent on the capacitive load that pin. The chart below lists all of the digital pins on the chip with the capacitive load, worst case clock to output time, setup time, and hold times. There is also a column which indicates what clock (cpu_clk, gio_clk, or masterout_clk) the signal is being used to flop the signal. Some signals are flopped with both the gio_clk and the cpu_clk. For signals that are a bus the times for the worst signal in that bus are given.

Signal	Clock	load (pf)	clk->q (ns)	setup (ns)	hold (ns)
parity_error(7:0)	cpu, gio			4.0	-1.5
cpu_sysad(31:0)	cpu	50	15.7	4.4	-2.1
cpu_sysadc(3:0)	cpu	60	15.7	4.6	-2.1
cpu_syscmd(8:0)	cpu	50	14.4	4.3	-2.2
cpu_syscmdp	cpu	50	14.3	4.2	-2.1
cpu_validin_n	cpu	50	12.9		
cpu_validout_n	cpu			3.6	-1.7
cpu_extrqst_n	cpu	50	12.7		
cpu_release_n	cpu			3.9	-1.6
cpu_wrrdy_n	cpu	50	12.6		
cpu_ivdack_n	cpu	50	12.9		
cpu_modeclk	master			3.7	3.0
cpu_eerom_dato	master	50	15.8		
cpu_eerom_dati	master			9.34	2.3
cpu_eerom_cs	master	50	15.8		
cpu_eerom_sck	master	50	16.1		
cpu_vccok	master	50	13.0		
cpu_cold_rst_n	master	50	12.9		
cpu_reset_n	master	50	12.5		
mem_addr(11:0)	cpu, gio	50	12.3		
mem_ras(7:0)	cpu, gio	50	12.0		
mem_cas(15:0)	cpu, gio	50	10.7		
mem_we	cpu, gio	50	12.7		
mux_gio_sel	gio	50	12.0		
mux_cpu_sel	cpu	50	11.9		
mux_cpu_push	cpu	50	11.9		
mux_cpu_mem_oe	cpu	50	11.7		
mux_data_sel(2:0)	cpu, gio	50	12.7		
mux_dir	cpu, gio	50	12.6		
mux_graphics(1:0)	gio	50	11.2		
mux_aen_mem	cpu, gio	50	12.7		
mux_ben_ctrl	cpu, gio	50	12.7		
mux_cen_fifo	cpu, gio	50	12.7		
mux_par_flush	cpu, gio	50	12.8		
mux_giostb	gio	50	11.7		
mux_mc_dly	gio	50	12.0		
eisa_ecp_n	gio			3.6	-1.4
eisa_eglock_n	gio			3.6	-1.4
eisa_gelock_n	gio	50	10.7		
eisa_memory	gio	50	10.6		
eisa_present_n	gio			3.6	-1.4
gio_ad(31:0)	gio	120	17.9	3.7	-1.3
gio_adp(3:0)	gio	120	16.8	3.7	-1.4
gio_vld_parity_n	gio	130	16.9	3.7	-1.4

Signal	Clock	load (pf)	clk->q (ns)	setup (ns)	hold (ns)
gio_as_n	gio	130	17.9	3.7	-1.4
gio_grx_as_n	gio	50	12.2		
gio_eisa_as_n	gio	50	11.8	3.7	-1.4
gio_read	gio	150	18.5	3.5	-1.4
gio_masdly	gio	150	18.5	3.5	-1.4
gio_slvdly	gio	150	18.5	3.5	-1.4
gio_dmasync_n	gio			3.6	-1.4
gio_grxdly0	gio			6.0	-1.6
gio_grxdly1	gio			6.0	-1.4
gio_grxdly2	gio			6.0	-1.4
gio_grxrst_n	gio	50	11.3		
gio_bpre_n	gio	100	13.9		
gio_hpc_req_n	gio			3.8	-1.4
gio_hpc_gnt_n	gio	50	11.2		
gio_exp0_req_n	gio			3.8	-1.4
gio_exp0_gnt_n	gio	50	11.3		
gio_exp1_req_n	gio			3.8	-1.4
gio_exp1_gnt_n	gio	50	11.3		
gio_grx_req_n	gio			3.8	-1.4
gio_grx_gnt_n	gio	50	11.4		
gio_eisa_req_n	gio			3.8	-1.4
gio_eisa_gnt_n	gio	50	11.1		
gio_gsize64	gio	100	13.8		
gio_ctl(3:0)	gio	80	14.6		
int_bus_err	gio	50	11.1		
int_dma_done	gio	50	11.0		
int_cpu_n(5:0)	cpu			3.9	-1.4
big_endian	cpu	50	11.4		
reset_out_n	cpu	30	9.2		
gio_reset_out_n	gio	50	14.9		
reset_in	master			3.9	2.8
jtdi	jtck			4.7	-1.8
jtdo	jtck	50	20.0		
jtms	jtck				
quick_boot	master			15.0	2.6
pll_reset_out_n	master	50	11.1		

6.7 MC Scan Chain and Pinout, 299 CPGA

This chart shows the package pin numbers for a 299 CPGA, the signal name, the type of signal, the number of the flop in the serial chain for the input, output and output enable, and the active level of the signal.

Pin Number	Signal Name	Type	In	Out	Enable	Active
A2	VSS	POWER				
A3	VDD	POWER				
A4	VSS	POWER				
A5	VDD4	POWER	0	0	0	
A6	CPU_SYSAD_10	BIDIR	216	215	219	LOW
A7	CPU_SYSAD_15	BIDIR	227	226	219	LOW
A8	CPU_SYSAD_18	BIDIR	235	234	238	LOW
A9	VDD	POWER				
A10	VSS	POWER				
A11	VDD	POWER				
A12	VSS	POWER				
A13	VDD	POWER				
A14	CPU_SYSAD_27	BIDIR	256	255	257	LOW
A15	VSS	POWER	0	0	0	
A16	CPU_SYSCMD_3	BIDIR	275	274	278	LOW
A17	CPU_SYSCMD_6	BIDIR	282	281	278	LOW
A18	VDD	POWER				
A19	VSS	POWER				
A20	VDD	POWER				
B1	VDD	POWER				
B2	MEM_ADDR_10	OUTPUT	0	188	0	
B3	CPU_SYSADC_0	BIDIR	191	190	201	LOW
B4	CPU_SYSAD_5	BIDIR	205	204	201	LOW
B5	CPU_SYSAD_7	BIDIR	209	208	201	LOW
B6	CPU_SYSAD_8	BIDIR	212	211	219	LOW
B7	CPU_SYSAD_13	BIDIR	223	222	219	LOW
B8	CPU_SYSAD_16	BIDIR	231	230	238	LOW
B9	CPU_SYSAD_20	BIDIR	240	239	238	LOW
B10	CPU_SYSAD_22	BIDIR	244	243	238	LOW
B11	VDD4	POWER	0	0	0	
B12	CPU_SYSADC_3	BIDIR	248	247	257	LOW
B13	CPU_SYSAD_25	BIDIR	252	251	257	LOW
B14	CPU_SYSAD_31	BIDIR	265	264	257	LOW
B15	CPU_SYSCMD_2	BIDIR	273	272	278	LOW
B16	CPU_SYSCMD_5	BIDIR	280	279	278	LOW
B17	CPU_SYSCMD_8	BIDIR	286	285	278	LOW
B18	CPU_VALIDIN_N	OUTPUT	0	290	0	
B19	CPU_COLD_RST_N	OUTPUT	0	303	0	
B20	VSS	POWER				
C1	VSS	POWER				
C2	MEM_ADDR_8	OUTPUT	0	186	0	
C3	VDD4	POWER	0	0	0	
C4	CPU_SYSAD_1	BIDIR	196	195	201	LOW
C5	CPU_SYSAD_6	BIDIR	207	206	201	LOW
C6	CPU_SYSADC_1	BIDIR	193	210	219	LOW
C7	CPU_SYSAD_11	BIDIR	218	217	219	LOW
C8	VSS	POWER	0	0	0	
C9	CPU_SYSAD_17	BIDIR	233	232	238	LOW
C10	CPU_SYSAD_19	BIDIR	237	236	238	LOW

Pin Number	Signal Name	Type	In	Out	Enable	Active
C11	CPU_SYSAD_26	BIDIR	254	253	257	LOW
C12	CPU_SYSAD_28	BIDIR	259	258	257	LOW
C13	CPU_SYSAD_29	BIDIR	261	260	257	LOW
C14	CPU_SYSCMD_0	BIDIR	269	268	278	LOW
C15	CPU_SYSCMD_4	BIDIR	277	276	278	LOW
C16	CPU_SYSCMD_7	BIDIR	284	283	278	LOW
C17	CPU_WRRDY_N	OUTPUT	0	288	0	
C18	CPU_CLK	CLOCK	0	0	0	
C19	CPU_PLL_LP2	PLL	0	0	0	
C20	VDD	POWER				
D1	VDD	POWER				
D2	MEM_ADDR_4	OUTPUT	0	182	0	
D3	MEM_ADDR_6	OUTPUT	0	184	0	
D4	VDD	POWER	0	0	0	
D5	CPU_SYSAD_3	BIDIR	200	199	201	LOW
D6	CPU_SYSAD_2	BIDIR	198	197	201	LOW
D7	CPU_SYSAD_9	BIDIR	214	213	219	LOW
D8	CPU_SYSAD_14	BIDIR	225	224	219	LOW
D9	CPU_SYSADC_2	BIDIR	229	228	238	LOW
D10	CPU_SYSAD_23	BIDIR	246	245	238	LOW
D11	VSS	POWER	0	0	0	
D12	CPU_SYSAD_30	BIDIR	263	262	257	LOW
D13	CPU_SYSCMDP	BIDIR	267	266	278	LOW
D14	CPU_IVDACK_N	OUTPUT	0	287	0	
D15	CPU_RESET_N	OUTPUT	0	301	0	
D16	VDD4	POWER	0	0	0	
D17	VDD3_4	POWER	0	0	0	
D18	CPU_PLL_VDD	POWER	0	0	0	
D19	CPU_VALIDOUT_N	INPUT	293	0	0	
D20	INT_CPU_N_0	INPUT	294	0	0	
E1	MEM_ADDR_1	OUTPUT	0	179	0	
E2	MEM_ADDR_2	OUTPUT	0	180	0	
E3	MEM_ADDR_3	OUTPUT	0	181	0	
E4	MEM_ADDR_9	OUTPUT	0	187	0	
E5	VSS	POWER	0	0	0	
E6	CPU_SYSAD_0	BIDIR	194	192	201	LOW
E7	CPU_SYSAD_4	BIDIR	203	202	201	LOW
E8	CPU_SYSAD_12	BIDIR	221	220	219	LOW
E9	VDD4	POWER	0	0	0	
E10	CPU_SYSAD_21	BIDIR	242	241	238	LOW
E11	CPU_SYSAD_24	BIDIR	250	249	257	LOW
E12	VDD4	POWER	0	0	0	
E13	CPU_SYSCMD_1	BIDIR	271	270	278	LOW
E14	CPU_EXTRQST_N	OUTPUT	0	289	0	
E15	CPU_VCCOK	OUTPUT	0	302	0	
E16	CPU_PLL_LP1	PLL	0	0	0	
E17	MASTEROUT_CLK	CLOCK	0	0	0	
E18	CPU_RELEASE_N	INPUT	291	0	0	
E19	INT_CPU_N_1	INPUT	295	0	0	
E20	INT_CPU_N_3	INPUT	297	0	0	
F1	MEM_RAS_6	OUTPUT	0	176	0	
F2	MEM_RAS_7	OUTPUT	0	177	0	
F3	MEM_ADDR_0	OUTPUT	0	178	0	
F4	MEM_ADDR_7	OUTPUT	0	185	0	
F5	MEM_ADDR_11	OUTPUT	0	189	0	

Pin Number	Signal Name	Type	In	Out	Enable	Active
F16	CPU_PLL_VSS	POWER	0	0	0	
F17	CPU_PLL_AGND	PLL	0	0	0	
F18	INT_CPU_N_2	INPUT	296	0	0	
F19	INT_CPU_N_5	INPUT	299	0	0	
F20	GIO_GRX_AS_N	OUTPUT	0	10	0	
G1	MEM_RAS_2	OUTPUT	0	172	0	
G2	MEM_RAS_4	OUTPUT	0	174	0	
G3	VSS	POWER	0	0	0	
G4	VDD	POWER	0	0	0	
G5	MEM_ADDR_5	OUTPUT	0	183	0	
G16	VSS	POWER	0	0	0	
G17	INT_CPU_N_4	INPUT	298	0	0	
G18		NC	0	0	0	
G19	VDD	POWER	0	0	0	
G20	RESET_OUT_N	OTHER	0	0	304	LOW
H1	MEM_CAS_14	OUTPUT	0	165	0	
H2	VDD	POWER	0	0	0	
H3	MEM_RAS_0	OUTPUT	0	170	0	
H4	MEM_RAS_3	OUTPUT	0	173	0	
H5	MEM_RAS_5	OUTPUT	0	175	0	
H16	VSS	POWER	0	0	0	
H17	PLL_RESET_IN_N	PLL	0	0	0	
H18	BIG_ENDIAN	OUTPUT	0	300	0	
H19	CPU_EEROM_SCK	OUTPUT	0	312	0	
H20	VSS	POWER				
J1	VSS	POWER				
J2	MEM_CAS_12	OUTPUT	0	163	0	
J3	MEM_CAS_15	OUTPUT	0	166	0	
J4	VSS	POWER	0	0	0	
J5	MEM_RAS_1	OUTPUT	0	171	0	
J16	VSS	POWER	0	0	0	
J17	PLL_RESET_OUT_N	OUTPUT	0	306	0	
J18	GIO_RESET_OUT_N	OUTPUT	0	1	0	
J19	CPU_EEROM_DATO	OUTPUT	0	309	0	
J20	VDD	POWER				
K1	VDD	POWER				
K2	MEM_CAS_10	OUTPUT	0	161	0	
K3	MEM_CAS_13	OUTPUT	0	164	0	
K4	MEM_CAS_9	OUTPUT	0	160	0	
K5	MEM_CAS_11	OUTPUT	0	162	0	
K16	CPU_EEROM_CS	OUTPUT	0	307	0	
K17	RESET_IN	INPUT	305	0	0	
K18	GIO_GRXRST_N	OUTPUT	0	2	0	
K19	VSS	POWER	0	0	0	
K20	VSS	POWER				
L1	VSS	POWER				
L2	MEM_CAS_8	OUTPUT	0	159	0	
L3	MEM_CAS_4	OUTPUT	0	155	0	
L4	VSS	POWER	0	0	0	
L5	MEM_CAS_6	OUTPUT	0	157	0	
L16	JTDO	JTAG	0	0	0	
L17	CPU_EEROM_DATI	INPUT	308	0	0	
L18	JTMS	JTAG	0	0	0	
L19	CPU_MODECLK	INPUT	311	0	0	
L20	VDD	POWER				

Pin Number	Signal Name	Type	In	Out	Enable	Active
M1	VDD	POWER				
M2	MEM_CAS_7	OUTPUT	0	158	0	
M3	MEM_CAS_2	OUTPUT	0	153	0	
M4	MEM_CAS_0	OUTPUT	0	169	0	
M5	VDD	POWER	0	0	0	
M16	GIO_GSIZE64	OUTPUT	0	3	0	
M17	TP1	TEST	0	0	0	
M18	ENTEI	OTHER	0	0	0	
M19	JTCK	JTAG	0	0	0	
M20	VSS	POWER				
N1	VSS	POWER				
N2	MEM_CAS_5	OUTPUT	0	156	0	
N3	MEM_CAS_1	OUTPUT	0	152	0	
N4	MUX_GIO_SEL	OUTPUT	0	131	0	
N5	MUX_CPU_PUSH	OUTPUT	0	148	0	
N16	GIO_HPC_GNT_N	OUTPUT	0	7	0	
N17	GIO_EXP1_GNT_N	OUTPUT	0	6	0	
N18	GIO_GRX_GNT_N	OUTPUT	0	4	0	
N19	QUICK_BOOT	INPUT	312	0	0	
N20	JTDI	JTAG	0	0	0	
P1	MEM_CAS_3	OUTPUT	0	154	0	
P2	MEM_WE	OUTPUT	0	150	0	
P3	MUX_CPU_SEL	OUTPUT	0	149	0	
P4	VSS	POWER	0	0	0	
P5	MUX_AEN_MEM	OUTPUT	0	132	0	
P16	EISA_ECP_N	INPUT	18	0	0	
P17	EISA_PRESENT_N	INPUT	20	0	0	
P18	GIO_GRX_REQ_N	INPUT	9	0	0	
P19	GIO_EXP0_GNT_N	OUTPUT	0	5	0	
P20	TP0	TEST	0	0	0	
R1	VSS	POWER	0	0	0	
R2	MUX_CPU_MEM_OE	OUTPUT	0	147	0	
R3	MUX_DATA_SEL_1	OUTPUT	0	129	0	
R4	MUX_CEN_FIFO	OUTPUT	0	135	0	
R5	MUX_PAR_FLUSH	OUTPUT	0	136	0	
R16	GIO_AS_N	BIDIR	23	24	37	LOW
R17	INT_BUS_ERR	OUTPUT	0	21	0	
R18	GIO_HPC_REQ_N	INPUT	13	0	0	
R19	GIO_EXP1_REQ_N	INPUT	12	0	0	
R20	GIO_BPRE_N	OUTPUT	0	8	0	
T1	MUX_DATA_SEL_2	OUTPUT	0	146	0	
T2	MUX_DATA_SEL_0	OUTPUT	0	128	0	
T3	MUX_GRAPHICS_1	OUTPUT	0	127	0	
T4	VDD	POWER	0	0	0	
T5	PARITY_ERROR_7	INPUT	145	0	0	
T6	GIO_PLL_LP1	PLL	0	0	0	
T7	GIO_PLL_AGND	PLL	0	0	0	
T8	GIO_MASDLY	BIDIR	118	117	119	LOW
T9	GIO_CTL_1	OUTPUT	0	108	0	
T10	GIO_AD_27	BIDIR	65	64	57	LOW
T11	GIO_AD_21	BIDIR	51	52	40	LOW
T12	GIO_AD_15	BIDIR	106	107	91	LOW
T13	GIO_AD_11	BIDIR	98	99	91	LOW
T14	GIO_AD_2	BIDIR	79	80	74	LOW
T15	GIO_ADP_3	BIDIR	31	35	36	LOW

Pin Number	Signal Name	Type	In	Out	Enable	Active
T16	GIO_ADP_1	BIDIR	29	33	36	LOW
T17	EISA_EGLOCK_N	INPUT	19	0	0	
T18	EISA_GELOCK_N	OUTPUT	0	16	0	
T19	EISA_GNT_N	OUTPUT	0	14	0	
T20	GIO_EXP0_REQ_N	INPUT	11	0	0	
U1	MUX_DIR	OUTPUT	0	134	0	
U2	MUX_GRAPHICS_0	OUTPUT	0	126	0	
U3	MUX_BEN_CTRL	OUTPUT	0	133	0	
U4	PARITY_ERROR_6	INPUT	144	0	0	
U5	PARITY_ERROR_5	INPUT	143	0	0	
U6	GIO_PLL_VSS	POWER	0	0	0	
U7	PARITY_ERROR_4	INPUT	142	0	0	
U8	GIO_CTL_3	OUTPUT	0	110	0	
U9	GIO_AD_31	BIDIR	73	72	57	LOW
U10	GIO_AD_25	BIDIR	60	61	57	LOW
U11	GIO_AD_23	BIDIR	55	56	40	LOW
U12	GIO_AD_17	BIDIR	44	43	40	LOW
U13	GIO_AD_13	BIDIR	102	103	91	LOW
U14	GIO_AD_10	BIDIR	96	97	91	LOW
U15	GIO_AD_0	BIDIR	75	76	74	LOW
U16	GIO_AD_3	BIDIR	81	82	74	LOW
U17	GIO_ADP_0	BIDIR	28	32	36	LOW
U18	INT_DMA_DONE	OUTPUT	0	22	0	
U19	EISA_MEMORY	OUTPUT	0	17	0	
U20	EISA_REQ_N	INPUT	15	0	0	
V1	VSS	POWER				
V2	MUX_GIOSTB	OUTPUT	0	125	0	
V3	GIO_CLK	CLOCK	0	0	0	
V4	GIO_PLL_VDD	POWER	0	0	0	
V5	PARITY_ERROR_2	INPUT	140	0	0	
V6	GIO_GRXDLY0	INPUT	121	0	0	
V7	GIO_DMASYNC_N	INPUT	120	0	0	
V8	GIO_CTL_2	OUTPUT	0	109	0	
V9	GIO_AD_29	BIDIR	69	68	57	LOW
V10	VDD	POWER	0	0	0	
V11	GIO_AD_19	BIDIR	47	48	40	LOW
V12	VSS	POWER	0	0	0	
V13	GIO_AD_18	BIDIR	45	46	40	LOW
V14	GIO_AD_14	BIDIR	104	105	91	LOW
V15	GIO_AD_8	BIDIR	92	93	91	LOW
V16	GIO_AD_5	BIDIR	85	86	74	LOW
V17	GIO_AD_1	BIDIR	77	78	74	LOW
V18	GIO_VLD_PARITY_N	BIDIR	26	27	25	LOW
V19	GIO_EISA_AS_N	BIDIR	38	39	37	LOW
V20	VDD	POWER				
W1	VDD	POWER				
W2	MUX_MC_DLY	OUTPUT	0	124	0	
W3	GIO_PLL_LP2	PLL	0	0	0	
W4	PARITY_ERROR_3	INPUT	141	0	0	
W5	PARITY_ERROR_0	INPUT	138	0	0	
W6	GIO_GRXDLY2	INPUT	123	0	0	
W7	GIO_SLVDLY	BIDIR	115	114	116	LOW
W8	GIO_AD_30	BIDIR	71	70	57	LOW
W9	GIO_AD_28	BIDIR	67	66	57	LOW
W10	VSS	POWER	0	0	0	

Pin Number	Signal Name	Type	In	Out	Enable	Active
W11	GIO_AD_26	BIDIR	63	62	57	LOW
W12	GIO_AD_24	BIDIR	58	59	57	LOW
W13	GIO_AD_20	BIDIR	49	50	40	LOW
W14	GIO_AD_16	BIDIR	42	41	40	LOW
W15	GIO_AD_9	BIDIR	94	95	91	LOW
W16	GIO_AD_6	BIDIR	87	88	74	LOW
W17	GIO_AD_4	BIDIR	83	84	74	LOW
W18	VDD	POWER	0	0	0	
W19	GIO_ADP_2	BIDIR	30	34	36	LOW
W20	VSS	POWER				
X1	VSS	POWER				
X2	VDD	POWER				
X3	VSS	POWER				
X4	PARITY_ERROR_1	INPUT	139	0	0	
X5	GIO_GRXDLY1	INPUT	122	0	0	
X6	GIO_READ	BIDIR	112	113	111	LOW
X7	GIO_CTL_0	OTHER	0	0	0	
X8	VDD	POWER				
X9	VSS	POWER				
X10	VDD	POWER				
X11	VSS	POWER				
X12	VDD	POWER				
X13	GIO_AD_22	BIDIR	53	54	40	LOW
X14	VDD	POWER	0	0	0	
X15	GIO_AD_12	BIDIR	100	101	91	LOW
X16	GIO_AD_7	BIDIR	89	90	74	LOW
X17	VSS	POWER				
X18	VDD	POWER				
X19	VSS	POWER				
X20	VDD	POWER				

6.8 MC Scan Chain and Pinout, 304 MQUAD

This chart shows the package pin numbers for a 304 MQUAD, the signal name, the type of signal, the number of the flop in the serial chain for the input, output and output enable, and the active level of the signal.

Pin Number	Signal Name	Type	In	Out	Enable	Active
1	VDD	POWER	0	0	0	
2	VDD4	POWER	0	0	0	
3	VSS	POWER	0	0	0	
4	CPU_SYSADC_0	BIDIR	191	190	201	LOW
5	CPU_SYSAD_0	BIDIR	194	192	201	LOW
6	CPU_SYSAD_1	BIDIR	196	195	201	LOW
7	CPU_SYSAD_2	BIDIR	198	197	201	LOW
8	CPU_SYSAD_3	BIDIR	200	199	201	LOW
9	CPU_SYSAD_4	BIDIR	203	202	201	LOW
10	CPU_SYSAD_5	BIDIR	205	204	201	LOW
11	VSS	POWER	0	0	0	
12	CPU_SYSAD_6	BIDIR	207	206	201	LOW
13	VSS	POWER	0	0	0	
14	CPU_SYSAD_7	BIDIR	209	208	201	LOW
15	VSS2_4	POWER	0	0	0	
16	VDD4	POWER	0	0	0	
17	CPU_SYSADC_1	BIDIR	193	210	219	LOW
18	VSS2_4	POWER	0	0	0	
19	CPU_SYSAD_8	BIDIR	212	211	219	LOW
20	CPU_SYSAD_9	BIDIR	214	213	219	LOW
21	VSS	POWER	0	0	0	
22	CPU_SYSAD_10	BIDIR	216	215	219	LOW
23	CPU_SYSAD_11	BIDIR	218	217	219	LOW
24	CPU_SYSAD_12	BIDIR	221	220	219	LOW
25	CPU_SYSAD_13	BIDIR	223	222	219	LOW
26	CPU_SYSAD_14	BIDIR	225	224	219	LOW
27	CPU_SYSAD_15	BIDIR	227	226	219	LOW
28	VDD4	POWER	0	0	0	
29	VSS	POWER	0	0	0	
30	CPU_SYSADC_2	BIDIR	229	228	238	LOW
31	CPU_SYSAD_16	BIDIR	231	230	238	LOW
32	CPU_SYSAD_17	BIDIR	233	232	238	LOW
33	CPU_SYSAD_18	BIDIR	235	234	238	LOW
34	CPU_SYSAD_19	BIDIR	237	236	238	LOW
35	CPU_SYSAD_20	BIDIR	240	239	238	LOW
36	CPU_SYSAD_21	BIDIR	242	241	238	LOW
37	CPU_SYSAD_22	BIDIR	244	243	238	LOW
38	CPU_SYSAD_23	BIDIR	246	245	238	LOW
39	VSS3_4	POWER	0	0	0	
40	VDD4	POWER	0	0	0	
41	VSS	POWER	0	0	0	
42	CPU_SYSADC_3	BIDIR	248	247	257	LOW
43	CPU_SYSAD_24	BIDIR	250	249	257	LOW
44	CPU_SYSAD_25	BIDIR	252	251	257	LOW
45	CPU_SYSAD_26	BIDIR	254	253	257	LOW
46	CPU_SYSAD_27	BIDIR	256	255	257	LOW
47	CPU_SYSAD_28	BIDIR	259	258	257	LOW
48	CPU_SYSAD_29	BIDIR	261	260	257	LOW
49	CPU_SYSAD_30	BIDIR	263	262	257	LOW

Pin Number	Signal Name	Type	In	Out	Enable	Active
50	CPU_SYSAD_31	BIDIR	265	264	257	LOW
51	VDD4	POWER	0	0	0	
52	VSS	POWER	0	0	0	
53	CPU_SYSCMDP	BIDIR	267	266	278	LOW
54	CPU_SYSCMD_0	BIDIR	269	268	278	LOW
55	CPU_SYSCMD_1	BIDIR	271	270	278	LOW
56	CPU_SYSCMD_2	BIDIR	273	272	278	LOW
57	CPU_SYSCMD_3	BIDIR	275	274	278	LOW
58	VSS	POWER	0	0	0	
59	CPU_SYSCMD_4	BIDIR	277	276	278	LOW
60	CPU_SYSCMD_5	BIDIR	280	279	278	LOW
61	VSS2_4	POWER	0	0	0	
62	CPU_SYSCMD_6	BIDIR	282	281	278	LOW
63	VSS2_4	POWER	0	0	0	
64	CPU_SYSCMD_7	BIDIR	284	283	278	LOW
65	VSS	POWER	0	0	0	
66	CPU_SYSCMD_8	BIDIR	286	285	278	LOW
67	VSS	POWER	0	0	0	
68	VDD4	POWER	0	0	0	
69	CPU_IVDACK_N	OUTPUT	0	287	0	
70	CPU_WRRDY_N	OUTPUT	0	288	0	
71	CPU_EXTRQST_N	OUTPUT	0	289	0	
72	CPU_VALIDIN_N	OUTPUT	0	290	0	
73	CPU_RESET_N	OUTPUT	0	301	0	
74	CPU_COLD_RST_N	OUTPUT	0	303	0	
75	CPU_VCCOK	OUTPUT	0	302	0	
76	VDD3	POWER	0	0	0	
77	VDD	POWER	0	0	0	
78	CPU_CLK	CLOCK	0	0	0	
79	CPU_PLL_LP1	PLL	0	0	0	
80	CPU_PLL_LP2	PLL	0	0	0	
81	CPU_PLL_VSS	POWER	0	0	0	
82	CPU_PLL_VDD	POWER	0	0	0	
83	CPU_PLL_AGND	PLL	0	0	0	
84	MASTEROUT_CLK	CLOCK	0	0	0	
85	VSS	POWER	0	0	0	
86	CPU_VALIDOUT_N	INPUT	293	0	0	
87	CPU_RELEASE_N	INPUT	291	0	0	
88	INT_CPU_N_0	INPUT	294	0	0	
89	VSS2	POWER	0	0	0	
90	INT_CPU_N_1	INPUT	295	0	0	
91	VSS2	POWER	0	0	0	
92	INT_CPU_N_2	INPUT	296	0	0	
93	INT_CPU_N_3	INPUT	297	0	0	
94	VSS	POWER	0	0	0	
95	INT_CPU_N_4	INPUT	298	0	0	
96	VDD	POWER	0	0	0	
97	INT_CPU_N_5	INPUT	299	0	0	
98	PLL_RESET_IN_N	PLL	0	0	0	
99	GIO_GRX_AS_N	OUTPUT	0	10	0	
100	VSS	POWER	0	0	0	
101	VDD	POWER	0	0	0	
102	PLL_RESET_OUT_N	OUTPUT	0	306	0	
103	BIG_ENDIAN	OUTPUT	0	300	0	
104	GIO_RESET_OUT_N	OUTPUT	0	1	0	

Pin Number	Signal Name	Type	In	Out	Enable	Active
105	RESET_OUT_N	OTHER	0	0	304	LOW
106	GIO_GRXRST_N	OUTPUT	0	2	0	
107	CPU_EEROM_SCK	OUTPUT	0	312	0	
108	CPU_EEROM_CS	OUTPUT	0	307	0	
109	CPU_EEROM_DATO	OUTPUT	0	309	0	
110	RESET_IN	INPUT	305	0	0	
111	VSS2	POWER	0	0	0	
112	VSS3	POWER	0	0	0	
113	VDD	POWER	0	0	0	
114	CPU_EEROM_DATI	INPUT	308	0	0	
115	CPU_MODECLK	INPUT	311	0	0	
116	JTDO	JTAG	0	0	0	
117	JTCK	JTAG	0	0	0	
118	JTMS	JTAG	0	0	0	
119	JTDI	JTAG	0	0	0	
120	ENTEI	OTHER	0	0	0	
121	QUICK_BOOT	INPUT	312	0	0	
122	TP1	TEST	0	0	0	
123	TP0	TEST	0	0	0	
124	GIO_GSIZE64	OUTPUT	0	3	0	
125	GIO_GRX_GNT_N	OUTPUT	0	4	0	
126	GIO_EXP1_GNT_N	OUTPUT	0	6	0	
127	GIO_EXP0_GNT_N	OUTPUT	0	5	0	
128	GIO_HPC_GNT_N	OUTPUT	0	7	0	
129	GIO_BPRE_N	OUTPUT	0	8	0	
130	VDD	POWER	0	0	0	
131	GIO_GRX_REQ_N	INPUT	9	0	0	
132	VSS	POWER	0	0	0	
133	GIO_EXP1_REQ_N	INPUT	12	0	0	
134	GIO_EXP0_REQ_N	INPUT	11	0	0	
135	VSS2	POWER	0	0	0	
136	GIO_HPC_REQ_N	INPUT	13	0	0	
137	VSS2	POWER	0	0	0	
138	EISA_GNT_N	OUTPUT	0	14	0	
139	VSS2	POWER	0	0	0	
140	EISA_REQ_N	INPUT	15	0	0	
141	EISA_GELOCK_N	OUTPUT	0	16	0	
142	VSS	POWER	0	0	0	
143	EISA_MEMORY	OUTPUT	0	17	0	
144	EISA_PRESENT_N	INPUT	20	0	0	
145	EISA_EGLOCK_N	INPUT	19	0	0	
146	EISA_ECP_N	INPUT	18	0	0	
147	INT_DMA_DONE	OUTPUT	0	22	0	
148	INT_BUS_ERR	OUTPUT	0	21	0	
149	GIO_EISA_AS_N	BIDIR	38	39	37	LOW
150	GIO_AS_N	BIDIR	23	24	37	LOW
151	GIO_VLD_PARITY_N	BIDIR	26	27	25	LOW
152	VDD	POWER	0	0	0	
153	VDD3	POWER	0	0	0	
154	GIO_ADP_0	BIDIR	28	32	36	LOW
155	GIO_ADP_1	BIDIR	29	33	36	LOW
156	GIO_ADP_2	BIDIR	30	34	36	LOW
157	GIO_ADP_3	BIDIR	31	35	36	LOW
158	VDD	POWER	0	0	0	
159	GIO_AD_0	BIDIR	75	76	74	LOW

Pin Number	Signal Name	Type	In	Out	Enable	Active
160	GIO_AD_1	BIDIR	77	78	74	LOW
161	GIO_AD_2	BIDIR	79	80	74	LOW
162	GIO_AD_3	BIDIR	81	82	74	LOW
163	GIO_AD_4	BIDIR	83	84	74	LOW
164	VSS	POWER	0	0	0	
165	GIO_AD_5	BIDIR	85	86	74	LOW
166	VSS2	POWER	0	0	0	
167	GIO_AD_6	BIDIR	87	88	74	LOW
168	VSS2	POWER	0	0	0	
169	GIO_AD_7	BIDIR	89	90	74	LOW
170	GIO_AD_8	BIDIR	92	93	91	LOW
171	GIO_AD_9	BIDIR	94	95	91	LOW
172	VDD	POWER	0	0	0	
173	GIO_AD_10	BIDIR	96	97	91	LOW
174	GIO_AD_11	BIDIR	98	99	91	LOW
175	GIO_AD_12	BIDIR	100	101	91	LOW
176	GIO_AD_13	BIDIR	102	103	91	LOW
177	GIO_AD_14	BIDIR	104	105	91	LOW
178	GIO_AD_15	BIDIR	106	107	91	LOW
179	GIO_AD_16	BIDIR	42	41	40	LOW
180	GIO_AD_17	BIDIR	44	43	40	LOW
181	VDD	POWER	0	0	0	
182	VSS	POWER	0	0	0	
183	GIO_AD_18	BIDIR	45	46	40	LOW
184	GIO_AD_19	BIDIR	47	48	40	LOW
185	GIO_AD_20	BIDIR	49	50	40	LOW
186	GIO_AD_21	BIDIR	51	52	40	LOW
187	GIO_AD_22	BIDIR	53	54	40	LOW
188	GIO_AD_23	BIDIR	55	56	40	LOW
189	GIO_AD_24	BIDIR	58	59	57	LOW
190	VSS3	POWER	0	0	0	
191	GIO_AD_25	BIDIR	60	61	57	LOW
192	GIO_AD_26	BIDIR	63	62	57	LOW
193	GIO_AD_27	BIDIR	65	64	57	LOW
194	VSS	POWER	0	0	0	
195	VDD	POWER	0	0	0	
196	GIO_AD_28	BIDIR	67	66	57	LOW
197	GIO_AD_29	BIDIR	69	68	57	LOW
198	GIO_AD_30	BIDIR	71	70	57	LOW
199	GIO_AD_31	BIDIR	73	72	57	LOW
200	GIO_CTL_0	OTHER	0	0	0	
201	GIO_CTL_1	OUTPUT	0	108	0	
202	GIO_CTL_2	OUTPUT	0	109	0	
203	GIO_CTL_3	OUTPUT	0	110	0	
204	GIO_SLVDLY	BIDIR	115	114	116	LOW
205	GIO_MASDLY	BIDIR	118	117	119	LOW
206	GIO_READ	BIDIR	112	113	111	LOW
207	VDD	POWER	0	0	0	
208	GIO_DMASYNC_N	INPUT	120	0	0	
209	VSS	POWER	0	0	0	
210	GIO_GRXDLY2	INPUT	123	0	0	
211	GIO_GRXDLY1	INPUT	122	0	0	
212	VSS2	POWER	0	0	0	
213	GIO_GRXDLY0	INPUT	121	0	0	
214	PARITY_ERROR_0	INPUT	138	0	0	

Pin Number	Signal Name	Type	In	Out	Enable	Active
215	VSS2	POWER	0	0	0	
216	PARITY_ERROR_1	INPUT	139	0	0	
217	PARITY_ERROR_2	INPUT	140	0	0	
218	VSS	POWER	0	0	0	
219	PARITY_ERROR_3	INPUT	141	0	0	
220	PARITY_ERROR_4	INPUT	142	0	0	
221	PARITY_ERROR_5	INPUT	143	0	0	
222	GIO_PLL_AGND	PLL	0	0	0	
223	GIO_PLL_VDD	POWER	0	0	0	
224	GIO_PLL_VSS	POWER	0	0	0	
225	GIO_PLL_LP2	PLL	0	0	0	
226	GIO_PLL_LP1	PLL	0	0	0	
227	GIO_CLK	CLOCK	0	0	0	
228	VDD3	POWER	0	0	0	
229	VDD	POWER	0	0	0	
230	PARITY_ERROR_6	INPUT	144	0	0	
231	PARITY_ERROR_7	INPUT	145	0	0	
232	MUX_MC_DLY	OUTPUT	0	124	0	
233	MUX_PAR_FLUSH	OUTPUT	0	136	0	
234	MUX_GIOSTB	OUTPUT	0	125	0	
235	MUX_CEN_FIFO	OUTPUT	0	135	0	
236	MUX_BEN_CTRL	OUTPUT	0	133	0	
237	MUX_AEN_MEM	OUTPUT	0	132	0	
238	VDD	POWER	0	0	0	
239	VSS	POWER	0	0	0	
240	MUX_GRAPHICS_0	OUTPUT	0	126	0	
241	MUX_GRAPHICS_1	OUTPUT	0	127	0	
242	VSS2	POWER	0	0	0	
243	MUX_DIR	OUTPUT	0	134	0	
244	MUX_DATA_SEL_0	OUTPUT	0	128	0	
245	VSS2	POWER	0	0	0	
246	MUX_DATA_SEL_1	OUTPUT	0	129	0	
247	MUX_DATA_SEL_2	OUTPUT	0	146	0	
248	VSS	POWER	0	0	0	
249	VDD	POWER	0	0	0	
250	MUX_CPU_MEM_OE	OUTPUT	0	147	0	
251	MUX_CPU_PUSH	OUTPUT	0	148	0	
252	MUX_CPU_SEL	OUTPUT	0	149	0	
253	MUX_GIO_SEL	OUTPUT	0	131	0	
254	VSS	POWER	0	0	0	
255	VDD	POWER	0	0	0	
256	MEM_WE	OUTPUT	0	150	0	
257	MEM_CAS_0	OUTPUT	0	169	0	
258	MEM_CAS_1	OUTPUT	0	152	0	
259	MEM_CAS_2	OUTPUT	0	153	0	
260	MEM_CAS_3	OUTPUT	0	154	0	
261	MEM_CAS_4	OUTPUT	0	155	0	
262	MEM_CAS_5	OUTPUT	0	156	0	
263	MEM_CAS_6	OUTPUT	0	157	0	
264	MEM_CAS_7	OUTPUT	0	158	0	
265	VSS	POWER	0	0	0	
266	VDD	POWER	0	0	0	
267	VSS2	POWER	0	0	0	
268	MEM_CAS_8	OUTPUT	0	159	0	
269	MEM_CAS_9	OUTPUT	0	160	0	

Pin Number	Signal Name	Type	In	Out	Enable	Active
270	MEM_CAS_10	OUTPUT	0	161	0	
271	MEM_CAS_11	OUTPUT	0	162	0	
272	MEM_CAS_12	OUTPUT	0	163	0	
273	MEM_CAS_13	OUTPUT	0	164	0	
274	MEM_CAS_14	OUTPUT	0	165	0	
275	MEM_CAS_15	OUTPUT	0	166	0	
276	VDD	POWER	0	0	0	
277	VSS	POWER	0	0	0	
278	MEM_RAS_0	OUTPUT	0	170	0	
279	MEM_RAS_1	OUTPUT	0	171	0	
280	MEM_RAS_2	OUTPUT	0	172	0	
281	MEM_RAS_3	OUTPUT	0	173	0	
282	MEM_RAS_4	OUTPUT	0	174	0	
283	MEM_RAS_5	OUTPUT	0	175	0	
284	VSS	POWER	0	0	0	
285	VDD	POWER	0	0	0	
286	MEM_RAS_6	OUTPUT	0	176	0	
287	VSS	POWER	0	0	0	
288	MEM_RAS_7	OUTPUT	0	177	0	
289	MEM_ADDR_0	OUTPUT	0	178	0	
290	VSS2	POWER	0	0	0	
291	MEM_ADDR_1	OUTPUT	0	179	0	
292	MEM_ADDR_2	OUTPUT	0	180	0	
293	VSS	POWER	0	0	0	
294	MEM_ADDR_3	OUTPUT	0	181	0	
295	VDD	POWER	0	0	0	
296	MEM_ADDR_4	OUTPUT	0	182	0	
297	MEM_ADDR_5	OUTPUT	0	183	0	
298	MEM_ADDR_6	OUTPUT	0	184	0	
299	MEM_ADDR_7	OUTPUT	0	185	0	
300	MEM_ADDR_8	OUTPUT	0	186	0	
301	MEM_ADDR_9	OUTPUT	0	187	0	
302	MEM_ADDR_10	OUTPUT	0	188	0	
303	MEM_ADDR_11	OUTPUT	0	189	0	
304	VDD3	POWER	0	0	0	

7.0 MC Revision B Fixes

This is a list of the changes made to revision B of the MC chip.

1. PLL reset in pin added to gain more control over pll reset signal mainly for testing.

2. Graphics address strobe added. LG2 board needs a decoded address strobe. This address strobe is an output only so it will only work with slave devices. This strobe is active for addresses 0x1f000000 - 0x1f3fffff.

3. Revision A of the MC chip would hang if a memory address that was not mapped by MC was read from while doing a dirty cache line write back. This has been fixed. In cpu_mc_rd_cmd sticky_invalid was f_sticky_invalid and f_cpu_invalid_bnk instead of an or function.

4. A true pll bypass mux that does not depend on the state of LP2 was added to both plls.

5. Added flops to boundary scan chain that were missing in revision A. All I/O pins can be controlled by the jtag controller except mem_cas.

6. Changed the chip revision field in the sysid register to 1.

7. Added a arc to the gio_memory_state fsm in the main_rd_stall state for preempted transfers. This arc was missing and if rd_col was set to two and a gio memory read was preempted while this fsm was in the main_rd_stall state the machine would hang.

8. Fixed writing MC registers that are preceded by a read with write forthcoming. The fsm in cpu_mc_command, dispatch_dispatch state needs to qualify validout = 1 with the fifo not being full. The MUX fifo was full so the write to MC was stalled by the R4000, but cpu_mc_command popped the fifo to execute the MC register write even though the MC fifo was empty. This caused the fifo to be completely full and hung the machine.

9. A vdma that was waiting for the GIO bus about to do a page table look up that got a GIO bus grant and preempt in the same cycle would hang and not deassert its bus request signal.

10. Valid parity was not always being driven high before being tri-stated. In the gio_pio_fsm drive gio_vld_parity high while in the own_state and waiting for a GIO operation.

11. Cpu memory error address is sometimes wrong if a write follows a read and the read gets a parity error. The write address was captured. This was because the fsm in cpu_memory_error would not reload the memory address if the memory controller was given back to back operations. The fsm_in_idle signal would not get asserted, which reloads the memory address register.

12. Delay cells between different clocks in scan chain have been changed to 40 ns to prevent clock delay problems between master_clk and the other clocks.

13. R4000 block writes that are not part of a dirty cache miss, (ie from the cache instruction), that have data on cpu_sysad(31:0) of the form 0x1faxxxxx

will hang the machine. This is because in cpu_mc_command, dispatch_dispatch state, if mc_space gets asserted the fsm will goto a state to wait for the MC register read or write to complete. The fsm's in cpu_mc_rd_cmd and cpu_mc_wr_cmd realize that this is not a real MC register read or write so never start a command, therefore they never send a complete signal that cpu_mc_command is waiting for.

8.0 Document Changes

Oct 12, 1990
1. Added config bit to allow RAS pins to come out encoded to support 32 simms. Added two more MEMCFG registers to support this.
2. Changed reset value for CPU_TIME and LB_TIME registers.

Feb. 4, 1991
1. Updated memory timing waveforms, and chip pin names.
2. Added Little bit to DMA descriptors.

Feb. 23, 1991
1. Fixed the memory timing waveforms. Also changed some of the reset values on the memory registers.
2. Changed address map.

April 12, 1991
1. Removed a lot old, wrong data.

July 11, 1991
1. Updated all but section 3. Graphics DMA section still needs work.

August 29, 1991
1. New arbiter and changes for EISA 2.5 µs problem.

September 6. 1991
1. Update time EISA holds GIO64 bus.

September 24, 1991
1. Changed Mux control signal names.
2. Changed Memory Map.

October 22, 1991
1. Added definition of GIO error registers.
2. Added restrictions to GIO PIO by the CPU.

October 28, 1991
1. Added section on R3000 support.

November 19, 1991
1. Fixed GIO_ERROR_STATUS register.
2. Added fixed endianess bits to CPU CTRL 1 register.

November 22, 1991
1. Changed memory map. The GIO expansion slots moved.

February 5, 1992
1. Removed R3K support.
2. Removed exclusive arbiter.
3. Removed most special support for EISA.

April 10, 1992
1. Fixed register definitions.
2. Added information about I/O pins.

Silicon Graphics
Computer Systems

MC Specification
R4000 Project

Revision 1.15
May 13, 2000

James Tornes

Table of Contents