Published in Proceedings of IEEE Compcon'96, pp.132-139, IEEE Computer Society Press, Los Alamitos, CA.

The Performance and PowerPC Platform™ Specification Implementation of The MPC106 Chipset

Christopher D. Bryant, Michael J. Garcia, Brian K. Reynolds, Laura A. Weber, Glen E. Wilson
Motorola Incorporated
6501 William Cannon Drive West
Austin, Texas 78735-8598

ABSTRACT

The MPC106 provides a PowerPC™ Platform specification compliant bridge between the family of PowerPC Microprocessors and the PCI bus. The MPC106's PCI support will allow system designers to rapidly design systems using peripherals already designed for PCI and the other standard interfaces available in the personal computer hardware environment. The MPC106 also integrates secondary cache control and a high-performance memory controller which supports various types of DRAM and ROM. The MPC106 is the second of a family of Motorola products that provide system level support for industry standard interfaces to be used with PowerPC microprocessors.

This paper describes the MPC106, its performance, and its implementation of the PowerPC Platform specification.

The PowerPC Platform specification is formally known as the Common Hardware Reference Platform or CHRP™.

ARCHITECTURAL OVERVIEW

The MPC106 connects directly to the PCI bus and the processor bus, and shares the data bus to system memory with the processor. The MPC106 is partitioned into four interfaces, the processor interface, the second level (L2) cache interface, the memory interface, and the PCI interface. A central control unit provides arbitration and coherency control between each of the interfaces. This central control unit supports concurrent operations on the processor/memory bus and the PCI bus. The processor interface is a high bandwidth, high performance, TTL compatible interface which supports any of the MPC60x PowerPC microprocessors. The processor interface supports multiprocessor configurations from 1 up to 4 processors. The L2 interface is highly programmable and flexible and supports both on chip and off chip interfaces. The on chip L2 interface supports four different sizes, 256KBytes, 512KBytes, 1MByte, and 2MBytes, and both write through and copy back modes. The memory interface supports either DRAM or EDO DRAM in sizes up to one gigabyte, which can be split into one to eight banks. The memory interface also supports page mode accesses and parity checking, Read Modify Write parity checking, or ECC checking of single and double bit errors and correction of single bit errors. The ROM interface supports both ROM and FLASH ROM and allows the ROM to be located on the memory interface or the PCI bus. The PCI interface is fully compliant with the PCI Local Bus Specification Revision 2.1 and all its supplements and functions as both a master and target device.

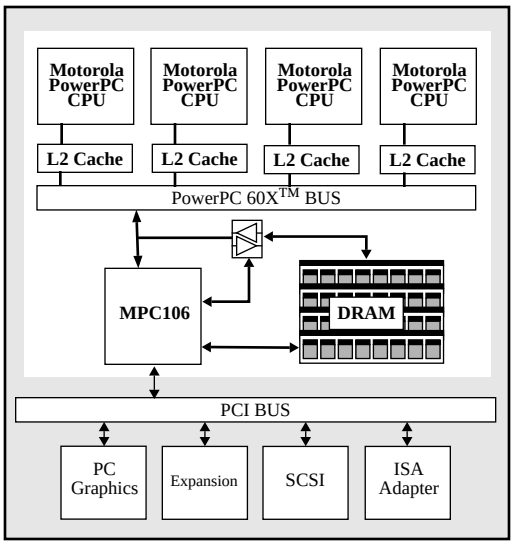

Figure 1 **High Performance system using the MPC106**

The MPC106 includes many features designed to optimize system performance. Since actual system performance numbers are highly dependent on the system configuration and the type of software running on it, system performance numbers are beyond the scope of this paper. Instead we will discuss the performance related features of the MPC106, give best case timings for a variety of system configurations, and discuss how to get an estimate of expected performance for different hardware limitations.

PERFORMANCE

One general performance strategy used in designing the MPC106 was to make processor accesses to memory as fast as possible. This was done by optimizing the address path for processor reads from memory, and by including page-mode operation. The L2 cache performance was also enhanced by changing the castout method compared to the previous chip set, the MPC105. For PCI accesses to memory, the performance of reads that cross multiple cache lines can be greatly improved by the speculative read option or by the use of the Memory Read Multiple command. For systems which do not require hardware-enforced coherency between the processor and PCI, the no-snoop mode was also added to improve the timing for PCI accesses to memory.

Fast Processor Accesses to Memory

The address path was designed to minimize the latency for processor reads from memory. When the chip is idle, the address from the processor bus is latched each cycle and the associated row address is driven on the address pins going to the memory. The row address is always driven from processor address bits 9-20 to speed up the generation of the row address. Thus, when TS_ is asserted, the transaction is decoded, the memory address driven the cycle after TS_, and RAS_ is asserted one cycle later if appropriate. This results in an initial latency of 8 clock cycles for a 60ns DRAM and a 66 MHz bus, as shown in Figure 2.

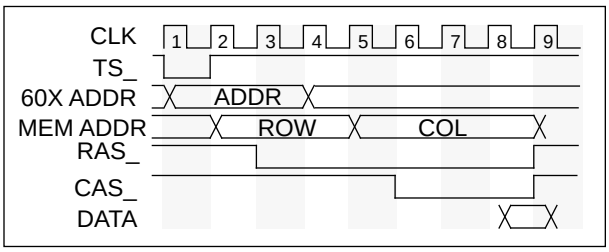

Figure 2 **Fastest Processor Read from Normal DRAM**

Page Mode

The MPC106 implements a one-level pipeline on the processor bus. This allows the decode of a second transaction to begin during the data phase of the first transaction. If the current transaction is a burst access to DRAM, and the pipelined transaction is also to DRAM (either single-beat or burst), then the MPC106 attempts a page-mode access. During the last clock cycle of the last data phase, the memory block compares the row addresses of the two transactions. If they are different, the RAS_ precharge begins immediately. If the row addresses are the same the RAS_ precharge and the RAS_ to CAS_ delay can be avoided, thus reducing the latency for the first data access of the second transfer. This implementation of page mode improves performance for page hits without incurring a penalty for page misses.

Fast L2 Cache Castout

Another aspect of performance for processor accesses to memory is the penalty incurred when a miss in the L2 cache requires a castout of modified data to memory. In this case, the data for the read cannot be fetched from memory until the L2 cache performs a write to memory. The MPC106 minimizes this latency by allowing the L2 cache to write into an internal buffer while starting the read transaction in the memory system.

As shown in Figure 3, the beginning of the transaction looks like a normal read. The processor's address phase is terminated normally with the assertion of AACK_, the row address is driven to the memory subsystem, and RAS_ is asserted. However, the assertion of CAS_ is delayed while the L2 castout data is transferred into the MPC106's internal buffer. This data transfer occurs at the fastest L2 cache access timing, before the address of the L2 castout is known. Once the L2 castout data has been transferred, CAS_ is asserted, and the processor read access completes. In the case shown, the initial read latency is 11 clock cycles, which is only a 3 cycle penalty for the L2 castout.

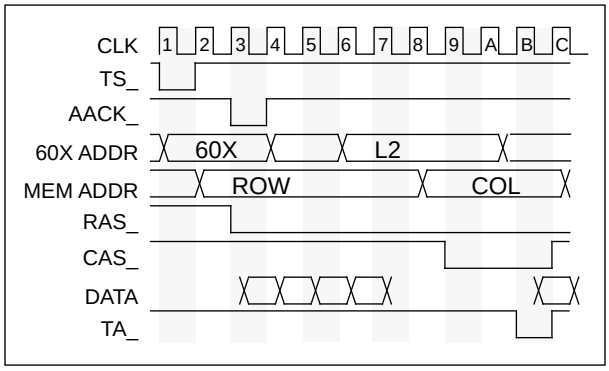

Figure 3 **L2 Castout Timing**

Speculative Reads

When acting as a target, the MPC106 disconnects at the end of each cache line to simplify the hardware enforced coherency. To reduce the latency due to this disconnect, the MPC106 includes a selectable speculative read feature used for multiple cache line transfers from memory to PCI, . When this feature is enabled, the MPC106 starts the snoop to the processor and L2 cache of the next sequential cache line address (32-bytes) when the current PCI read is accessing the third double-word of the current cache line. Once the speculative snoop response is known and PCI has completed the read, the data at the speculative address is fetched from memory and loaded in the buffer in anticipation of the next PCI request. If a different address is requested, the speculative operation is halted and any data latched in the PCI read buffer is invalidated. If the speculative read feature is enabled the MPC106 will attempt speculative reads for all PCI Memory Read transaction types including Memory Read Line transactions. Because PCI Memory Read Multiple transactions, by definition, are a request for multiple cache lines of data the MPC106 will attempt a speculative read operation regardless of whether or not the speculative read feature is enabled.

No-Snoop Mode

The MPC106 provides hardware-enforced coherency between the primary and secondary caches and PCI devices. This requires all PCI accesses to memory to be snooped on the processor bus. Although the snoops do not affect the best-case timings for the PCI accesses to memory, when the processor bus is busy and the snoop delayed, PCI performance can be significantly degraded. For applications which do not require the hardware-enforced coherency, snooping can be turned off to eliminate some of the delay.

Best Case Performance of Processor Memory Accesses

The processor memory access performance numbers were generated with the following assumptions. All of the tests were generated with the PowerPC 604™ microprocessor using a bus speed of 66 MHz. All of the internal MPC106 timing parameters were set at their minimum possible for the tested

situation. No other transactions are occuring in the system. Parity checking is disabled. The page mode cycle time for 70ns DRAM is 45ns, 40ns for 60ns DRAM. The second set of performance numbers generated for the 60ns DRAMS was achieved by using DRAMS with fast CAS precharge access times resulting in a 5ns reduction in page mode cycle times.

TABLE 1 Processor Reads and Writes from Memory

Memory Type and Speed	Buffer Type	First Burst Write	First Burst Read	Pipelined Read Page Hit	Pipelined Read Page Miss
70ns DRAM	Flow-through	9-5-5-5	9-5-5-5	5-5-5-5	11-5-5-5
70ns DRAM	Trans. Latch	8-4-4-4	9-4-4-4	5-4-4-4	10-4-4-4
60ns DRAM	Trans. Latch	8-4-4-4	8-4-4-4	5-4-4-4	8-4-4-4
60ns DRAM	Flow-through	8-4-4-4	8-4-4-4	4-4-4-4	9-4-4-4
60ns DRAM*	Trans. Latch	7-3-3-3	8-3-3-3	4-3-3-3	8-3-3-3
60ns EDO	Trans. Latch	7-3-3-3	8-3-3-3	5-3-3-3	8-3-3-3
50ns EDO	Trans. Latch	6-2-2-2	7-2-2-2	4-2-2-2	6-2-2-2

* Fast CAS precharge access DRAMs

TABLE 2 Processor Read followed by a Write

Memory Type and Speed	Buffer Type	First Burst Read Access	Pipelined Write - Page Hit	Pipelined Write - Page Miss
70ns DRAM	Flow-through	9-5-5-5	8-5-5-5	11-5-5-5
70ns DRAM	Trans. Latch	9-4-4-4	6-4-4-4	9-4-4-4
60ns DRAM	Trans. Latch	8-4-4-4	6-4-4-4	7-4-4-4
60ns DRAM	Flow-through	8-4-4-4	7-4-4-4	9-4-4-4
60ns DRAM*	Trans. Latch	8-3-3-3	5-3-3-3	7-3-3-3
60ns EDO	Trans. Latch	8-3-3-3	5-3-3-3	6-3-3-3
50ns EDO	Trans. Latch	7-2-2-2	4-2-2-2	5-2-2-2

* Fast CAS precharge access DRAMs

L2 Performance

The L2 performance numbers were generated using the following assumptions: all of the tests were generated with the PowerPC 604 microprocessor using a bus speed of 66 MHz, all of the internal MPC106 timing parameters were set at their minimum possible for the tested situation. 60ns DRAMs with normal CAS accesses times were uesd.

TABLE 3 L2 Access Timings

Transfer Type	Asynch. SRAMs	Burst SRAMs	Pipelined Burst SRAMs
Burst Read Hit	3-2-2-2/2-2-2-2	3-1-1-1/1-1-1-1	4-1-1-1/1-1-1-1
Read Miss w/Castout (*Latched Buffers*)	3-2-2-2/6-4-4-4	3-1-1-1/6-4-4-4	5-1-1-1/6-4-4-4
Read Miss w/Castout (*Flow through Buffers*)	3-2-2-2/5-4-4-4	3-1-1-1/5-4-4-4	5-1-1-1/5-4-4-4
Burst Write Hit	4-2-2-2	3-1-1-1	3-1-1-1
Read/ Write	3-2-2-2/3-2-2-2	3-1-1-1/2-1-1-1	4-1-1-1/2-1-1-1

Performance of Processor Accesses to PCI

The processor to PCI access performance numbers were generated using the following assumptions. All of the tests were generated with the PowerPC 604 microprocessor using a bus speed of 66 MHz. All of the internal MPC106 timing parameters were set at their minimum possible for the tested situation. The MPC106 was parked on the PCI bus. The MPC106 was in 2 to 1 clocking mode, making the PCI bus frequency 33 MHz. For reads the target provided data 2 cycles after FRAME# was asserted, inserted no wait states on TRDY#, and used fast DEVSEL# timing. For writes the target could sink the data with fast DEVSEL# timing and no wait states on TRDY#.

For processor accesses to PCI, burst reads, single beat writes and burst writes are affected by the MPC106 buffering scheme. Single beat reads are not affected. Burst reads have a roughly doubled penalty to the first beat of data because the data is gathered into a cache line size buffer and not forwarded to the processor until all data has been received. The data is not forwarded to prevent a deadlock scenario in which the processor read is retried on the PCI bus and a different PCI master initiates a transaction that needs to be snooped.

The MPC106 has 2 processor to PCI write buffers that are each 16 bytes wide. For single beat writes, the first 2 writes get buffered in 4 cycles each. Subsequent writes by the processor get held off until a buffer becomes available. Burst writes, 32 bytes in length, use both buffers as one continuous buffer. If the buffers are empty the burst write can occur very quickly. Subsequent burst writes from the processor get held off until the write buffers are empty.

Burst reads by the processor from the PCI bus are done at about 98 MBytes/S on the PCI bus. Burst writes by the processor are done at about 108 MBytes/S.

TABLE 4 Processor Accesses to PCI

Processor Access	Performance	Maximum Throughput
Single-beat Reads	13	20 MBytes/S
Burst Reads	27-1-1-1	71 MBytes/S
Single-beat Writes	4-4-5-8-8-8...	33 MBytes/S
Burst Writes	4-1-1-1-23-1-1-1	82 Mbytes/S

Published in Proceedings of IEEE Compcon'96, pp.132-139, IEEE Computer Society Press, Los Alamitos, CA.

PCI Performance

The theoretical maximum for a 32-bit, 33Mhz PCI bus is 132MBytes/S. There are some assumptions made with this maximum. First, the address phase is not accounted for in writes. For reads, neither the address phase nor the turn-around phase is counted. Second, it is assumed that the burst is long enough so that an address phase doesn't have any impact on the transaction. Third, it is assumed that neither the master nor the target inserts wait states on IRDY# or TRDY#, and that four bytes of data are transferred every PCI clock. Finally, this maximum assumes that memory speeds can keep up with delivering data for reads, or that a good buffering scheme is used to mask the memory overhead. This maximum can be calculated by using the equation:

$$A = \frac{B}{C \times D}$$

where; A = Throughput (bytes/second)
B = Number of PCI bytes
C = Number of cycles for the entire transaction
D = Cycle time (seconds), for 33Mhz D ~ 30ns

For example, to transfer 256 bytes of data with no wait states;

$$\left(A = \frac{256}{64 \bullet 30 \times 10^{-9}} \right)$$

A ≈133 MBytes/S

Adding in the overhead of an address phase for every transaction, the equation for writes becomes *Aw* and for reads becomes *Ar* as described below. Using the same 256 bytes to transfer it shows that *Aw* and *Ar* drop accordingly from the optimal 132 MBytes/S. Note that if one uses a 4K byte transfer, which takes a minimum number of 1000 cycles, then it can be shown that the overhead of an address phase is negligible.

$$Aw = \frac{256}{(64 + 1) \bullet 30 \times 10^{-9}}$$

$Aw \approx 131$ MBytes/S

$$Ar = \frac{256}{(64 + 2) \bullet 30 \times 10^{-9}}$$

$Ar \approx 129$ MBytes/S

When acting as a master there are some other points that can limit system performance. They are the delay from the assertion of GNT# to the actual start of the address phase, wait states on the IRDY# signal, and the time to get off the bus due to disconnects (STOP# asserted). Not included in this equation is the arbitration penalty for REQ# to GNT# assertion. Typically arbitration overhead can be masked by changing GNT# signals while the bus is busy. Adding in these factors, the throughput equations become:

$$Aw = (B / [C' \bullet D] \text{ and } Ar = (B / [(C' + 1) \bullet D]$$

where

$$C' = t_{Cmin} + t_{ap} + t_{gd} + t_{iw} + t_{sd} + t_{tad}$$

t_{Cmin} = Minimum data transfer cycles
t_{ap} = Address phase
t_{gd} = Grant delay
t_{iw} = IRDY wait cycles
t_{sd} = STOP delay
t_{tad} = Turn-around delay

When acting as a target, limiting factors are DEVSEL# assertion timing, TRDY# wait states, and the buffering scheme. The buffering scheme determines how much data the target can source/sink before asserting STOP#, and how much overhead the PCI bus sees due to memory speeds. It is easy to change *C'* above to reflect DEVSEL# assertion and TRDY# wait states but the buffering scheme is a little more difficult. The buffering scheme actually plays a role in another wait state category, this being the PCI retry overhead. If the buffers are full then the target must issue a retry before sinking new data from the PCI bus.

t_{dev} = DEVSEL assertion
t_{tw} = TRDY wait cycles
N_r = Number of retries
t_{ro} = Retry overhead

The equation for *C'* then becomes;

$$C' = t_{Cmin} + t_{ap} + t_{gd} + t_{iw} + t_{dev} + t_{tw} + t_{sd} + t_{tad} + (N_r \bullet t_{ro})$$

Lets take an example case. A master wants to transfer 512 bytes to a target. Lets assume the target has medium DEVSEL# assertion (one idle cycle between the assertion of FRAME# and the assertion of DEVSEL#), inserts no wait states on TRDY#, asserts STOP~# at 256 bytes, and has to retry the master one time to finish flushing the buffer. The master has a one cycle delay from GNT# assertion to an address phase, it inserts 1 wait state for every 8 data transfers, and has a 2 cycle delay to negate FRAME# when STOP# is asserted. Assume that the arbiter keeps the master parked on the bus, and that the masters latency timer is programmed to handle this type of transfer. The minimum number of cycles to transfer 512 bytes would be 128. There are 3 address phases to account for, the first to start the transaction, the second after being disconnected at 256 bytes, and the third because the master got retried. There are 2 grant

delay cycles that must be accounted for which correspond to the second and third address phases. There are 14 IRDY# wait states, 3 cycles of DEVSEL# forced delay, 0 TRDY# wait states, 3 cycles of STOP# delay for the disconnect, 3 cycles of STOP# delay for retry, and 0 cycles for the third STOP# assertion due to the fact that the master would already be negating FRAME#. There are 2 turn-around cycles after STOP# is asserted. Note that we took the retry and folded that overhead into the STOP# delay, the turn-around delay, and the DEVSEL# assertion. With these numbers put into the equation it can be shown that this master/target performance is 108MBytes/S.

$$Aw = \frac{512}{(128 + 3 + 2 + 14 + 3 + 0 + 6 + 2) \bullet 30\times10^{-9}}$$

$$Aw \approx 108 \text{ MBytes/S}$$

To calculate an estimated performance number of a system using the MPC106 a system designer would have to know all the information in the example above. When the MPC106 is acting as a master on the PCI bus it has a 1 cycle delay from seeing GNT# asserted to it and the assertion of FRAME#. The MPC106 never inserts wait states on IRDY# and has a 1 cycle delay from the assertion of STOP# to the negation of FRAME# if doing a burst. If STOP# is asserted while FRAME# is already negated there is no penalty. When the MPC106 is acting as a target on the PCI bus it uses the fast DEVSEL# timing, this means that there is no forced delay due to DEVSEL# assertion. The MPC106 has 0 wait states on TRDY# for writes but will disconnect every 32 bytes. For reads the number of wait states depends on memory speed.

TABLE 5 and TABLE 6 represent some estimated numbers for a system using the MPC106 as a target. Since these are best case simulated numbers there are some inherent assumptions made. First, there is no arbitration overhead. Second, there is no other traffic in the system which avoids contention for system memory. Third, the PCI master inserts 0 wait states on IRDY#. Finally, PCI is running at 33Mhz and the processor bus is running at 66Mhz. Some memory timing assumptions are listed here for reference. A 60ns DRAM was simulated with 8-4-4-4 timing and 70ns DRAM simulated with 9-5-5-5 timing, 50ns EDO DRAM with 7-2-2-2 timing and 60ns EDO DRAM with 8-3-3-3 timing. In TABLE 5, Spec_Rd reflects the setting of the configuration bit that enables speculative reads. In TABLE 6, FB2B reflects the setting of the configuration bit that enables fast back to back PCI cycles.

TABLE 5 Read Throughput

Memory	Spec_Rd = 0		Spec_Rd = 1	
	With Disconnects	Without Disconnects	With Disconnects	Without Disconnects
100% L1 Hit	57 MB/S	60 MB/S	N/A	N/A
100% L2 Hit	57 MB/S	60 MB/S	79 MB/S	85 MB/S
70ns DRAM	54 MB/S	57 MB/S	74 MB/S	76 MB/S
60ns DRAM	60 MB/S	63 MB/S	78 MB/S	82 MB/S
60ns EDO	60 MB/S	63 MB/S	78 MB/S	84 MB/S
50ns EDO	64 MB/S	67 MB/S	80 MB/S	85 MB/S

TABLE 6 Write Throughput

Memory	With Disconnects	Without Disconnects	
		FB2B = 0	FB2B = 1
70ns DRAM	85 MB/S	91 MB/S	88 MB/S
60ns DRAM	100 MB/S	96 MB/S	99 MB/S
60ns EDO	100 MB/S	108 MB/S	119 MB/S
50ns EDO	100 MB/S	108 MB/S	119 MB/S

POWERPC PLATFORM SPECIFICATION COMPLIANCE

The MPC106 is the first Motorola bridge chip to be PowerPC Platform specification compliant. That is, it meets the requirements specified in the architecture document for I/O bridges and memory controllers in the PowerPC Platform specification . The PowerPC Platform specification allows systems to be designed and built that can be used with a variety of common Operating Systems (O/S's) because the O/S's can be written to a standard interface. It also allows systems to differentiate themselves, yet still support the application and O/S software written for other compliant systems.

There are several aspects to the PowerPC Platform specification, the most significant ones being the Address Map definition, the Processor and Memory Architecture specification, the I/O bridge specification, the Interrupt Controller requirements, the Run-Time Abstraction Services definitions, the Non-Volatile Memory specification, the I/O device requirements, the Error and Event Notification specification, and the Power Management options. Of these apsects, the MPC106 is affected by four of them, the Address Map, the I/O bridge specification, the Memory Architecture, and in a limited way by the Power Management options and the Error Notification specification.

Address Map

The Address Map of the PowerPC Platform specification is designed to be flexible enough to support systems from portables to servers and has optional features to help support legacy x86 software and emulation of x86 code. The Address Map defines five types of areas which are System Memory, Peripheral Memory, Peripheral I/O, System control, and Undefined or Reserved. The System Memory area refers to memory, typically a form of dynamic random access memory, which is used for temporary storage of programs and data being used by the processor. The Peripheral Memory area refers to the range of addresses that are assigned to the Host Bridge (an I/O bus that is connected to the processor bus) to be used by devices in the memory space of the I/O bus that is generated by the Host Bridge. The Peripheral I/O space is similar to the Peripheral Memory space except that it refers to the address range used by devices in the I/O space of the I/O bus generated by the Host Bridge. The System Control area refers to the range of addresses which contain the system ROM and other system dependent code and data such as the firmware and Run-time Abstraction Services.

The MPC106 implements the System Memory space from address 0x0000_0000 to 0x3FFF_FFFF, or 1 Gbyte in size. This is because the MPC106 supports a maximum of 1 Gbyte of

DRAM or EDO DRAM and the PowerPC Platform specification requires that the first System Memory area must begin at address 0x0 and be contiguous. The Top of System Memory (0x3FFF_FFFF) for the MPC106 could have been programmable as allowed by the PowerPC Platform specification, but in the interest of address decode speed from the processor address bus, it is fixed at 0x7FFF_FFFF, or 2 Gbytes-1. Accesses from the processor that are to an address less than 0x7FFF_FFFF, but greater than the actual amount of real, physical system memory, generate an error that can be reported to the processor.

The Peripheral Memory space of the MPC106 is from 0x8000_0000 to 0xFDFF_FFFF, or 2 Gbytes - 32 MBytes in size. The starting and ending addresses are not configurable because the MPC106 tries to maximize the amount of address space that devices may use without sacrificing address decode performance for accesses to System Memory.

The Peripheral I/O space begins immediately after the Peripheral Memory space and is from address 0xFE00_0000 to 0xFEBF_FFFF, or 12 MBytes in size. The first 8 MBytes of the Peripheral I/O space are programmable to be either contiguous, or discontiguous in 32 byte chunks that are aligned on 4KByte boundaries.

The System Control area ranges from 0xFEC0_0000 to 0xFFFF_FFFF. The range from 0xFEC0_0000 to 0xFEFF_FEFF is used for configuring the MPC106 and other devices in the system except for the processor. The range from 0xFF00_0000 to 0xFFFF_FFFF is interpreted by the MPC106 as ROM space, which in the MPC106 can be either controlled by the MPC106 or treated like a normal PCI transaction depending on the system configuration.

In addition to the requirements for the five major sections of the Address Map, the primary Host Bridge in the system, which the MPC106 is intended to be, has the following optional address ranges, the Compatibility holes and the Initial Memory Alias spaces.

The Compatibility holes are to be used by software that needs compatibility with PC systems. The first Compatibility hole, the I/O-hole, is used for PCI initiated transactions. It is an address range from 0x000A_0000 to 0x000F_FFFF, or 640KByte to 1MByte-1. When enabled, the Host Bridge will not respond to PCI initiated transactions with addresses in the hole address range. The other Compatibility hole, the processor hole, is used for processor initiated transactions. It exists from address range 0x000A_0000 to 0x00BF_FFFF, or 640KByte to 768KByte-1. When enabled, processor initiated transactions to the processor hole address range will not go to System Memory, but instead will be sent to the PCI bus. The MPC106 implements both holes and each is programmable whether it is enabled or not.

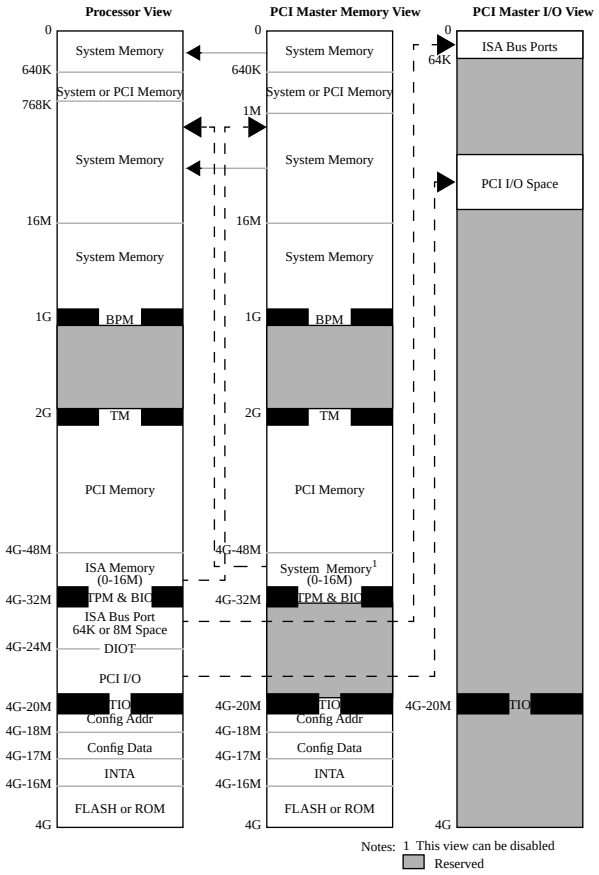

Figure 4 **MPC106 Address Map.**

The Initial Memory Alias spaces are areas in the Peripheral Memory space and allow accesses to the first 16 MBytes of System Memory from a different address range on the PCI bus, and allow accesses to the first 16 MBytes of Peripheral Memory from the processor. The MPC106 implements both of the aliases. The system-memory-alias as implemented on the MPC106, when enabled, allows devices on PCI to access system memory that is at address 0x0000_0000 to 0x00FF_FFFF by using addresses 0xFD00_0000 to 0xFDFF_FFFF. The peripheral-memory-alias on the MPC106 allows the processor to access Peripheral Memory that is located at 0x0000_0000 to 0x00FF_FFFF by using addresses 0xFD00_0000 to 0xFDFF_FFFF. The peripheral-memory-alias on the MPC106 is always enabled. The reason for these additional address ranges is to allow compatibility with devices and software that expect the compatibility holes that are described above. The memory alias spaces provide mechanisms to still access memory that is located in the address holes, but at an aliased address. This enables the use of legacy devices and code in PowerPC Platform specification compliant systems.

Emulation Options

PowerPC Platform specification also defines a set of changes in the system architecture that are intended to improve the performance of emulation of current PC software. These

changes are a Top of Emulated Memory, the I/O-hole must be enabled, System Memory that exists above the Top of Emulated Memory where emulator code can reside that is not accessible from I/O devices, the system-memory-alias must be disabled, and an exception-relocation register to define where exceptions can be relocated to in System Memory.

The Top of Emulated Memory defines an address below which the System Memory matches that of a PC being emulated. The System Memory above the Top of Emulated Memory is accessible by the processor to run the emulation software. That System Memory is not accessible by I/O devices, and transactions to it by I/O devices will not be accepted by the MPC106. The system-memory-alias is disabled to prevent accidental modification of either the emulated memory or the emulator code itself by I/O devices. The io-hole is enabled to allow I/O bus master and third party DMA accesses to legacy devices such as VGA.

The exception-relocation-register is used to map accesses by the processor to addresses 0xFFF0_0000 to 0xFFFF_FFFF to System Memory. This is needed to allow fast access to exception vectors by the processor. Usually, the PowerPC microprocessors are programmed to access the exception vectors at address offset 0x0, but in emulation of a PC, address 0x0 is used as System Memory for applications. This means that the processor needs to be set to access the exception vectors at offset 0xFFF0_0000, but this is a ROM address which is a slow access. The exception-relocation-register function then is to map accesses to the exception vectors to the faster System Memory. The MPC106 implements all of the above features.

I/O Bridge Requirements

The requirements for the primary I/O bridges in the PowerPC Platform specification are mainly restrictions and clarifications over what is specified in the PCI Local Bus Specification. These clarifications deal primarily with data buffering and transaction ordering.

The first buffering requirement is that Host Bridges which include buffers must make sure that these are transparent to software. The MPC106 complies with this by keeping all transactions from the processor to I/O in order. The only exception to this is if store gathering is enabled in the MPC106. If store gathering is enabled, the MPC106 will potentially take a number of sequential writes from the processor to PCI and generate one PCI transaction. The MPC106 also recognizes the eieio and sync instructions from the processors. The MPC106 will flush all of its buffers upon receiving one of these transactions and will not grant the bus to the processor until all buffers are flushed.

The first ordering requirement is that data from a DMA read (I/O initiated) must be allowed to complete prior to a processor Load or Store operation that was queued previously. The PCI interface of the MPC106 handles this requirement by having independent master and target state machines. That is, the data for the DMA read is being handled by the target state machine and the Load or Store operation is being handled by the master state machine. These state machines operate independent of each other, so the DMA read completion is not dependent on any other transaction.

The next buffering/ordering requirement is that during a proceesor Load operation, the Host Bridge must not prevent a subsequent DMA (I/O initiated) write from being posted in the Host Bridge. The MPC106 meets this requirement by having separate buffers for processor reads from PCI and PCI writes to System Memory so there are not resource conflicts.

The third ordering rule is that a DMA read or write from an I/O device to system Memory must not be passed to the processor side of the Host Bridge before the data for a previous I/O DMA write has been flushed to the processor side. The MPC106 handles this requirement by keeping all snoops to the processor due to PCI initiated transactions in order. This does not mean that data is written or read from memory in the same order that its transaction occurred. That is not a requirement because all the MPC106 buffers are compared with incoming transactions to make sure that coherency is kept. In other words, once a transaction from PCI to memory has been snooped to the processor and any copybacks due to the snoop have been completed, then the buffer in the MPC106 is effectively System Memory.

The other I/O bridge requirements are the ability to run in either Big-Endian or Little-Endian mode and the ability to support a PCI Interrupt Acknowledge cycle by providing a 1-byte address that the processor can read to generate an Interrupt Acknowledge on the PCI bus. The MPC106 handles the Endian requirement by keeping System Memory in the same format that the processor keeps its caches. That is, Big-Endian format in Big-Endian mode and in PowerPC Little-Endian format in Little-Endian mode. The conversion to true Little-Endian occurs at the MPC106's PCI interface, and the method of the conversion is dependent on the mode that the MPC106 is operating in. The MPC106 provides the Interrupt Acknowledge address at address 0xFEF0_0000 in the System control area. Also, in order to speed up the decode of the processor initiated read to the Interrupt Acknowledge address, the MPC106 decodes any processor read in the address range from 0xFEF0_0000 to 0xFEFF_FFFF0 as an Interrupt Acknowledge cycle.

Error Notification and Power Management Options

The PowerPC Platform specification specifies how the system is to handle error and event notification and any power management that the system wishes to do. The MPC106 has features designed in to help the system meet these specifications. The MPC106 provides a set of error notification registers that contain information about errors detected by the MPC106. These registers include the address of the transaction that caused the error, the transaction type that caused the error, and the specific type of error that was detected. The MPC106 detects the following errors: PCI, L2 cache, and memory parity errors, accesses to non-existent System Memory, illegal transactions from the processor, ECC errors, and system errors reported by PCI. Once an error is detected, software can read the MPC106's notification registers to get more information about the specific error. The MPC106 also provides a set of power management registers that control the level of activity in the MPC106. By programming the registers in various ways, the MPC106 can be put into power management states that vary

from shutting off the clock internally to the MPC106 to only shutting off internal state machines and waiting for a system transaction that needs servicing to wake up.

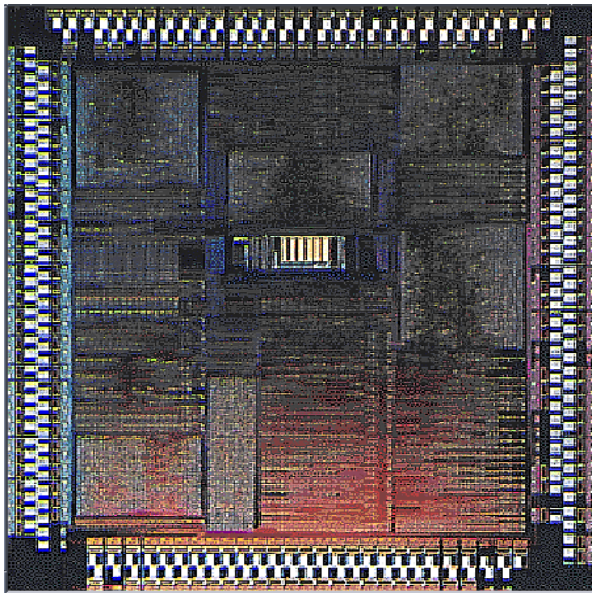

Figure 5 **MPC106 die photo.**

SUMMARY

The MPC106 is the first PowerPC Platform specification compliant Motorola bridge chip. It meets all of the requirements specified in the architecture document of the PowerPC Platform specification for I/O bridges and memory controllers. It also supports some optional features, such as power management, error notification, and emulation performance enhancement. The MPC106 is a high performing chip set. It has optimized processor accesses, a fast L2 cache castout algorithm, and page mode memory accessing. It also greatly utilizes the PCI bandwidth, speeds up read accesses with the speculative read mode, and speeds up all accesses in systems that don't require hardware-enforced coherency by supporting the no-snoop option. The MPC106 is flexible enough to be used in a wide range of systems, from lap tops to low end servers. The performance features will enable the design of high performing PowerPC Platform specification compliant systems that will support a variety of common Operating Systems.

ACKNOWLEDGMENTS

The authors would like to acknowledge the hard work and dedication put forth by the rest of the MPC106 staff: Daphne Bazile-Octave, Mike Carlson, Jen Yeu Chen, Todd Dukes, Tom Elmer, Andrew Harris, C.S. Hui, Quang Le, C.K. Leung, Dina McKinney, Jim Schafer, Raymond Tang, Jim Wenzel. We would also like to thank the other members of Somerset who helped throughout the development process as well as: the packaging team, the tools group, the manufacturing team, and all the other support staff.

REFERENCES

[1] PCI Special Interest Group, "PCI Local Bus Specification Revision 2.1", June 1, 1995.

[2] Morgan Kaufmann Publishers, Inc., "PowerPC Microprocessor Common Hardware Reference Platform: A System Architecture", November, 1995

The authors may be reached at chrisb@ibmoto.com, garcia@ibmoto.com, brianr@ibmoto.com, lauraw@ibmoto.com, gew@ibmoto.com