

ZERO WAIT STATE CACHE USING NON-INTERLEAVED BANKS OF ASYNCHRONOUS
STATIC RANDOM ACCESS MEMORIES.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to the field of computer systems specifically, to second level cache memories which represent the level of the memory hierarchy in a computer system situated between the central processing unit (CPU) cache and the main memory.

2. Prior Art

Historically, the demands of microprocessor technology have been increasing at a faster rate than the support technologies, such as DRAM and TTL/programmable logic. Recent trends are further aggravating this mismatch in the following ways. First, microprocessor clock rates are rapidly approaching, and in some cases exceeding the clock rates of standard support logic. In addition, the clocks per instruction rate is rapidly decreasing, putting a very high bandwidth demand on memory. Newer designs such as RISC architectures, are demanding even more memory bandwidth to accomplish the same amount of work. The memory bandwidth demand has been further aggravated by the need for Direct Memory Access (DMA) by devices such as coprocessor and multi-processors. Finally, the rate at which new devices are being introduced into the market place is accelerating--further exacerbating all the above.

As a result of these trends, performance bottlenecks have emerged that continue to influence the way that systems are designed. Memory bandwidth, as a performance limiter, has already forced the use of cache memories in many microprocessors systems. By way of example, the use of cache memories is common place in the 386TM generation of microprocessors manufactured by Intel Corporation. Also, Intel's Intel486TM and i860TM (also referred to as the

i860XP™) processors include on-chip caches for enhanced performance. It is clear that further changes in the memory hierarchy (primary cache, secondary cache, DRAM architectures, etc.) will be required to sustain performance increases in the future generations. (Intel, 386, Intel486, i860, and i860XP™ are all trademarks of the Intel Corporation.)

A comprehensive description of caching concepts is found in a publication entitled "Cache Tutorial", available from Intel Corporation, Literature Sales, P.O. Box 7641, Mt. Prospect, IL 60056-7641. Known cache memories are designed at two levels-First Level Caches and Second Level Caches. A First Level Cache is a single layer of high speed memory between the microprocessor and main system DRAM memory. First Level Caches hold copies of code and data most frequently requested by the microprocessor. First Level caches are typically small (4 Kbytes to 64 Kbytes) in size. First Level Cache may be either internal or external to the device (e.g. a microprocessor) embodying the processor.

A second level cache is a second layer of high speed memory between the First Level Cache and main system DRAM memory. Second Level Caches also hold copies of code and data frequently requested by the microprocessor. Second Level Caches are typically external to the device embodying the processor. A Second Level Cache handles the more random memory requests that the First Level Cache misses. In order to simplify the handling of requests the First Level Cache misses, the Second Level Cache typically includes all the data of the First Level Cache and more. As a result, a Second Level Cache is almost always larger (64 Kbytes to 512 Kbytes) than a First Level Cache.

To achieve optimal performance, it is desirable that the cache provides zero wait state performance. Zero wait states implies that the processor does not have to wait for data from cache memory. For example, the timing with which the cache memory provides valid data on the data bus does not require the processor to waste cycles waiting for valid data. Known systems provide zero wait state performance by using a special type of Static Random Access Memory (SRAM) called "Burst SRAM" or synchronous SRAM, or by using interleaved banks of SRAM that are alternatively accessed. A negative aspect of using "Burst SRAM" is its cost. "Burst SRAM" is more expensive than commodity asynchronous SRAM.

Interleaved banks of SRAM add control logic and circuitry to a cache controller. Further, use of an interleaving structure results in data bus contention when switching between the banks. This has effects on system timing because in the time it takes the one bank to turn off and another bank to turn on, both banks may be driving data on the same bus line. This reduces the long term reliability of the SRAM components themselves. Further, caches implemented using interleaved SRAM banks cost more than those using non-interleaved banks, although lower in cost than Burst SRAM.

Thus it is, a primary object of the present invention to provide a cache controller with zero wait states performance without the high cost of Burst SRAMs and reduced reliability of an interleaved cache structure. It is further object of the present invention to provide a caching system that utilizes commodity asynchronous SRAM devices, thus reducing the overall system cost.

THIS SHEET HAS BEEN
REPLACED

-4-

SUMMARY

A zero wait state system cache memory for use in computer systems is disclosed. The present invention utilizes banks of non-interleaved asynchronous static random access memories (SRAM) which are each independently controlled. The components used are commodity level asynchronous SRAM chips. The use of a non-interleaving data access scheme eliminates data contention. Further, independent control of the banks of memories allows the cache controller to support a greater number of cache sizes, as well as providing for greater scope of economies since all banks need not be used.

Zero wait state performance is achieved by correlating cache memory control signals relative to the clock and control signals used by the processor. In the currently preferred embodiment of the present invention, correlation is accomplished by 1) clocking cache operations at twice the clock rate of the microprocessor; and 2) correcting control signal skew resulting from the circuit paths for each of the cache control signals. Correction of control signal skew is accomplished by identifying the control signal(s) with maximum delays and then introducing delay circuitry into the circuit path of control and clock signals with lesser delays.

Processor burst access to memory is supported by an internal sequencer which predicts subsequent burst addresses based on an initial address provided by the processor.

This Sheet Has Been Replaced



BRIEF DESCRIPTION OF THE FIGURES

Figure 1 is a block diagram of a computer system in which the currently preferred embodiment of the present invention may be embodied.

Figure 2 is a block diagram of a chipset which may be used to provide a basis for a computer system as illustrated in Figure 1, the currently preferred embodiment of the present invention being a component of such a chipset.

Figure 3 illustrates the signal interfaces for a Cache DRAM Controller (CDC) component of the currently preferred embodiment of the present invention.

Figure 4 is a block diagram illustrating the signal interface of the cache controller with the banks of Static Random Access Memory (SRAM) of the currently preferred embodiment of the present invention.

Figure 5 is a block diagram of the functionality of the cache controller of the currently preferred embodiment of the present invention.

Figure 6 is a timing diagram for a read hit , with zero wait states and in write through or write back mode, in the currently preferred embodiment of the present invention.

Figure 7 is a timing diagram for a write hit, with zero wait states and in write through mode, in the currently preferred embodiment of the present invention.

Figure 8 is a timing diagram for a write hit, with zero wait states and in write back mode, in the currently preferred embodiment of the present invention.

THIS SHEET HAS BEEN
REPLACED

-7-

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A circuit which implements a controller for a cache system is described. In the following description, numerous specific details are set forth, such as specific numbers of bytes, bits, devices, etc., in order to provide a thorough understanding of the preferred embodiment of the present invention. It would be apparent to one skilled in the art that the present invention may be practiced without these specific details. Also, well known circuits have been shown in block diagram form, rather than in detail in order to avoid unnecessarily obscuring the present invention.

In the course of describing the present invention, frequent reference may be made to the use of the invented cache controller in conjunction with certain specific CPU architectures and /or microprocessor types, such as the Intel Intel486DX microprocessor. These implementations merely reflect the currently preferred embodiment of the present invention. It should be understood that the concepts embodied in the present invention are applicable, or may be extended to other processor types and architectures.

In addition, in describing the present invention, reference is made to signal names peculiar to the currently preferred embodiment. A description of each of these signals is provided in the text and in Appendix A. Reference to these specific signal names should not be construed as a limitation on the spirit of scope of the present invention.

This Sheet Has Been Replaced



Overview of a Computer System
Of the Currently Preferred Embodiment

The currently preferred embodiment of the present invention may be embodied for use in a computer system such as the one illustrated in Figure 1. The present invention may be implemented on a general purpose microcomputer, such as one of the members of the IBM compatible Personal Computer family, or one of several work-station or graphics computer devices which are presently commercially available. A computer system as may be utilized by the preferred embodiment generally comprises a bus or other communication means 101 for communicating information, a processor means 102 coupled with said bus 101 for processing information, a random access memory (RAM) or other storage device 103 (commonly referred to as a main memory) coupled with said bus 101 for storing information and instructions for said processor 102, a read only memory (ROM) or other static storage device 104 coupled with said bus 101 for storing static information and instructions for said processor 102, a data storage device 105, such as a magnetic disk and disk drive, coupled with said bus 101 for storing information and instructions, an alphanumeric input device 106 including alphanumeric and other keys coupled to said bus 101 for communicating information and command selections to said processor 102, a cursor control device 107, such as a mouse, track-ball, cursor control keys, etc., coupled to said bus 101 for communicating information and command selections to said processor 102 and for controlling cursor movement. Additionally, it is useful if the system includes a hard copy device 109, such as a printer, for providing permanent copies of information. The hard copy device 109 is coupled with the processor 102 through bus 101.

Note that in the foregoing description, the bus 101 is used in a generic sense to denote a coupling between the various components in the computer system. As will be described in more detail below, the bus 101 may be actually a plurality of bus structures, each of which are interconnected, and each of which is used to support a specific class of devices.

Overview of the System Embodiment

The currently preferred embodiment of the present invention is embodied in a computer system component termed a Cache DRAM Controller (CDC). DRAM is a well known term that refers to memory devices known as a Dynamic Random Access Memory. DRAMs are commonly used as a computer systems main memory. In any event, the CDC is part of a chipset. A chipset refers to a collection of a closely designed integrated circuits which provide the core functionality for a computer system. Use of chipsets significantly reduces the effort involved in designing, manufacturing and testing of a computer system.

The chipset of the currently preferred embodiment is described with reference to Figure 2. The components of a chipset are the CDC 202, a microprocessor 201, a Data Path Unit (DPU) 203 and a System I/O (SIO) unit 204. The chipset of the present invention interconnects three different bus structures. A host bus is internally designed for specific devices, e.g. the processor, co-processor and system memory. A Peripheral Component Interface (PCI) bus is to couple PCI compliant I/O devices to the system. Finally an Industry Standard Architecture (ISA) bus interface is provided to allow connection of an existing class of I/O devices to the system. Generally speaking, the PCI bus will operate faster than the ISA, and provide different functionality. It should further be noted that each of these buses have data, address and control sub-buses.

The DPU 203 provides data bus connections for the microprocessor, local bus and DRAM data buses. The SIO 204 serves as a bridge from the PCI local bus to an ISA bus. Additionally, it integrates an ISA compatible DMA controller, timer/counter and interrupt controller, as well as integrating PCI local bus arbitration logic. The microprocessor 201 of the currently preferred embodiment is a 25 or 33 Mhz i486SX, i487SX, i486DX, i486DX2 microprocessors, and other bus compatible Intel microprocessors or upgrade processors, all of which are available from the Intel Corporation of Santa Clara, California.

In the currently preferred embodiment, the CDC 202 includes a dual ported DRAM controller that interfaces DRAM 206 to the host CPU bus and the PCI bus. The Dram Controller sends addresses 213 and control 214 signals to DRAM 206, as well as DPU control signals 215 to DPU 203. This results in the transfer of data to the DPU 203 and subsequently to processor 201. The CDC 202 supports a two-way interleaved DRAM organization. The CDC 202 also integrates a cache controller that supports 64 Kbytes, 128 Kbytes, 256 Kbytes or 512 Kbytes of a secondary cache using standard asynchronous SRAMs. The CDC's integrated cache controller supports either write through or write-back cache protocols. The CDC provides other functionality to the computer system, including system clock generation and providing system reset logic. As will be described in greater detail below, the system clock generation plays a role in achieving the zero wait state cache performance.

Referring back to Figure 2, a host bus 205 is used to couple the processor 201 to critical system components such as DRAM 206, cache memory arrangement 207 and a co-processor 208. Note that the cache memory

arrangement includes all the TAG, DATA and other storage areas needed by the cache system. A second bus, here a PCI BUS 209, is used to couple a second tier of components. In the PCI architecture components may be either master devices 210 or slave devices 211. A master device is one that can drive and control the PCI bus 209. A slave device is one that responds to requests from a master device on the PCI bus 209, or from another component defined as a master, that may be physically located on another bus. Finally, an ISA bus 212 is used to couple ISA compliant devices.

The present invention is directed towards the cache controller portion of the CDC 202. However, it would be apparent to one skilled in the art that the present invention could be practiced as a discrete component, or packaged as a component with even greater functionality. The present invention may be practiced as either a first level, or second level cache. Moreover, the present invention is directed towards a cache system that is external to the microprocessor. In the currently preferred embodiment, the cache will be a second level cache. This is because the microprocessor supported in the preferred embodiment include an internal first level cache. However, it will become apparent to one skilled in the art to practice the present invention as a first level cache.

Signal Interfaces

Figure 3 illustrates the interface modules and signal interfaces for the CDC component of the currently preferred embodiment.

Referring to Figure 3, a host CPU interface module 301 is used to couple to the host processor. Specific signals are provided to an upgrade processor via

upgrade processor interface module 302. An upgrade processor is one that replaces an original processor. An upgrade processor provides enhanced performance with over the original processor. The cache control Interface 303 is used to interface with the banks of cache memory. DRAM Control Interface 304 is used to interface with the system DRAM. PCI local bus interface 305 provides an interface to the local bus. Reset and clock module 306 provides system clock signals, as well as receiving a 2X clock signal. Finally DPU control interface 307 provides an interface to the DPU unit of the chipset. A complete description of all the signals is provided in Appendix A.

Figure 4 is a block diagram illustrating the coupling of the cache controller interface signals with the various other components. The Cache Write Enable signals (CWE3#, CWE2#, CWE1#, CWE0#) and the chip output enable signal (COE#) are coupled to each of the banks of SRAMs, bank 401 and 402, respectively. A chip select signal (CCS1# and CCS2#, respectively) are coupled to enable each of the banks 401 and 402. The tag portion of each of the banks of SRAMs, here illustrated at 403 and valid bits 404, are coupled to receive the write enable (TWE#) and output enable (TOE#) signals. The write enable signal allows the tag portion to be updated. The output enable signal allows the contents of the tag portion to be read and used. The tag portions 403 and valid bits 404 are also coupled to tag fields of the address provided by the processor (HA[31:2]).

Dirty bits 406 corresponding to lines in the banks of SRAMs are used to signal that a corresponding line in the SRAM has been modified. This is controlled by the signals DWE#, DIRTYD AND DIRTYQ.

Further, cache addresses CA(3:2), CA (18:4) are coupled to the banks 401 and 402 for accessing locations in the cache banks. Finally, the cache latch 405 enable signal (CALE) is coupled to a latch which is used to create addresses on a cache address bus (CA[18:4]).

Cache System of the Currently Preferred Embodiment

In the currently preferred embodiment, a first level caching system is integrated into the preferred '486 microprocessor. The cache organization is a 4-way set associative and each cache line is 16 bytes wide. The first level cache memory is physically split into four 2-Kbytes blocks each containing 128 lines. Associated with each 2-Kbyte block are 128 21-bit tags. The tags are used to identify the contents of a corresponding entry in the cache memory. The writing strategy used is write-through. Further details on the 'Intel486 first level cache may be found in the publication Microprocessors Volume I, available from the Intel Corporation via Intel Literature Sales, P.O. Box 7641, Mt. Prospect, IL 60056-7641.

The second level cache is a direct map, where each cache line is 16 bytes wide. The cache may be 64 Kbytes to 512 Kbytes in size. The write strategy may be either write through or write back. Writing strategies are discussed briefly below.

In the currently preferred embodiment, the writing strategy for the second level cache may be different than for a first level cache. However, an implementation of a first and second level caches using write back strategies is not supported. In the write-through strategy, every hit operation to the cache is

accompanied by a write of the same data to main memory. In a write back strategy, the cache is updated during the write operation, but actual main memory is updated when the line is discarded from the cache.

The cache controller portion of the CDC provides independent control signals to banks of asynchronous Static Random Access Memory (SRAM) devices. SRAM devices are desirable for use in a caching systems due to their high data access speeds. It is desirable to use asynchronous SRAMs due to their lower cost. Further, since the banks are non-interleaved and independently controlled, the same controller may be used to support various cache sizes. In order to achieve zero wait state performance, with non-interleaved banks of SRAM, the cache controller correlates control signals to the banks of the SRAMs with the system clock. Recall that the CDC is used to derive the system clock signal.

Figure 5 is a block diagram of the functionality of the cache controller. As noted above, the CDC provides the timing for the system clock. As noted in Figure 5, a 2X clock signal 501 is provided to CDC 510, where it is converted into a 1X signal by 2X to 1X converter 504. Means for converting a 2X clock signal to a 1X clock signal are known in the art. Any such well known means may be utilized. The signal from the converter 504 is then provided to a delay equalizer element 505. The delay equalizer element 505 introduces prescribed delays to create a 1X clock signal 502 that has zero delays with respect to cache control signals 507. The 1X clock signal is then provided to a microprocessor 503.

The 2X clock signal is used to drive control signals 507 for the cache memory 506. The 2X clock signal is provided to a cache control signal generation

means 508. The cache control signal generation means 508 generates cache control signals. The output of the cache control signal generation means 508 is then provided to a delay equalizer element 509. The delay equalizer element 509 is used to correct for skewing of control signals out of the cache control signal generation means 508.

While the currently preferred embodiment of the present invention utilizes a 2X clock signal to drive the cache control signal generation means 508, it would be apparent to one skilled in the art to use other clock signals of different frequencies. What is key is that the cache control signals have a zero delay with respect to the system clock. This is termed a correlated clock scheme. The manner in which the correlated clock scheme is achieved in the currently preferred embodiment is discussed below.

Correlated Clock Scheme

The correlated clock scheme provides for correction of skewing that may occur with respect to the provision of control signals out of the CDC. This skewing occurs because of inherent delays resulting from the circuit path also of the various control signals going through the CDC chip. In order to improve performance, these skews must be corrected. By correcting for this skew, timing of signals can be more precisely controlled. This precise control, and by clocking the cache control signals at a 2X clock, zero delay in the provision of a system clock signal and the cache controller signal is achieved.

The correlated clock scheme has several premises. These include:

1. The system clock generation is done on the chip.

2. The delay between any input and resulting output on an integrated circuit chip can vary widely as a result of the manufacturing process, operating temperature and voltage.
3. All input signal to output signal paths on the same ship will have the same variations due to the manufacturing process, operating temperature and voltage.
4. Output signals which share common or similar circuits and are derived from a common input signal will have identical delay characteristics and minimal skew with respect to each other. This skew will remain small across process, temperature and voltage variations because intrinsic circuit delays will "track" each other in the given circuits.

Using the premises, the circuit paths of the control signals are modified to eliminate the skew in provision of the control signals. Skew is undesirable since it causes timing delays which require the introduction of wait states. Wait states degrade system performance.

Correlation of the timing signals of the currently preferred embodiment has been achieved through iterations of timing simulations and the introduction of gate delays in particular circuit paths. First, the control signal with the longest delay is determined. When this is determined, the circuit path for other control signals may be extended as needed. This extension may occur by for example, introducing pairs of inverters into the circuit path of the control signals. Such means are illustrated above in Figure 5 as delay equalizer elements 505 and 509.

Address Order Prediction

Address order prediction provides burst mode transfer of cache data. Note that the burst mode of transfer is a feature of the Intel i486 family of microprocessors. Essentially, it provides for fast transfer of data based on a provided address. Note that in known implementations, the device must still read the address of all the data to be accessed from the address bus.

Ideally, the burst mode of operation uses 5 clock cycles, two for obtaining the first data, and each of the next data provided in the next cycle. A problem in supporting bus mode transfer is that an address provided by the CPU may lag the time needed for reading or writing the data. Address Order Prediction eliminates the need to look at every address. A prediction is made based on an observation of data addressing patterns by the i486. What are examined are the second and third low order address bits (i.e. A2 and A3) of the first address. This is because the bits A0 and A1 are ignored by the microprocessors of the currently preferred embodiment (e.g. the Intel486) for cache line addressing purposes. For the particular values of A2 and A3 a corresponding first address pattern is identified. This correspondence is;

00	=	0
01	=	4
10	=	8
11	=	C

Recall that the cache is addressed as lines of double words (i.e. 4 bytes). So the first doubleword is at line address locations 0, the second doubleword at line address locations 4, the third doubleword at line address location 8 and the fourth double word at line address locations C. From the identified first address the corresponding sequence of addresses is as follows:

0	4	8	C
4	0	C	8
8	C	0	4
C	8	4	0

Such sequences occur because a memory address may occur at any point in the cache line. The appropriate data is then accessed according to the appropriate address sequence and provided on each subsequent processor cycle.

Timing Diagrams

What follow are descriptions of timing diagrams for various cache operations. They are not meant to be exhaustive, but merely representative of operations of the code system. Note that in order to fully convey the sequence of operation, signals not generated by the cache controller have been included. Also, all the signals for the cache controller are illustrated but may not have use for a particular operation. Again this is done for the sake of completeness.

Figure 6 is a timing diagram for a level two Cache Hit Read at Zero wait states. Level two Cache Hit Reads are handled the same way whether the level two cache is in Write Thorough or Write Through or Write Back mode. The cycle is initiated by ADS# 601 and terminated by BRDY# 602. The ADS# 601 signal is provided by the processor, whereas the BRDY# 602 signal is generated by the CPU interface portion of the CDC (as illustrated in Figure 2). Note that the BRDY# 602 signal is used for burst mode accesses. Tag comparison is completed before the falling edge of the T2 clock cycle 603 to determine whether it is a cache Hit or Miss. If it is a Hit, BRDY# 602 is generated in the middle of the T2 clock cycle, on the falling edge of CLK1 604 and is sampled by the CPU at the end of the T2 clock cycle 603. The subsequent DWORDS in the Burst read are sampled

at the end of consecutive clock cycles to complete the Burst read in 2-1-1-1 clock cycles. During level two Cache Hit Reads the Tag, Valid and Dirty bits corresponding to the DRAMS in cache memory being read, remain unchanged. The CALE signal 605 is used to latch the CPU address into external transparent latches. The lower address bits CA(3:2) 606 are driven directly by the CDC utilizing the Address Order Prediction described above.

Figure 7 is a timing diagram of a level two cache hit Write at Zero wait states in the level two Write Through mode. The cycle is initiated by ADS# 701 and completes in zero wait states due to write posting at the DPU (from where the data will be subsequently written to DRAM memory) RDY# signal 704 is asserted back to the CPU in the T2 clock cycle 702. As in the case of the Hit Read, the Tag comparison is completed before the falling edge of the T2 clock cycle 701 and is used to determine whether the CWE[3:0]# signals 703 need to be activated in the T2 clock cycle 702. If it is a Hit, CWE[3:0] # are asserted during the second half of the T2 clock cycle and the level two cache line is updated at the same time that it is posted in the DPU. The CDC then initiates a DRAM write cycle which is completed a few clock cycles later using the posted address in the CDC and the posted data in the DPU (timing of the DRAM write cycle is not illustrated).

Figure 8 is a timing diagram of a Level two cache Hit Write at Zero wait states in the level two Write Back mode. The cycle is very similar to the level two Write Through case, except that the CDC does not initiate a DRAM Write cycle immediately. The new data is written into the level two cache line and the Dirty bit corresponding to that line is marked "Dirty" to indicate that the data is different from the data in DRAM. The DIRTYD 801 and DWE# 802 signals are used for this purpose. The DWE# signal 802 has the same timing as the

CWE[3:0]# 803 signals in that it cannot be asserted before the Cache Hit / Miss information is valid and that it also has to be deserialized before the end of the zero wait state cycle. No data is posted in the DPU for this type of cycle.

•

As becomes clear from the above timing diagrams, the cache control signals must be precisely aligned with the system CLK1. Although not illustrated, the cache control signals are triggered according to the 2X clock signal provided to the CDC. Given this clocking along with the close signal correlation described above, zero wait state performance can be achieved.

Thus, a zero wait state system memory cache, utilizing non-interleaved banks of SRAM has been disclosed.

APPENDIX A CACHE DRAM CONTROLLER SIGNALS

Host CPU Interface Signals

Pin/ Signal Name	Type	Description
CPURST	Output	The <i>CPU reset</i> signal forces the CPU to begin execution at a known state. The CPURST signal is asserted whenever the PWROK signal is low, or the CPURST is enabled through the TRC register. When the CDC is programmed for a CPU type other than an upgrade processor, this synchronous signal is also asserted whenever the SRESET# signal is active or a shutdown cycle is detected. The CDC insures at least 1 use wide CPURST pulse width, however 1 msec power-on reset pulse width requirement of CPU must be satisfied by the PWROK signal inactive time.
A[31:2]	Input/ Output	A[31:2] are the <i>address lines</i> of the CPU. A[31:2], together with the byte enables BE[3:0]#, define the physical area of memory or input/ output space accessed. A[31:4] are used to drive addresses into the CPU and secondary cache to perform cache snoop (inquire) cycles. A[3:2] are not used by the CPU during snoop cycles, however they are driven low by the CDC- instead of allowing these signals to float.
BE[3:0]#	Input	The <i>byte enable</i> signals indicate active bytes during read and write cycles. BE3# applies to D[31:24], BE2# applies to D[23:16], BE1# applies to D[15:8] and BE0# applies to D[7:0].
HLOCK#	Input	The <i>bus lock</i> signal indicates that the current CPU bus cycle is locked. HLOCK#, driven by the CPU, normally goes active in the first clock of the first locked bus cycle and goes inactive on the last clock of the last locked cycle. HLOCK# goes active for read-modify-write operations. A locked operation is a combination of one or more READ cycles followed by one or more WRITE cycles. For the upgrade processor before a locked cycle is generated, the upgrade processor checks if the line is in M state. If it is the upgrade processor does a write back and then runs the locked cycle on the external bus.

PLOCK#	Input	The <i>pseudo-lock</i> signal indicates that the current CPU bus transaction requires more than one bus cycle to complete. Examples of such operations are floating point long reads and writes (64 bits), segment table descriptor reads (64 bits), and upgrade processor cache line read and write-back cycles (256 bits). The PLOCK# is driven active until the addresses for the last bus cycle of the transaction is driven.																																				
M/IO# D/C# W/R#	Input Input Input	The memory /<i>input-output</i>, <i>data/control</i> and <i>write/read</i> lines are the primary bus cycle definition signals. The following table describes the bus cycles. <table><tr><td><u>M/IO</u></td><td><u>D/C#</u></td><td><u>W.R#</u></td><td><u>Bus Cycle Initiated</u></td></tr><tr><td>Low</td><td>Low</td><td>Low</td><td>Interrupt Acknowledge</td></tr><tr><td>Low</td><td>Low</td><td>High</td><td>Halt / Special Cycle</td></tr><tr><td>Low</td><td>High</td><td>Low</td><td>I/O Read</td></tr><tr><td>Low</td><td>High</td><td>High</td><td>I/O Write</td></tr><tr><td>High</td><td>Low</td><td>Low</td><td>Code Read</td></tr><tr><td>High</td><td>Low</td><td>High</td><td>Reserved</td></tr><tr><td>High</td><td>High</td><td>Low</td><td>Memory Read</td></tr><tr><td>High</td><td>High</td><td>High</td><td>Memory Write</td></tr></table>	<u>M/IO</u>	<u>D/C#</u>	<u>W.R#</u>	<u>Bus Cycle Initiated</u>	Low	Low	Low	Interrupt Acknowledge	Low	Low	High	Halt / Special Cycle	Low	High	Low	I/O Read	Low	High	High	I/O Write	High	Low	Low	Code Read	High	Low	High	Reserved	High	High	Low	Memory Read	High	High	High	Memory Write
<u>M/IO</u>	<u>D/C#</u>	<u>W.R#</u>	<u>Bus Cycle Initiated</u>																																			
Low	Low	Low	Interrupt Acknowledge																																			
Low	Low	High	Halt / Special Cycle																																			
Low	High	Low	I/O Read																																			
Low	High	High	I/O Write																																			
High	Low	Low	Code Read																																			
High	Low	High	Reserved																																			
High	High	Low	Memory Read																																			
High	High	High	Memory Write																																			
ADS#	Input	The <i>address input</i> indicates that a valid bus cycle definition, byte enables and addresses are available on their corresponding pins.																																				
RDY#	Output	The <i>non-burst ready</i> output indicates that the current bus cycle is complete. RDY# indicates that the system has presented valid data to the CPU in response to a read, or that the system has accepted data from the CPU in response to a write.																																				
BRDY#	Output	The <i>burst ready</i> output performs the same function during a burst cycle that RDY# performs during a non-burst cycle. BRDY# indicates that the system has presented valid data in response to a read, or that the system has accepted data in response to a write.																																				

BOFF#	Output	The backoff output signal forces the CPU to float its bus in to the next clock. the CPU remains in bus hold state until BOFF# is negated. BOFF# is used to abort a CPU cycle when the system resource being accessed is not readily available and there is a possibility of a dead-lock. If a bus cycle was in progress when BOFF# was asserted the cycle will be restarted by the CPU from where it left off. For the upgrade processor, if a dirty line write-back due to an external snoop is pending within the processor, it will be sent out after BOFF# is negated, but before the interrupted cycle is started.
AHOLD	Output	The <i>address hold</i> output signal forces the CPU to float its address bus in the next clock. The CDC asserts this signal in preparation to perform cache invalidation (i486SX, i487SX, i486DX, i486DX2) or cache inquiry (upgrade processor) cycles. The CDC always drives the address on the host bus starting from the third clock on which AHOLD is asserted and continues to drive the addresses until AHOLD is negated.
EADS#	Output	This signal indicates that a valid <i>external address</i> has been driven onto the CPU address lines. This address will be used to perform an internal cache invalidate or cache inquire cycle.
KEN#	Output	The <i>cache enable</i> output signal is used to indicate whether the current cycle is cacheable in the CPU internal (primary or first level) cache.
PWT	Input	The <i>page write-through</i> input signal indicates the current memory cycle can be cached in the secondary cache -- if enabled -- in the write-through mode.
PCD/	Input	The <i>page cache disable</i> input pin when active indicates the current cycle can not be cached in the secondary cache during a cache line fill operation. When PCD is asserted the lien will not be cached in L1 or L2.
HITM# /DIRTYQ	Input	The hit <i>modified cache</i> input pin indicates that a hit to a modified data cache has occurred. HITM# is sampled two clocks after EADS# is asserted.

PCD/ CACHE#	Input	The cache pin is active along with the first ADS# until the first transfer. On line fills the functionality of the CACHE# signal is identical to that of PCD signal. CACHE# will be asserted for each half of cache line fill. During write-back cycles. CACHE# will be asserted only for the first half of the line fill. The beginning of a write-back cycle uniquely identified by ADS#, W/R# and CACHE# together. Beginning of the snoop write-back is identified by the ADS#, W/R#, CACHE# and HITM# being active together.
INV /DWE	Output	The <i>invalidate</i> output signal specifies which final state (invalid or shared) a cache line will transition to in the event of a cache hit during an inquire cycle to the CPU.
INIT	Output	The initialization signal forces the CPU to begin execution at a known state. The processor's internal state is the same as the state after RESET, except that the internal caches, machine check registers and floating point registers retain whatever value they had prior to INIT. INIT is asserted by the CDC whenever SRESET# is sampled active a shutdown cycle is detected or a warm reset sequence is initiated through the TRC register. The CDC ensures a minimum pulse width of 4 clocks for the INIT signal.

Level 2 Cache Control Interface

Pin/ Signal Name	Type	Description										
LCA[3:2] HCA[3:2]	Output	The cache address [3:2] signals generate the burst sequence required by the CPU during secondary cache accesses. The CDC latches the starting burst address and internally generates subsequent dword addresses for the entire cache line. These signals are correlated to the clock. A separate set of CA[3:2] signals is provided for the lower bank and higher bank so as to keep the captive loading constant on LCA[3:2] and HCA[3:2].										
CCS1#	Output	The lower bank cache chip select output signals indicate that the lower cache data bank is selected for the current cache operation. <table><tr><th><u>Cache Size</u></th><th><u>Address range for LCCS# active</u></th></tr><tr><td>64k</td><td>0-32k</td></tr><tr><td>128k</td><td>0-128k</td></tr><tr><td>256k</td><td>0-128k</td></tr><tr><td>512k</td><td>0-512k</td></tr></table>	<u>Cache Size</u>	<u>Address range for LCCS# active</u>	64k	0-32k	128k	0-128k	256k	0-128k	512k	0-512k
<u>Cache Size</u>	<u>Address range for LCCS# active</u>											
64k	0-32k											
128k	0-128k											
256k	0-128k											
512k	0-512k											

CCS2#	Output	The <i>Higher bank cache chip</i> select output signals indicate which cache data bank is selected for the current cache operation. Selected secondary cache size defines the address range in which these signals are driven active. <table><tr><th><u>Cache Size</u></th><th><u>Address Range for LCCS# active</u></th></tr><tr><td>64k</td><td>32k-64k</td></tr><tr><td>128k</td><td>not active</td></tr><tr><td>256k</td><td>128k-256k</td></tr><tr><td>512k</td><td>not active</td></tr></table>	<u>Cache Size</u>	<u>Address Range for LCCS# active</u>	64k	32k-64k	128k	not active	256k	128k-256k	512k	not active
<u>Cache Size</u>	<u>Address Range for LCCS# active</u>											
64k	32k-64k											
128k	not active											
256k	128k-256k											
512k	not active											
CWE[3:0]#	Output	The <i>cache write enable</i> output signals provide byte wide write capability to the secondary cache during cache line fills or cache write hits.										
COE#	Output	The <i>cache output enable</i> output signal is used to perform read cycles from the cache data SRAMs. This signal is connected to the output enable pins of the cache data SRAMs.										
CALE	Output	The <i>cache address latch enable</i> output signal provides the proper control timing to the latches that create the cache address bus CA[18:4] from the host CPU address bus A[18:4].										
TWE#	Output	The <i>tag write enable</i> signal is connected to the tag SRAM write enable (WE#) pin. TWE# is active during CPU read-miss cycles when the cache is updated.										
TOE#	Output	The <i>tag output enable</i> signal controls the output enable pin of the tag SRAMs. When active, tag address and valid bits are driven into the CDC. This signal is normally active, and driven inactive only during tag update.										
VALID	Input/ Output	The <i>valid</i> signal indicates the validity of data in the cache data SRAMs on a line by line basis. VALID is used along with the tag address to make the cache hit/miss decision by the CDC. If sampled low during a CPU memory read cycle, data is not valid in the cache. During an invalidate cycle, the VALID signal is forced low indicating data is not valid. When Valid is sampled 0 during the rising edge of PWROK then the CDC enters the test mod defined by TA[7:4] lines. When valid is sampled high during the rising edge of PWROK the CDC enters the normal mode of operation.										
DIRTYD	Output	The <i>dirty-in</i> output signal indicates whether the data in the secondary cache is being marked as modified. This signal is connected to the data input of the dirty SRAM.										

HITM# /DIRTYQ	Output	The <i>dirty-out</i> signal indicates whether the data in the secondary cache was marked as modified. This signal is connected to the data output of the dirty SRAM.																
INV /DWE#	Output	The dirty bit write enable output signal goes active when the CPU does a WRITE cycle to the secondary write-back cache. This signal is connected to the WE# input of the dirty SRAM.																
TA[7:0]	Input/ Output	The tag address signals are directly connected to the tag SRAM data bus. The following table defines the relation between tag address and CPU address as a function of the secondary cache size. This mapping is performed by the CDC based on the cache size configuration. <table><tr><td><u>Cache Size</u></td><td><u>TA[7:0] corresponds to...</u></td></tr><tr><td>64 KBytes</td><td>A[23:16]</td></tr><tr><td>128 KBytes</td><td>A[24:17]</td></tr><tr><td>256 KBytes</td><td>A[25:18]</td></tr><tr><td>512 KBytes</td><td>A[26:19]</td></tr></table> These pins are also used as inputs to define strapping options and test modes of the CDC. TA[7:4] signal levels are sampled at the rising edge of the PWROK signal to define the test mode. Refer to section 5 "Testability Considerations" for a description of the test modes supported by the CDC. Similarly, TA[2:0] signal levels sampled at the rising edge of the PWROK signal to define the strapping options described below: <table><tr><td><u>Signal</u></td><td><u>Strapping options</u></td></tr><tr><td>TA[1:0]</td><td>Host CPU speed, HCS register bits [1:0].</td></tr><tr><td>TA2</td><td>Cache read/write wait state, SCC bit [2].</td></tr></table>	<u>Cache Size</u>	<u>TA[7:0] corresponds to...</u>	64 KBytes	A[23:16]	128 KBytes	A[24:17]	256 KBytes	A[25:18]	512 KBytes	A[26:19]	<u>Signal</u>	<u>Strapping options</u>	TA[1:0]	Host CPU speed, HCS register bits [1:0].	TA2	Cache read/write wait state, SCC bit [2].
<u>Cache Size</u>	<u>TA[7:0] corresponds to...</u>																	
64 KBytes	A[23:16]																	
128 KBytes	A[24:17]																	
256 KBytes	A[25:18]																	
512 KBytes	A[26:19]																	
<u>Signal</u>	<u>Strapping options</u>																	
TA[1:0]	Host CPU speed, HCS register bits [1:0].																	
TA2	Cache read/write wait state, SCC bit [2].																	

DRAM Control Interface

Pin/ Signal Name	Type	Description
EMAO, OMAO, MA[10:1]	Output Output Output	The <i>multiplexed DRAM address</i> bus provides row and column address information to the DRAMs. External buffering is required. Two signals provide the least significant multiplexed DRAM address lines, one for even dword (EMAO) and one for the odd dword (OMAO).
RAS[3:0]#	Output	Each <i>Row address strobe</i> output signal corresponds to one DRAM row of eight bytes. These signals are used to latch the row addresses on the EMAO, OMAO and MA[10:1] bus into the DRAMs. These signals drive the DRAMs directly, without any external buffers.
CAS[7:0]#	Output	<i>Column address strobe</i> output signals are used to latch the column addresses on the EMAO, OMAO and MA[10:1] bus into the DRAMs. CAS [7:0]# correspond to byte lanes 7:0 of the eight bytes wide, two way interleaved, DRAM array. These signals drive the DRAMs directly, without any external buffers.
WE#/ MR/W#	Output	The <i>write enable</i> output signal is externally buffered to drive the write enable (WE#) inputs of DRAMs.
PCIRST	Output	<i>PCI Bus Reset</i> forces the PCI interfaces of each device to a known state. The CDC tri-state all of its bi-directional and open collector PCI signals, and drives REQ# and MEMACK# signals to their inactive states. PCIRST is driven during power up and when hard reset sequence is initiated through the TRC register.
AD[31:0]	Input/ Output	<i>PCI address</i> signals. When the CDC acts as a master on the PCI bus, the AD[31:0] signals are in the output mode, during the first PCI clock period of a local bus cycle AD[31:0] contain physical address. During subsequent clocks the AD [31:0] lines are floated by the CDC. When the CDC acts as a target on the PCI bus the AD[31:0] signals contain the Address during the first clock and data subsequent clocks, but the data path for PCI initiated cycles to the CDC is not through the CDC.

C/BE[3:0]#	Input/ Output	PCI bus command and byte enable signals which are multiplexed on the same pins. During the address phase of a transaction, C/BE[3:0]# define the bus command. During the data phase C/BE[3:0]# signals are used as byte enables. The byte enables determine which byte lanes carry meaningful data. PCI local bus command encoding and types are listed below. <table><tr><th><u>C/BE[3:0]#</u></th><th><u>Command Type</u></th></tr><tr><td>0000</td><td>Interrupt Acknowledge</td></tr><tr><td>0001</td><td>Special Cycle</td></tr><tr><td>0010</td><td>I/O Read</td></tr><tr><td>0011</td><td>I/O Write</td></tr><tr><td>0100</td><td>Reserved</td></tr><tr><td>0101</td><td>Reserved</td></tr><tr><td>0110</td><td>Memory Read</td></tr><tr><td>0111</td><td>Memory Write</td></tr><tr><td>1000</td><td>Reserved</td></tr><tr><td>1001</td><td>Reserved</td></tr><tr><td>1010</td><td>Configuration</td></tr><tr><td>1011</td><td>Configuration Write</td></tr><tr><td>1100</td><td>Reserved</td></tr><tr><td>1101</td><td>Memory Write and Invalidate</td></tr><tr><td>1110</td><td>Memory Read Long</td></tr><tr><td>1111</td><td>Postable Memory Write</td></tr></table>	<u>C/BE[3:0]#</u>	<u>Command Type</u>	0000	Interrupt Acknowledge	0001	Special Cycle	0010	I/O Read	0011	I/O Write	0100	Reserved	0101	Reserved	0110	Memory Read	0111	Memory Write	1000	Reserved	1001	Reserved	1010	Configuration	1011	Configuration Write	1100	Reserved	1101	Memory Write and Invalidate	1110	Memory Read Long	1111	Postable Memory Write
<u>C/BE[3:0]#</u>	<u>Command Type</u>																																			
0000	Interrupt Acknowledge																																			
0001	Special Cycle																																			
0010	I/O Read																																			
0011	I/O Write																																			
0100	Reserved																																			
0101	Reserved																																			
0110	Memory Read																																			
0111	Memory Write																																			
1000	Reserved																																			
1001	Reserved																																			
1010	Configuration																																			
1011	Configuration Write																																			
1100	Reserved																																			
1101	Memory Write and Invalidate																																			
1110	Memory Read Long																																			
1111	Postable Memory Write																																			
FRAME#	Input Output	Cycle frame is an output when the CDC acts as a master on the PCI bus. FRAME# is driven by the CDC to indicate the beginning and duration of an access. FRAME# is asserted to indicate a bus transaction is beginning. While FRAME# is asserted data transfers continue. When FRAME# is deasserted the transaction is in the final data phase. FRAME# is an input when the CDC acts as a PCI slave. When the CDC acts as a target on the PCI bus it latches the C/BE[3:0] and the AD[31:0] signals on the clock edge on which it samples FRAME# active.																																		
TRDY#	Input Output	Target ready is an input when CDC acts a master on the PCI bus. Assertion of TRDY# indicates the target agent's ability to complete the current data phase of the transaction. For read cycles TRDY# indicates that the target has driven valid read data onto the PCI local bus. For a write cycle TRDY# indicates that the target is prepared to accept write data form the PCI bus. TRDY# is an output when the CDC acts as a PCI slave.																																		

IRDY#	Input Output	<i>Initiator ready</i> is an output when CDC acts as an PCI MASTER. The assertion of IRDY# indicates the current PCI bus master's ability to complete the current data phase of the transaction. For read cycles IRDY# indicates that the PCI master is prepared to accept the read data on the following rising edge of the PCI clock. For a write cycle IRDY# indicates that the master has driven valid write data on the PCI bus. IRDY# is an input when the CDC acts as a PCI slave.
LOCK#	Input Output	<i>Lock</i> indicates and exclusive bus operation and may require multiple transactions to complete. When LOCK# is asserted, non-exclusive transitions may proceed. A grant to start a transaction on the PCI local bus does not guarantee control of the LOCK# signal. Control of the LOCK# is obtained under its own protocol in conjunction with the GNT# signal. It is possible for different agents to use the PCI local bus, while a single master retains ownership of the LOCK# signal.
STOP#	Input Output	<i>Stop</i> indicates that the current bus master must immediately terminate its current local bus cycle at the next clock edge and release control of the local bus. STOP# is used to disconnect, retry and abort sequences on the PCI bus.
REQ#	Output	<i>Request</i> indicates to the PCI local bus arbiter that the CDC desires use of the PCI bus.
GNT#	Input	<i>Grant</i> indicates that access to the PCI local bus has been granted to the CDC.
PAR	Input Output	<i>Parity</i> is driven by the CDC when it acts as a PCI master during address and data phases for a write cycles, and during the address phase for a read cycle. When the CDC acts as PCI slave parity is driven by the CDC for the data phase for a PCI read cycle. Parity is even parity across AS[31:0] and C/BE[3:0]. The CDC does not do any parity checking.
SERR#	Open Collector output pin	<i>System Error</i> signal, when driven by the CDC, indicates that a either a parity error on DRAM or cache has occurred or CDC received a target abort. The CDC receives the parity error indication fro the DPU (DPUPE#), qualifies it with the cycle type and if there is a parity error it is signaled through the SERR# signal. SERR# is also pulsed if the CDC Receives target abort on a CDC initiated PCI cycle

DEVSEL#	Input Output	<i>Device select</i> , when active, indicates that a PCI slave device has decoded its address as the target of the current access. The CDC drives DEVSEL# based on the DRAM address range being accessed by a PCI master. As an input it indicates whether any device on the bus has been selected.
FLSHREQ#	Input	<i>Flush Request</i> signal instructs the CDC to flush its CPU to PCI posted write buffers and disable further posting to these buffers as long as the FLSHREQ# remains active. The CDC acknowledges completion of the write buffer flush operation by asserting the MEMACK# signal. This signal is driven by the expansion bus bridge, and used to avoid dead-lock conditions on the PCI bus.
MEMREQ#	Input	<i>Memory request</i> signal instructs the CDC to flush its host to DRAM and host to PCI posted write buffers and keep the host CPU under address hold (AHOLD) as long as the MEMREQ# signal remains active. The CDC acknowledges completion of the flush operations by asserting the MEMACK# signal. This signal is driven by the expansion bus bridge, and used to provide minimum access latency during ISA master to DRAM cycles.
MEMACK#	Output	<i>Memory acknowledge</i> signal indicates completion of the operations requested by an active FLSHREQ# and/or MEMREQ# signals.

DPU Control Interface

Pin/ Signal Name	Type	Description
DPUPE#	Input	<i>DPU Parity Error</i> signal indicates that a DRAM or cache data parity error has occurred during a data read operation. DPUPE# is active for one clock period from the clock edge at which read data is sampled by the CPU or PCI master. A parity error on any one of the four bytes will be indicated as an active DPUPE#.
PCIDP	Input	<i>PCI Data Parity</i> signal is an input to the CDC. During PCI read cycles from the memory or a CPU write to PCI, the DPU generates an even parity for the PCI data forwards this information to the CDC through the PCIDP pin. The CDC combines this data parity information with the BE[3:]# parity information and generates the PCI parity -- the PAR signal.

HM/P#	Output	<i>Host Bus to Memory/PCI</i> signal is the data source / destination select for the host interface of the DPU. HM/P# when low, indicates that data is to be read or written to the PCI bus. When this signal is high, the data will be read or written to memory. This signal, with the HSTB# and HW/R# signals also control the latching to data in the host post buffers and selecting data from the memory or PCI read buffers.
HW/R#	Output	<i>Host Bus Write/Read</i> signal is used to distinguish between host CPU write and read cycles. HW/R# when low, indicates a host CPU and read cycle. When this signal is high, it indicates a host write cycle. This signal along with the HM/P# and HSTB# signals, also control the latching of data in the host post buffers and selecting data from the memory PCI read buffers.
HSTB#	Output	<i>Host Bus Data Strobe</i> signal is used to control the latching of data in the host post buffers during write cycles, and selecting data from the memory or PCI read buffers during read cycles. When this signal is asserted during a write cycle, the appropriate post buffer latch will be enabled on the CLK1 rising edge if selected by HM/P#. During a read cycle, HSTB# asserted at the rising edge of CLK1 selects one of the two read latches.
PM/H#	Output	<i>PCI Bus to Memory/Host</i> signal is the data source / destination select for the PCI interface of the DPU. PM/H#, when low, indicates that data is to be read or written by the host bus. When this signal is high, the data will be read or written to memory. This signal along with the PSTB# and PW/R# signals, also controls the latching of data in the PCI post buffers and selecting data from the memory read buffers or host post buffers.
PW/R#	Output	<i>PCI Bus Write/Read</i> signal is used to distinguish between PCI write and read cycles. PW/R#, when low, it indicates a PCI read cycle. When this signal is high, it indicates a PCI write cycle. This signal along with the PM/H# and PSTB# signals, also controls the latching of data in the PCI post buffers and selecting data from the memory read buffers or host post buffers.

PSTB#	Output	<i>PCI Bus Data Strobe</i> signal is used to control the latching of data in the PCI post buffers during the write cycles, and selecting data from the memory read buffers or host post buffers during read cycles. When this signal is active during a write cycle, the appropriate post buffers latch will be enabled on the rising clock edge. During a read cycle, PSTB# active will cause the appropriate read data to be selected on the rising clock edge. When PSTB# is deasserted, no change is made in the latch control or read data selected.
MH/P#	Output	<i>Memory bus to Host/PCI</i> signal is the data source/destination select for memory interface of the DPU. MH/P#, when low, indicates that data is to read or written by the PCI bus. When this signal is high, the data will be read or written by the host bus. During memory read cycles, this signal along with the MSTB# signal controls the latching of data in the memory read buffer. During memory write cycles, the MH/P#, MR/W#, and MSTB# signals select either host CPU or PCI post buffers.
WE#/MR/W#	Output	<i>Memory Bus Read/Write</i> signal is used to distinguish between memory read and write cycles. MR/W#, when low, indicates a memory write cycle. When this signal is high, it indicates a memory read cycle. This signal along with the MSTB# signal, also controls the latching of data in the memory read buffers and selecting data from the host or PCI post buffers.
MSTB#	Output	<i>Memory Bus Data Strobe</i> signal is used to control the latching of data in the memory read buffers during read cycles, and selecting data from the host or PCI post buffers during write cycles. When this signal is active during a memory read cycle, the appropriate read buffer latch will be enabled. During a memory write cycle, the appropriate post buffer data will be selected as determined by the MH/P# signal and driven onto the MD bus. When MSTB# is deasserted, no change is made in the read buffer control or post buffer data select.

Reset and Clock Signals

Pin/ Signal Name	Type	Description
PWROK	Input	The <i>power good</i> input forces all internal register and state machines to their default state. This input is asynchronous, but must meet set up and hold specifications for recognition's in any specification clock.
SRESET#	Input	The <i>soft reset</i> input forces all the CDC's internal state machines to their default states; however the contents of the configuration registers remain unchanged. This input is asynchronous, but must meet setup and hold specifications for recognition in any specific clock.
CLK2	Input	The <i>2x Clock</i> input is internally divided by two to generate CPU, DPU and PCI device clocks and driven out on six clock output signals (CLK1A - CLK1F). A CMOS level clock signal running at twice the operating speed of the CDC must be provided on this signal.
CLK1	Input	The <i>1x Clock</i> input provides the fundamental timing and the internal operating frequency for the CDC. This signal is typically connected as a feedback from one of six clock output pins of the CDC (CLK1A - CLK1F). The CDC operates at 25 or 33 MHz clock frequencies. Most of the external timing parameters are specified with respect to the rising or falling edge of CLK1. Maximum skew between the CDC and CPU clocks is ± 1 nsec. Maximum clock skew between the CDC and the PCI agents in ± 2 nsec.
CLK1A CLK1F	Output	These six <i>1x Clock</i> output signals provide the clock reference for the CPU, DPU, and PCI devices of the system. The CDC divides the clock provided on CLK2 by two generate these signals. The CLK1 input signal must be connected to one of these signals -- preferably the one used for the CPU clock.



CLAIMS

1. A computer system comprising:
 - a) a processor unit having an internal first level cache;
 - b) system random access memory;
 - c) one or more data input devices;
 - d) one or more data output devices;
 - e) system clock generation means;
 - f) a second level cache comprising:
 - one or more asynchronous cache memory banks;
 - a cache controller comprising:
 - independent control means for each of said one or more asynchronous cache memory banks; and
 - means for providing control signals to each of said one or more asynchronous cache memory banks so that they are correlated to said system clock signal.
2. The computer system as recited in Claim 1 wherein said cache controller is further comprised of means for predicting subsequent read address from a provided first read address.
3. A caching system comprising:
 - a) one or more banks of non-interleaved cache memory means;
 - b) system clock generation means for generating a system clock signal;
 - c) means for correlating control signals to said one or more banks of cache memory means with said system clock signal; and





d) means for providing burst data from said one or more banks of non-interleaved cache memory means.

4. The caching system as recited in Claim 3 wherein said one or more banks of non-interleaved cache memory means is comprised of asynchronous Static Random Access Memory devices.

5. The caching system as recited in Claim 4 wherein said system clock generation means is further comprised of means for generating a 1X clock signal from a 2X clock signal.

6. The caching system as recited in Claim 5 wherein said means for correlating control signals to said one or more banks of cache memory means with said system clock signal is comprised of:

a first control signal having a greater intrinsic delay than a second control signal; and

delay means for delaying said first control signal so that the delay of said first control signal equals the delay of said second control signal.

7. The caching system as recited in Claim 6 wherein said delay means is further comprised of an even number of inverters inserted into the circuit path of said first control signal.

8. The caching system as recited in Claim 5 wherein said means for providing burst data from said one or more banks of non-interleaved cache memory means is comprised of





a) means for predicting subsequent addresses from a first provided address; and

b) means for providing data from said predicted subsequent addresses in consecutive system clock signal cycles.

9. The caching system as recited in Claim 8 wherein said means for predicting subsequent addresses from a first provided address is further comprised of:

a) means for extracting a plurality of bits from said first provided address; and

b) means for identifying a predetermined sequence of addresses associated with the content of said extracted plurality of bits.

10. A method for providing cache control signals in a caching subsystem comprising the steps of:

a) generating a 1X clock signal from a 2X clock signal;

b) providing said 1X clock signal as a system clock;

c) providing processor control signals based on said 1X clock signal;

d) providing said 2X clock signal to a cache controller; and

e) providing control signals from said cache controller so that there is zero delay with respect to said system clock and said processor control signals.

11. The method for providing cache controls signals as recited in Claim 10 wherein said step of providing control signals from said cache controller so that there is zero delay with respect to said system clock and said processor control signals is further comprised of the steps of:





- a) identifying a first control signal that has a greater intrinsic delay than a second control signal;
 - b) delaying the provision of said first control signal so that it has a delay equal to said second control signal; and
 - c) providing said first control signal and said second control signal based on said 2X clock signal.
12. The method as recited in Claim 10 is further comprised of the steps of:
- a) identifying a burst read request;
 - b) identifying a plurality of subsequent addresses from a first provided address; and
 - c) providing control signals for reading data from said first provided address and said plurality of subsequent addresses.
13. A cache controller comprising:
- a) receiving means for receiving a 2X clock signal;
 - b) means for generating a 1X clock signal from said 2X clock signal and providing said 1X clock signal as a system clock signal; and
 - c) means for providing cache controller signals having a zero delay relative to said provided system clock signal.
14. The cache controller as recited in Claim 13 wherein said means for providing cache controller signals having a zero delay relative to said provided system clock signal is further comprised of delay means inserted into the circuit path of cache control signals having shorter delays than the circuit path of other cache control signals.





15. The cache controller as recited in Claim 14 wherein said delay means is comprised of an even number of inverters.
16. The cache controller as recited in Claim 13 is further comprised of means for providing a plurality of data in response to a burst read request comprised of:
- a) means for identifying a burst read request;
 - b) means for predicting a plurality of subsequent addresses from a provided first address;
 - c) means for providing said subsequent address on consecutive system clock cycles.
17. The cache controller as recited in Claim 16 wherein said means for predicting a plurality of subsequent addresses from a first provided address is further comprised of:
- a) means for extracting a plurality of bits from said first provided address; and
 - b) means for identifying a predetermined sequence of addresses associated with the content of said extracted plurality of bits.

