A method and apparatus for reducing the snooping requirements of a cache system and for reducing latency problems in a cache system. When a snoop access occurs to the cache, and if snoop control logic determines that the previous snoop access involved the same memory location line, then the snoop control logic does not direct the cache to snoop this subsequent access. This eases the snooping burden of the cache and thus increases the efficiency of the processor working out of the cache during this time. When a multilevel cache system is implemented, the snoop control logic directs the cache to snoop certain subsequent accesses to a previously snooped line in order to prevent cache coherency problems from arising. Latency reduction logic which reduces latency problems in the snooping operation of the cache is also included. After every processor read that is transmitted beyond the cache, i.e., cache read misses, the logic gains control of the address inputs of the cache for snooping purposes. The cache no longer needs its address bus for the read cycle and thus the read operation continues unhindered. In addition, the cache is prepared for an upcoming snoop cycle.
### FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

<table>
<thead>
<tr>
<th>Country Code</th>
<th>Country Name</th>
<th>Country Code</th>
<th>Country Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT</td>
<td>Austria</td>
<td>FR</td>
<td>France</td>
</tr>
<tr>
<td>AU</td>
<td>Australia</td>
<td>GA</td>
<td>Gabon</td>
</tr>
<tr>
<td>BB</td>
<td>Barbados</td>
<td>GB</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>BE</td>
<td>Belgium</td>
<td>GN</td>
<td>Guinea</td>
</tr>
<tr>
<td>BF</td>
<td>Burkina Faso</td>
<td>GR</td>
<td>Greece</td>
</tr>
<tr>
<td>BG</td>
<td>Bulgaria</td>
<td>HU</td>
<td>Hungary</td>
</tr>
<tr>
<td>BJ</td>
<td>Benin</td>
<td>IE</td>
<td>Ireland</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
<td>IT</td>
<td>Italy</td>
</tr>
<tr>
<td>CA</td>
<td>Canada</td>
<td>JP</td>
<td>Japan</td>
</tr>
<tr>
<td>CF</td>
<td>Central African Republic</td>
<td>KP</td>
<td>Democratic People's Republic of Korea</td>
</tr>
<tr>
<td>CG</td>
<td>Congo</td>
<td>KZ</td>
<td>Kazakhstan</td>
</tr>
<tr>
<td>CH</td>
<td>Switzerland</td>
<td>LK</td>
<td>Sri Lanka</td>
</tr>
<tr>
<td>CI</td>
<td>Côte d'Ivoire</td>
<td>LU</td>
<td>Luxembourg</td>
</tr>
<tr>
<td>CM</td>
<td>Cameroun</td>
<td>MC</td>
<td>Mongolia</td>
</tr>
<tr>
<td>CS</td>
<td>Czech Republic</td>
<td>MG</td>
<td>Madagascar</td>
</tr>
<tr>
<td>CZ</td>
<td>Czech Republic</td>
<td>ML</td>
<td>Mali</td>
</tr>
<tr>
<td>DE</td>
<td>Germany</td>
<td>MN</td>
<td>Mongolia</td>
</tr>
<tr>
<td>DK</td>
<td>Denmark</td>
<td>MR</td>
<td>Mauritania</td>
</tr>
<tr>
<td>ES</td>
<td>Spain</td>
<td>MW</td>
<td>Malawi</td>
</tr>
<tr>
<td>FI</td>
<td>Finland</td>
<td>NL</td>
<td>Netherlends</td>
</tr>
<tr>
<td>NO</td>
<td>Norway</td>
<td>NZ</td>
<td>New Zealand</td>
</tr>
<tr>
<td>PL</td>
<td>Poland</td>
<td>PT</td>
<td>Portugal</td>
</tr>
<tr>
<td>RO</td>
<td>Romania</td>
<td>RU</td>
<td>Russian Federation</td>
</tr>
<tr>
<td>SD</td>
<td>Sudan</td>
<td>SE</td>
<td>Sweden</td>
</tr>
<tr>
<td>SK</td>
<td>Slovak Republic</td>
<td>SN</td>
<td>Senegal</td>
</tr>
<tr>
<td>SU</td>
<td>Soviet Union</td>
<td>TD</td>
<td>Chad</td>
</tr>
<tr>
<td>TG</td>
<td>Togo</td>
<td>UA</td>
<td>Ukraine</td>
</tr>
<tr>
<td>US</td>
<td>United States of America</td>
<td>VN</td>
<td>Viet Nam</td>
</tr>
</tbody>
</table>
CACHE SNOOP REDUCTION AND
LATENCY PREVENTION APPARATUS

The present invention relates to microprocessor cache subsystems in computer systems, and more specifically to a method and apparatus for decreasing the snooping requirements and reducing latency problems in a cache system.

The driving force behind computer system innovation has been the demand for faster and more powerful personal computers. A major bottleneck in personal computer speed has historically been the speed with which data can be accessed from memory, referred to as the memory access time. The microprocessor, with its relatively fast processor cycle times, has generally been delayed by the use of wait states during memory accesses to account for the relatively slow memory access times. Therefore, improvement in memory access times has been one of the major areas of research in enhancing computer performance.

In order to bridge the gap between fast processor cycle times and slow memory access times, cache memory was developed. A cache is a small amount of very fast, and expensive, zero wait state memory that is used to store a copy of frequently accessed code and data from main memory. The microprocessor can operate out of this very fast memory and thereby reduce the number of
wait states that must be interposed during memory accesses. When the processor requests data from memory
and the data resides in the cache, then a cache read
hit takes place, and the data from the memory access
can be returned to the processor from the cache without
incurring wait states. If the data is not in the
cache, then a cache read miss takes place. In a cache
read miss, the memory request is forwarded to the
system, and the data is retrieved from main memory, as
would normally be done if the cache did not exist. On
a cache miss, the data that is retrieved from memory is
provided to the processor and is also written into the
cache due to the statistical likelihood that this data
will be requested again by the processor.

An efficient cache yields a high "hit rate", which
is the percentage of cache hits that occur during all
memory accesses. When a cache has a high hit rate, the
majority of memory accesses are serviced with zero wait
states. The net effect of a high cache hit rate is
that the wait states incurred on a relatively
infrequent miss are averaged over a large number of
zero wait state cache hit accesses, resulting in an
average of nearly zero wait states per access. Also,
since a cache is usually located on the local bus of
the microprocessor, cache hits are serviced locally
without requiring use of the system bus. Therefore, a
processor operating out of its local cache has a much
lower "bus utilization." This reduces system bus
bandwidth used by the processor, making more bandwidth
available for other devices, such as intelligent bus
masters, which can independently gain access to the
bus.

Another important feature of caches is that the
processor can operate out of its local cache when it
does not have control of the system bus, thereby
increasing the efficiency of the computer system. In systems without microprocessor caches, the processor generally must remain idle while it does not have control of the system bus. This reduces the overall efficiency of the computer system because the processor cannot do any useful work at this time. However, if the processor includes a cache placed on its local bus, it can retrieve the necessary code and data from its cache to perform useful work while other devices have control of the system bus, thereby increasing system efficiency.

Important considerations in cache performance are the organization of the cache and the cache management policies that are employed in the cache. A cache can generally be organized into either a direct-mapped or set-associative configuration. In a direct-mapped organization, the physical address space of the computer is conceptually divided up into a number of equal pages, with the page size equaling the size of the cache. The cache is partitioned into a number of sets, with each set having a certain number of lines. The line size is generally a plurality of dwords, wherein a dword is 32 bits. Each of the conceptual pages in main memory has a number of lines equivalent to the number of lines in the cache, and each line from a respective page in main memory corresponds to a similarly located line in the cache. An important characteristic of a direct-mapped cache is that each memory line from a conceptual page in main memory, referred to as a page offset, can only reside in the equivalently located line or page offset in the cache. Due to this restriction, the cache only need refer to a certain number of the upper address bits of a memory address, referred to as a tag, to determine if a copy of the data from the respective memory address resides
in the cache because the lower order address bits are pre-determined by the page offset of the memory address.

Whereas a direct-mapped cache is organized as one bank of memory that is equivalent in size to a conceptual page in main memory, a set-associative cache includes a number of banks, or ways, of memory that are each equivalent in size to a conceptual page in main memory. Accordingly, a page offset in main memory can be mapped to a number of locations in the cache equal to the number of ways in the cache. For example, in a 4-way set associative cache, a line or page offset from main memory can reside in the equivalent page offset location in any of the four ways of the cache. As with a direct-mapped cache, each of the ways in a multiple way cache is partitioned into a number of sets each having a certain number of lines: A set-associative cache also generally includes a replacement algorithm, such as a least recently used (LRU) algorithm, that determines which bank, or way, with which to fill data when a read miss occurs.

Cache management is generally performed by a device referred to as a cache controller. One cache management duty performed by the cache controller is the handling of processor writes to memory. The manner in which write operations are handled determines whether a cache is designated as "write-through" or "write-back." When the processor initiates a write to main memory, the cache is first checked to determine if a copy of the data from this location resides in the cache. If a processor write hit occurs in a write-back cache design, then the cache location is updated with the new data, and main memory is only updated later if this data is requested by another device, such as a bus master. Alternatively, the cache maintains the
correct or "clean" copy of data thereafter, and the main memory is only updated when a flush operation occurs. In a write-through cache, the main memory location is generally updated in conjunction with the cache location on a processor write hit. If a processor write miss occurs to a write-through cache, the cache controller may either ignore the write miss or may perform a "write-allocate," whereby the cache controller allocates a new line in the cache in addition to passing the data to the main memory. In a write-back cache design, the cache controller generally allocates a new line in the cache when a processor write miss occurs. This generally involves reading the remaining entries from main memory to fill the line in addition to allocating the new write data.

The cache controller includes a directory that holds an associated entry for each set in the cache. In a write-through cache, this entry generally has three components: a tag, a tag valid bit, and a number of line valid bits equaling the number of lines in each cache set. The tag acts as a main memory page number, and it holds the upper address bits of the particular page in main memory from which the copy of data residing in the respective set of the cache originated. The status of the tag valid bit determines whether the data in the respective set of the cache is considered valid or invalid. If the tag valid bit is clear, then the entire set is considered invalid. If the tag valid bit is true, then an individual line within the set is considered valid or invalid depending on the status of its respective line valid bit. In a write-back cache, the entries in the cache directory are generally comprised of a tag and a number of tag state bits for each of the lines in each set. As before, the tag comprises the upper address bits of the particular page.
in main memory from which the copy originated. The tag state bits determine the status of the data for each respective line, i.e., whether the data is invalid, modified (owned), or clean.

A principal cache management policy is the preservation of cache coherency. Cache coherency refers to the requirement that any copy of data in a cache must be identical to (or actually be) the owner of that location's data. The owner of a location's data is generally defined as the respective location having the most recent or the correct version of data. The owner of data is generally either an unmodified location in main memory, or a modified location in a write-back cache.

In computer systems where independent bus masters can access memory, there is a possibility that a bus master, such as a direct memory access controller, network or disk interface card, or video graphics card, might alter the contents of a main memory location that is duplicated in the cache. When this occurs, the cache is said to hold "stale," "dirty" or invalid data. Also, when the processor executes a cache write hit operation to a write-back cache, the cache receives the new data, but main memory is not updated until a later time, if at all. In this instance, the cache contains a "clean" or correct version of the data and is said to own the location, and main memory holds invalid or "dirty" data. Problems would arise if the processor was allowed to access dirty data from the cache, or if a bus master was allowed to access dirty data from main memory. Therefore, in order to maintain cache coherency, i.e., in order to prevent a device such as a processor or bus master from inadvertently receiving incorrect or dirty data, it is necessary for the cache controller to monitor the system bus for bus master
accesses to main memory when the processor does not control the system bus. This method of monitoring the bus is referred to as snooping.

In a write-back cache design, the cache controller must monitor the system bus during memory reads by a bus master because of the possibility that the cache may own the location, i.e., the cache may contain the only correct copy of data for this location, referred to as modified data. This is referred to as read snooping. On a read snoop hit where the cache contains modified data, the cache controller generally provides the respective data to main memory, and the requesting bus master generally reads this data en route from the cache controller to main memory, this operation being referred to as snarfing. Alternatively, the cache controller provides the respective data directly to the bus master and not to main memory. In this alternative scheme, the main memory would perpetually contain erroneous or "dirty" data until a cache flush occurred.

In both write-back and write-through cache designs, the cache controller must also monitor the system bus during bus master writes to memory because the bus master may write to or alter a memory location having data that resides in the cache. This is referred to as write snooping. On a write snoop hit to a write-through cache, the cache entry is generally marked invalid in the cache directory by the cache controller, signifying that this entry is no longer correct. In a write-back cache, the cache is updated along with main memory, and the tag states bits are set to indicate that the respective cache location now includes a clean copy of the data. Alternatively, a write-back cache may invalidate the entire line on a snoop write hit. Therefore, in a write-back cache design, the cache controller must snoop both bus master
reads and writes to main memory. In a write-through cache design, the cache controller need only snoop bus master writes to main memory.

The process of snooping generally entails that the cache controller latch the system bus address and perform a cache look-up in the tag directory corresponding to the page offset location where the memory access occurred to see if a copy of data from the main memory location being accessed also resides in the cache. If a copy of the data from this location does reside in the cache, then the cache controller takes the appropriate action depending on whether a write-back or write-through cache design has been implemented, or whether a read or write snoop hit has occurred. This prevents incompatible data from being stored in main memory and the cache, thereby preserving cache coherency.

However, the requirement that a cache snoop every non-processor memory access, or every memory write in a write-through cache, considerably impairs the efficiency of the processor working out of its cache during this time because it is continually being interrupted by snoop accesses. This snooping requirement degrades system performance because it prevents the processor from efficiently operating out of its cache while it does not have control of the system bus. Therefore, a method and apparatus is desired to reduce the snooping requirements of a cache so that the processor can more efficiently operate out of its cache when the processor does not have control of the bus.

Another problem that occurs where cache systems are utilized is that, when the respective processor is not in control of the system bus, the cache must be able to both service local requests from the processor
and snoop the system bus for memory accesses by other
devices. Latency problems can arise where the
processor is operating out of the cache and a snooping
operation is required due to a pending bus master
memory access cycle on the system bus. If the cache is
busy servicing a processor access while a bus master
memory access is occurring on the bus, the processor
access may not complete before the respective bus
master cycle completes. If this occurs, the cache will
miss a snoop cycle, thus resulting in potential
erroneous data in the cache and possible erroneous
operation. This condition is exacerbated when logic
external to the cache controller controls cache snoop
accesses to the system bus. Therefore, a method and
apparatus is desired to prevent latency problems from
occurring in a cache resulting from its dual
requirement that it simultaneously service processor
accesses and snoop accesses.

Background on multilevel cache systems is deemed
appropriate. Caches have generally been designed
independently of the microprocessor. The cache is
placed on the local bus of the microprocessor and
interfaced between the processor and the system bus
during the design of the computer system. However,
with the development of higher transistor density
computer chips, many processors are currently being
designed with an on-chip cache in order to meet
performance goals with regard to memory access times.
The on-chip cache used in these processors is generally
small, an exemplary size being 8 kbytes in size. The
smaller, on-chip cache is generally faster than a large
off-chip cache and reduces the gap between fast
processor cycle times and the relatively slow access
times of large caches.
In computer systems that utilize processors with on-chip caches, an external, second level cache is often added to the system to further improve memory access time. The second level cache is generally much larger than the on-chip cache, and, when used in conjunction with the on-chip cache, provides a greater overall hit rate than the on-chip cache would provide by itself.

In systems that incorporate multiple levels of caches, when the processor requests data from memory, the on-chip or first level cache is first checked to see if a copy of the data resides there. If so, then a first level cache hit occurs, and the first level cache provides the appropriate data to the processor. If a first level cache miss occurs, then the second level cache is then checked. If a second level cache hit occurs, then the data is provided from the second level cache to the processor. If a second level cache miss occurs, then the data is retrieved from main memory.

Write operations are similar, with mixing and matching of the operations discussed above being possible.

In many instances where multilevel cache hierarchies exist with multiple processors, a property referred to as multilevel inclusion may be implemented in the hierarchy. Multilevel inclusion provides that the second level cache is guaranteed to have a copy of what is inside the first level, or on-chip cache. When this occurs, the second level cache is said to hold a superset of the first level cache. When multilevel inclusion is implemented, and certain criteria are met, it is possible for the second level cache to perform the snooping responsibilities for both caches. For more information on this feature, please see related application Serial No. 07/538,874 filed June 15, 1990 titled "Multilevel Inclusion in Multilevel Cache"
Hierarchies," which is hereby incorporated by reference. In multilevel cache systems where multilevel inclusion is not implemented, it is generally necessary for each cache to snoop the system bus during memory accesses by other bus masters in order to maintain cache coherency.

The present invention comprises a method and apparatus for reducing the snooping requirements and reducing latency problems in a cache system. The present invention is incorporated into a computer system which includes a first level and second level cache. The first level cache is a write-through cache and the second level cache is preferably a write-back cache. Multi-level inclusion is not incorporated, and thus the first level cache would generally be required to snoop all bus master write operations to main memory. Snoop control logic according to the present invention in the second level cache controller directs the snooping operations of the first level cache such that the first level cache is not required to snoop all write operations. Thus, the snooping requirements of the first level cache are reduced.

The first level cache includes a line size comprising a plurality of words, preferably four words or 128 bits in the preferred embodiment. When a snoop write access occurs to the first level cache, either a snoop write miss occurs, in which case the data does not reside in the cache, or a snoop write hit occurs, wherein the first level cache, being a write-through cache, simply invalidates the entire line. If the snoop control logic according to the present invention determines that a subsequent snoop write access involves the same memory location line as the
immediately previous access, then the logic does not direct the first level cache to snoop these subsequent bus master writes because the respective line has either been invalidated in a previous operation, or the requested data does not reside in the cache. Therefore, the first level cache is generally only required to snoop one memory write per cache line by a bus master and will not be required to snoop subsequent write operations to this line. This eases the snooping burden of the first level cache and thus increases the efficiency of the processor working out of the first level cache during this time.

As an exception to the above method, the snoop control logic according to the present invention directs the first level cache to snoop certain subsequent bus master writes to a previously snooped line in order to prevent cache coherency problems from arising. For example, if a snoop write access occurs to a line in the cache as was discussed above, and the processor then reads an address corresponding to a dword in that line, the first level cache will retrieve the data from the second level cache or main memory. Assuming that this data resides in the second level cache, and that this data was modified due to the previous snoop write access, then the first level cache will obtain this data from the second level cache without a bus access. Having retrieved this modified data from the second level cache, the first level cache would now have valid or clean data. However, if the first level cache was prevented from snooping subsequent snoop write accesses to this line, the data in this line could be changed in the second level cache or main memory unbeknownst to the first level cache. This would result in the first level cache maintaining
an erroneous or dirty copy of data which it believes to be clean, resulting in probable erroneous operation.

In the present invention, the snoop control logic does not prevent or block snoop requests, i.e., it instructs the first level cache to snoop, on subsequent bus master writes to a memory location line wherein an immediately previous write occurred to this memory line in all cases where a CPU read hit to a modified location in the second level cache occurs prior to the write. In addition, in certain timing situations where a snoop write request has been blocked and a CPU read hit to a modified location occurs immediately thereafter, i.e., a first level cache fill of modified data from the second level cache, the snoop control logic designates as non-cacheable the address that the first level cache attempts to allocate. This prevents the first level cache from allocating a clean copy of data requested by the processor to a line after the snoop control logic has blocked a potentially important snoop request to the first level cache for this line.

The present invention also includes logic which reduces latency problems in the snooping operation of the first level cache. As discussed in the background of the invention, latency problems may result in cache systems from the dual requirement that a cache both service accesses from the processor and snoop system bus memory accesses. In the preferred embodiment of the invention, in every processor read that is transmitted to the second level cache, i.e., first level cache read misses, latency reduction logic in the second level cache controller gains control of the address inputs of the first level cache for snooping purposes. Since the first level cache has already presented the respective address of the data that it requests to the second level cache, the first level
cache no longer needs its address bus for the read cycle anyway. Thus, the read operation continues unhindered, and in addition, the first level cache is prepared for an upcoming snoop cycle.

Therefore, a method and apparatus for reducing the snooping requirements of a cache system are disclosed. In addition, a method and apparatus for reducing latency problems in the snooping operations of a cache system are disclosed.

A better understanding of the present invention can be obtained with the following detailed description of the preferred embodiment is considered in conjunction with the following drawings, in which:

Figure 1 is an exploded, perspective illustration of a modular computer system incorporating snoop control logic and latency prevention logic according to the present invention;

Figure 2 is a block diagram illustrating the first level and second level cache systems;

Figure 3 is a more detailed block diagram of the second level cache controller of Figure 2;

Figure 4 illustrates SAMELINE generation logic in the memory controller of Figure 1;

Figures 5 and 7 are schematic logic diagrams of snoop control logic in the snoop control block of Figure 3;

Figure 6 is a timing diagram illustrating operation of the logic of Figure 5;

Figure 8 is a schematic logic diagram of snoop control logic in the processor bus interface block of Figure 3; and
Figure 9 is a schematic logic diagram of latency reduction logic in the processor bus interface block of Figure 3.

The present invention is incorporated into a modular computer system in the disclosed embodiment. However, it is noted that the invention can be incorporated into virtually any type of computer system, and the following description is intended to describe one environment in which the present invention operates. In the description that follows, a signal name followed by an L indicates that the signal is asserted when it has a logic low value and is the inverse of the signal without an L.

A modular computer system generally referred to by the letter C is shown in Figure 1. The system C includes a system board S, which includes a number of devices and a series of connectors or slots 20A, 20B and 21. The slots 20A and 20B are connected together by means of a system bus or host bus 80 (Figure 2), and the slots 21 are connected together by an input/output (I/O) bus referred to as the Extended Industry Standard Architecture (EISA) bus (not shown). The circuitry located on the system board S includes a bus controller 22, referred to as the EISA bus controller (EBC), which controls operations on the EISA bus and interfaces between the host bus 80 and the EISA bus. Related to the bus controller 22 is an integrated system peripheral (ISP) 24 which contains an interrupt controller, various timers, a direct memory access
(DMA) controller, non-maskable interrupt logic refresh and EISA bus arbitration and prioritization logic. In addition, various data latches/buffers and address latches/buffers referred to as EISA bus buffers (EBBs) 26 and 28 are provided to couple to the EISA bus. Further, a random logic chip 30, commonly referred to as a system glue chip (SGC), is provided to reduce the overall space and component count of the system board S.

An I/O board I is connected to the system board S by a connector 32. The I/O board I contains certain input/output related functions and other functions as commonly developed on the X bus of a personal computer system according to the ISA or EISA architecture. For example, read only memory (ROM) 34 is located on the I/O board I. Additionally a real time clock (RTC) and CMOS memory unit 36, a floppy disk controller (FDC) 38 and a multiple peripheral controller (MPC) 40, which incorporates two serial ports, a parallel port and a hard disk interface, are also located on the I/O board I. Further, a keyboard controller (not shown) is located on the I/O board I.

A processor card P is located in the interchangeable slot 20A. The processor card P includes a central processing unit (CPU) or microprocessor 42. The CPU 42 is preferably the i486 processor by Intel Corporation. The processor card P also includes a multilevel cache system in the disclosed embodiment. Although the disclosed embodiment includes a multilevel cache system, it is noted that the present invention can be incorporated into a system which includes only a single cache system. The multilevel cache system comprises a first level cache system (Fig. 2), which is preferably located on the CPU chip 42, and a second level cache
system 44. The second level cache system 44 includes logic according to the present invention which reduces the snooping requirements and reduces latency problems in the first level cache system. The processor card P also includes a memory controller 46 and a data buffer/latch (EBB) 48. Various miscellaneous support logic (not shown) is also included on the processor board P. Additionally, an amount of base memory 50, for example, 4 Mbytes, is preferably located on the processor board P. This memory 50 is utilized with the buffer/latch 48 and is directly controlled by the memory controller 46.

Due to space limitations and the number of complex components on the processor board P, a separate memory board M is preferably located in the interchangeable slot 20B. The memory board M preferably contains a pair of data buffers/latches (EBBs) 48. Additionally, row address strobe (RAS) logic 52 and various other buffering logic 58 are located on the memory board M. Finally, a series of locations 54 for receiving memory are provided on the memory board M. This allows memory expansion to be easily developed on the memory board M. The control signals for the memory board M are transmitted from the memory controller 46 on the processor board P through the host bus on the system board S and up to the memory board M.

The computer system C also may contain a plurality of input/output related interchangeable cards. For example, in the system shown in Figure 1, the interchangeable card 56 preferably is a video card which is interconnected with a monitor. In addition, the card 57 is preferably an intelligent bus master which is capable of operating cycles on both the EISA bus and the host bus and can access memory located on the host bus. Numerous other cards can be installed as
is conventional. The bus master card 57, as well as other devices (not shown), may execute cycles on the host bus to access memory when the CPU 42 is not in control of the bus. For the remainder of this description, non-processor host bus cycles are referred to simply as bus master cycles for convenience.

Referring now to Figure 2, a diagram illustrating the second level cache 44 coupled between the CPU 42 and the host bus 80 is shown. As previously mentioned, the CPU 42 is preferably the i486 microprocessor from Intel Corporation in the disclosed embodiment. The 486 processor 42 includes an on-chip cache 70, also referred to as the first level cache. The cache 70 is preferably a write-through cache, although it is noted that the snoop reduction method of the present invention may be implemented where the cache 70 is a write-back cache. If the cache 70 is implemented as a write-back cache, the cache 70 would not be allowed to update or modify data on snoop write hits, but rather would only be allowed to update or modify data on processor write hits. Otherwise, cache coherency problems would occur. In addition, if the cache 70 is implemented as a write-back cache, the snoop reduction method could include host bus snoop read cycles as well as write cycles.

The 486 processor on-chip cache 70 includes a cache enable input referred to as KENL (not shown), which is used to determine if the address in a current cycle is cacheable. When the 486 processor 42 generates a cycle that can be cached and the KENL input is asserted low, the cycle becomes a cache line fill cycle. For more information on the KENL input to the 486 processor 42, please see the Intel i486

The 486 processor on-chip cache 70 also includes two inputs referred to as A HOLD and EADSL (both not shown) used in snoop cycles or cache invalidation cycles. A cache invalidation cycle in the cache 70 is controlled by logic external to the CPU 42 and comprises two steps. First, when a host bus write cycle is being performed which must be snooped, for example, a write cycle by the bus master 57, the external logic asserts the A HOLD input to the 486 CPU 42, forcing the processor 42 to immediately relinquish its address bus. Next, the external logic asserts the external address signal EADSL which indicates, when asserted low, that a valid address is on the 486 processor's address pins. The cache 70 then reads the address and performs a tag compare cycle, also referred to as a cache invalidation cycle. If a snoop write hit occurs, i.e., if the cache 70 determines that a copy of data from this location resides in the cache, then the respective cache line is invalidated. For more information on cache invalidation cycles in the 486 processor on-chip cache, please see the Intel i486 Microprocessor Handbook, pgs. 111-113, referenced above.

It is noted that running an invalidation cycle in the cache 70 prevents the cache 70 from satisfying other requests from the CPU 42. Thus the snooping responsibilities of the cache 70 reduce the efficiency of the processor 42 when the processor 42 does not have control of the host bus 80. Therefore it is generally desired that invalidation or snoop cycles should only be run by the cache 70 when absolutely necessary.
-20-

The second level cache 44 includes a cache controller 82 and cache memory comprising a cache data RAM 84. The CPU 42 includes a 32 bit data bus 83 comprising signals P_D<31..0>. The CPU 42 also includes an address bus 85 comprising address lines P_A<31..2>. The host bus 80 includes a data bus 87 and address bus 89 comprising signals H_D<31..0> and H_A<31..2>, respectively. The P_D<31..0> signals from the CPU 42 are coupled through a transceiver 86 to the H_D<31..0> signals. The P_D<31..0> signals are also provided from the CPU 42 to the cache data RAM 84. In the preferred embodiment, the cache data RAM 84 is organized as a 2-way set associative cache, and thus the P_D<31..0> signals are provided to each way of the cache, as shown. The P_A<31..2> signals are provided from the CPU 42 to the cache controller 82 and are also provided through a latchable transceiver 88 to the cache data RAM 84 through a cache address bus 91. The cache controller 82 provides a gating signal 90 to the transceiver 88. The cache controller 82 is also connected to the H_A<31..2> signals. Various processor control and status lines 92 are provided from the cache controller 82 to the cache data RAM 84. In addition, various control and status signals 94 are provided between the cache controller 82 and the on-chip cache 70 located in the CPU 42.

The cache controller 82 snoops addresses during cycles on the host bus 80 when the CPU 42 is not in control of the host bus 80. The cache controller 82 is preferably organized as a write-back cache, and thus the cache controller 82 must snoop both write and read operations. In the disclosed embodiment, the on-chip or first level cache 70 is a write-through cache. As explained in the background, a write-through cache is only required to snoop host bus write operations. In
the disclosed embodiment, the cache controller 82 includes logic which directs the snooping operations of the first level cache 70. When the first level cache is required to snoop a bus cycle, the second level cache controller 82 provides the current host bus address over the PA<31..2> lines to the first level cache 70, and the logic asserts respective control signals directing the on-chip cache 70 to snoop during certain bus master write operations, as is explained below.

Referring now to Figure 3, the cache controller 82 of the second level cache system 44 is shown. The cache controller 82 is interfaced between a processor local bus 98 of the CPU 42, the host bus 80, and a cache bus comprising the cache address bus 91 and cache control bus 92 (Fig. 2). The cache bus, in turn, is coupled to the cache data RAM 84 (Fig. 2).

The cache controller 82 includes a processor bus interface block 102 which tracks the states of the CPU 42 in order to maintain correct synchronization with the CPU 42. The processor bus interface block 102 is connected to the processor local bus 98 and generates a cache enable signal referred to as PKENL, which is provided to the KENL input of the on-chip cache 70. When the cache controller 82 negates the PKENL signal to the on-chip cache 70, the cache 70 is prevented from allocating a line to the respective address that has been presented to the cache 70. As described below, the PKENL signal is generated in conjunction with other logic in the cache controller 82 and is used according to the present invention to designate as noncacheable certain first level cache allocations where snooping
has been disabled and cache coherency problems could occur.

The processor bus interface block 102 also generates two signals referred to as PAHOLD and PEADSL, which are provided to the AHOLD and EADSL inputs of the 486 processor on-chip cache 70. The cache controller 82 uses the PAHOLD and PEADSL signals to control the snooping operations of the cache 70. The PEADSL and PAHOLD signals are generated in a manner to reduce the snooping requirements of the cache 70 and reduce latency problems, as is explained below.

The processor bus interface block 102 is connected to a cache bus interface block 104. The cache bus interface block 104 is connected to the cache bus and provides addresses and signals which control CPU and host bus access to the cache data RAM 84 (Fig. 2). An address multiplexor (address MUX) block 106 is coupled to the processor local bus 92, the cache bus, and is also coupled to tag RAM 108. The tag RAM 108 holds the tags or upper address bits of the address locations corresponding to the data stored in the cache data RAM 84. The address MUX block 106 provides the proper address information to the tag memory 108.

The cache controller 82 includes a tag scheduler block 110. The tag scheduler block 110 is partitioned into sub-units referred to as CPU interface, controller and state scheduler, snoop control interface, host bus control, and protocol. The controller and state schedule portion and the protocol portion are connected to the tag RAM 108. The controller and state scheduler portion is also connected to the address MUX block 106. The CPU interface portion is connected to the processor bus interface block 102 and the cache bus interface block 104. The snoop control interface portion is connected to a logic block referred to as the snoop
control block 112. The host bus control portion is connected to logic blocks referred to as the host bus control block 114 and the host bus interface block 116. The primary function of the tag scheduler block 110 is to arbitrate priority for accesses to the tag RAM 108 between the CPU 42 and the host bus 80. This function is primarily performed by the controller and state scheduler portion and the protocol portion. The CPU interface portion is primarily concerned with handling CPU read and write requests to the cache data RAM 84. The snoop interface portion is primarily concerned with snoop accesses to the tag RAM 108. The host bus control portion is primarily concerned with coordinating write-back cycles to the host bus 80.

The primary function of the snoop control block 112 is to watch the host bus 80 for any read or write memory cycles to cacheable main memory locations and help direct the snooping operations of the first level and second level caches 70 and 44. The snoop control block 112 notifies the tag scheduler block 110 of any valid snoop activity occurring on the host bus 80 to gain access to the tag RAM 108 for snooping purposes. The snoop control block 112 also includes logic according to the present invention which generates a first level cache snoop request signal referred to as FLSNPREQ to the processor bus interface block 102. The snoop control block 112 asserts the FLSNPREQ signal to indicate that the first level cache 70 should perform a snoop cycle or invalidation cycle. The FLSNPREQ signal is used by the processor bus interface block 102 in the generation of the PREADSL signal, as is explained below.

The snoop control block 112 receives a signal referred to as SAMELINE from SAMELINE generation logic (Figure 4) in the memory controller 46. The SAMELINE signal is provided from the memory controller 46.
through the host bus 80 to the snoop control block 112. The SAMELINNL signal indicates, when asserted low, that the snoop address of the current bus master host bus write cycle is within the same memory line as the immediately previous bus master write cycle. The SAMELINNL signal is used to determine if snoop write cycles must be forwarded to the first level cache 70 for snooping purposes. As explained below, the snoop control block 112 uses the SAMELINNL signal to aid in generating the FLUSHREQ signal.

The snoop control block 112 also provides a signal referred to as NOTKERNEL to the processor bus interface block 102. When the NOTKERNEL signal is asserted low, the processor bus interface block 102 deasserts the PKENL signal to the first level cache 70, directing the first level cache 70 to not allocate the current address provided to the first level cache 70. As described below, the NOTKERNEL signal is asserted in certain situations where the first level cache was directed not to snoop a write cycle on the host bus 80 and cache coherency problems could occur.

The host bus control block 114 arbitrates for control of the host bus 80 in response to a request from the tag scheduler block 110. The host bus control block 114 runs cycles compatible with the CPU cycles which it is designed to emulate. The host bus control block 114 is connected to the host bus control portion in the tag scheduler block 110 and is also connected to the host bus interface block 116. The host bus interface block 116 is connected to the host bus control portion in the tag scheduler 110, the host bus control block 114, and to the host bus 80. The host bus interface block 116 acts as an interpreter between the tag scheduler 110 and the host bus control block 114.
In the disclosed embodiment, a multilevel cache system is disclosed, and the logic that controls the snooping operation of the first level cache 70 according to the present invention resides in the second level cache controller 82. It is noted that the present invention may also be incorporated into a system which includes only a single cache. In a single cache system, the various logic referenced above which asserts the PAHOLD, PEADSL, and PKENL signals would be separate from, or part of, the cache system 70 and coupled to the host bus 80.

For a more complete understanding of the logic which follows, a brief review of the concepts behind the operation of the snoop reduction method of the present invention is deemed appropriate. When a snoop write hit occurs to a line in the on-chip cache 70, the system will invalidate the entire line, i.e., will invalidate all of the words in the line where the snoop write hit occurred. In addition, when a snoop write miss occurs to a line in the on-chip cache 70, it can be assumed that immediately subsequent writes to the same memory location line will also be snoop misses. Therefore, the logic according to the present invention determines whether a current snoop write access is to the same memory line in which an immediately previous snoop access occurred. If so, then the cache controller 82 does not direct the first level cache 70 to snoop this operation. This reduces the snooping requirements of the cache 70, allowing the CPU 42 to operate out of the cache 70 with greater efficiency when it does not have control of the host bus 80.
However, a problem can arise using the method described above in a multilevel cache system wherein the first level cache is not directed to snoop subsequent write cycles to a line in which an immediately previous snoop access has occurred. For example, assume that a snoop access occurs to a line in the first level cache, resulting in either an invalidation of the entire line or a snoop write miss. If the processor 42 then reads the address location corresponding to that line, the first level cache 70 will attempt to retrieve the information from the second level cache 44. If the second level cache 44 does not include this data, then the first level cache 70 will have to wait for access to the host bus 80, no first level cache allocations will occur, and no problems will result. However, if the second level cache 44 contains this data, the first level cache 70 will have immediate access to this data. This data will have been previously modified due to the previous CPU write access. Having retrieved this modified data from the second level cache, the first level cache 70 would now have valid or clean data. If a subsequent bus master write operation were to occur to this line, the first level cache 70 would not be directed to snoop this operation. This failure of the first level cache 70 to snoop a write operation to a location in which it contains valid data could result in possible cache coherency problems i.e., the first level cache 70 might contain erroneous or incorrect data after this unsnoop write cycle.

One possible solution to this problem would be to track this sequence of events doing a full address comparison and then to designate as noncacheable any allocation by the first level cache 70 when this sequence occurs. However, this solution would require
a large amount of address compare and decode logic, and the number of gates required would be obtrusive. Therefore, in the preferred embodiment, logic in the snoop control block 112 determines if the first level cache 70 has performed an allocation cycle during the interval between a first snoop access and immediately subsequent write cycles on the host bus 80. If a cache allocation has occurred during this interval, the first level cache 70 is directed to snoop subsequent write operations to the same memory line location. This is because the cache allocation may be to the line where a previous snoop access occurred, which can cause the cache coherency problems previously described. Due to certain timing problems, if an allocation occurs in the first level cache 70 immediately after a snoop request to the first level cache 70 has been blocked, then the attempted cache allocation is thwarted by the snoop control logic, which declares this location noncacheable.

It is again noted that the cache coherency problems described above only result from the use of the present invention in a multilevel cache system. If the present invention is incorporated into a single cache system, the processor or first level cache would always require use of the host bus to access data it does not have, and thus the cache coherency problems described above would not occur.

Referring now to Figure 4, SAMELINE generation logic in the memory controller 46 is shown. The H_A<31..4> signals are provided to inputs of a latch 120. A latching signal referred to as LNLEN is provided to the gating input of the latch 120. The outputs of the latch 120 are latched versions of the
H_A<31..4> signals referred to as LHA<31..4>. During bus master write cycles, the LNLEN signal goes active several clock cycles after the H_A<31..4> signals are valid from the bus master. For example, the LNLEN signal goes high after the STARTL signal on the EISA bus goes high. The STARTL signal is an EISA signal which provides timing control at the start of a cycle and is asserted after the LA<31..2> signals become valid. Therefore, the LHA<31..4> signals from the previous bus master cycle are present for several processor clock cycles after the H_A<31..4> signals for the new cycle are provided. This allows sufficient comparison time to develop the snooping signals.

The LHA<31..4> signals and the H_A<31..4> signals are provided to compare logic 122. The compare logic 122 asserts the SAMELINEL signal if it determines that the current host bus address is within the same memory line as the address of the previous bus master access. The compare logic 122 only uses the H_A<31..4> signals for comparison purposes and does not use the H_A<3..2> signals because the logic 122 determines whether the same memory line is being accessed, not the same memory location. Since address comparison logic such as that represented by block 122 is well known to those in the art, details of its implementation are omitted for simplicity.

Referring now to Figure 5, a portion of the snoop control block 112 which generates the FLSNPREQ signal and the NOTKENL signal to the processor bus interface block 102 according to the present invention is shown. A brief description of the signals used in this logic is deemed appropriate. A signal referred to as CLK1 is a clocking signal having a frequency of 33 MHz and is
preferably the clock signal provided to the CPU 42. A signal referred to as CLK2 is a clocking signal having a frequency of 66 MHz or twice the CLK1 signal. Preferably, the CLK2 signal has a rising edge at the same time the CLK1 signal has a rising edge. A signal referred to as PLOCKL is a locking signal output from the CPU 42 which indicates, when asserted low, that the local bus cycle currently active is locked. The asserted PLOCKL signal indicates that the CPU access must gain control of the host bus 80 and stay active until the host bus access is completed. Locked cycles are generated by the 486 CPU 42 curing certain cycles, such as read-modify-write cycles, as well as others. For more information on 486 locked cycles, please see the 486 Microprocessor Reference Guide referenced above.

A signal referred to CPU RDHITM is asserted high when a CPU read hit occurs to a modified location in the second level cache 44. This signal is provided by the tag scheduler block 110. As previously discussed, the SAMELINEL signal is a signal from the memory controller 46 which indicates that the snoop address of the current host bus cycle is within the same memory line as the previous snoop cycle. A signal referred to as HSSTBL is a host bus snoop strobe signal which indicates, when asserted low, that the current host bus access cycle must be snooped. The HSSTBL signal is asserted for one CLK1 signal cycle when the H_A<31..2> signals are becoming valid. This signal is provided by the memory controller 46. By means of circuitry not shown, the asserted HSSTBL signal sets a signal referred to as SNOOPREQL, which remains asserted until the snoop is actually serviced. A signal referred to as HWR is a host bus write/read signal which is asserted high during write cycles and negated low
during read cycles. A signal referred to as ENDPCYCL indicates that a processor bus cycle has completed. The ENDPCYCL signal is generated by logic in the processor bus interface block 102. A signal referred to as RESETL is a system reset signal which is asserted low.

Referring again to Figure 5, the PEADSL signal and the CLK1L signal are connected to the inputs of a two input OR gate 140 whose output is connected to the input of a two input NAND gate 142. The other input of the NAND gate 142 is connected to a signal referred to as BLKSAMELINE. The BLKSAMELINE signal is output from the Q output of a D-type flip-flop 146. The PEADSL signal, the PLOCKL signal, and the CPURDHITM signal are connected to inputs of a three input NAND gate 148. The output of the NAND gate 148 and the output of the NAND gate 142 are connected to the inputs of a two input NAND gate 144, whose output is a signal referred to as BLKSAME. When high, the BLKSAME signal guarantees that the snoop write cycles are provided to the first level cache 70. The BLKSAME signal is provided to the D input of the D-type flip-flop 146. The clock input of the flip-flop 146 receives the CLK2 signal, and the inverted CLR input receives the RESETL signal. The Q output of the flip-flop 146, which is the BLKSAMELINE signal, and the SAMELINEL signal are connected to the inputs of a two input OR gate 150 whose output is a signal referred to as FLDONTSNPL. The FLDONTSNPL signal indicates that the first level cache 70 should not be directed to snoop the current host bus write cycle.

Therefore, the FLDONTSNPL signal is active or low when the SAMELINEL signal is asserted low and the BLKSAMELINE signal is negated low. Thus the FLDONTSNPL signal is asserted when the current bus master write is
to the same memory location line as the immediately
previous bus master write and an unlocked processor
read hit to a modified location in the secondary cache
has not occurred.

The HSSTBL signal and the SNOOPREQ signal are
provided to the inputs of a two input NAND gate 152
whose output is connected to an input of a 7 input NAND
gate 154. The SAMELINEL signal is connected through an
inverter 156 to an input of the NAND gate 154. Other
inputs to the NAND gate 154 are the PLOCKL, CPURDHTM,
CLK1, and HWR signals, and the inverted Q output from
the flip-flop 146, the BLKSAMELINEL signal. The output
of the NAND gate 154 is a signal referred to as
SETBLKENL. The SETBLKENL signal indicates that a snoop
write request was blocked by the snoop control logic
during a bus master write cycle and a CPU read hit to a
modified location in the second level cache has begun.
The ENDPYCWL signal and a NOTKEN signal are provided to
the inputs of a two input NAND gate 158. The output of
the NAND gate 158 is a signal referred to as HLDBLKENL.

The SETBLKENL signal and the HLDBLKENL signal are
connected to inputs of a two input NAND gate 160 whose
output is a signal referred to as ENBLKKEN. The
ENBLKKEN signal is connected to the D input of a D-type
flip-flop 170. The clock input of the flip-flop 170
receives the CLK2 signal, and the inverted clear input
receives the system reset signal. The Q output of the
flip-flop 170 is the NOTKEN signal, and the inverted Q
output provides the NOTKEN signal.

The NOTKENL signal indicates to logic in the
processor bus interface logic 102 that it is possible
that a snoop write request was blocked by this logic
just before a CPU read modify write cycle was started.
As discussed above, this situation could cause possible
cache coherency problems. The SETBLKENL signal sets
the flip-flop 170 to assert the NOTKENL signal, and the HLDBLKLEN signal maintains the NOTKENL signal asserted for the remainder of the processor bus cycle.

Referring now to Figure 8, the NOTKENL signal is connected to an input of a two input NAND gate 240. The other input to the NAND gate 240 receives a signal referred to as KEN, which is asserted high to indicate that other criteria for a cacheable address have been met, such as the cycle is a memory cycle and the address resides in a cacheable address range that has not otherwise been designated noncacheable, etc, as normally developed to indicate cacheable cycles. The output of the NAND gate 220 is the PKENL signal which, when asserted low, informs the first level cache 70 that the address which it is proposing to allocate is a cacheable address which can indeed be cached. Therefore, when the NOTKENL signal is negated high, the KEN signal flows through the NAND gate 220 to assert the PKENL signal. When the NOTKENL signal is asserted low, the signal negates the PKENL signal high to indicate that the address should not be cached.

Referring now to Figure 6, a timing diagram illustrating the situation where NOTKENL signal is asserted, is shown. When the BLKSAMELINEL signal is negated high, then the SAMELINEL signal passes through the OR gate 150, and snoop write accesses are not passed to the first level cache 70. The negated BLKSAMELINEL signal between time T1 and T3 prevents the assertion of the PEADSFL signal at time T4. However, if a CPU read hit to a modified location in the second level cache 44 had started at time T2, then it is possible that the cache line where a snoop access was blocked may have been allocated, resulting in possible cache coherency problems. Therefore, the NOTKENL signal is asserted at time T3 to prevent any first
level cache line from being allocated when this situation occurs. The asserted NOTKENL signal deasserts the PKENL signal for this period, preventing any first level cache allocations from occurring.

Referring now to Figure 7, the FLDONTSNPL signal is connected to the input of a two input NAND gate 202. The other input to the NAND gate 202 receives a signal referred to as SNOOP, which represents various conditions required to indicate that a snoop access should occur. These other conditions include signals indicating a memory write cycle, the snoop strobe signal being asserted, the address being a cacheable address, the cache being enabled or turned on, etc. The output of the NAND gate 202 is connected to an input of a three input NAND gate 204. The output of the NAND gate 204 is connected to the D input of a D-type flip-flop 206. The clock input of the flip-flop 206 receives the CLK2 signal, and the inverted clear input receives the RESETL signal. The Q output of the flip-flop is the FLSNPREQ signal, which is provided to the processor bus interface logic 102. The FLSNPREQ signal is also connected to an input of a two input NAND gate 208. The other input of the NAND gate 208 receives the CLK1L signal. The FLSNPREQ signal and a signal referred to as FLSNPDONE, which represents that a snoop cycle has completed in the first level cache 70 and is developed from the processor bus interface block 102, are connected to inputs of a two input NAND gate 210. The output of the NAND gates 208 and 210 are connected to inputs of the NAND gate 204.

Therefore, when the SNOOP signal is asserted, indicating an otherwise valid snoop condition to the first level cache 70, and the FLDONTSNPL signal is negated high, meaning that snooping is not being blocked by the snoop control logic of Figure 5, then
the flip-flop 206 is set, and the FLSNPREQ signal is asserted high to request that the first level cache 70 snoop the current host bus cycle. Once the FLSNPREQ signal is asserted high, the gate 208 maintains the signal asserted high while the CLK1 signal is low. The NAND gate 210 maintains the FLSNPREQ signal asserted until the first level cache 70 completes the invalidation cycle, signified by the FLSNPDONE signal being asserted.

Referring again to Figure 8, logic in the processor bus interface block 102 which receives the FLSNPREQ and NOTKENL signals from the snoop control block 112 and generates the PEADSL and PKENL signals to the first level cache 70, is shown. The FLSNPREQ signal is connected to an input of a four input NAND gate 220. Other inputs to the NAND gate are a signal referred to as AHOOLDDLY, the CLK1 signal, and the PEADSL signal. The AHOOLDDLY signal is a version of the PAHOLD signal delayed one CLK1 signal cycle. The AHOOLDDLY signal indicates, when asserted high, that the AHOOLD input of the 486 CPU 42 has been asserted for one CLK1 signal cycle. The output of the NAND gate 220 is a signal referred to as SETEADSL, which is connected to an input of a two input NAND gate 222. The Q output of the flip-flop 226 is a signal referred to as PEADS, which is connected to an input of a two input NAND gate 224. The other input of the NAND gate 224 receives the CLK1L signal. The output of the NAND gate 224 is a signal referred to as EADSPH2L which is connected to the other input of the NAND gate 222. The output of the NAND gate 222 is connected to the D input of the flip-flop 226. The clock input of the flip-flop 226
receives the CLK2 signal, and the inverted clear input receives the RESETL signal. The Q output of the flip-flop 226 is connected through an inverter 228 to form the PEADSL signal.

As previously noted, the PEADSL signal is asserted to the EADSL input of the 486 processor 42 to begin a snoop or invalidation cycle. The NAND gate 220 is responsible for setting the flip-flop 226 and asserting the PEADSL signal. The SETEADSL signal is asserted low when the FLSNPREQ signal is asserted high, indicating a snoop request; the AHOLOADLY signal is asserted, guaranteeing that the address pins of the 486 CPU 42 are ready to receive the address for invalidation purposes; the CLK1 signal is high; and the PEADSL signal is currently negated high. Once the flip-flop 226 is set and the PEADSL signal is asserted low, the NAND gate 224 maintains the PEADSL signal asserted for the next phase of the CLK1 signal when the CLK1 signal is low.

Referring now to Figure 9, the processor bus interface block 102 includes logic according to the present invention which reduces latency problems in the first level cache 70. This logic asserts the PAHOLD signal to the AHOLOAD input of the CPU 42 after every CPU read cycle that is transmitted beyond the first level cache 70. A brief review of the signal names used in this logic is deemed appropriate. A signal referred to as PMIO is a processor memory-input/output signal which is high during processor memory cycles and is low during processor I/O cycles. A signal referred to as PADSL is the processor address status signal. A signal referred to as PWR is the processor write/read signal.
A signal referred to as HBDONEDLY is indicative of a host bus cycle completing. The HBDONEDLY signal is asserted one CLK2 cycle after the host bus ready signal is returned by the system. A signal referred to as ALCATEDLY indicates that an allocation is occurring to the second level cache.

A signal referred to as BYPRDIP indicates that a bypass read is in progress, meaning that the CPU 42 is running a read cycle to the host bus for data that will be provided to the processor 42 and that will not be cached in either of the first level or second level caches 70 and 44. These cycles include noncacheable address or NCA cycles, and locked cycles, which are not cached. A signal referred to as HLSTRDYLY is a ready signal indicating that the last cycle of a burst transfer or BYPRDIP cycle has completed. A signal referred to as HHoldA is a host hold acknowledge signal from the host bus 80 which indicates that the host bus 80 has acknowledged a hold request.

A signal referred to as T2L indicates, when asserted low, that a CPU cycle is in its T2 state, meaning that the cycle is in progress and the processor address strobe signal PADS is already been asserted. When a cycle is in the T2 state, the CPU 42 is either sending data or waiting to receive data. A signal referred to as T2 is an inverted version of the T2L signal. A signal referred to as PBOFFL is a signal provided to the 486 back-off or BOFFL input. The asserted PBOFFL signal forces the 486 processor 42 to float its bus in the next CLK1 signal cycle. A signal referred to as CPULINEDLY indicates, when asserted high, that a read hit has occurred in the secondary cache 44 and the CPU 42 is retrieving data from the secondary cache 44. A signal referred to as LBAL indicates, when asserted low, that the CPU 42 is
performing a local bus access cycle on its processor bus 98 (Fig. 3).

The PMIO signal and the PAHOLDL signal are connected to inputs of a two input NAND gate 302. The output of the NAND gate 302 is connected to an input of a four input OR gate 304. Other inputs to the OR gate 304 are the PADSL signal, the PWR signal, and the CLK1L signal. The output of the OR gate 304 is a signal referred to as SETRDAHDL. The HBDONEDLY and the ALCATEDLY signals are connected to inputs of a two input NAND gate 308 whose output is connected to an input of a five input NAND gate 310. The BYPRDIP signal is connected through an inverter 312 to an input of a four input NOR gate 314. Other inputs to the NOR gate 314 are the HLSTRDYDL signal, the HWR signal, and the HHOLDA signal.

The CLK1L signal, the T2L signal, and the PBOFDL signal are connected to inputs of a three input NOR gate 316. The outputs of the NOR gates 314 and 316 are connected to inputs of a two input NOR gate 318 whose output is connected to the NAND gate 310. The CPULINEDLY signal and the CLK1 signal are connected to inputs of a two input NAND gate 320 whose output is connected to the NAND gate 310. The LBAL signal is connected through an inverter 322 to an input of a two input NAND gate 324. The other input to the NAND gate 324 receives the T2 signal. The output of the NAND gate 324 is a signal referred to as LBACYCL, which is connected to an input of the NAND gate 310. The LBACYCL signal indicates, when asserted low, that a local bus access cycle is occurring, and the cycle is in the T2 state. A signal referred to as CPURDAHLDL is connected to the remaining input of the NAND gate 310.

The CPURDAHLDL signal and the CLK1L signal are connected to inputs of a two input NAND gate 326.
SETRDAHLDL signal output from the OR gate 304 and the
outputs from the NAND gates 326 and 310 are connected
to inputs of a three input NAND gate 306. The output
of the NAND gate 306 is connected to the D input of a
D-type flip-flop 330. The clock input of the flip-flop
330 receives the CLK2 signal and the inverted clear
input receives the system reset signal RESETL. The Q
output of the flip-flop 330 is the CPURDAHLD signal.
The CPURDAHLD signal is connected to an input of a two
input OR gate 332. The other input to the OR gate
receives a signal referred to as SETPAHOLD. The
SETPAHOLD signal indicates other conditions where the
PAHOLD signal should be asserted. One condition is
during a power-up reset where the PAHOLD signal is
asserted to direct the 486 CPU 42 to begin its power-on
self test or POST procedure. Another condition is
during snoop read hits to the second level cache 44
which require write-back cycles. Other conditions may
also be included.

The output of the OR gate 332 is a signal referred
to as ENAHLD, which is connected to an input of a two
input AND gate 334. The LBACYCL signal is connected to
the other input of the AND gate 334. The output of the
AND gate 334 is the PAHOLD signal. The LBACYCL signal
prevents the PAHOLD signal from being asserted when a
processor local bus access cycle is occurring and the
cycle is in the T2 state. The PAHOLD signal is
connected through an inverter 336 to form the PAHOLDL signal.

The PAHOLD signal is provided to the AHOLD input
of the 486 CPU 42 and is asserted after every CPU read
cycle that advances beyond the first level cache 70.
The OR gate 304 which generates the SETRDAHLDL signal
is responsible for setting the flip-flop 330, and hence
for asserting the PAHOLD signal, after every CPU read
advancing beyond the first level cache 70. Once the flip-flop 330 is set, the CPU/RDAHL signal is asserted. The NAND gate 326 maintains the flip-flop 330 asserted during the period when the CLK1 signal is low. The NAND gate 310 maintains the flip-flop 330 asserted thereafter.

Certain conditions can occur which clear the flip-flop 330 and deassert the PAHOLD signal. These conditions are represented by the gates 308, 314, 316, and 320. The NAND gate 308 deasserts the PAHOLD signal when an allocate cycle completes to the second level cache 44. The NOR gate 314 deasserts the PAHOLD signal when the last ready signal is returned from either a burst cycle or a bus ready in progress cycle. The NOR gate 316 deasserts the PAHOLD signal when the PBOFF signal is asserted low. When the PBOFFL signal is asserted, the CPU 42 is forced into its back-off state, and thus the PAHOLD signal no longer need be asserted. The NAND gate 320 deasserts the PAHOLD signal during a CPU read hit to the secondary cache. When this occurs, the data from the second level cache will be immediately provided to the first level cache 70 and the CPU 42, and thus there will be insufficient time to allow any snoops by the first level cache 70 during this time. Therefore, when this situation occurs, the PAHOLD is deasserted. Finally, the PAHOLD signal is not asserted when the LBACYCL signal is asserted low because local bus access cycles performed by the CPU 42 should not be interrupted.

Therefore, a method and apparatus for reducing the snooping requirements of a cache system are disclosed. When a snoop access occurs to a line of the cache, further snoop accesses to that line are unnecessary
when the same memory line is being accessed, and thus the cache is relieved of this snooping burden. Because a multilevel cache system is implemented in the disclosed embodiment, the first level cache is directed to snoop in all cases where a read hit occurs to a modified location in the second level cache. In addition, an attempted first level cache allocation is declared non-cacheable if it occurs within a period of time after a snoop write request has been blocked.

Also, a method and apparatus for reducing the latency of a cache system is disclosed. When a cache system has the dual requirement of servicing processor accesses and snooping the host bus, latency problems can occur during snoop accesses. Logic according to the present invention gains access to the address inputs of the cache system after every processor read that propagates beyond the cache system so that the address inputs are available for snooping purposes.

The foregoing disclosure and description of the invention are illustrative and explanatory thereof, and various changes in the components and details of the illustrated circuitry, as well as the construction and method of operation of the invention, may be made without departing from the spirit of the invention.
CLAIMS:

1. An apparatus for reducing the snooping requirements of a cache system in a computer system, comprising:
   a system bus, wherein bus cycles of a first data length execute on said system bus;
   a processor coupled to said system bus;
   a cache system coupled between said system bus and said processor, said cache system including a cache memory storing a plurality of lines of data and having a line size comprising a plurality of units of said first data length, wherein said cache system invalidates an entire line on a snoop hit;
   a main memory coupled to said system bus including a plurality of lines storing data corresponding to said cache memory lines, wherein said processor can access said main memory using said system bus;
   a device coupled to said system bus which can access said main memory using said system bus;
   means coupled to said cache system and said system bus for directing said cache system to snoop said system bus during a first device main memory access to a first main memory location; and
   means coupled to said means for directing said cache system to snoop and said system bus for preventing operation of said means for directing said cache system to snoop during a subsequent device main memory access to a main memory location in the same line as said first main memory location, wherein no processor main memory accesses occur between said first device main memory access and said subsequent device main memory access.
2. The apparatus of claim 1, wherein said means for preventing further includes means coupled to said system bus for determining if said subsequent device main memory access is to the same main memory line as said first main memory location and generating a signal indicative thereof.

3. The apparatus of claim 2, wherein said subsequent device main memory access occurs immediately after said first device main memory access.

4. The apparatus of claim 1, wherein said cache system is a write-through cache system and said device main memory accesses are main memory write accesses.

5. The apparatus of claim 1, wherein said cache system is a write-back cache system, and said cache system invalidates a cache line on a snoop write hit.

6. The apparatus of claim 5, wherein said device main memory accesses are either main memory read or main memory write accesses.

7. The apparatus of claim 1, further comprising:
   a second cache system coupled between said cache system and said system bus, wherein said means for preventing is further coupled to said cache system and does not prevent operation of said means for directing said cache system to snoop after a cache allocation occurs in said cache system from said second cache system.

8. The apparatus of claim 7, wherein said means for directing said cache system to snoop provides an indication of a cacheable or non-cacheable address to
said cache system during cache allocations, said means
for directing said cache system to snoop providing a
non-cacheable indication to said cache system if said
cache system attempts an allocation from said second
cache system a period of time after said means for
preventing prevents operation of said means for
directing said cache system to snoop.

9. The apparatus of claim 7, wherein said means
for directing said cache system to snoop and said means
for preventing are located in said second cache system.

10. A method for reducing the snooping
requirements of a cache system in a computer system,
the computer system comprising:
    a system bus, wherein bus cycles of a first
data length execute on said system bus;
    a processor coupled to said system bus;
    a cache system coupled between said processor
and said system bus, said cache system including a
    cache memory storing a plurality of lines of data and
having a line size including a plurality of units of
said first data length, wherein said cache system
    invalidates an entire line on a snoop hit;
    main memory coupled to said system bus
    including a plurality of lines storing data
    corresponding to said cache memory lines, wherein said
processor can access said main memory using said system
bus; and
    a device coupled to said system bus which can
access said main memory using said system bus,
the method comprising:
    directing said cache system to snoop
said system bus during a first device main memory
access to a first memory location; and
24 preventing execution of said step of
25 directing during a subsequent device main memory access
26 to a memory location in the same line as said first
27 memory location, wherein no processor main memory access
28 occur between said first device main memory access and
29 said subsequent device main memory access.

11. An apparatus for reducing latency problems in
a cache system, comprising:
   a system bus;
   memory coupled to said system bus;
   a processor coupled to said system bus which
   can access said memory using said system bus;
   a cache system coupled between said processor
   and said system bus which generates a read allocation
   request to allocate data into the cache system and
   snoops addresses presented to its address inputs,
   wherein said cache system also services requests from
   said processor when said processor is not using said
   system bus; and
   means coupled to said cache system for
   gaining access to said cache system address inputs for
   snooping purposes immediately after said cache system
   generates a read allocation request.

12. The apparatus of claim 11, further
comprising:
   means coupled to said cache system for
   terminating access to said cache system address inputs
   when said read allocation is completed.

13. The apparatus of claim 12, further
comprising:
   a second cache system coupled between said
   cache system and said system bus; and
means coupled to said cache system and to said second cache system for terminating access to said cache system address inputs when a read hit occurs to said second cache system.
SNOOP CONTROL

FIG. 7

PROCESSOR BUS INTERFACE

FIG. 8
# INTERNATIONAL SEARCH REPORT

## I. CLASSIFICATION OF SUBJECT MATTER
According to International Patent Classification (IPC) or to both National Classification and IPC

<table>
<thead>
<tr>
<th>Int.Cl.</th>
<th>Classification Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>G06F12/08</td>
</tr>
</tbody>
</table>

## II. FIELDS SEARCHED

<table>
<thead>
<tr>
<th>Classification System</th>
<th>Classification Symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Int.Cl. 5</td>
<td>G06F</td>
</tr>
</tbody>
</table>

Documentation Searched other than Minimum Documentation to the extent that such Documents are Included in the Fields Searched

## III. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of Document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to Claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>EP,A,0 090 575 (WESTERN ELECTRIC COMP.) 5 October 1983 see page 3, line 6 - page 4, line 22; figure 3</td>
<td>1, 4, 10</td>
</tr>
<tr>
<td>A</td>
<td>see page 3, line 6 - page 4, line 22; figure 3</td>
<td>2, 3, 5-9</td>
</tr>
<tr>
<td>X</td>
<td>EP,A,0 288 649 (INTERNATIONAL BUSINESS MACHINES) 2 November 1988 see abstract see column 12, line 56 - column 13, line 24</td>
<td>1, 4</td>
</tr>
<tr>
<td>A</td>
<td>ELECTRO CONFERENCE RECORD vol. 16, 16 April 1991, NEW YORK US pages 283 - 288 HANDY 'Practical cache design techniques for today's RISC and CISC CPUs' see page 287, right column, line 7 - page 288, right column, line 8</td>
<td>11-13</td>
</tr>
</tbody>
</table>

## IV. CERTIFICATION

- Date of the Actual Completion of the International Search
  
  **03 JUNE 1993**

- Date of Mailing of this International Search Report
  
  **11. 06. 93**

- International Searching Authority
  
  **EUROPEAN PATENT OFFICE**

- Signature of Authorized Officer
  
  **NIELSEN O.P.**

---

*Special categories of cited documents:

- **X**: document defining the general state of the art which is not considered to be of particular relevance
- **A**: earlier document but published on or after the international filing date
- **L**: document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
- **O**: document referring to an oral disclosure, use, exhibition or other means
- **P**: document published prior to the international filing date but later than the priority date claimed

**T** later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

**X** document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

**Y** document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

**&** document member of the same patent family
<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of Document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to Claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>IRE WESCON CONVENTION RECORD vol. 34, November 1990, NORTH HOLLYWOOD US pages 90 - 94 CANTRELL 'Futurebus+ Cache Coherence' see page 92, right column</td>
<td>11,12</td>
</tr>
</tbody>
</table>
This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CA-A- 1186804</td>
<td>07-05-85</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GB-A, B 2117148</td>
<td>05-10-83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP-C- 1688928</td>
<td>11-08-92</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP-B- 3045407</td>
<td>11-07-91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP-A- 58212694</td>
<td>10-12-83</td>
</tr>
</tbody>
</table>

For more details about this annex: see Official Journal of the European Patent Office, No. 12/82