

Cache controller and associated method for remapping cache
address bits

FIELD OF THE INVENTION

This invention relates to the field of cache memories for microprocessor systems, and particularly to a cache controller for an associative cache in a system having a page mapped memory.

BACKGROUND OF THE INVENTION

A significant barrier to improving the performance of a microprocessor system is the access time of system memory. Although the speed of semiconductor memories has improved over time, the speed of DRAM devices has not kept pace with the speed of the processors. Consequently, when executing most applications, a processor will experience numerous wait states while system memory is accessed. A frequently employed solution to this problem is the incorporation in the microprocessor system of a high-speed cache memory comprising SRAM devices. In general, a cached system will experience significantly fewer wait states than a non-cached system.

The simplest form of cache is generally referred to as a direct-mapped cache, wherein contents of the system memory are retrieved and stored in cache locations having the same low-order address. For example, if an 8K cache is provided, the thirteen lowest order address bits of the system memory location to be

retrieved define the cache storage location. A significant disadvantage of a direct-mapped cache is that the cache contents will be overwritten whenever there is an access request to a system memory location having the same low order address but a different high order address.

To overcome this disadvantage, a set associative cache structure is sometimes used. For example, with a two-way set associative cache, the cache memory is physically divided into two banks of SRAMs. Thus, a two-way set associative 8K cache would comprise two 4K banks of SRAM. Data retrieved from system memory may be mapped into either one of the two banks since the two banks have identical low order addresses. A cache hit in one bank causes a least recently used (LRU) flag to be set for the corresponding address in the other bank. Thus, cache writes may be directed to the cache bank whose contents were least recently used, thereby preserving the more recently used data for subsequent accesses by the CPU. An associative cache significantly improves the cache hit rate and thus improves overall system performance.

Modern microprocessor systems have enormous virtual addressing capabilities. For example, the protected virtual address mode of the Intel 386TM SX processor supports a virtual memory of 64 terabytes (2^{46} bytes). Management of such a large

virtual address space is facilitated by an expanded memory system (EMS) such as specified in the Lotus/Intel/Microsoft (LIM) 4.0 standard. Under this standard, address space is partitioned into 16K byte pages that are mapped into the physical address space. Use of EMS complicates cache management since any changes to the EMS mapping will invalidate cached data. To preserve cache integrity, prior art systems must either make EMS pages non-cacheable and suffer a loss in system performance or flush the contents of the cache whenever an EMS mapping change invalidates the contents.

One of the objects of the present invention is to provide a cache controller that overcomes many of the prior art disadvantages of operating in an EMS environment.

SUMMARY OF THE INVENTION

The present invention provides a cache controller for a set associative cache that selectively remaps predetermined bits of the cache address so as to confine data from a single memory page to a particular block of the cache memory. When changing a memory page, only the particular block of the cache in which data from that page may be stored is flushed, thereby preserving the remaining contents of the cache.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a functional block diagram of a microprocessor system embodying the present invention.

Figure 2 is a simplified functional block diagram of a prior art EMS mapper implementation.

Figure 3 is a simplified block diagram of an alternative prior art EMS mapper implementation.

Figure 4 illustrates prior art cache mapping of a memory page.

Figure 5 illustrates cache mapping of a memory page according to the present invention.

Figure 6 is a logic circuit diagram for the cache address mapping of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following description, for purposes of explanation and not limitation, specific mappings, bit assignments, etc. are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced in other embodiments that depart from these specific details. In other instances, detailed descriptions of well known aspects of microprocessor systems are omitted so as to not obscure the description of the present invention with unnecessary detail.

System Overview

The present invention is advantageously employed in a microprocessor design optimized for applications requiring low power consumption and compact size. Such applications particularly include small personal computers of the types commonly referred to as laptops and notebooks. One embodiment of such a microprocessor is briefly described below; however, it is to be understood that the present invention is not limited to this particular microprocessor design, and may be incorporated in virtually any processor design.

Referring to Figure 1, a processor system 10 is shown. System 10 comprises three main processor components, designated as GENCPU, GENIO and GENVGA. GENCPU is an expanded central processing unit including the CPU 12, a memory controller 14, a cache controller 16, ISA bus control logic 18 and line buffers 20.

In the described embodiment, CPU 12 is essentially a 386TM SX CPU manufactured by Intel Corporation, the corporate assignee of this invention. Throughout this description, certain terminology relating to the 386TM SX CPU, such as register names, signal nomenclature, etc., is employed to describe the present invention. Such terminology is understood by practitioners in the field of microprocessor design and will therefore not be explained at length herein. For further information relating to the internal structure of CPU 12, reference is made to the "386TM SX Microprocessor Hardware Reference Manual" published by Intel Corporation as publication number 240332 and related publications.

GENIO is a single chip input/output unit comprising parallel ports 22, dual serial ports 24a, 24b, real time clock unit 26, dual programmable interrupt controllers 28a, 28b, dual programmable timers 30a, 30b, dual DMA controllers 32a, 32b and memory mapper 34.

GENVGA is a single chip graphics interface comprising VGA graphics controller 36, video memory controller 38 and interface 40 for a flat panel display unit.

All of the processor units communicate with one another and with other system components (such as expansion slots, keyboard controller, disk controllers, etc.) via ISA bus 42.

External to the three main components of system 10 are system memory 44, optional cache memory 46 and optional video memory 48. An optional PAL/DAC unit 50 provides a video interface with a conventional VGA monitor.

Cache Controller Implementation

Prior art systems that include both a cache memory and an EMS mapping function may be implemented in one of two ways. Figure 2 illustrates a system wherein CPU address bus 102 from CPU 100 is coupled to EMS mapper 104. Remapped addresses on bus 106 are then provided to both cache controller 108 and memory controller 110. A system implemented as illustrated in Figure 2 has the advantage of being fully cacheable since remapped addresses are provided to both the cache controller and the memory controller. However, the system of Figure 2 has the disadvantage of inserting a delay in the address path to the cache controller when EMS is being used and thus will likely

cause CPU 100 to experience additional wait states.

An alternative implementation is illustrated in Figure 3. In this system, CPU address bus 102 is coupled both to EMS mapper 104 and to cache controller 108. Remapped address bus 106 from the EMS mapper is coupled to memory controller 110 as in the system illustrated in Figure 2. The address path to the memory controller is thus identical in both of the systems illustrated in Figures 2 and 3. The system of Figure 3 has the advantage of eliminating any wait states caused by the EMS mapper to cache access requests. However, this implementation has the disadvantage of requiring that the cache be flushed whenever EMS is enabled, disabled, EMS page mapping is modified, or the EMS register set is changed. This can significantly degrade cache performance. Yet another alternative is to make all EMS memory areas non-cacheable, but this imposes an even more serious performance penalty.

The present invention is implemented generally in accordance with the system architecture illustrated in Figure 3, but is done in such a manner that EMS memory may be cached without sacrificing cache performance. The performance objective of the present invention is based upon two assumptions, namely that the EMS function will be enabled or disabled relatively infrequently and that a high percentage (on the order of 90%) of hardware EMS

users use only one EMS register set. Thus, the most frequently occurring event that requires cache flushing is the remapping of an EMS page. The cache controller of the present invention handles this situation by allowing the selective flushing of the cache in 16K byte blocks, corresponding to the size of an EMS page, regardless of cache associativity. Thus, when and if an EMS page is changed, only that 16K block of the cache which corresponds to the page being remapped will need to be flushed, rather than the entire cache as in prior art systems.

Implementation of the present invention may be conceptualized with the aid of Figures 4 and 5. Figure 4 diagrammatically illustrates the structure of a conventional 64K 4-way set associative cache. Using a 16 bit (2 byte) data word structure, each set of the cache comprises 8K word addresses. Let X specify a location in main memory and W a location equal to $X+16K$ so that X and W are in different 16K memory pages, but at the same relative position in their respective pages. Data retrieved from location X or W of main memory may be placed into the cache in any one of the four sets at the locations indicated as $X1-X4$. Data from main memory locations $Y = X+8K$ or $Z = W+8K$ (where Y and Z are in the same memory pages as X and W, respectively) can be placed into the cache in any one of the four sets at the locations designated as $Y1-Y4$. Thus, data from a single 16K page of memory may be randomly distributed throughout

the cache structure. Consequently, when a page is remapped, it is necessary to flush the entire contents of the cache in order to assure the complete deletion of the old data from cache memory and thus insure cache coherency.

Referring now to Figure 5, the mapping of data from memory into the cache by the cache controller of the present invention is illustrated. The incoming addresses to the cache controller are remapped so that all of the addresses within a single 16K page are collapsed into a 2K block of cache word addresses. As shown, data from location X of a main memory page may be placed into the cache in any one of the sets at the locations designated X1'-X4'. Likewise, data from locations W, Y and Z (as defined above) can be placed into any one of the sets at the locations designated as W1'-W4', Y1'-Y4' and Z1'-Z4', respectively. By this remapping of the incoming addresses to the cache controller, all cached data from a single 16K page of main memory will be confined to a contiguous 16K block of the cache, i.e., one 2K block of word addresses across each of the four sets. As a result, when a main memory page is remapped, it is necessary only to flush the 16K block of the cache in which data from that page has been stored. Data from other pages stored in the remaining 48K of the cache are unaffected. Therefore, cache performance is significantly improved in comparison to prior art systems that must flush the entire cache.

The detailed implementation of the present invention is illustrated in Figure 6. This is used in conjunction with the system illustrated in Figure 1 which is designed to accommodate a 16K, 32K or 64K cache. A high state on input 202 indicates a 16K cache, whereas a high state on input 204 indicates a 32K cache. If neither input is set high, a 64K cache is assumed.

The physical address provided from CPU12 comprises address bits A1-A23. For a 64K byte four-way set associative cache, bits A1-A13 define one of 8K cache word addresses and higher order bits A12-A23 comprise a tag field used by cache controller 16 to determine if there is a cache hit. CPU address lines A12-A15 are remapped by the cache controller whenever EMS is enabled as indicated by a high state of signal EMEMSE. CPU address lines A12-A15 are remapped by gate circuits 206 to develop cache address bits A12'-A15' in accordance with the following table:

EMEMSE	16K	32K	A12'	A13'	A14'	A15'
0	X	X	A12	A13	A14	A15
1	1	0	A12	A13	A14	A15
1	0	1	A14	A12	A13	A15
1	0	0	A14	A15	A12	A13

It will be observed that the CPU address bits are not remapped if EMS is disabled. Likewise, remapping is not performed in the case of a 16K cache since this is equal in size to the EMS page and therefore requires that the entire cache be flushed when the EMS page is remapped.

It will be recognized that the above described invention may be embodied in other specific forms without departing from the essential characteristics of the disclosure. Thus, it is understood that the invention is not to be limited by the foregoing illustrative details, but rather is to be defined by the appended claims.

CLAIMS

1. An improved cache controller for use in a microprocessor system having a set associative cache and a page mapped memory, wherein the improvement comprises address mapping means for remapping at least one cache address bit such that all addresses within a single page of memory are mapped into a block of said cache comprising less than the entirety of said cache.
2. The device of claim 1 wherein said address mapping means interchanges at least one cache address bit and at least one cache tag bit.
3. The device of claim 1 wherein each page of memory comprises 16K bytes, said cache comprises a 64K byte four-way associative cache, and all addresses within a single page of memory are mapped into a block of said cache comprising 16K bytes.
4. The device of claim 3 wherein a memory address is specified by address bits A1 to AN inclusive, where N is greater than 15 and A1 is the least significant bit, a cache address is specified by address bits A1 to A13 inclusive, and said address mapping means remaps address bits A12 to A15 according to:



15

A12 in place of A14

A13 in place of A15

A14 in place of A12

A15 in place of A13.

5. The device of claim 1 wherein each page of memory comprises 16K bytes, said cache comprises a 32K byte four-way associative cache, and all addresses within a single page of memory are mapped into a block of said cache comprising 16K bytes.

6. The device of claim 5 wherein a memory address is specified by address bits A1 to AN inclusive, where N is greater than 15 and A1 is the least significant bit, a cache address is specified by address bits A1 to A13 inclusive, and said address mapping means remaps address bits A12 to A14 according to:

A12 in place of A13

A13 in place of A14

A14 in place of A12.

7. In a microprocessor system having a set associative cache and a page mapped memory, a method of generating cache addresses comprising the step of remapping at least one cache address bit such that all addresses within a single page of



memory are mapped into a block of said cache comprising less than the entirety of said cache.

8. The method of claim 7 wherein the step of remapping comprises interchanging at least one cache address bit and at least one cache tag bit.

9. The method of claim 7 wherein each page of memory comprises 16K bytes, said cache comprises a 64K byte four-way associative cache, and all addresses within a single page of memory are mapped into a block of said cache comprising 16K bytes.

10. The method of claim 9 wherein a memory address is specified by address bits A1 to AN inclusive, where N is greater than 15 and A1 is the least significant bit, a cache address is specified by address bits A1 to A13 inclusive, and the step of remapping comprises remapping address bits A12 to A15 according to:

A12 in place of A14

A13 in place of A15

A14 in place of A12

A15 in place of A13.



11. The method of claim 7 wherein each page of memory comprises 16K bytes, said cache comprises a 32K byte four-way associative cache, and all addresses within a single page of memory are mapped into a block of said cache comprising 16K bytes.

12. The method of claim 11 wherein a memory address is specified by address bits A1 to AN inclusive, where N is greater than 15 and A1 is the least significant bit, a cache address is specified by address bits A1 to A13 inclusive, and the step of remapping comprises remapping address bits A12 to A15 according to:

A12 in place of A14
A13 in place of A15
A14 in place of A12
A15 in place of A13.



* C L M S *