

ZPOS

MEMORY CONTROLLER WITH QUEUE AND SNOOP TABLES

Technical Field:

This invention relates to a memory controller for snooping in multiple processor systems and more particularly to a memory controller including queue and snoop tables for efficiently snooping which reduces the number of snoop cycles and effectively increases the available processor bus bandwidth.

Background of the Invention:

The limiting performance factor in most multiple processor systems is processor bus bandwidth. Since most multiple processor systems use processor caches, a significant percentage of the processor bus bandwidth is consumed in performing snoops of these caches during I/O operations. These snoop operations have a negative effect on overall system performance since they require a significant portion of the processor bus bandwidth.

Snoops are used during I/O operation to determine by means of a controller if the most recent copy in memory of a data item also resides in a processor cache. Systems which snoop, do so on a processor cache-line basis. To snoop a 4K byte page, in a system which has a 16 byte cache line, requires 256 snoop cycles. In systems that snoop, it is also required that the I/O and processor buses be synchronized for each of the snoop cycles. Regarding the mechanism referred to as synchronization. In the example of a computer system with two busses, i.e., the processor bus and the I/O bus, if there is no transfer of information from one bus to the other bus then these busses can run independently of each other. However, if and when information is to be transferred from one bus to the other bus, then a mechanism must be created to allow this transfer and this can be done through synchronization. Synchronization can be implemented a number of ways and the following are two which are typical.

In a first example the busses are run in lock step. This allows the transfer of information to occur at anytime. There are a number of disadvantages to the lock step design. The key being that as processor busses become faster, due to improvement in processor technology, this improvement may not be implemented or taken advantage of because of the requirement that the processor bus be synchronized (run at the same speed or at a multiple (2X, 3X, etc.) speed) to the I/O bus. A second typical example is

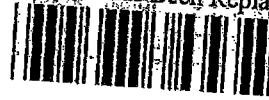
of a latch interface. This is a storage device that is placed between the two busses. When data is to be transferred, one bus places the data into the latch and signals the other bus. The other bus then can access the storage device and signals the first bus that it has accessed the information. This need for synchronization can detrimentally lengthen the time required for the snoops, thereby increasing the load that snooping places on the multiple processor bus bandwidth.

One possible way of eliminating the snoop cycles on the processor bus would be to use processor caches which are store-thru. Unfortunately any bandwidth saved by eliminating the snoops, would be more than lost by the increase write-to-memory traffic. Therefore such a solution is not practicable and is not readily useful for multiple processor designs.

Store-thru, also called write-through strategy, provides for all memory writes from the microprocessor to be passed along immediately by a cache controller to also update the main system memory. The result is that the main system memory always contains valid data. Any location in the cache can be overwritten, i.e., updated, immediately without data loss. Further discussion of other related cache operations, upon which the specification relies, can be found in a booklet entitled "Cache Tutorial" available from Intel Corporation, Literature Sales, Mt. Prospect, IL. The booklet is dated 1991 and the order number is 296543-002.

It would be advantageous to provide for snooping in a more efficient manner particularly for multiple processor systems without hindering access to the bus and unduly limiting bus bandwidth.

This Sheet Has Been Replaced



Summary of the Invention:

This invention introduces a memory controller which in conjunction with a system architecture supports the concept of block snoops to efficiently ascertain the need for snooping. Block snoops make the cache snooping operations more efficient by reducing the amount of traffic on the multiple processor bus. This is accomplished in part by incorporation of request queue and snoop tables in the memory controller. Not only is the performance improved, but also an increased the number of processors can be supported at a reasonable level of performance, with a given bus design.

These significant features of the memory controller are employed in combination with a bus interface unit or BIU which attaches between the caches of the processors and the multiple processor bus, and which includes two block snoop control registers. In a contemplated embodiment there is a BIU for each processor/cache subsystem; two unique signals which interface between the BIUs and the system memory controller; and enhanced function within the memory controller to support not only I/O devices which support the block snooping function, but also other I/O devices which do not support the new block snooping function, which can be characterized as more vintage type I/O devices. This combination effectively uses the snoop table within the system memory controller to indicate which address of the physical memory can be safely transferred to I/O without the need for snooping during I/O operations.

This innovation introduces an architecture and there is a software impact to its implementation. The software impact is minimal and is containable to just a set of device drivers and any busmaster I/O adapters which are set to implement this function. The contemplated architecture is also backward compatible and busmaster devices which do not use or anticipate this function can still be used within a system which incorporates the invention.

The BIU which exists for each of the processor complexes, which complexes are connected to the multiple processor bus, is prepared with the following functions.

There exists an interface to the processor complex, a start snoop address register, a block size register, a control logic to manage and control the block snooping function, including the two unique signals and also an interface to the multiple processor bus.

The aspects of the BIU allows the multiple processor bus to be processor independent. Therefore the concepts of the architecture of this invention can be applied to both x86 (Intel Corporation series processors and their compatibles) as well as RISC (reduced instruction set computing) processors. The BIUs interface to the memory controller through the multiple processor bus and the two unique signals. These two signals comprise a signal from the BIU to the memory controller to indicate that the block snoop for this processor complex is completed, and a signal from the memory controller to all the BIUs indicating that each BIU should reset its respective start snoop address register and block size register.

In the present invention, when an I/O device begins an I/O operation, it notifies the memory controller of the I/O starting address. The information is brought into the snoop request queue. A request at the top is compared to snoop table. If it is noted that the address of the requested memory is listed in the snoop table, than that area of main memory need not be snooped. When listed, the appropriate adapter provides for the execution of the I/O operation. When there is no present listing, the area of memory must be snooped and the memory controller executes the snoop operation.

In addition, the present innovation is operational without a need to synchronize the busses. The present invention provides that when a block of data is to be moved between memory and an I/O device, wherein the address and number of bytes to be transferred are known, the system broadcasts on the processor bus this information to the processors, including the address and number of bytes. At each processor is provided a BIU which will independently generate a sequence of addresses causing the

cache within the processor to be snooped. Depending on the result of each snoop operation, the caches will do one of three things. Under conditions where there is no match, the cache does nothing. Otherwise, where there is a match and the cache was not modified since the last time it received it, the cache invalidates the data in the cache. To invalidate is to mark the data in the cache and indicate that if this data is needed the processor will have to go to memory and get the data in its most current state. The third possibility is that there is a match, but that however, the cache was modified since the time it was received. In this instance, the cache will write the data out to memory, since this data is the most up-to-date copy of data, and then invalidate the data in the cache. Once this operation is completed, the BIU signals that the block has been snooped. Once all the BIUs report and the total process is completed, the system allows the I/O transfer between the memory and the I/O to occur.

Note that since the actual I/O transfer occurs after the snoop operation on the processor bus, there is no need to synchronize the two busses and in this manner a significant feature of the present inventive contribution is achieved, i.e., nonsynchronous block snooping. Synchronization would be needed for operations where the busses are working concurrently. In the contemplated applicability of the present invention, the busses, do not work concurrently but rather, first the processor bus is active then the I/O bus is active.

Brief Description of the Drawings

The invention is pointed out with particularity in the appended claims. The stated advantages of the invention may be better understood by referring to the following detailed description in the specification in conjunction with the drawings in which:

FIG. 1 is a schematic diagram of the bus interface unit subsystem according to the principles of the present invention;

FIG. 2 is a schematic block diagram of a multiple processor system depicting the logical system structure and the control signal paths according to the principles of the present invention;

FIG. 3 is a bus signal timing diagram of control signals during a block snoop operation according to the principles of the present invention;

FIG. 4 illustrates the logical structure of the memory controller according to the principles of the present invention;

FIG. 5 illustrates a block diagram of the logic functions of the memory/I/O controller according to the principles of the present invention; and

FIG. 6 illustrates a typical I/O request queue element.

Detailed Description of the Preferred Embodiment(s)

FIG. 1 shows a general overview of a bus interface unit (BIU) 10, which includes the features of the block start address register 12 and the block size register 14. These registers 12 and 14 are used in the performance of the block snoop operation. The meaning of block, when used in the context of storage and block snooping, refers to a collection of consecutive bytes of data that, as a group, are read-from or written-to storage. When a block snoop operation is to be performed, the block start address register 12 is loaded with the starting address of the area of physical storage that is to be snooped. The block size register 14 is loaded with the number of bytes of storage which are to be snooped.

In a preferred implementation of the memory controller, each loading of the block size register 14 automatically causes the BIU, as typically illustrated by BIU 10, to start the block snoop operation. The respective BIU 10 will issue snoops to each respective

cache starting at the address contained in its block start address register 12. The BIU 10 will increment the block start address register 12 and decrement the block size register 14 by the number of bytes contained in a processor cache line. (i.e., the BIU 10 will issue one snoop per cache line.) The block snoop of the BIU 10 is completed when the contents of the block size register 14 is zero. If during the block snoop, a cache castout occurs the BIU 10 will send the castout to the memory controller 43 on multiple processor bus 48.

Cache castout is existent and contemplated in the following manner. When a cache is a store-in (write-back), the cache may contain the most up-to-date copy of data. When the processor writes to a memory location and that location is found in the cache, the copy in the cache is updated but the update, at the time the write occurs, is not written out to memory. This creates a recognized condition referred to as dirty data. It is expected that there will come a time when the dirty data will be written out to memory. This can be due to two events. In the first event: since a cache is a mapping of memory, there is the possibility that the cache will want to use the location in cache to store a different address, and in this event, since the cache has the most up to date copy of the current data, the cache will write the data out to memory.

In the second event, a snoop of this address occurs. When this address is snooped it is discoverable that this cache has the most up to date copy of the data. Again the cache will write the data out to memory. The occurrence of the writing out of data, is referred to as a cache castout.

The BIU 10 also supports the two mentioned unique signals. The first signal is used by the BIU 10 to indicate to the memory/IO controller subsystem 43, as seen in FIG. 2, that the typical BIU 10 in FIG. 1 and each of the exemplary BIUs 38, 40, 42 and 44 in FIG. 2, has completed a block snoop. The second signal 21, or as seen in FIG. 2 as line 50, from the memory controller 43, is used by the memory/IO unit 43 to tell the respective exemplary BIU 10 to reset its block snoop complete signal. This

architecture operational within the controller 43 has the added benefit of eliminating the need for the processor bus 48 and I/O buss 46 to be synchronized for the snoop operations.

FIG. 2 illustrates an exemplary system layout of the system with multiple exemplary processors 22, 24, 26 and 28, multiple exemplary cache units, 30, 32, 34 and 36 and the corresponding multiple BIUs 38, 40, 42 and 44, according to the principles of this invention. The memory/I/O subsystem 43 may be composed of multiple memory and IO subsystems. However, this is transparent to the user. The memory/I/O subsystem 43 determines whether a request from any of the processors 22, 24, 26 or 28 is to physical storage (cachable memory 30, 32, 34 or 36) or is directed to an, exemplary, I/O device 45 on the I/O bus 46. It is obvious that a plurality of I/O devices can be present.

This implementation supports only one outstanding block snoop at a time, i.e., a block snoop must be completed before another block snoop can be initiated. This implementation is chosen for its simplification. It should be appreciated that alternative implementations which support multiple outstanding block snoop operations are not precluded.

Since there are multiple I/O devices contemplated, is it possible that there can be multiple I/O operations that occur concurrently. Using the present invention, as the requests to perform I/O are received they are queued up in the request queue. As the requests are serviced, the snoop tables, as illustrated in **FIG. 4**, are checked to see if a snoop is required. If the snoop is needed, that snoop operation is executed. Otherwise the I/O operation is simply allowed to execute. As described herein, the I/O operation involves the device I/O bus and the memory. Since the snoop is completed before the I/O transfer, the processor bus and I/O do not need to be synchronized, as is noted herein.

FIG. 3 illustrates the timing of the control signals during a block snoop operation. At some time T_0 , on the X axis of **FIG. 3**, a block snoop is initiated. The block address and block size are broadcasted to all BIUs 10 and the multiple processor bus 48 is released. The BIUs 10 will then each independently issue a series of snoop cycles to their associated processors and caches, as best seen in **FIG. 2**. If any cache castouts are generated they are sent to the memory controller 43 on multi processor bus 48. When all signals are set, as shown by the elevated signal level on each of the exemplary P1, P2, P3 and P4 lines corresponding to the output terminals in **FIG. 2**, the AND unit 49 of **FIG. 2** goes high and the memory controller 43 knows that the snooping operation is complete and the memory controller 43 pulses the reset (RST) line 50, from the signal on line 21, which causes all the exemplary BIUs 38, 40, 42, and 44 to reset their signals to the memory controller 43. This reset is illustrated by the lowering of the signal level on each of the P1, P2, P3 and P4 lines at the same time on the X axis of **FIG. 3**. Once reset is issued the system is ready for another block snoop operation. Any time differential between the pulse of the reset signal and the reset of each P line is typically due to propagation delay.

FIG. 4 illustrates the logical structure of the memory controller 43. This memory controller 43 illustrates unique functions and includes a request queue 52 and snoop tables 54. The snoop tables 54 indicate which addresses can be safely transferred to/from an I/O device, such as I/O device 45, without snooping while the transfer is occurring. The snoop table 54 contains an entry for each 4K byte page of cacheable memory. This entry is a single bit. For a system with 64 Mbytes of physical storage, a snoop table of 2K bytes is required. When a block snoop operation is completed for a given page, the snoop table 54 marks the page as snooped. Whenever one of the processors 22, 24, 26 or 28 does a read operation which goes to memory 56, the page of cacheable memory on which that data is contained is marked as not-snooped in the snoop table 54.

When an I/O device 45 wishes to issue a I/O operation it passes the I/O starting address and the number of bytes to transfer, to the memory control unit 43. This information is placed in the snoop request queue 52. When the request is at the top of the queue 52 it is compared against the snoop table 54. If the address of the requested memory is listed in the snoop table 54 then that area of memory 56 does not need to be snooped. If the area is listed in the snoop table 54, the I/O device 45 is informed that it is free to execute its I/O operation. The I/O operation is also free to execute at the rated speed of the I/O bus 46. If on the other hand the address of the requested memory is not listed in the snoop tables 54, then that area must be snooped. The memory control unit 43 then executes a block snoop operation as described.

As each of the BIU complexes completes its snoop operation, it sets its signal, as shown in FIG. 3. When all have completed their snoops the memory controller 43 tells all the BIUs 38, 40, 42 and 44 to reset. At this time the I/O device 45 is informed that it is free to execute its I/O operation. The I/O operation is also free to execute at the rated speed of the I/O bus.

Further, with regard to system initialization of the snoop tables, the following provides additional explanation of the procedures contemplated and followed in the present inventive method. The snoop tables can be viewed, or considered, as a map of main memory. This map indicates that if an I/O operation occurs to a specific address in main memory then that address may not need to be snooped for a number of reasons. Possibly the area is non-cacheable memory. Normally the system allow parts of memory to be defined as non-cacheable. In this case, when a processor accesses this address, it will not place the data in caches and since the data is never in the cache it does not have to be snooped. Another alternative, is that the area has been snooped. This is set when the present invention is used. All other areas of memory are marked as, needs-to-be snooped. When the system is first initialized, this table is set. Defining non-cacheable memory is currently part of system bring up. All other memory (cacheable) is marked as needs-to-be snooped.

The following is the sequence of operations that occur:

/* The follow are the operations executed by the memory controller */

Memory controller selects requests from queue

Memory controller compares the address against the snoop table

IF required area of storage is snooped

THEN signal I/O device to perform I/O transfer operation

ELSE /* Required area of storage has not been snooped */

Do

Perform a block snoop of required area of storage

**When snoop operation completed then signal I/O device to
perform I/O transfer operation**

END

**/* NOTE WHILE THE I/O TRANSFER OPERATION is going on, the
memory controller can snoop another block */**

/* The follow are the operations executed by the I/O adapter */

I/O device issues a request to execute a block I/O operation.

**I/O device passes the start address and number of bytes to memory
controller**

Request is placed into request queue

While request not granted wait

When request granted perform I/O TRANSFER OPERATION

In FIG. 5 is a detail description of how the supporting mechanism for the snoop table 54 operates. The figure illustrates a block diagram of the logic functions of the memory/I/O controller. A processor BIU 60 interfaces the memory/I/O subsystem 43 to the processor bus 48. Similarly an IO/BIU 62 interfaces the memory/I/O subsystem 43 to the I/O bus 46. On the other side of FIG. 5 a DRAM interface 69 interfaces the

memory/I/O subsystem 43 to the DRAM arrays 56. Memory controller 66 contains the necessary logic to carry out the functions. A local storage 68, has data storage addressable only to the memory controller 66. The local storage 66 is used to store control information and the following data structures.

In the illustrated snoop table 54 in FIG. 5, there is anticipated a change in the granularity of the snoop table 54 from that presented earlier herein with regard to FIG. 4 where the snoop table entry represents a 4 Kbyte page. For the present illustration a snoop table entry represents a processor cache line (typical size is 32 bytes). This change is simply to make the block snooping mechanism more efficient in the case of block snoops which are less than 4 Kbytes. The snoop table 54 is an addressable storage array. The snoop table 54 contains an entry for each cache line (32 bytes) of addressable memory. An entry is a single bit. This bit indicates whether or not this 32 bytes portion of addressable memory needs to be snooped. The I/O REQUEST QUEUE 52 is a logical data structure which can be implemented as an addressable storage array, or as a linked list within a local storage of the memory/I/O controller.

Whenever a device wishes to execute an I/O operation to memory, the device first communicates with the I/O controller. An I/O request is constructed by the I/O controller and this request is placed into the I/O queue. This request is referred to as an I/O queue element, the format of an I/O queue element is shown in FIG. 6.

The following is a pseudo-code explanation of the operation of the memory controller 66 for the servicing of the I/O request queue 52.

```

WHILE I/O_request_queue NOT EMPTY
  GET_NEXT_QUEUE_ELEMENT /* take the next request off the queue */
  transfer_request = queue_element
  WHILE queue_element.count NOT zero /* Has the whole request be
                                     checked? */
    If (CHECK_SNOOP_TABLE(queue_element.address) = 1) THEN
      /* Get the bit in the snoop table that corresponds to the
         cache line that contain the address referenced in the
         queue_element. If the bit in the snoop table is set
         then this cache line must be snooped. If the bit is 0
         then this cache line does not need to be snooped)
      */
      DO
        SET_FOUND = TRUE
        BREAK /* BREAK OUT OF INNER WHILE LOOP */
      END
      DECREMENT (queue_element.count)
      IF (queue_element.count NOT ZERO) THEN
        DECREMENT(queue_element.address)
      END WHILE
      IF (SET_FOUND = TRUE)
        ISSUE_SNOOP(transfer_request.address, transfer_request.count)
        RESET_SNOOP_TABLE(transfer_request.address Xfer_request.count)
        PERFORM_TRANSFER (transfer_request)
      END WHILE

```

As is shown, this operation resets corresponding snoop table entries. The snoop table 54 is set when a processor performs a memory read operation to cacheable memory. When a processor performs a memory read operation 54 to cacheable memory the corresponding bit in the snoop table will be set. If a processor performs a read to non-cacheable memory, the bit in the snoop table is not set since the data will not be cached in the processor and therefore does not need to be snooped.

For I/O devices 45 which do not support the use of the block snoop architecture, the system will simply handle the snoops on a cache line basis, as systems currently do. The system will also force the processor bus 48 and I/O busses, represented by exemplary I/O bus 46, to synchronize during the snoop operations. Therefore the system can support a mix of I/O adapters which both anticipate and do not anticipate the present invention.

Although the present invention and its advantages have been described in detail, it should be understood that various changes, substitutions and alterations can be made herein without departing from the spirit and scope of the invention as defined by the appended claims.



Claims:

16

1. A memory controller for a computer system which includes a memory, a multiprocessor bus, an I/O bus for coupling to I/O devices and a complex of a multitude of independent microprocessing units, a multitude of cache storage units, each cache storage unit of the multitude of cache storage units coupled to one of the multitude of microprocessing units and a multitude of bus interface units (BIUs), each BIU of the multitude of BIUs coupling each respective microprocessing unit through each respective cache storage unit to the multiprocessor bus, and wherein the memory controller is connectable to the multiprocessor bus and the I/O bus for processing requests from any one of the I/O devices for determining which addresses need to be snooped and for implementing a cache storage unit block snoop operation when a determination is made that an address needs to be snooped and, alternatively, an I/O operation when a determination is made that the address does not need to be snooped, comprising:

a memory port for communicating with the memory of the computer system;

an I/O bus port for receiving the processing request from the any one of the I/O devices;

a request queue table for listing the received processing request from the any one of the I/O devices;

a snoop table for listing addresses of the memory;

comparing means for determining if the listing in the request queue table is listed in the snoop table; and

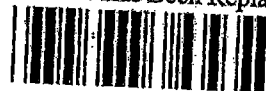
operational means for signaling for the operation of the processing request providing the listing in the request queue is in the snoop table, otherwise for signaling the commencement of a snoop operation.





2. The memory controller as defined in Claim 1, wherein the comparing means compares the listing at the top of the list of the request queue table with the listings in the snoop table.
3. The memory controller as defined in Claim 1, wherein the memory controller receives respective signals from each of the multitude of BIUs in the complex after a block snooping operation has been completed in each corresponding cache of the complex of microprocessors, cache storage units and BIUs and sends a reset signal to all of the BIUs for resetting of registers in each BIU.
4. A method of memory controller operation for a memory controller in a computer system including a multitude of independent microprocessor units, a multiprocessor bus, a multitude of cache storage units, each cache storage unit of the multitude of cache storage units coupled to one of the multitude of microprocessing units, a multitude of bus interface units (BIUs), each BIU of the multitude of BIUs coupling each respective microprocessing unit through each respective cache storage unit to the multiprocessor bus, an I/O bus for coupling to I/O devices and a memory controller having a queue table and a snoop table, and for coupling to the multiprocessor bus and the I/O bus for processing I/O requests from any one of the multitude of independent microprocessing units, comprising the steps of:
 - receiving an I/O request;
 - listing the I/O request in the queue table in the memory controller;
 - comparing the I/O request in the queue table with a listing in the snoop table;
 - signaling for the operation of the I/O processing request providing the listing in the queue table is in the snoop table, otherwise for signaling the commencement of a snoop operation.





5. A computer system including a memory, a multiprocessor bus, an I/O bus for coupling to I/O devices, in addition to a complex of a multitude of independent microprocessing units, a multitude of cache storage units, each cache storage unit of the multitude of cache storage units coupled to one of the multitude of microprocessing units and a multitude of bus interface units (BIUs), each BIU of the multitude of BIUs coupling each respective microprocessing unit through each respective cache storage unit to the multiprocessor bus, and a memory controller connected to the multiprocessor bus and the I/O bus for processing requests from any one of the I/O devices for determining which addresses need to be snooped and for implementing a cache storage unit block snoop operation when a determination is made that an address needs to be snooped and, alternatively, an I/O operation when a determination is made that the address does not need to be snooped, comprising:

a memory port for communicating with the memory of the computer system;

an I/O bus port for receiving the processing request from the any one of the I/O devices;

a memory controller local storage for storing control information including a request queue table for listing the received processing request from the any one of the I/O devices and a snoop table for listing addresses of the memory;

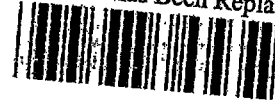
comparing means for determining if the listing in the request queue table is listed in the snoop table; and

operational means for signaling for the operation of the processing request providing the listing in the request queue is in the snoop table, otherwise for signaling the commencement of a snoop operation.



* C L M S *

This Sheet Has Been Replaced



6. The computer system as defined in Claim 5, further including an IO/BIU for processing the processing request from any one of the I/O devices, a memory interface for communicating with the memory of the system and a processor BIU for interfacing with the processor bus.
7. The computer system as defined in Claim 5, wherein the snoop table represents processor cache lines.
8. The computer system as defined in Claim 5, wherein the snoop table is an addressable storage array and the snoop table contains an entry for each cache line of addressable memory.
9. The computer system as defined in Claim 5, wherein the I/O request queue is a logical data structure implemented as addressable storage array.
10. The computer system as defined in Claim 5, wherein the I/O request queue is a logical data structure implemented as a linked list within the memory controller local storage.
11. The computer system as defined in Claim 6, wherein the processing request from any one of the I/O devices is constructed by the IO/BIU as an I/O queue element and is placed into the I/O request queue table.

