

BACKGROUND OF THE INVENTIONA SYSTEM LEVEL MECHANISM FOR INVALIDATING DATA STORED IN THE EXTERNAL CACHE OF A PROCESSOR IN A COMPUTER SYSTEM1. FIELD OF THE INVENTION:

5 The present invention pertains to the field of computer systems. More particularly, this invention relates to a computer system that includes a system level mechanism for invalidating data stored in the external cache of a processor in the computer system.

10 2. BACKGROUND:

 Prior computer systems typically include one or more processor subsystems and one or more input/output subsystems coupled to a main memory subsystem. Such a main memory subsystem typically provides storage areas for instructions and data for the processors, as well as storage areas for data transfer to and from input/output devices coupled to the input/output subsystem. Typically, the processor subsystems and the input/output subsystems are coupled to the main memory subsystem via a physical interconnect that provides a common bus or a switched interconnect data path.

20

 A processor subsystem in such a system typically includes an associated external cache memory. Such an external cache memory usually stores selected blocks from the main memory subsystem. An external cache memory usually provides fast processor access to code and data by avoiding external fetches to the main memory subsystem for datablocks that are stored in the cache. In addition, such a computer system typically implements a protocol for maintaining data coherency between the main memory and the corresponding datablocks in the external cache memory. For example, each external cache

memory typically includes a set of tags that indicate the validity status of the datablocks stored in the external cache.

Typically, such a computer system provides a mechanism for performing
5 coherent data input/output. For example, an input/output subsystem in such a computer system commonly injects new data into the system by issuing write invalidate (WRI) operations to the main memory subsystem. Such write invalidate operations may occur, for example, when a new page of data is read from a disk device coupled to the input/output subsystem.

10

In addition, a processor subsystem commonly injects new data into the computer system using WRI operations during block copy operations. Such block copy operations occur, for example, during data structure initialization or during file copy operations in the computer system. Such block copy operations
15 occur at a relatively high rate in a typical prior computer system.

Such WRI operations usually require the invalidation of corresponding data blocks stored in an external processor caches to maintain data coherency in the system. Typically, such an invalidate operation on an external cache is only
20 necessary when the tags for that external cache indicate that the newly injected datablock is present.

In some prior computer systems, such an invalidation operation is performed within each processor subsystem. In such a system, a processor reads
25 the corresponding external tags to determine whether an invalidate operation is required during a write invalidate transaction. However, such a mechanism requires that each processor perform a tag read operation and potentially a tag read-modify-write operation for each WRI operation issued to the memory

subsystem. Such WRI operations occur at a frequent rate in typical prior systems and interfere with normal instruction execution by the processors, thereby slowing the overall performance of such a system. Moreover, processor subsystems that perform such self-invalidation operations may be incompatible
5 with systems that provide directory based tags in the memory subsystem.

Some computer systems may implement a worst case approach to such invalidate operations. For example, a system may blindly perform an external cache invalidate operation for every write invalidate transaction whether or not
10 the datablock specified by the write invalidate transaction is present in the external cache. Such an invalidate operation is unnecessary whenever the specified datablock is not stored in the external cache. Unfortunately, such unnecessary invalidate operations interfere with normal operations of the processor.

15

In addition, such invalidate operations may be required during some types block copy operations while not during other types of block copy operations. For example, a processor subsystem may require an invalidate operation during block copy operations that generate new memory datablocks
20 that the processor subsystem will not later access. On the other hand, block copy operations that write datablocks to stable storage in the computer system typically do not require an invalidate operation to the external cache if the processor retains control over the datablocks transferred to stable storage. Prior systems that blindly perform invalidate operations usually cause the processor
25 subsystem to later perform an external fetch to the memory subsystem for the specified datablocks. Unfortunately, such unnecessary external fetches to the memory subsystem slows the overall throughput of such a computer system.

SUMMARY OF THE INVENTION

A computer system is disclosed including a memory subsystem and a processor subsystem having an external cache wherein the memory subsystem
5 includes a mechanism for invalidating datablocks in the external cache. The computer system also includes a mechanism for reducing false invalidation operations by providing hints to the memory subsystem of the state of data in the external cache.

10 The processor subsystem issues a write invalidate message to the memory subsystem that specifies a datablock and that includes an invalidate advisory indication. The invalidate advisory indication indicates whether the datablock is present in the external cache. The invalidate advisory indication determines whether the memory subsystem transfers an invalidate message to
15 the processor subsystem that invalidates the datablock in the external cache.

The mechanism in the memory subsystem for invalidating the external cache of the processor simplifies the design of the processor to external cache interface and reduces the impact on processor performance of write invalidate
20 operations in the system. Such an external mechanism for cache invalidation also eliminates read-modify-write operations by the processor to the external cache during write invalidate operations that would otherwise impact processor performance. In addition, such an external mechanism for cache invalidation provides a processor subsystem that is compatible with multiprocessor systems
25 that maintain directory based or duplicate tags and with uniprocessor systems that do not maintain directory based or duplicate tags.

Other features and advantages of the present invention will be apparent from the accompanying drawings, and from the detailed description that follows below.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate
5 similar elements, and in which:

Figure 1 illustrates a computer system 10 for one embodiment including a processor subsystem, an input/output port, and a memory interconnect subsystem;
10

Figure 2 illustrates a processor subsystem which includes a processor, an external cache and associated set of external tags, and a memory interconnect port interface;

15 Figure 3 illustrates the input/output port for one embodiment including a standard input/output bus interface and a port interface that enables coherent input/output direct memory access to the main memory;

20 Figure 4 illustrates one embodiment of a request packet from one of the memory ports Port 1 - Port N to the system controller;

Figure 5 illustrates the handling of a block store operation by the processor subsystem for one embodiment.

DETAILED DESCRIPTION

Figure 1 illustrates a computer system 10 for one embodiment. The computer system 10 comprises a processor subsystem 20, and an input/output port 22, and a memory interconnect subsystem 18. The memory interconnect subsystem 18 provides shared access to a main memory 14 to the processor subsystem 20 and the input/output subsystem 22. Elements of the computer system 10 communicate with the memory interconnect subsystem 18 via a set of memory ports 1-N. For example, port 1 for the processor subsystem 20 comprises a data path 40, an address path 42, and a control path 44, and port N for the input/output subsystem 22 comprises a data path 46, an address path 48 and a control path 50.

The memory interconnect subsystem 18 includes a data path switch 12 to the main memory 14, and a system controller 16. For one embodiment, the data path switch 12 is a cross bar switch controlled by the system controller 16 via a set of control signals 54. The data path switch 12 enables data path access to the main memory 14 via the data path of each port 1-N including the data path 40 and the data path 46. The system controller 16 functions as a memory controller by decoding the addresses received from the processor subsystems 20-22 and by generating memory control signals including row address select (RAS) and column address select (CAS) signals to the main memory 14 via a memory control bus 56.

The system controller 16 receives memory access requests from the subsystems 20-22 via the address path of each port 1-N including the address path 42 and the address path 48. The system controller 16 arbitrates among the requests from the subsystems 20-22 and configures the data path switch 12 to

provide the necessary connections for access to the main memory 14 via a memory data bus 52 or for subsystem to subsystem data transfers. The system controller 16 responds with cache state transition control messages to the processor subsystem 20 when necessary.

5

The memory access requests include, for example, write invalidate (WRI) requests. The subsystems 20-22 use a WRI request to the system controller 16 to write a specified datablock into the main memory 14. The system controller 16 processes a WRI request by configuring the data path switch 12 to provide the proper source for the specified datablock. If the WRI request originates with the processor subsystem 20, then the system controller 16 transfers messages via the control path 44 to control the cache state transitions in the external cache of the processor subsystem 20. For example, the system controller 16 transfers invalidate messages via the control path 44 to invalidate datablocks in the external cache of the processor subsystem 20 when appropriate:

In another embodiment, the system controller 16 maintains a set of duplicate tags that correspond to the external tags. In such a multiprocessor embodiment, the system controller 16 accesses the duplicate tags via a snoop bus coupled between the system controller 16 and the duplicate tags. For an illustration of such an embodiment, refer to application serial no._____, filed on March 31, 1995, and entitled **Packet Switched Cache Coherent Multiprocessor Computer System**.

The input/output port 22 enables communication with a set of peripheral devices 30-33 via a standardized input/output bus 58. The standardized input/output bus 58 may comprise, for example, a SBUS or a peripheral component interconnect (PCI) standardized bus. The input/output

port 22 implements a direct memory access mechanism that provides coherent input/output operations to the main memory 14 through the system controller 16.

5 **Figure 2** illustrates the processor subsystem 20 for one embodiment. The processor subsystem 20 comprises a processor 70, an external cache 74, and an associated set of external tags 76 along with a port interface 72.

10 The processor 70 issues memory load and memory store operations to the external cache 74. The port interface 72 issues a cache requests to the system controller 16 using a message packet transferred via the address pass 42. The cache requests include WRI transactions during block copy operations by the processor 70. Each message packet transferred via the address pass 42 includes a header that specifies the write invalidate request type along with an address that
15 specifies the datablock for the write invalidate transaction. The system controller 16 processes a write invalidate transaction by configuring the datapath switch 12 to transfer the corresponding datablock to the main memory 14.

20 For one embodiment, the WRI request packet to the system controller 16 includes an invalidate advisory (IVA) indication. The IVA indication determines whether the system controller 16 issues an invalidate request back to the processor subsystem 20 to invalidate the corresponding datablock in the external cache 74. In addition, the processor subsystem 20 performs a copyback
25 operation to the memory interconnect subsystem 18 in response to the invalidate request if the specified invalidated datablock is present in the external cache 74.

Figure 3 illustrates the input/output port 22 for one embodiment. The input/output port 22 comprises a standard input/output bus interface 80 along with a port interface 82. The port interface 82 enables coherent input/output direct memory access to the main memory 14 through the system controller 16.

5

Figure 4 illustrates one embodiment of a request packet from one of the memory ports Port 1 - Port N to the system controller 16. For this embodiment, the request packet is transferred in two cycles via the appropriate memory path to the system controller 16. The physical address of the request packet specifies a datablock of the main memory 14 for the request packet. During a WRI transaction from the processor subsystem 20, the transaction type field of the first cycle of the request packet indicates the write invalidate transaction type. The IVA bit indicates whether the processor subsystem 20 requires an invalidate request for the corresponding write invalidate transaction to invalidate the datablock in the external cache 74 and tags 76.

15

Figure 5 illustrates a block store operation by the processor subsystem 20 for one embodiment. The block store operation may occur, for example, during a block copy by the processor subsystem 20. At block 100, the processor 70 issues a block store operation to the external cache 74.

20

At block 102 the port interface 72 generates a write invalidate request packet that specifies the datablock of the block store operation from the processor 70. At decision block 104, the processor 70 determines whether the specified datablock is being retained. If the specified datablock is not being retained by the processor 70 at decision block 104, then control proceeds to block 106.

25

At block 106, the processor 70 clears the IVA bit in the write invalidate request packet assembled in the port interface 72. Thereafter, at block 108 the port interface 72 issues the write invalidate request to the system controller 16 via the address path 42.

5

If the specified datablock is being retained by the processor 70 at decision block 104, then control proceeds to decision block 110. At decision block 110, the processor 70 determines whether the specified datablock is present in the external cache 74 by performing a tag look-up operation to the external tags 76.

10 If the specified datablock is not present in the external cache 74, then control proceeds to block 108 to issue the write invalidate request to the system controller 16. Otherwise control proceeds to block 112.

At block 112 the processor 70 sets the IVA bit in the write invalidate
15 request packet to indicate that a write invalidate request for the datablock is required from the system controller 16. Thereafter, control proceeds to block 108 to issue the write invalidate request packet to the system controller 16.

In the foregoing specification the invention has been described with
20 reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. The specification and drawings are accordingly to be regarded as illustrative rather than a restrictive sense.



12

THIS SHEET HAS BEEN
REPLACED

CLAIMS

This Sheet Has Been Replaced



What is claimed is:

1. A computer system comprising a processor subsystem having an external
5 cache and further comprising a memory subsystem with a mechanism that
invalidates a datablock in the external cache.
2. The computer system of claim 1, wherein the memory subsystem
invalidates the datablock in the external cache during a write invalidate
10 operation the specifies the datablock.
3. The computer system of claim 2, wherein the processor subsystem issues
the write invalidate operation to the memory subsystem that specifies the
datablock and that includes an invalidate advisory indication that indicates
15 whether the datablock is present in the external cache.
4. The computer system of claim 3, wherein the invalidate advisory
indication determines whether the memory subsystem transfers an invalidate
message to the processor subsystem that invalidates the datablock in the
20 external cache.
5. A computer system, comprising:
processor subsystem having an external cache memory including
external tags;
25 memory interconnect including a mechanism that invalidates a
datablock in the external cache;
wherein the processor subsystem issues a write invalidate operation to
the memory subsystem that specifies the datablock and that includes an



13

THIS SHEET HAS BEEN
REPLACED

invalidate advisory indication that indicates whether the datablock is present in the external cache.

6. The computer system of claim 5, wherein the memory interconnect
5 maintains a set of duplicate tags that correspond to the external tags.

7. The computer system of claim 5, wherein the memory interconnect
invalidates the datablock in the external cache in response to the write
invalidate operation the specifies the datablock.

10

8. The computer system of claim 7, wherein the invalidate advisory
indication determines whether the memory interconnect transfers an
invalidate message to the processor subsystem that invalidates the datablock in
the external cache.

15

9. A method for invalidating a datablock stored in an external cache of a
processor in a computer system, comprising the steps of transferring a write
invalidate operation that specifies the datablock to a memory interconnect and
then invalidating the datablock in response to an invalidate message received
20 from the memory interconnect.

10. The method of claim 9, wherein the write invalidate operation to the
memory interconnect specifies the datablock and includes an invalidate
advisory indication that indicates whether the datablock is present in the
25 external cache.

This Sheet Has Been Replaced



14

THIS SHEET HAS BEEN
REPLACED

11. The method of claim 10, wherein the invalidate advisory indication determines whether the memory interconnect transfers the invalidate message to the processor that invalidates the datablock in the external cache.

This Sheet Has Been Replaced

