(54) Title: METHOD FOR CONTROLLING CACHE SYSTEM COMPRISING DIRECT-MAPPED CACHE AND FULLY-ASSOCIATIVE BUFFER

(57) Abstract: A method for controlling a cache system is provided. The cache system is organized as a direct-mapped cache configured with a small block size and a fully associative spatial buffer configured with a large block size, consisting of a plurality of small blocks. Here, if accesses to the direct-mapped cache and the fully associative buffer are a miss, data of the accessed address and data of the adjacent addresses are copied to the large block in the fully associative spatial buffer according to a first-in-first-out (FIFO) algorithm. Furthermore, if one or more small data blocks accessed before exist among its corresponding large block of data to be expelled from the fully associative buffer, the small block(s) accessed is copied to the direct-mapped cache.
METHOD FOR CONTROLLING CACHE SYSTEM COMPRISING DIRECT-MAPPED CACHE AND FULLY-ASSOCIATIVE BUFFER

Technical Field

The present invention relates to a method for controlling a cache system used for reducing the access time of a central processing unit (CPU), and more particularly, to a method for controlling a cache system constructed with a direct-mapped cache and a fully associative buffer to exploit both temporal and spatial locality adaptively.

Background Art

Since access time to a memory upon the request by a CPU for instructions or data can engender considerable delays, as shown in FIG. 1, a memory hierarchy arranged according to the order in which the CPU accesses data from different memories in the hierarchy is needed. Referring to FIG. 1, the CPU accesses memories in the order of a register 10, a cache system 11, a main memory 12, a disk 13, and a tape 14. Here, the register 10 having the fastest access time is referred to as the highest hierarchical level, and the tape 14 having the slowest access time is referred to as the lowest hierarchical level.

Among the above memories, the cache system 11 is accessed before accessing to the main memory 12, and therefore its structure and controlling method may significantly impact the execution speed and power consumption of a CPU. The cache system 11 is designed to exploit the principle of locality.

The locality is divided into spatial locality and temporal locality. The spatial locality refers to the tendency for adjacent or nearby memory locations to be referenced close together in time. The temporal locality is the likelihood that data retrieved once will be retrieved again soon.

Cache systems exploit temporal locality by retaining recently referenced data, and spatial locality by fetching multiple words as a cache block whenever a miss occurs. These two approaches for optimizing each type of locality contradict each other when cache capacity is fixed. This is because the increment in the block size is inversely proportional to the number of cache blocks. Therefore, as the size of a block increases, more data adjacent to the accessed memory address are copied in the cache system. In this case, data blocks reside shorter in the cache system because of the reduced number of
cache blocks. Thus, if the storage capacity of the cache system is fixed at a predetermined level, as the size of a block increases, the cache system has a higher spatial locality but a lower temporal locality. Conversely, as the size of a block decreases, the cache system has a lower spatial locality but a higher temporal locality.

To reduce the above conflicts as much as possible, a cache system including two cache memories, which are separately controlled, has been suggested. According to a conventional cache control method, complex mechanisms were used to exploit two localities, e.g., methods of using locality prediction table, compiler, locality detection unit, prefetching, and so forth. These conventional cache control methods have problems in that they offer high design complexity and high hardware cost.

Disclosure of the Invention

To solve the above problems, it is an object of the present invention to provide a cache system that consists of two caches with different configurations and its control method to exploit each type of locality, resulting in reduced miss ratio and power consumption.

Accordingly, to achieve the above object, the present invention provides a method for controlling a cache system, constructed as a direct-mapped cache configured with a small block size and a fully associative spatial buffer configured with a large block size. Temporal locality is exploited by caching candidate small blocks selectively into the direct-mapped cache. Spatial locality is enhanced using a large fetch size. Selection mechanism for temporal locality is based on a time interval for choosing the blocks to store into the direct-mapped cache. Instead of placing every missed block directly into the direct-mapped cache, this method places a large block including the missed small block into the fully associative spatial buffer. Then the missed block is moved into the direct-mapped cache at the moment when the large block is replaced from the fully associative spatial buffer, according to a first-in-first-out (FIFO) algorithm.

When a miss occurs in both the direct-mapped and the fully associative spatial buffer, a large block consisting of the corresponding group of small blocks is fetched into the spatial buffer. If a reference misses in the direct-mapped cache, but hits in the spatial buffer, its corresponding small block is fetched from the spatial buffer and its hit bit is set. Using this hit bit enables the
cache system to selectively determine those blocks showing temporal locality. Thus, data may reside in the direct-mapped cache for a long period, and therefore the temporal locality is raised, thereby reducing miss rates and power consumption.

5

Brief Description of the Drawings

FIG. 1 is a block diagram of a memory hierarchy arranged according to the order in which a central processing unit (CPU) accesses different memories in the hierarchy;

FIG. 2 is a block diagram showing the structure of a cache system to which a cache control method according to the present invention is applied;

FIG. 3 is a flowchart of the method for controlling the cache system of FIG. 2 according to the present invention;

FIG. 4 is a graph showing miss ratios of a victim cache system according to its own cache control method and of a cache system according to a cache control method of this invention, respectively;

FIG. 5 is a graph showing average access times of a victim cache system according to its own cache control method and of a cache system according to a cache control method of this invention, respectively;

FIG. 6 is a graph showing normalized power consumption of a victim cache system according to its own cache control method and of a cache system according to a cache control method of this invention, respectively; and

FIG. 7 is a block diagram showing the other structure of a cache system to which a cache control method according to the present invention is applied.

Best mode for carrying out the Invention

Referring to FIG. 2, a cache system, to which a cache control method according to the present invention is applied, consists of a direct-mapped cache 21 and a fully associative spatial buffer 22. The direct-mapped cache 21 includes a data storage unit 211 and a control bit storage unit 212. The data storage unit 211 is configured such that data accessed by a central processing unit (CPU, not shown) are stored in an 8-byte small block SUB. The control bit storage unit 212 stores a group of bits, i.e., a 1-bit valid bit V, a 1-bit dirty bit D, and an n-bit tag T, for each small block SUB in response to an index signal DMI input through an address bus 201. A comparator 203 checks to see if the value of a tag signal DMT input from the CPU through the address bus 201
exists in the tags T of the control bit storage unit 212, and generates an access result signal DMH indicative of whether access to the direct-mapped cache 21 is a hit or miss. The access result signal DMH is input to the CPU through a control bus (not shown). A multiplexer 204 selectively inputs to the data storage unit 211 of the direct-mapped cache 21 one data word DW among 8 bytes of data words DWs from a data bus 202 and 8 bytes of data words DWs from small blocks 220, 221, 222, and 223 of the fully associative spatial buffer 22.

The fully associative spatial buffer 22 is configured such that data of an address accessed by the CPU and data of adjacent addresses are stored in a 32-byte large block consisting of the four small blocks 220, 221, 222, and 223. A content addressable memory CAM 227 in the fully associative spatial buffer 22 generates a valid bit V 226 for the large block 220, 221, 222, and 223 and a dirty bit D 225 and a hit bit H 224 for each of the small blocks 220, 221, 222, and 223 in response to a tag signal SBT input from the CPU through the address bus 201. An AND gate 205 checks whether an address generated in the content addressable memory 227 is valid and generates an access result signal SBH indicative of whether access to the fully associative spatial buffer 22 is a hit or a miss. The access result signal SBH is input to the CPU through the control bus.

Two-bit offset control signal SBO from the address bus 201 selectively enables inputs and outputs of the small blocks 220, 221, 222, and 223 by issuing each bank enable signal BE through a multiplexer 207. Furthermore, 8 bytes of data words DWs from the data bus 202 or from a subordinate memory such as a main memory are input to the selectively enabled small blocks 220, 221, 222, and 223 through a multiplexer 206.

A method for controlling the cache system according to the present invention will now be described with reference to FIGS. 2 and 3.

First, the direct-mapped cache 21 and the fully associative spatial buffer 22 are accessed in parallel at the same level by read or write operation upon a write or read request from the CPU (step S301). Then, a check is made as to whether access to the direct-mapped cache 21 is a hit (step S302). If read access to the direct-mapped cache 21 is a hit (step S303), the read data are transmitted to the CPU (step S315) to terminate the process. If write access to the direct-mapped cache 21 is a hit (step S303), a dirty bit D for the small block
SUB accessed in the direct-mapped cache 21 is set (step S304) to terminate the process.

On the other hand, while accessing to the direct-mapped cache 21, a check is made as to whether access to the fully associative spatial buffer 22 is a hit (step S305) at the same time. If read access to the fully associative spatial buffer 22 is a hit (step S306), a hit bit H3, H2, H1 or H0 for each small block 220, 221, 222 or 223 accessed is set (step S314) and the read data are transmitted to the CPU (step S315), thereby terminating the process. If write access to the fully associative buffer 22 is a hit (step S306), a dirty bit D3, D2, D1, or D0 and a hit bit H3, H2, H1 or H0 for each small block 220, 221, 222 or 223 accessed are set (steps S307 and S314), thereby terminating the process. Write back of any dirty small block 220, 221,222, or 223 in the fully associative spatial buffer cannot directly occur at the fully associative spatial buffer because any more modified or referenced small block 220, 221,222, or 223 is always copied at the direct-mapped cache 21 before this block is replaced.

If accesses to both direct-mapped cache 21 and fully associative spatial buffer 22 are a miss, a check is made as to whether an empty large block 220, 221, 222, and 223 exists in the fully associative spatial buffer 22 (step S308). Here, if the states of all valid bits V 226 in the fully associative spatial buffer 22 are set, no empty large block 220, 221, 222, and 223 exists therein. If the state of one of the valid bits 226 is not set, an empty large block 220, 221, 222, and 223 exists in the fully associative spatial buffer 22.

If an empty large block 220, 221, 222, and 223 exists in the fully associative spatial buffer 22, a large data block is copied to the empty large block 220, 221, 222, and 223 in the fully associative spatial buffer 22 according to a first-in-first-out (FIFO) algorithm (step S313). Then, a hit bit H3, H2, H1 or H0 for each small block 220, 221, 222 or 223 accessed is set (step S314). Here, in the case of a read operation, the read data are transmitted to the CPU (step S315).

On the other hand, if no empty large block 220, 221, 222, and 223 exists in the fully associative spatial buffer 22, a check is made as to whether there is a small block of data 220, 221, 222, or 223 accessed within the large block 220, 221, 222, and 223 of data to be expelled from the fully associative spatial buffer 22 according to a FIFO algorithm (step S309). In the step S309, if at least one among the hit bits H3, H2, H1, and H0 of the small blocks 220, 221, 222, and 223 is set, it is determined that a corresponding small block of
data has been accessed. Also a large block of data in the main memory is copied to an empty large block 220, 221, 222, and 223 in the fully associative spatial buffer 22 according to a FIFO algorithm (step S313). Then, a hit bit H3, H2, H1, or H0 of the small block of data 220, 221, 222, or 223 accessed is set (step S314). Here, in the case of a read operation, the read data are transmitted to the CPU (step S315).

If the small data block accessed exists in the step S309, a check is made as to whether the small block of data SUB to be expelled from the direct-mapped cache has been written directly to the direct-mapped cache 21 (step S310). In the step S310, if a dirty bit D of the corresponding small block SUB is set, the small block of data SUB is determined as written directly to the direct-mapped cache 21 (See the step S307) If the small block of data SUB written directly to the direct-mapped cache 21 exists, the small block of data SUB is copied to a main memory that is a subordinate memory (step S311).

Furthermore, the small block of data 220, 221, 222, or 223 that has been accessed is copied into the direct-mapped cache 21 (step S312). Then, a large data block is copied to an empty large block 220, 221, 222 and 223 in the fully associative spatial buffer 22 according to a FIFO algorithm (step S313). Then, a hit bit H3, H2, H1, or H0 of the small block of data 220, 221, 222, or 223 accessed is set while they are accessed (step S314). Here, in the case of a read operation, the read data is transmitted to the CPU (step S315).

FIG. 4 shows miss ratios of a victim cache system according to its own cache control method and of a cache system according to a cache control method of this invention, respectively. FIG. 5 shows average access times of a victim cache system according to its own cache control method and of a cache system according to a cache control method of this invention, respectively. FIG. 6 shows normalized power consumption of a victim cache system according to its own cache control method and of a cache system according to a cache control method of this invention, respectively. Go, tomcatv, gcc, jpeg, compress, applu, vortex, and m88ksim denote different benchmark programs having different temporal and spatial localities for simulations.

In a victim cache system, a direct-mapped cache is configured the same as a victim buffer in terms of size of blocks. Reference numerals 411, 421, 431, 441, 451, 461, 471, 481, 491, 511, 521, 531, 541, 551, 561, 571, 581, and 591 denote graphs of a victim cache system including a direct-mapped cache and a victim buffer, each having 8-byte blocks. Reference numberals 412, 422,
432, 442, 452, 462, 472, 482, 492, 512, 522, 532, 542, 552, 562, 572, 582, and
592 denote graphs of a victim cache system including a direct-mapped cache
and a victim buffer each having 16-byte blocks. The reference numberals 413,
423, 433, 443, 453, 463, 473, 483, 493, 513, 523, 533, 543, 553, 563, 573, 583,
593, 611, 621, 631, 641, 651, 661, 671, 681, and 691 denote graphs of a victim
cache system including a direct-mapped cache and a victim buffer having 32-
byte blocks. The reference numberals 414, 424, 434, 444, 454, 464, 474, 484,
494, 514, 524, 534, 544, 554, 564, 574, 584, 594, 612, 622, 632, 642, 652, 662,
672, 682, and 692 denote graphs of the cache system according to the present
invention including the direct-mapped cache 21 of FIG. 2 having 8-byte small
blocks SUB of FIG. 2 and the fully associative spatial buffer 22 of FIG. 2 having
the 32-byte large blocks 220, 221, 222, and 223 of FIG. 2. As shown in FIGS.
4, 5, and 6, the cache system, to which the cache control method according to
this invention is applied, has low miss ratio, low average access time, and low
power consumption compared to conventional victim cache systems.

FIG. 7 is a block diagram showing the other structure of a cache system
to which a cache control method according to the present invention is applied.
Reference numerals of FIG. 7 which are the same as those of FIG. 2 indicate
identical elements. A reference numeral AG indicates an address generator,
228 indicates a multiplexer, 209 indicates an inverter, and 232 indicates an AND
gate. The cache memory subsystem of FIG.7 is similar to the system of FIG.2
except having an additional feature for prefetching mode described hereunder.

In case of the prefetching mode, if a prefetch bit P corresponding to the
large block is still reset, a prefetch operation is initiated when a hit occurs in any
bank of the fully associative spatial buffer 22 and at least one among the hit bits
H3, H2, H1, and H0 corresponding to its large block SUB3, SUB2, SUB1 and
SUB0 is already set. At the same time, the tags of the spatial buffer are
searched for a prefetch address to check whether it is already present. If the
address is not in the spatial buffer, the prefetch controller 208 generates a
prefetch signal PFS and a target address PFA to fetch the large block into the
prefetch buffer 231 from the subordinate memory. And also the prefetch bit P of
the large block generating the prefetch signal is set. The main object of this
prefetch bit P is to prevent the prefetch controller 208 from searching the
already used large block. That is, if the prefetch bit P of the large block is set,
the sequential large block (i.e., prefetch target block) must present in the fully
associative spatial buffer 22 or the prefetch buffer 231. Therefore, there is no
necessity for searching the tags of the fully associative spatial buffer 22 whether it is present.

Further, in the prefetching mode, the prefetch controller 208 generates a prefetch signal when multiple hit bits are set. Then two operations are performed consecutively by the prefetch controller 208. The first operation is to search the tag part of the fully associative spatial buffer 22 when a hit occurs for the \(i\)-th large block stored in the fully associative spatial buffer 22, in order to detect whether the \((i+1)\)-th large block already exists in the fully associative spatial buffer 22. A one cycle penalty is present in this case, but this overhead is negligible because prefetching initiates only about 1.5% ~ 2.5% of the total number of addresses generated by the CPU. Thus, the average MCPI(Memory Cycles Per Instruction) is increased by about 0.06 %. If the \((i+1)\)-th large block does not exist in the fully associative spatial buffer 22, the second operation is performed: the \((i+1)\)-th large block is prefetched into the prefetch buffer 231. If misses occur at both the direct-mapped cache 21 and the fully associative spatial buffer 22, a cache controller initiates its miss handling process. While this miss handling occurs, a block that was already in the prefetch buffer 231 is transferred into the fully associative spatial buffer 22. Therefore, the transfer time can be totally hidden because there is plenty of time, i.e., 19 clock cycles for handling a miss, to perform this block move.

The system of the FIG.7 has two modes determined by user selection, that is, the non-prefetching mode and the prefetching mode. The prefetching mode guarantees further performance gain with low overhead. On every memory access, both the direct-mapped cache 21 and the fully associative spatial buffer 22 are accessed at the same time. The different cases for the operational model are explained hereunder. In either non-prefetching mode or prefetching mode, if a small block (e.g., 8-byte block size) is found in the direct-mapped cache 21, the actions are the same as any conventional cache hit. The requested data item is sent to the CPU without delay. In either non-prefetching mode or prefetching mode, when a memory address is generated by the CPU, some address bits are used to select one bank among several in the fully associative spatial buffer 22. If the size of a small block is 8-bytes and the size of a large block is 32-bytes as a design example, the number of banks is assumed to be four as in FIG. 7. Most two bits of large block offset are used to enable one of four banks in the fully associative spatial buffer 22. Thus, power consumption can be decreased by activating only one bank at a time. In this
case, the block size of one bank is 8-bytes and it is equal to the size of a block in the direct-mapped cache 21. All of the small block entries in each bank hold a hit bit to check whether a particular small block within a large block has been referenced before or not. If a hit occurs in the fully associative spatial buffer 22, the hit bit of that small block becomes set to mark it as a referenced block.

In the case of prefetching mode, if the prefetch bit P corresponding to the large block is still reset, a prefetch operation is initiated when a hit occurs in any bank of the fully associative spatial buffer 22 and one or more of the hit bits corresponding to its large block are already set. At the same time, the tags of the fully associative spatial buffer 22 are searched for the prefetch address to check whether it is already present. If the address is not in the fully associative spatial buffer 22, the prefetch controller 208 generates the prefetch signal and target address PFS to fetch the large block into the prefetch buffer 231 from the subordinate memory. And also, the prefetch bit P of the large block generating the prefetch signal is set. The main object of this prefetch bit P is to prevent the prefetch controller 208 from searching the already used large block. That is, if the prefetch bit P of the large block is set, the sequential large block (i.e., prefetch target block) must present in the fully associative spatial buffer 22 or the prefetch buffer 231. Therefore there is no necessity for searching the tags of the fully associative spatial buffer 22 whether it is present. Whenever either cache misses, then while the cache controller is handling the miss, a large block is loaded into the fully associative spatial buffer 22 from the prefetch buffer 231.

If the fully associative spatial buffer 22 is full, the oldest large block is replaced. Each small block whose hit bit is set in the about-to-be-evicted large block is loaded into the direct-mapped cache 21, because those blocks are marked as showing temporal locality. This loading time is hidden by the time it takes to transfer the missed block into the fully associative spatial buffer 22. Finally, if either cache misses while a prefetch operation is being performed, the miss handling process occurs after the ongoing prefetch operation completes. And all hit bits of the prefetched block are set to zero.

In either non-prefetching mode or prefetching mode, if a miss occurs in both the direct-mapped cache 21 and the fully associative spatial buffer 22, a large block including a missed small block is brought into the fully associative spatial buffer 22 from the subordinate memory. An example is chosen such that a small block size is 8-bytes and a large block size is 32-bytes, and those four sequential small blocks belong to a 32-byte block boundary. Two cases are
considered depending on whether the fully associative spatial buffer 22 is full. If at least one entry in the fully associative spatial buffer 22 is in the invalid state, a large block is fetched and stored in the fully associative spatial buffer 22. When a particular small block is accessed by the CPU, the corresponding hit bit is marked as one. Thus the hit bit of the small block identifies it as a referenced block. The fully associative spatial buffer 22 has the FIFO replacement policy. If the fully associative spatial buffer 22 is full, the oldest entry is replaced. Then the blocks in the entry whose hits bit are set are moved into the direct-mapped cache 21. These actions exploit temporal locality by selectively increasing the lifetime of a small block. Moreover this mechanism reduces conflict misses and the thrashing effect. Because these actions are accomplished while the cache controller is handling a miss, this operation does not introduce any additional delay.

The move operations between the direct-mapped cache 21 and the fully associative spatial buffer 22 are illustrated as follows. When a memory address is generated, such as 0000 0000 0000 0000 0000 0000 1000 0000, in the direct-mapped cache 21 (e.g., 8KB cache size with 8-byte block size), the tag field is 19-bits, the index field T is 10-bits, and the offset field is 3-bits. Therefore the tag, index, and offset values are 000 0000 0000 0000, 00 0001 0000, and 000 respectively. In the fully associative spatial buffer 22 (e.g., 1KB cache size with 32-byte block size), the tag field is 27-bits and the offset field is 5-bits. Therefore the tag values of the fully associative spatial buffer 22 are 000 0000 0000 0000 0000 0000 0100, and the high order two bits of the offset are 00. Also these bits are used to search the four banks selectively. If a miss occurs in both the direct-mapped cache 21 and the fully associative spatial buffer 22, data corresponding to the most significant two bits of the offset (e.g., 11, 10, 01, 00 bits) are fetched and only the hit bit of the first of the four small blocks is set. When this large block in the fully associative spatial buffer 22 is replaced and if the hit bit of the first small block is the only one set, the bits 00 corresponding to the first small block are added to the tag value of the fully associative spatial buffer 22 by the address generator AG. Therefore, a new memory address without offset is formed, and corresponding tag and index values for the direct-mapped cache 21 are created as 000 0000 0000 0000 0000 and 00 0001 0000 respectively through the process of redecoding. Cache write-back cannot directly occur at the fully associative spatial buffer 22 because any modified or referenced small block is always moved to the direct-mapped cache 21 before it
large block is replaced. A write back operation for a conventional direct-mapped cache or victim cache with the same fetch block size (e.g., 32-bytes) must be performed with a 32-byte block size even though only one word requires write-back. In contrast, the SMI cache executes this write back operation only for the marked 8-byte small blocks. Therefore write traffic into memory is reduced.

**Industrial Applicability**

As described above, according to the cache control method of this invention, a small block of data accessed among a large block of data to be expelled from the fully associative spatial buffer is copied to the direct-mapped cache. Using hit bit information enables the cache system to selectively determine those small blocks showing strong temporal locality. The cache control method exploits hit bit information about utilization of the small blocks that is obtained during a time interval proportional to the number of entries in the spatial buffer. Accordingly, the time duration that data reside in the direct-mapped cache is increased by increased temporal locality while maintaining spatial locality, thereby reducing miss ratios and power consumption of a cache system.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined by the appended claims.
What is claimed is:

1. A method for controlling a cache system constructed with a direct-mapped cache configured with a small block size and a fully associative spatial buffer configured with a large block size, consisting of a plurality of small blocks, the method comprising the steps of:
   
   (a) copying data of the accessed address and data of the adjacent addresses to the large block in the fully associative buffer according to a first-in-first-out (FIFO) algorithm if accesses to the direct-mapped cache and the fully associative spatial buffer are a miss; and
   
   (b) if one or more small blocks accessed before exist among the large block of data to be expelled from the fully associative spatial buffer in the step (a), copying corresponding small block(s) accessed to the direct-mapped cache.

2. A method for controlling a cache system constructed with a direct-mapped cache configured with a small block size and a fully associative spatial buffer configured with a large block size, consisting of a plurality of small blocks, the method comprising the steps of:
   
   (a) checking whether an empty large block exists in the fully associative buffer if accesses to the direct-mapped cache and the fully associative buffer are a miss;
   
   (b) if the empty large block exists in the step (a), performing step (c7);
   
   (c) if no empty block exists in the step (a), performing steps (c1) –(c7);
   
   (c1) checking whether a small data block, which has been written directly to the fully associative spatial buffer, exists among its corresponding large data block to be expelled according to a first-in-first-out (FIFO) algorithm;
   
   (c2) if the small data block which has been written directly to the fully associative spatial buffer exists belonging to its large data block to be expelled, copying the corresponding small data block into the direct-mapped cache;
   
   (c3) checking whether the small block of data that has been accessed exists among the large block of data to be expelled according to a FIFO algorithm;
   
   (c4) if the small block of data accessed exists, checking whether the small block of data to be expelled has been written directly to the direct-mapped cache;
   
   (c5) if the small block of data to be expelled has been written directly to the direct-mapped cache or to the fully associative buffer, copying the small
block of data to be expelled into the main memory;

(c6) copying the small block of data confirmed to have been accessed
in the step (c3) to the direct-mapped cache; and

(c7) copying data of an address accessed in the step (a) and data of
adjacent addresses in the large block to the fully associative buffer according to
a FIFO algorithm.
FIG. 1

- Register
- Cache System
- Main Memory
- Disk
- Tape
FIG. 3

Start

Both caches are accessed at the same time S301

Direct-mapped cache / Spatial buffer

Direct-mapped cache hit S302

Write operation? No S303

Yes S304

Dirty bit of small block in direct-mapped cache is set

Spatial buffer hit S305

Spatial buffer full? No S308

S309

Hit bit corresponding to the replaced small blocks in the bank is at least one or more set S310

No

Corresponding block of direct mapped cache dirty? No S311

Yes

Write back dirty block of direct-mapped cache

S312

Corresponding small blocks are copied at direct-mapped cache

S313

One large block including a missed small block is placed at each bank of the spatial buffer

S314

Hit bit of requested block of one bank in buffer is only set

S315

If read operation, requested data is transferred to the CPU

End
FIG. 6
INTERNATIONAL SEARCH REPORT

A. CLASSIFICATION OF SUBJECT MATTER
   
   IPC7 G06F 12/02, G06F 12/08
   According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
   
   Minimum documentation searched (classification system followed by classification symbols)
   IPC7 G06F 13/00, G06F 12/12,

   Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched:
   - Korean Patents and Applications for Inventions since 1975
   - Korean Utilities and Applications for Utility models since 1975

   Electronic data base consulted during the international search (name of data base and, where practicable, search terms used):
   http://www.kipo.go.kr(Domestic Search System in the Korean Intellectual Property Office) "internet broadcast server link"

C. 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>A</td>
<td>KR99-0019298(HYUNDAI ELECTRONIC CORP.) 15 JUNE 1999 * abstract &amp; claim</td>
<td>1-2</td>
</tr>
<tr>
<td>A</td>
<td>US 2264577(HEWLETT PACKARD CO.) 01 SEPTEMBER 1993 * abstract &amp; claim</td>
<td>1-2</td>
</tr>
<tr>
<td>A</td>
<td>US 04493026(International Business Machines Corporation) 08 JANUARY 1985 * abstract &amp; claim</td>
<td>1-2</td>
</tr>
<tr>
<td>A</td>
<td>US 06047357(Digital Equipment Corporation) 04 APRIL 2000 * abstract &amp; claim</td>
<td>1-2</td>
</tr>
</tbody>
</table>

☐ Further documents are listed in the continuation of Box C. X See patent family annex.

* Special categories of cited documents:
  "A" document defining the general state of the art which is not considered to be of particular relevance
  "B" earlier application or patent 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 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

"I" 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

"K" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"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

Date of the actual completion of the international search: 27 JULY 2001 (27.07.2001)

Date of mailing of the international search report: 27 JULY 2001 (27.07.2001)

Name and mailing address of the ISA/KR
Korean Intellectual Property Office
Government Complex-Daejon, Dunsan-dong, Seo-gu, Daejeon
Metropolitan City 302-701, Republic of Korea
Facsimile No. 82-42-472-7140

Authorized officer
LIM, Young Heui
Telephone No. 82-42-481-5779

Form PCT/ISA/210 (second sheet) (July 1998)
<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>US 2264577</td>
<td>01-09-1993</td>
<td>DE 4306077</td>
<td>02-09-1993</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GB 2264577</td>
<td>01-09-1993</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 6012323</td>
<td>21-01-1994</td>
</tr>
<tr>
<td>US 04493026</td>
<td>08-01-1985</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>US 06047357</td>
<td>04-04-2000</td>
<td>NONE</td>
<td></td>
</tr>
</tbody>
</table>