

METHOD AND APPARATUS TO CONTROL MEMORY ACCESSES

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention is related to the field of electronics. In particular, the present invention is related to a method and apparatus for controlling memory accesses.

Description of the Related Art

[0002] Dynamic Random Access Memories (DRAMs) have memory precharge, activate, read, and write operations. In particular, a memory controller that addresses a bank of memory must first precharge the memory bank, then the addressed page within the bank must be activated before the addressed column in that page is accessed (read or written). A "DRAM page open" or a "page hit" indicates the memory being accessed has already been precharged and activated, and data may be read or written from the page without having to precharge or activate the memory during each memory access. When a "page miss occurs" (i.e., data is accessed from a page in memory other than from the page that is open), the open page must be written back to the DRAM chip from the sense amps. Next, the new memory page has to first be precharged and activated before being accessed. Writing the old page to DRAM, and precharging and activating the new DRAM pages takes time and slows down memory accesses resulting in an inefficient use of the memory bus (reduced bandwidth) and a loss in performance of an apparatus (e.g., a computer) employing DRAM.

[0003] Figure 1 illustrates a front side bus (FSB) controller that schedules memory accesses according to a prior art embodiment. As illustrated in Figure 1, the FSB

controller 105 includes a FSB scheduler 125 that schedules accesses (reads and writes) from a processor 110 to DRAM 120 via a FSB access queue 130. The FSB controller is coupled to processor 110 via a L2 miss request bus and to memory controller 115 via a front side bus. The memory controller 115 is coupled to DRAM 120 via a memory bus.

5 Processor 110 comprises an out-of-order core 135 and a hardware prefetcher (HWP) 140. The out-of-order core 135 uses a pipelining technique wherein multiple instructions are overlapped in execution in an effort to improve the overall performance of the processor. The HWP prefetches data needed by execution units in the out-of-order core from DRAM.

[0004] The FSB controller 105 and in particular the FSB scheduler 125 schedules DRAM
10 accesses from the processor 110 (i.e., from the out-of-order core and from the HWP) based upon the relative age of the instructions that caused the access (i.e., a load or store resulting in a request for reading or writing data to DRAM). Hence the accesses in the FSB access queue 130 are scheduled to DRAM in program order. In particular, DRAM accesses from the out-of-order core have priority over DRAM accesses from the HWP,
15 and accesses generated by the out-of-order core are scheduled in program order. Thus, the FSB controller minimizes the DRAM access latency by first scheduling the accesses associated with the oldest instruction in the FSB access queue. However, scheduling DRAM accesses based upon minimizing the latency of DRAM accesses is not always the optimum solution, as successive DRAM accesses may not always access open DRAM
20 pages. Accesses from a closed DRAM page involve writing the currently open page to DRAM, and precharging and activating a new DRAM page. Thus, accessing a closed DRAM page takes time and slows down memory accesses, thereby rendering the processor's DRAM bus inefficient.

BRIEF SUMMARY OF THE DRAWINGS

[0005] Examples of the present invention are illustrated in the accompanying drawings.

The accompanying drawings, however, do not limit the scope of the present invention.

Similar references in the drawings indicate similar elements.

5 [0006] **Figure 1** illustrates a front side bus (FSB) controller that schedules memory accesses according to a prior art embodiment;

[0007] **Figure 2A** illustrates a front side bus (FSB) controller that schedules memory accesses according to one embodiment of the invention;

10 [0008] **Figure 2B** illustrates parameters associated with memory accesses according to one embodiment of the invention.

[0009] **Figure 3** illustrates a flow diagram of memory accesses according to one embodiment of the invention;

[0010] **Figure 4** illustrates one embodiment of a computer system in which the present invention operates.

15

DETAILED DESCRIPTION OF THE INVENTION

[0011] Described is a method and apparatus to control memory accesses. In the following description, numerous specific details are set forth in order to provide a thorough
20 understanding of the present invention. It will be apparent, however, to one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well-known architectures, steps, and techniques have not been shown to avoid unnecessarily obscuring the present invention.

[0012] Parts of the description are presented using terminology commonly employed by those skilled in the art to convey the substance of their work to others skilled in the art.

Also, parts of the description will be presented in terms of operations performed through the execution of programming instructions. As well understood by those skilled in the art,

5 these operations often take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, and otherwise manipulated through, for instance, electrical components.

[0013] **Figure 2A** illustrates a front side bus (FSB) controller that schedules memory

accesses according to one embodiment of the invention. As illustrated in the embodiment

10 of Figure 2A, the FSB controller 205 includes a FSB scheduler 225 that schedules DRAM accesses from a processor 210 to DRAM 220 using a FSB access queue 230. The FSB

controller is coupled to processor 210 via a L2 miss request bus and to memory controller

215 via a FSB. The memory controller 215 is coupled to DRAM 220 via a memory bus.

Processor 210 comprises an out-of-order core 235 and a hardware prefetcher (HWP) 240.

15 The out-of-order core 235 uses a pipelining technique wherein multiple instructions are overlapped in execution in an effort to improve the overall performance of the processor.

The HWP prefetches DRAM accesses that are predicted to be eventually needed by execution units in the out-of-order core.

[0014] The FSB controller 205 comprises a FSB scheduler 225 and a DRAM page open

20 predictor 250. In one embodiment, the FSB scheduler 225 is coupled to the DRAM page

open predictor 250, and the FSB scheduler is coupled to a FSB access queue 230. Alternate embodiments may have both the DRAM page open predictor and the FSB scheduler

coupled to the FSB access queue 230. In one embodiment, the FSB scheduler 225

schedules accesses from the processor 210 (i.e., from the out-of-order core and from the

25 HWP) to DRAM based upon the application, or portions of the application, that is being

processed by the processor. For example, if the application being processed is latency tolerant, the FSB scheduler 225 schedules memory accesses for maximizing the memory bus bandwidth. Alternately, if the application being processed is latency intolerant, the FSB scheduler 225 schedules memory accesses to minimize memory access latency. In other words, the FSB scheduler arranges memory accesses in the FSB access queue 230 based upon the characteristics of the application being processed by processor 210. An application may be latency intolerant e.g., when a memory access is needed to resolve a branch missprediction and therefore the speculative work performed by the core when a branch is mispredicted is useless until the memory access is complete and the processor fetches instructions from the correct branch of program execution. In order to determine whether an application is "latency tolerant" or "latency intolerant", the FSB scheduler 225 monitors the memory accesses generated by the HWP 240 and by the out-of-order core 235. The FSB does this by monitoring the entries in the FSB access queue 230.

[0015] Figure 2B illustrates parameters associated with memory accesses according to one embodiment of the invention. As Figure 2B illustrates, each DRAM access (entry) in the FSB access queue 230 has associated with it at least the following information: a) The instruction number e.g., the program instruction number, 260, of the DRAM access (i.e., the elapsed time since the DRAM access was generated); b) The memory address, 262, of the DRAM access; c) A number, 264, indicating the position of the DRAM access in the FSB queue (older accesses are placed higher in the queue hierarchy than later instructions); d) A flag, 266, that indicates whether the access was created by the out-of-order core or by the HWP; e) A flag, 268, that indicates whether the entry created by the HWP was also generated/used by the core; f) A flag, 270, that indicates whether the access is predicted to access an open DRAM page; and g) A flag, 272, that indicates whether the memory access is for reading or writing memory.

[0016] In one embodiment of the invention, if a majority of the memory accesses scheduled by the FSB scheduler on to the memory bus are generated by the HWP 240, determined by checking whether flag 268 is set, and if the majority of these generated accesses are utilized by the out-of-order core 235, determined by checking whether flag 266 is set, then the application determined to be latency tolerant. For a latency tolerant application, in one embodiment of the invention, the FSB scheduler 225 may re-order the entries in the FSB queue 230 for maximizing the memory bus bandwidth. This means that the FSB scheduler prioritizes DRAM accesses that are predicted to access data from the same DRAM page. In the embodiment of Figure 2B, in the maximize bandwidth mode the FSB scheduler schedules memory accesses in accordance with the following queue entries 1, 6, 7, 4, 0, 8, 9, 11, 2, 12, 5, and 3. Thus, accesses by the out-of-order-core have priority over accesses by the HWP for accessing open DRAM pages. However, in the minimize latency mode the FSB scheduler schedules memory accesses in accordance with the following queue entries 5, 3, 1, 6, 7, 4, 0, 8, 9, 11, 2, and 12. Thus, older instructions from the core are scheduled by older instructions from the HWP regardless of whether a DRAM page is open. Details of this are provided with respect to Figure 3 below.

[0017] In order to schedule DRAM accesses that are predicted to access data from an open DRAM page, the FSB scheduler 225 utilizes the DRAM page open predictor 250. In one embodiment, the DRAM page open predictor keeps track of the history of DRAM pages that are accessed, and based on the history of the DRAM pages that are accessed, the DRAM page open predictor 250 generates a list of DRAM pages predicted to be open in DRAM. For example, if DRAM pages 12, 14, 16 and 18 were accessed in the past, and the memory system allows for 4 DRAM pages to be simultaneously open, then the DRAM page open predictor predicts that a new access to page 12 will hit, and a new access to page 13 will miss. In one embodiment of the invention, the DRAM page open predictor 250

generates or updates the list of DRAM pages predicted to be open in DRAM at least every memory bus cycle, in other embodiments, the DRAM page open predictor may generate or update the list of DRAM pages predicted to be open in DRAM every few (e.g., 2 or more) memory bus cycles. In one embodiment, the memory controller may send the list of open
5 DRAM pages to the DRAM page open predictor over the FSB. In another embodiment, the memory controller is on the processor core and therefore the list of open DRAM pages, needed by the memory controller, can be utilized as the DRAM page open predictor. In another embodiment, the memory controller predicts which DRAM pages may be speculatively opened based on previous access patterns. For example, the memory
10 controller may use an algorithm (e.g., a curve fit algorithm) that uses the addresses of previously opened DRAM pages to predict future DRAM pages that may be opened. The DRAM page open predictor can be made aware of this algorithm (e.g., by programming the DRAM page open predictor at the time of manufacture), and infer from the accesses being sent to the memory controller which pages the memory controller will speculatively open.
15 This eliminates the need for the memory controller to send the list of DRAM pages that are predicted to be open to the DRAM page open predictor. In one embodiment of the invention, the FSB scheduler uses the list of DRAM pages predicted to be open, generated by the DRAM page hit predictor, to set flag 270 for each memory access.

[0018] Figure 3 illustrates a flow diagram of memory accesses according to one
20 embodiment of the invention. The DRAM page open predictor may predict the DRAM accesses that will access open DRAM pages and the accesses that will access closed DRAM pages. Using this information (i.e., if a DRAM access is predicted to access an open page) the FSB scheduler sets flag 270 (i.e., the predicted DRAM page open flag) for entries in the FSB access queue.

[0019] At 310, the FSB scheduler determines if the HWP generates most of the DRAM accesses utilized by the out-of-order core. In one embodiment of the invention, determining whether the HWP generates most of the DRAM accesses utilized by the out-of-order core implies determining whether flags 266 and 268 are set in some unit time interval. In another embodiment of the invention, determining whether the HWP generates most of the DRAM accesses utilized by the out-of-order core implies determining whether the percentage of accesses that have both flags 266 and 268 set exceeds some threshold e.g., a programmable percentage. In one embodiment of the invention, the programmable percentage is programmed at the time of manufacture of the FSB controller. In alternate embodiments, the programmable percentage is programmed by a user or automatically by the program being executed. In one embodiment of the invention, when the HWP generates the DRAM access, the flag 266 is set (e.g., by the HWP) indicating that the HWP generated the DRAM access. Subsequently, if the out-of-order core also generates the same DRAM access, flag 268 is set (e.g., by the out-of-order core), indicating the out-of-order core utilized the access generated by the HWP. Thus, by monitoring flags 266 and 268 for DRAM accesses in the FSB queue, the following may be inferred if both flags are set for most DRAM accesses: 1) The HWP is generating the required DRAM accesses, and 2) The HWP is not generating unused DRAM accesses. Based at least upon conditions 1) and 2) above it may be inferred that the memory bus is saturated. This assumption is based upon the fact that the HWP requests entering the FSB queue are stalled in the FSB queue long enough such that the out-of-order core then generates the same DRAM access.

[0020] If the conditions described above are met, the application may be described as latency tolerant and the bandwidth of the memory bus may be assumed to be constrained. Once the FSB scheduler determines that the application being processed is latency tolerant and that the bandwidth of the memory bus is constrained, the FSB scheduler switches to

maximize bus bandwidth mode. Otherwise, the FSB scheduler assumes that the application being processed is not latency tolerant and switches to the minimize latency mode. In one embodiment of the invention, if the FSB scheduler is in the maximize bus bandwidth mode, the FSB scheduler re-orders the entries in the FSB access queue to maximize the memory bus bandwidth.

[0021] In the minimize latency mode, at 320, the FSB scheduler schedules FSB accesses such that the memory accesses that are generated by earlier instructions in the out-of-order core are scheduled to access DRAM before accesses generated by later instructions. Also, the FSB scheduler schedules DRAM accesses generated by the out-of-order core before DRAM accesses generated by the HWP. In the minimize latency mode when scheduling entries that are earliest in time (i.e., accesses associated with the oldest instruction in the FSB queue), the FSB scheduler does not consider whether entries in the FSB access queue are accessed from the same DRAM page or from different DRAM pages.

[0022] When in maximize bandwidth mode, the FSB scheduler searches the FSB access queue for an entry associated with the oldest instruction (instructions from the out-of-order core have priority over instructions from the HWP) that is predicted to access an open DRAM page. In one embodiment of the invention, there are 2 parameters used when performing this search. The first parameter, the "search window size", describes how many entries will be searched when searching for an entry that accesses an open DRAM page. Starting with the list of entries, sorted (e.g., when the entries are entered into the FSB queue) by the age of the instruction associated with the entry (older instruction first), the FSB scheduler searches the "search window size" entries on this list for an access that is predicted to access an open DRAM page. If none of the "search window size" entries are predicted to access an open DRAM page, then the first entry on the list is scheduled.

[0023] The second parameter, the "Max Bypass Count", determines how many entries are allowed to bypass the first entry on the list, before the first entry on the list is chosen even though it is predicted to access a closed DRAM page. If an entry is scheduled that is not the first entry on the list, a "Bypass Counter" is incremented. If the first entry on the list is chosen, the "Bypass Counter" is reset to 0. If the value of the Bypass Counter is greater than or equal to the "Max Bypass Count", the first entry on the list is chosen even though it is predicted to access a closed DRAM page (resetting the Bypass Counter to 0). In one embodiment of the invention, the maximum bypass count, and the search window size may be programmed e.g., at the time of manufacture of the FSB scheduler.

[0024] In the maximize bandwidth mode, at 330, a determination is made whether the bypass counter is greater than the bypass count. If the bypass counter is greater than the bypass count then at 350, the first entry in the FSB is scheduled for a DRAM access and the bypass counter is reset e.g., to 0. If at 330, the value of the bypass counter is not greater than the maximum bypass count, at 335, a determination is made whether the first entry in the search window (the search window is defined as the entries in the FSB access queue equal to the search window size) is predicted to access an open DRAM page. If the first entry in the search window is predicted to access an open DRAM page at, 350, the FSB scheduler schedules the first entry for a DRAM access and the bypass counter is reset.

[0025] However, if the first entry in the search window is not predicted to access an open

DRAM page, at 340, a determination is made whether any of the entries in the search window in the FSB access queue is predicted to access an open DRAM page. If none of the entries in the search window is predicted to access an open DRAM page at 350 the first entry in the FSB access queue is scheduled. However, if at 340 at least one of the entries in the search window is predicted to access an open DRAM page, at 345, the FSB scheduler

schedules oldest entry in the search window that is predicted to access an open DRAM page and increments the bypass counter.

[0026] In one embodiment of the invention, if the FSB scheduler schedules DRAM accesses in the maximize bus bandwidth mode, and if the FSB scheduler determines that the percentage of accesses that are scheduled on the FSB access queue are generated by the HWP and requested by the out-of-order core are below the threshold percentage, the FSB scheduler automatically changes the mode of scheduling DRAM accesses from the maximize bus bandwidth mode to the minimize latency mode. So also, if the FSB scheduler schedules DRAM access in the minimize latency mode, and if the FSB scheduler determines that the percentage of accesses that are scheduled on the FSB access queue are created by the HWP and requested by the out-of-order core are equal to or above the threshold percentage the FSB scheduler automatically changes the mode of scheduling DRAM accesses from the minimize latency mode to the maximize bus bandwidth mode. Thus, the change of modes from one to the other is done dynamically (i.e., on-the-fly) based upon the application or portion thereof being processed.

[0027] Although the embodiment of Figure 3 accesses DRAM, alternate embodiments may access other types of memory e.g., Synchronous DRAM etc. Although the embodiment described illustrates the FSB controller as a separate device, in other embodiments, a processor or even a memory controller may incorporate the functionality of the FSB controller described with respect to Figure 3.

[0028] Figure 4 illustrates one embodiment of a computer system in which the present invention operates. In general, the computer system 400 illustrated by Figure 4 includes a processing unit 402 (e.g., a microprocessor) coupled through a bus 401 to a system memory 413. System memory 413 comprises a read only memory (ROM) 404 and a random access memory (RAM) 403. ROM 404 comprises Basic Input Output System (BIOS) 416, and

RAM 403 comprises operating system 418, application programs 420, and program data 424. Processing unit 402 communicates with system memory 413 via FSB controller 422, and FSB controller 422 communicates with system memory 413 via a memory controller (not shown). The FSB controller 422 operates in accordance with the flow diagram described with respect to Figure 3.

[0029] Computer system 400 includes mass storage devices 407, input devices 406 and display device 405 coupled to processing unit 402 via bus 401. Mass storage devices 407 represent a persistent data storage device, such as a floppy disk drive, fixed disk drive (e.g., magnetic, optical, magneto-optical, or the like), or streaming tape drive. Mass storage devices store program data including application programs 428 and operating system 426. Processing unit 402 may be any of a wide variety of general purpose processors or microprocessors (such as the Pentium® family of processors manufactured by Intel® Corporation), a special purpose processor, or even a specifically programmed logic device.

[0030] Display device 405, provides graphical output for computer system 400. Input devices 406, such as a keyboard or mouse, may be coupled to bus 401 for communicating information and command selections to processing unit 402. Also coupled to processing unit 402 through bus 401 are one or more network devices 408 that can be used to control and transfer data to electronic devices (printers, other computers, etc.) connected to computer system 400. Network devices 408 also connect computer system 400 to a network, and may include Ethernet devices, phone jacks, and satellite links.

[0031] Thus, a method and apparatus have been disclosed to control memory accesses.

While there has been illustrated and described what are presently considered to be example embodiments of the present invention, it will be understood by those skilled in the art that various other modifications may be made, and equivalents may be substituted, without departing from the true scope of the invention. Additionally, many modifications

may be made to adapt a particular situation to the teachings of the present invention without departing from the central inventive concept described herein. Therefore, it is intended that the present invention not be limited to the particular embodiments disclosed, but that the invention include all embodiments falling within the scope of the appended

5 claims.

What is claimed is:

1. An apparatus for accessing memory comprising:
a page open predictor to predict open memory pages;
a front side bus scheduler to switch memory access modes from a first mode to a
5 second mode, said front side bus scheduler coupled to the page open predictor; and
a front side bus access queue, coupled to the front side bus scheduler, to contain
memory accesses.
2. The apparatus of claim 1 further comprising a processor coupled to the front side
bus access queue, wherein the processor comprises an out-of-order core and a hardware
10 prefetcher.
3. The apparatus of claim 2 further comprising a memory controller coupled to the
front side bus access queue.
4. The apparatus of claim 1 wherein the page open predictor predicts memory
accesses that are predicted to occur from a page in memory.
- 15 5. The apparatus of claim 1 wherein the front side bus scheduler to determine whether
an application being processed is latency tolerant or latency intolerant.
6. The apparatus of claim 1 wherein the first mode comprises the front side bus
scheduler to schedule memory accesses to minimize memory access latency, and the

second mode comprises the front side bus scheduler to schedule memory accesses to maximize memory bus bandwidth.

7. The apparatus of claim 6 wherein the minimize memory access latency mode comprises the front side bus scheduler to first schedule memory accesses associated with
5 earlier instructions.

8. The apparatus of claim 6 wherein maximize memory bus bandwidth mode comprises the front side bus scheduler to schedule memory accesses that are predicted to access data from an open memory page.

9. The apparatus of claim 1 wherein in the front side bus scheduler searches a
10 predetermined number of entries in the front side bus access queue to find entries that are predicted to access data from an open memory page, prior to scheduling the first entry in the front side bus access queue.

10. The apparatus of claim 5 wherein the front side bus scheduler automatically switches from the first mode to the second mode and vice versa.

15 11. The apparatus of claim 10 wherein the front side bus scheduler switches from the minimize memory access latency mode to the maximize memory bus bandwidth mode if for more than X percentage of accesses generated by the hardware prefetcher, more than Y percentage of accesses generated by the hardware prefetcher are also requested by the out-of-order core, based on any one of time and last n accesses and a weighted combination of
20 time and last n accesses.

12. A method comprising:
monitoring memory accesses generated by a hardware prefetcher;
determining whether the memory accesses from the hardware prefetcher are used
by an out-of-order core; and
5 switching memory accesses from a first mode to a second mode if a percentage of
the memory accesses generated by the hardware prefetcher are used by the out-of-order
core.
13. The method of claim 12 wherein the percentage of the memory accesses generated
by the hardware prefetcher that is used by the out-of-order core is at least 50 percent in a
10 unit time.
14. The method of claim 12 wherein the first mode comprises memory accesses to
minimize memory access latency, and the second mode comprises memory accesses to
maximize memory bus bandwidth.
15. The method of claim 14 wherein the minimize memory access latency mode
15 comprises the front side bus scheduler first scheduling memory accesses that are generated
by earlier instructions.
16. The method of claim 15 wherein accesses from the out-of-order core are scheduled
before accesses from the hardware prefetcher.

17. The method of claim 14 wherein the maximizing memory bus bandwidth mode comprises the front side bus scheduler scheduling memory accesses that are predicted to access data from an open memory page.

18. The method of claim 17 further comprising the front side bus scheduler searching
5 a predetermined number of entries in the front side bus access queue to find entries that are predicted to access data from an open memory page, prior to scheduling the first entry in the front side bus access queue.

19. The method of claim 12 further comprising switching from the minimize memory access latency mode to the maximize memory bus bandwidth mode if from the percentage
10 of memory accesses that are generated by the hardware prefetcher in a unit time, more than 50 percent of the memory accesses generated by the hardware prefetcher in the unit time are used by the out-of-order core.

20. A computer system comprising:

a bus;

15 a processor coupled to the bus;

a front side bus controller coupled to the bus, said front side bus controller comprising a page open predictor, and a front side bus scheduler, said front side bus scheduler to switch memory access modes from a first mode to a second mode, said front side bus scheduler coupled to the page open predictor; and

20 a front side bus access queue, coupled to the front side bus scheduler, to contain memory accesses.

21. The computer system of claim 20 wherein the processor comprises an out-of-order core and a hardware prefetcher.

22. The computer system of claim 20 wherein the page open predictor predicts memory accesses that are likely to occur from a page in memory.

5 23. The computer system of claim 20 wherein the front side bus scheduler switches memory access modes from a first mode to a second mode.

24. The computer system of claim 23 wherein the first mode comprises memory accesses to minimize memory access latency, and the second mode comprises memory accesses to maximize memory bus bandwidth.

10 25. The computer system of claim 24 wherein the minimizing memory access latency mode comprises the front side bus scheduler to first schedule memory accesses generated by earlier instructions.

26. The computer system of claim 24 wherein maximizing memory bus bandwidth mode comprises the front side bus scheduler to schedule memory accesses that are
15 predicted to access data from an open memory page.

27. The computer system of claim 24 wherein in the maximizing bus bandwidth mode the front side bus scheduler searches a predetermined number of entries in the front side bus access queue to find entries that are predicted to access data from an open memory page, prior to scheduling the first entry in the front side bus access queue.

28. The computer system of claim 22 wherein the front side bus scheduler switches from the minimize memory access latency mode to the maximize memory bus bandwidth mode if from the percentage of memory accesses that are generated by the hardware prefetcher in a unit time, more than 50 percent of the memory accesses generated by the
5 hardware prefetcher in the unit time are used by the out-of-order core.

29. The computer system of claim 26 wherein the front side bus scheduler switches from the minimize memory access latency mode to the maximize memory bus bandwidth mode dynamically.

30. The computer system of claim 22 wherein the front side bus scheduler switches
10 from the maximize memory bus bandwidth mode to the minimize memory access latency mode if from the percentage of memory accesses that are generated by the hardware prefetcher in a unit time, less than 50 percent of the memory accesses generated by the hardware prefetcher in the unit time are used by the out-of-order core.