BESCHWERDEKAMMERN DES EUROPÄISCHEN PATENTAMTS

BOARDS OF APPEAL OF THE EUROPEAN PATENT OFFICE

CHAMBRES DE RECOURS DE L'OFFICE EUROPEEN DES BREVETS

Internal distribution code:
(A) [ ] Publication in OJ
(B) [ ] To Chairmen and Members
(C) [ ] To Chairmen
(D) [X] No distribution

Datasheet for the decision of 24 May 2013

Case Number: T 2246/08 - 3.5.06

Application Number: 04781494.2

Publication Number: 1665054

IPC: G06F 12/08

Language of the proceedings: EN

Title of invention:
Decoupled store address and data in a multiprocessor system

Applicant:
Cray Inc.

Headword:
Store address buffer/CRAY

Relevant legal provisions (EPC 1973):
EPC Art. 56

Keyword:
"Inventive step (no)"

Decisions cited:
-

Catchword:
-

Europäisches Patentamt | European Patent Office | Office européen des brevets

Beschwerdekammern | Boards of Appeal | Chambres de recours

Case Number: T 2246/08 - 3.5.06

D E C I S I O N
of the Technical Board of Appeal 3.5.06
of 24 May 2013

Appellant: (Applicant)
Cray Inc.
411 First Avenue South
Suite 600
Seattle, Washington 98104-2860 (US)

Representative:
Kenrick, Mark Lloyd
Marks & Clerk LLP
1 New York Street
Manchester M1 4HD (GB)

Decision under appeal: **Decision of the Examining Division of the European Patent Office posted 9 July 2008 refusing European patent application No. 04781494.2 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chairman: D. H. Rees
Members: S. Krischer
W. Sekretaruk

Summary of Facts and Submissions

I. The appeal is directed against the decision of the examining division, posted on 9 July 2008, to refuse the application 04781494 for lack of novelty and inventive step over documents:

D1 D. Abts et al.: "So Many States, So Little Time: Verifying Memory Coherence in the Cray X1", Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03), April 2003, Piscataway/NJ, USA, IEEE, pages 1-10, XP10645295, ISBN 0-7695-1926-1, downloadable from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1213087, a more readable version downloadable from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.99&rep=rep1&type=pdf.

D2 J. L. Hennessy et al.: "Computer Architecture - A quantitative Approach", 2002, Morgan Kaufmann Publishers, San Francisco/CA, USA, pages 390-423, XP2318184.

In an Obiter Dictum of the decision (section 12), it was objected that there was a lack of clarity of the terms "shared memory" and "store address buffer" in claim 1.

II. A notice of appeal was received on 2 September 2008. The fee was received the same day. A statement of the grounds of appeal was received on 18 November 2008. Claim sets of a main request (identical to the refused sole request) and a first auxiliary request were filed. Oral proceedings were requested.

III. In its summons to oral proceedings, the board gave reasons for its preliminary opinion that claim 1 of the two requests lacked an inventive step over D1.

IV. In a letter dated 22 May 2013, the appellant withdrew the request for oral proceedings, filed a second auxiliary request and requested a decision.

V. The appellant *requests* that the decision be set aside and a patent be granted on the basis of the main request (claims 1-15) filed on 9 May 2008 (the refused sole request), the first auxiliary request (claims 1-15) filed with the grounds of appeal, or the second auxiliary request (claims 1-14) filed on 22 May 2013. The further text on file is: description pages 1, 3, 4, 6, 8-11 as published, pages 2, 5, 7, 12 as filed on 15 January 2007, pages 2a, 2b as filed on 9 May 2008; drawing sheets 1-5 as published.

VI. Claim 1 of the main request reads as follows:

"1. A method of decoupling an address from write data in a store to shared memory (16) of a computer system (10) having a plurality of processors (12, 14) connected to the shared memory (16), the method comprising:

generating a write request address for a memory write, wherein the write request address points to a memory location in shared memory (16);

issuing (50) a write request to the shared memory (16), wherein the write request includes the write request address;

noting the write request address in the shared memory (16), wherein the shared memory includes a store

address buffer (19) and wherein noting includes writing the write request address in the store address buffer;

comparing (56), in the shared memory (16), addresses in subsequent load and store requests to the write request address written in the store address buffer;

transferring the write data to the shared memory (16);

matching, within the shared memory, the write request address to the write data; and

storing the write data into the shared memory as a function of the write request address."

VII. Claim 1 of the first auxiliary request differs from claim 1 of the main request in that the last two steps read (differences marked in *italics*):

"matching, within the shared memory, the write request address *written in the store address buffer* to the write data; and

storing the write data into the shared memory as a function of the write request address *written in the store address buffer*."

VIII. Claim 1 of the second auxiliary request differs from claim 1 of the main request in that after the fourth step of comparing the following step is inserted:

"stalling subsequent read requests to the write request address until the write data is written into the shared memory;"

Reasons for the Decision

1. *Overview*

1.1 The application *relates* to the shared memory of a multiprocessor computer. It is often the case that the address for a write operation is known many clock cycles before the data of the write operation is available (original description page 5, paragraph 4). If another processor's memory accesses must be ordered after the first processor's write operations, then a conventional system may require waiting until the data is produced and the write is performed. The invention overcomes this problem by splitting the write operation into two parts (paragraph 5): a write address request and a write data request. The write address request is transferred first and its address is stored in a "store address buffer" (SAB) which is located in the shared memory (claim 1 of the three requests; and original description, page 5, lines 23-26; page 2, last paragraph; page 7, paragraph 7; page 9, paragraph 4). The addresses contained in subsequent memory access requests are compared against the entries in this buffer (claim 1; page 6, first paragraph).

1.2 The board agrees with the appellant that the arguments set out in the appealed decision (sections 4-7) do not convincingly demonstrate that claim 1 lacks novelty or inventive step. Nonetheless on the basis of its own analysis starting from the same closest prior art document D1 as the examining division, the board has come to the conclusion that claim 1 of all three requests lacks an inventive step.

2. *Novelty of claim 1*

The appealed decision gives two approaches leading to an objection of lack of novelty of claim 1 over D1, based on two interpretations of the term "store address buffer" (SAB).

2.1 In the *first* approach (section 4), the SAB is regarded as an entry in the memory directory of D1 (page 4, right column, first paragraph). In section 4.2, passages discussing the "Ecache" and the "memory directory" of D1 are cited to show the correspondence between the features of claim 1 and those of D1. However, the board finds that neither the Ecache nor the memory directory can be designated as being included in the shared memory of D1. In particular, a memory directory is a data structure for maintaining the set of sharers and the access permissions of Ecache lines (D1, page 2, left column, paragraph 5, around line 5), and is implemented in a dedicated chip, called M chip (page 3, left column, paragraph 6, around line 5; and page 4, right column, first paragraph, lines 1-4, 9-11).

2.1.1 Although the application as filed mixes up the included/located relationships between "shared memory", "cache" and SAB (e.g. original claim 4: the cache is *included in* the shared memory; original description, page 7, lines 29, 30: the SAB is *located in* the cache; page 8, lines 22, 23: the SAB is *stored in* the cache), it is clear that current claim 1 relates to an embodiment where the SAB is in the shared memory (in its proper sense), *and not in the cache*. This embodiment can be found in figures 1A, 1B and 4 (in

contrast to figure 3 with the SAB in the Ecache "EC"), and in the original description at page 5, line 23-26:

> "In the embodiment shown in Fig. 2, write address requests are sent to [shared] memory 16 at 50, where they are held in the memory system at 52, either by changing the state of the associated cache lines in a cache, *or by saving them in some structure*." (emphasis added by the board)

Further at page 7, lines 24, 25:

> "In one embodiment, [shared] memory 16 includes a store address buffer 19 for storing write addresses while the memory waits for the associated write data."

And at page 9, lines 20-21:

> "In the embodiment shown in Fig. 4, the store address buffer could be stored in either cache 24 *or shared memory 26*" (emphasis added by the board)

2.1.2 All claims and most (but not all - see e.g. page 8, lines 22-27; page 9, lines 20, 21 and page 10, lines 10-20) of the passages in the original description relating to the embodiment with the SAB in the cache (e.g. original claims 4, 14, 24; original description, page 5, lines 25, 26; page 7, lines 29, 30) have been deleted. Although the Ecache (and its E chip) and the M chip are mentioned in the description (page 9, lines 5, 7, 15), they support the deleted claims and relate to their embodiment with the SAB in the cache, and not to the embodiment with the SAB included in the shared memory of claim 1.

2.1.3 Since the aforementioned passages cited in the appealed decision (section 4.2) are only about the Ecache and the memory directory, they cannot disclose the SAB *included in the shared memory* of claim 1.

2.1.4 Thus, the first approach of the decision (section 4) does not convincingly show a lack of novelty of claim 1 over D1.

2.2 In the *second* approach (section 5), the SAB is regarded as the "transient buffer" of D1, page 4, right column, paragraph 4. Here again, this transient buffer relates to the cache and/or the memory directory, and not to the shared memory. For example, section 3.3 which contains the paragraph about the transient buffer is entitled "Cache Organisation and Memory Directory".

Thus, the second approach of the decision (section 4) also does not convincingly show a lack of novelty of claim 1 over D1.

3. *Inventiveness of claim 1*

3.1 In the *third and fourth* approaches of the decision (sections 6 and 7), the SAB is regarded as the "write buffer" of the second document D2, page 420, paragraph 3. The term "write buffer" is used in the present description in the sense of a buffer for addresses, allowing the decoupling of address and data (page 1, line 31 to page 2, line 2). However, the board finds that there is no suggestion in D2 of such decoupling. Indeed it is clear that address and data writing are regarded as atomic - see e.g. D2, page 405, lines 8-10 ("If the 'victim' was modified, its data and address are sent to the Victim Buffer. (This structure

is similar to a *write buffer* in other computers.)") or page 420, lines 29-31 ("If the write buffer is empty, the data and the full address are written in the buffer, and the write is finished from the CPU's perspective; ..."). For this reason, the board considers that the write buffer of D2 cannot be equated with the SAB as defined in the present claims and consequently the reasoning in the decision is unconvincing.

3.2 The board's analysis

3.2.1 Claim 1 of the first auxiliary request and the main request merely differ in the explicit addition of "written in the store address buffer" in the steps of matching and of storing. However, the board considers this features as implicitly present in the main request. Therefore, the requests are treated together in what follows.

3.2.2 Claim 1 of the second auxiliary request and the main request differ in the addition of the stalling step after the comparing step.

3.2.3 The board considers D1 to be novelty destroying for the deleted original claims 4, 14, 24 which all related to the embodiment with the SAB in the cache. As in these deleted claims, an Ecache line of D1 enters a "WaitFor[V]Data" state when the decoupled write address arrives. In D1, there is an additional "V" in "WaitForData". It corresponds to vector data; see table 2 and section 3.3.2, first paragraph. This correspondence with WaitForData was already pointed out during the International phase (the IPRP dated 13 July 2005, sections 3.2 and 4.4) for the original claims.

3.2.4 However, the IPRP considered original claim 1's step of comparing not to be present in D1. The board disagrees on that point. First, this step does not completely solve a memory access conflict as it might appear from the formulation of the problem solved given in section 3.5 of the IPRP, since the step of comparing is merely a first preparatory step in achieving a memory access conflict resolution. After a conflict is recognised, the step of stalling the subsequent memory request (in original claim 3 or 4) is necessary to resolve the conflict. Note that current claim 1 of the main and the first auxiliary request also does not contain the stalling step. Only claim 1 of the second auxiliary request contains that step (like original claim 3).

3.2.5 Second, there are at least three passages in D1 which would lead the skilled person to consider that the steps of comparing and stalling are inherent in the method disclosed by D1:

- page 5, left column, paragraph 3/point 2.: "Writes to the same address are serialized."
- paragraph 5/point 4.: "No SSP can read a value written by *another* MSP before that value becomes globally visible."
- same column, section 4 "Correctness Properties", lines 9-12: "For example, a hardware error in an arbiter may allow subsequent memory references to overtake earlier ones and thus violate the property of preserving individual program order."

Further it is clear that packets such as the "Inval" packet discussed in section 3.3.2 must include an

address which is compared to addresses in the cache. The same would obviously be true of a read access.

3.2.6 With the steps of comparing and stalling being implicitly disclosed in D1, original claims 1 and 4 were not new over D1.

3.2.7 Current claim 1 of the three requests differs from original claim 4 in that the SAB is included in the shared memory, and not in the cache. Therefore, claim 1 *differs* from D1 in that feature, since it is not disclosed in D1, as shown above.

3.2.8 The objective technical *problem* would then be the same as in the decision (6.6), namely how to provide an alternative solution for releasing a processor and reducing memory latency after a write request.

3.2.9 The board considers it obvious for a skilled person to select the shared memory as an alternative place for the SAB: The SAB needs to be shared by all processors in order to allow them to send their write addresses requests to the SAB. The Ecache of D1 is one such shared memory region. The (proper) shared memory is another one and is therefore the straightforward candidate for an alternative location of the SAB. There is no indication of any advantage or unexpected technical effect in choosing to locate the SAB in the shared memory. In addition, in a system in which the global shared memory is not accessed through a cache dedicated to the shared memory as a whole, which would be perfectly reasonable, there would be no alternative but to store the SAB in the shared memory itself.

3.2.10 Therefore, claim 1 of the three requests is not inventive in the sense of Article 56 EPC 1973.

3.3 Arguments of the appellant

3.3.1 In his letter dated 22 May 2013 (in response to the summons for oral proceedings), the appellant argued that there was a difference between current claim 1 and original claim 4 (and thus D1) in the *way* in which the write request addresses are noted (page 2, paragraph 3). In current claim 1, the address was written in the store address buffer whereas in original claim 4 only a state in a cache line was changed. It was argued that, consequently, "in the arrangement of original claim 4 a cache is provided ... and there is no store address buffer and no writing of an address in the store address buffer." However, on the first point, the original description frequently calls the storage locations used in the cache a "store address buffer". As already stated in the summons (4.1.1.), there are many passages in the description disclosing two equivalent locations for the SAB, namely the cache or the shared memory. This means that the cache lines used are an SAB as defined by the application. On the second point, in order that a state in a cache line can be changed, the address has to have been written in that cache line at some point, either when data was read from the addressed location or at the point of the write, if there has been no previous read. Therefore there is inevitably a "writing of the address in the store address buffer".
Therefore, the board is not convinced by this argument.

3.3.2 As to the second auxiliary request, the appellant rightly states that the objection in the summons (5.2.3)

no longer holds that claim 1 does not resolve memory access conflicts, since the step of stalling is missing. However, as explained above, the board considers the memory access resolution to be implicitly disclosed in D1, including the indispensable stalling step. (The board notes that the summons already gave its reasons for the view that original claim 4, which included stalling, lacked novelty with respect to D1, i.e. that stalling was implicit in the disclosure of D1.) Therefore, the board is also not convinced by this argument.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

B. Atienza Vivancos D. H. Rees