

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
19 February 2009 (19.02.2009)

PCT

(10) International Publication Number
WO 2009/023076 A1

(51) International Patent Classification:
G06F 9/46 (2006.01)

(21) International Application Number:
PCT/US2008/008625

(22) International Filing Date: 14 July 2008 (14.07.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/838,630 14 August 2007 (14.08.2007) US

(71) Applicant (for all designated States except US):
HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P. [US/US]; 20555 S.h. 249, Houston, Texas 77070 (US).

(72) Inventors; and

(75) Inventors/Applicants (for US only): **IMARK, Robert R.** [US/US]; 3404 East Harmony Road, Fort Collins, Colorado 80528 (US). **GASSER, Raymond A.** [US/US]; 3404 East Harmony Road, Fort Collins, Colorado 80528 (US).

(74) Agents: **WEBB, Steven L** et al.; Hewlett-Packard Company, Intellectual Property Administration, P.O. Box 272400, Mail Stop 35, Fort Collins, Colorado 80527-2400 (US).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

- as to the identity of the inventor (Rule 4.17(i))
- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))
- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))

[Continued on next page]

(54) Title: SYNCHRONIZATION OF PROCESSORS IN A MULTIPROCESSOR SYSTEM

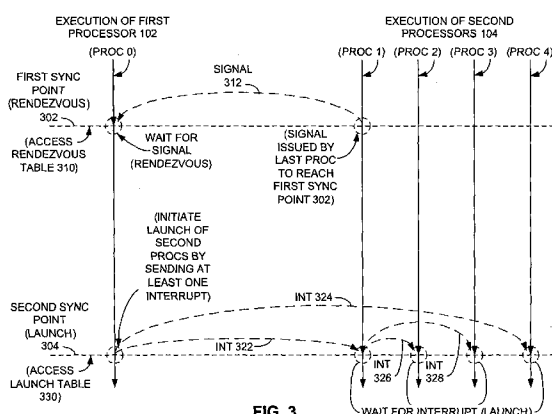


FIG. 3

(57) Abstract: A method (200) for synchronizing a first processor and multiple second processors is presented. In the method (200), each of the second processors waits (202) at a second synchronization point after reaching a first synchronization point. The last of the second processors to reach the first synchronization point sends (204) a signal to the first processor. The first processor waits (206) at the first synchronization point until it receives the signal. After receiving the signal, the first processor initiates (208) a launch of the second processors by launching at least one of the second processors. At least one of the second processors launched by the first processor launches another of the second processors in response to being launched by the first processor. Each of the second processors continues (210) execution from the second synchronization point in response to being launched.

WO 2009/023076 A1



Published:

- *with international search report*
- *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

SYNCHRONIZATION OF PROCESSORS IN A MULTIPROCESSOR SYSTEM

BACKGROUND

[0001] Designers of computer systems consistently strive for increased processing capacity with each product generation. Many different approaches have been adopted to achieve the computing speeds currently enjoyed by users of such systems. Increased system clock speeds, integrated circuit design advances, wider data paths, and various other technological developments have all contributed to increasing the processing throughput of a single processor.

[0002] To further enhance computer capability, pipelined and parallel arrangements of multiple processing units have been pursued successfully. Parallel processing generally began with the use of “single instruction stream, multiple data stream” (SIMD) architectures, in which multiple processors perform identical operations on different data. In such a system, a single program line, or “thread,” of instructions is executed. More advanced “multiple instruction stream, multiple data stream” (MIMD) systems allow each processor to execute a completely diverse set of instructions, or a separate copy of the same set of instructions.

[0003] However, even in an MIMD system, some communication or cooperation between the various processors is typically required. One type of communication involves synchronizing two or more of the processors by requiring each of the processors to halt execution at a predetermined point in its execution thread (called “rendezvous”), and then begin execution again at the same or another predetermined location (termed “launch”). One or more such synchronization points may be employed depending on the general nature and specific requirements of the overall task to be performed.

[0004]

[0005] Typically, a multiprocessing computer system supports the use of synchronization points by way of complicated, specialized and nonstandard hardware functions. For example, a dedicated hardware implementation of a specialized broadcast interrupt may be supplied to support the use of launch points so that a single processor may inform all other processors of a launch quickly and efficiently. However, other multiprocessor systems may not provide such hardware due to the design and implementation expense involved in supporting such a specialized hardware construct.

BRIEF DESCRIPTION OF THE DRAWINGS

[0006] Fig. 1 is a block diagram of a multiprocessor system according to an embodiment of the invention.

[0007] Fig. 2 is flow diagram of a method for synchronizing the processors of Fig. 1 according to an embodiment of the invention.

[0008] Fig. 3 is a graphic representation of execution of the processors in the multiprocessor of Fig. 1 according to another embodiment of the invention.

[0009] Fig. 4 is a diagram of a rendezvous table employed in the processor execution of Fig. 3 according to an embodiment of the invention.

[0010] Fig. 5 is a diagram of a launch table employed in the processor execution of Fig. 3 according to an embodiment of the invention.

DETAILED DESCRIPTION

[0011] Fig. 1 provides a simplified block diagram of a multiprocessor system 100 including a first processor 102 and multiple second processors 104. In other embodiments, as few as two second processors 104, or many more than that depicted in Fig. 1, may be included. The multiprocessor system 100 may employ any type of multiprocessor architecture in which synchronization of two or more of the processors is desired. One possible architecture for the multiprocessor system may be a symmetric multiprocessing (SMP) system, although any of a plethora of other

[0012]

[0013] multiprocessor architectures may benefit from the inventive concepts described below.

[0014] While the first processor 102 is distinguished from the second processors 104, the first processor 102 and the second processors 104 may all be equivalent in terms of physical and electronic construction. The first processor 102 is instead distinguished in terms of its role in the synchronization of the processors 102, 104 from that of the second processors 104, as described in greater detail below. Further, each of the second processors 104 serve a similar synchronization function. However, the second processors 104 may or may not be similar in design and construction to each other. Only the functionality of the processors 102, 104 as described below is relevant to the embodiments presented herein.

[0015] Fig. 2 presents by way of a flow diagram a method 200 of synchronizing the processors 102, 104 of Fig. 1. In the method 200, each of the second processors 104 waits at a second synchronization point after reaching a first synchronization point (operation 202). The last of the second processors 104 to reach the first synchronization point sends a signal to the first processor 102 (operation 204). The first processor 102 waits at the first synchronization point until receiving the signal (operation 206), and initiates a launch of the second processors 104 after receiving the signal (operation 208). More specifically, the first processor initiates the launch by launching at least one of the second processors 104, wherein the second processor 104 being launched launches another one of the second processors 104 in response to being launched. Each of the second processors 104, in response to being launched, continues execution from the second synchronization point (operation 210).

[0016] Fig. 3 graphically characterizes the execution of the first processor 102 and four separate second processors 104 according to a more detailed embodiment of the invention. While four separate second processors 104 are employed in this particular example, any number of second processors 104 greater than one may be utilized in other embodiments. For identification purposes, the first processor 102 is identified as Processor 0, while the second processors 104 are labeled Processors 1-4. Execution of each of the processors 102, 104 is depicted as progressing from top to

[0017]

[0018] bottom in relation to a first synchronization point 302 and a second synchronization point 304. In one embodiment, the first synchronization point 302 is a rendezvous or gathering point for the processors 102, 104, while the second synchronization point 304 is a launching point from which the processors 102, 104 continue execution. Although the synchronization points 302, 304 are represented in Fig. 3 as being located in completely separate areas of the execution stream of each of the processors 102, 104, the first synchronization point 302 and the second synchronization point 304 may represent approximately the same point in the execution thread of any or all of the processors 102, 104 in another example.

[0019] In Fig. 3, the first processor 102 executes until reaching the first synchronization point 302, whereupon the first processor 102 waits for a signal 312 from one of the second processors 104. Meanwhile, each of the second processors 104 executes through the first synchronization point 302 to the second synchronization point 304. Each of the second processors 104 accesses a rendezvous table 310 at the first synchronization point 302 to indicate that the accessing second processor 104 has reached the first synchronization point 302. The second processor 104 that is last to reach the first synchronization point 302 issues the signal 312 to the first processor 102 so that it may continue execution.

[0020] Fig. 4 illustrates one specific example of the rendezvous table 310 that each of the second processors 104 accesses to indicate its arrival at the first synchronization point 302. The rendezvous table 310 includes several entries, wherein each entry is depicted in Fig. 3 as a row. Further, each entry is associated with an address 402, and includes three fields: a processor count field 404, a processor threshold field 406, and a next address field 408. Also shown in Fig. 4 is the entry of the rendezvous table 310 that is associated with each of the processors 102, 104, identified by the address 402 of the entry.

[0021] As each second processor 104 reaches the first synchronization point 302, that second processor 104 accesses the entry of the rendezvous table 310 to which it is assigned. The second processor 104 then performs an atomic read-modify-write operation of the processor count field 404 of that entry and compares the value

[0022]

[0023] read from the processor count field 404 to the processor threshold field 406. If the values are equal, the second processor 104 then accesses the rendezvous table 310 entry indicated in the next address field 408 in the same fashion as the previous entry. This process continues for each of the second processors 104 until an accessed processor count field 404 is found to be less than its associated processor threshold field 406.

[0024] Using the scenario depicted in Figs. 3 and 4 as an example, Processors 1-3 are each assigned the rendezvous table 310 entry located at address 7066200. Each of these second processors 104 reads the processor count field 404 of that entry and compares it to the processor threshold field 406. Thus, the first of Processors 1-3 that reaches the first synchronization point 302 and accesses the rendezvous table 310 entry at 7066200 reads a value of zero from the processor count field 404 and writes the incremented value of one back to the processor count field 404. That processor also compares the value of zero read from the processor count field 404 to the processor threshold 406 of two, and ceases this particular access of the rendezvous table 310 as a result. Similarly, the second of the Processors 1-3 to reach the first synchronization point 302 accesses the same entry of the rendezvous table 310, resulting in a value of two being stored for the processor count field 404 at address 7066200.

[0025] Continuing in this manner, the third of the Processors 1-3 to pass through the first synchronization point 302 accesses the same entry. However, after reading a value of two from the processor count field 404 and writing back a three thereto, that particular second processor 104 compares the two to the processor threshold 406 of two, and after finding that they are equal, accesses the next address field 408, which stores an address of 7066100. The last of the Processors 1-3 reaching the first synchronization point 302 then accesses the entry of the rendezvous table 310 at the address of 7066100 and repeats the process. Assuming this processor 104 reaches the first synchronization point 302 before Processor 4 (the operation of which is addressed below), the processor count field 404 of zero is read, a one is written back thereto, and the zero is compared to the processor threshold field 406 of

[0026]

[0027] one. As a result, the last of these second processors 104 (i.e., Processors 1-3) ceases its access of the rendezvous table 310.

[0028] Proceeding with the example of Figs. 3 and 4, the second processor 104 referred to as Processor 4 also accesses the entry of the rendezvous table 310 at address 7066100 when it reaches the first synchronization point 302. In this case, Processor 4 is the last second processor 104 to reach the first synchronization point 302. After reading the value of one from the processor count field 404 (written therein by the last of Processors 1-3, as described above), Processor 4 compares the one to the value of one written in the processor threshold field 406. With these two values being equal, Processor 4 reads the next address field 408 that holds the value of 7066000. Processor 4 then reads the rendezvous table 310 entry at that address, reads the processor count field 404 of zero, stores the incremented value of one thereto, and compares the zero to the zero stored in the processor threshold field 406 of that entry. With the compared values being equal, Processor 4 then reads the next address field 408 of that entry, which holds the value 0000000. In this embodiment, reading all zeros indicates to Processor 4 that it is the last of the second processors 104 to reach the first synchronization point 302.

[0029] In response to being the last of the second processors 104, Processor 4 sends a signal 312 to Processor 0 (i.e., the first processor 102 of Fig. 3). The signal 312 may be issued in a number of ways. In one embodiment, the act of Processor 4 writing a one to the processor count field 404 of the rendezvous table 310 entry at address 7066000 may serve as the signal 312. In that case, Processor 0 may poll the processor count field 404 for a one to be written thereto, interpreting the one as the signal 312. In another implementation, Processor 4 writes a separate memory location, sends a message to Processor 0, or performs some other operation to implement the signal 312 to Processor 0. In addition, Processor 0 or Processor 4 may clear the processor count field 404 of each entry of the rendezvous table 310 to initialize the table 310 for the next time the first synchronization point 302 or other similar rendezvous point is employed.

[0030] Write operations to the processor count fields 404 as described above specifically employ an atomic read-modify-write operation often used in multiprocessor systems for processor intercommunication so that conflicts in accessing the field 404 will not arise between two or more of the second processors 104. For example, use of an atomic operation eliminates the possibility that two of the second processors will read the same value from the same processor count field 404. Other memory accesses that prevent such conflicts, such as standard “semaphore” or “mailbox” operations, could be utilized in other embodiments.

[0031] By employing multiple rendezvous table 310 entries, a hierarchical structure of memory locations is formed by which the second processors 104 indicate reaching the first synchronization point 302. Thus, by spreading the access to the rendezvous table 310 by each of the second processors 104 across multiple memory locations, access contention for those locations, which potentially is exacerbated by the atomic memory operations, is greatly decreased, resulting in faster signaling of the first processor 102. In addition, the particular embodiment of Fig. 4 allows generation of the signal 312 by way of a single access of the processor count field 404 of address 7066000, which the first processor 102 may be polling, as described above. In contrast to a single counter being updated by each of the second processors 104 while the first processor 102 polls the same memory location, the hierarchical model of Fig. 4 provides a more efficient method for processor synchronization.

[0032] A similar hierarchical model employing interrupts is utilized in Fig. 3 in launching the second processors 104 from the second synchronization point 304. More specifically, after the first processor 102 (i.e., Processor 0) receives the signal 312, the first processor 102 resumes execution to the second synchronization point 304, at which time it initiates a launch of each of the second processors 104 (i.e., Processors 1-4) by sending a first interrupt 322 to one of the second processors 104, which is Processor 1 in the embodiment of Fig. 3. In this specific example, Processor 0 also issues a second interrupt 324 to Processor 4. In turn, after receiving the first interrupt 322, Processor 1 issues a third interrupt 326 to Processor 2, and a fourth interrupt 328 to Processor 3. In response to receiving one of the interrupts 322-328, each of the second processors 104 then continues execution from the second

[0033]

[0034] synchronization point 304, along with the first processor 102. Thus, the first synchronization point 302 and the second synchronization point 304 are employed together to rendezvous and subsequently launch the processors 102, 104.

[0035] Fig. 5 graphically shows an embodiment of a launch table 330 accessed by each of the Processors 0-4 to perform the hierarchically-structured launch. As with the rendezvous table 310, the launch table 330 includes a number of entries, each of which is accessible via an address 502. Within each entry reside one or more interrupt indicators 504, wherein each indicator 504 is associated with a particular launch interrupt. In one embodiment, each interrupt indicator 504 is an interrupt mask and enable value that specifies an interrupt associated with a particular second processor 104. Thus, the processor 102, 104 issuing the interrupt may merely write the data stored at the interrupt indicator 504 to a location that causes the processor 102, 104 to generate the interrupt. Other indicators, such as a simple binary value indicating the specific second processor 104 to be launched, may be used in other examples. Fig. 5 also indicates the launch table 330 entry address and interrupt associated with each of the processors 102, 104.

[0036] In the particular example of Fig. 5, Processor 0 (i.e., the first processor 102) accesses its assigned launch table 330 address 7066700 after reaching the second synchronization point 304. The entry at address 7066700 indicates that interrupts are to be issued to Processor 1 (by way of interrupt 322) and Processor 4 (via interrupt 324), as shown in the interrupt indicators 504 of that entry. Further according to the embodiment of Fig. 5, an interrupt indicator 504 of zero denotes the end of the list of interrupts to be issued by the processor accessing the associated entry of the launch table 330.

[0037] After receiving its launch interrupt 322, Processor 1 accesses its assigned launch table entry 330 at address 7066800, which indicates that Processor 1 is to issue launch interrupts 326 and 328 for Processors 2 and 3, respectively. Similarly, Processor 4 accesses its assigned entry at address 7066900 in response to receiving interrupt 324 from Processor 0. However, the entry at address 7066900 lists no further launch interrupts to be issued. Similarly, Processors 2 and 3 access the

[0038]

[0039] same entry in response to receiving interrupts 326, 328, and find that they are not responsible for issuing any launch interrupts, either. As a result, each of the second processors 104 has received a launch interrupt, even though the first processor 102 has only issued two launch interrupts directly. As a result, the work required to issue the interrupts has been distributed among the processors 102, 104 in a hierarchical fashion, thus hastening the launch process.

[0040] In one embodiment, the rendezvous table 310, the launch table 330, and the assigned table addresses for each of the processors 102, 104 are initialized by one or more of the processors 102, 104. According to another implementation, a separate processor not discussed above may perform this function. Once the tables 310, 330 are initialized, further setup of the tables 310, 330 may not be required during the use of multiple synchronizations of the processors 102, 104.

[0041] The use of the two synchronization points 302, 304 may be advantageous in a number of processing contexts. In one example, each of the processors 102, 104 may act as a producer of service requests before the first synchronization point 302 and after the second synchronization point 304. In response, the first processor 102 may then process or consume the service requests after the first synchronization point 302 and before the second synchronization point 304 while the second processors 104 remain idle at the second synchronization point 304. The service requests may constitute requests for any service that may be provided by the first processor 102, including but not limited to generating billing records, or searching and retrieving items in a database.

[0042] In another example, the first synchronization point 302 may be used to implement a "join" operation employed in a MIMD multiprocessing system, while the second synchronization point 304 may be utilized as part of a "fork" operation. In such an environment, the first processor 102 may execute a single thread after the first synchronization point 302 while the second processors 104 wait to be called on at their second synchronization point 304. The first processor 104 may then execute a fork operation to spawn the second processors 104 by way of the launch process from the second synchronization point 304, as described above. As each of the second

[0043]

[0044] processors 104 then finishes the work to which it was assigned, execution in that second processor 104 reaches the next first synchronization point 302. The last of the second processors 104 to reach the first synchronization point 302 then issues the signal 312 to the first processor 102, thus joining the execution of the processors back together. The first processor 102 may then operate as a lone thread until another fork operation is undertaken at another second synchronization point 304.

[0045] Various embodiments of a multiprocessor system and method as discussed above may provide significant benefits. Since a combination of simple shared memory communication and individual interrupts is employed to effectuate the rendezvous and launch processes, the embodiments may be implemented on most multiprocessing systems. In addition, the use of a logical processor hierarchy among the second processors 104, with the first processor 102 residing at the top of the hierarchy, facilitates quick execution of both the rendezvous and launch phases of the synchronization.

[0046] While several embodiments of the invention have been discussed herein, other embodiments encompassed by the scope of the invention are possible. For example, while many embodiments as described above specifically involve the use of a handful of processors within a multiprocessor system, other embodiments employing many more processors coupled together within a single system may exhibit even greater advantages over more serially oriented solutions due to the hierarchical nature of the synchronization mechanisms distributing the required work among many more processors. Further, aspects of one embodiment may be combined with those of alternative embodiments to create further implementations of the present invention. Thus, while the present invention has been described in the context of specific embodiments, such descriptions are provided for illustration and not limitation. Accordingly, the proper scope of the present invention is delimited only by the following claims and their equivalents.

[0047]

CLAIMS

What is claimed is:

1. A method (200) for synchronizing a first processor and second processors, the method (200) comprising:

in each of the second processors, waiting (202) at a second synchronization point after reaching a first synchronization point;

in the last of the second processors to reach the first synchronization point, sending (204) the first processor a signal;

in the first processor, waiting (206) at the first synchronization point until receiving the signal, and initiating (208) a launch of the second processors after receiving the signal by launching one of the second processors, wherein the one of the second processors launches another one of the second processors in response to being launched; and

in each of the second processors, continuing (210) execution from the second synchronization point in response to being launched.

2. The method (200) of claim 1, wherein:

initiating (208) the launch of the second processors comprises initiating a plurality of interrupts;

launching the one of the second processors comprises sending a first of the interrupts to the one of the second processors; and

launching the other one of the second processors comprises sending one of the remaining interrupts to the other one of the second processors in response to receiving the first interrupt.

3. The method (200) of claim 2, further comprising:

in each of the second processors, after receiving one of the interrupts, sending another one of the interrupts to another one of the second processors if indicated in a memory location assigned to the second processor sending the other one of the interrupts.

4. The method (200) of claim 3, wherein sending another one of the interrupts comprises writing to a memory-mapped address indicated in the memory location assigned to the second processor sending the other one of the interrupts.
5. The method (200) of claim 1, further comprising:
 - in each of the second processors, after reaching the first synchronization point, updating at least one of a plurality of memory locations, wherein the memory locations indicate whether all of the second processors have reached the first synchronization point.
6. The method (200) of claim 1, further comprising:
 - in the first processor, continuing execution from the first synchronization point to the second synchronization point in response to receiving the signal, and initiating the plurality of interrupts after reaching the second synchronization point.
7. The method (200) of claim 6, further comprising:
 - in the processors, generating service requests for the first processor during execution before the first synchronization point and after the second synchronization point;
 - wherein continuing execution in the first processor from the first synchronization point to the second synchronization point in response to receiving the signal comprises processing the service requests.
8. The method (200) of claim 1, wherein the first synchronization point comprises the second synchronization point.
9. A computer-readable storage medium comprising instructions executable on a first processor and second processors for employing a method (200) for synchronizing the processors, the method (200) comprising:
 - in each of the second processors, waiting (202) at a second synchronization point after reaching a first synchronization point;
 - in the last of the second processors to reach the first synchronization point, sending (204) the first processor a signal;
 - in the first processor, waiting (206) at the first synchronization point until receiving the signal, and initiating (208) a launch of the second processors after

receiving the signal by launching one of the second processors, wherein the one of the second processors launches another one of the second processors in response to being launched; and

in each of the second processors, continuing (210) execution from the second synchronization point in response to being launched.

10. A multiprocessor system (100), comprising:

a first processor (102) configured to wait at a first synchronization point (302) until receiving a signal (312); and

second processors (104), wherein each of the second processors (104) is configured to wait at a second synchronization point (304) after reaching the first synchronization point (302), and to send the signal (312) to the first processor (102) if last of the second processors (104) to reach the first synchronization point (302);

wherein the first processor (102) is configured to initiate a launch of the second processors (104) after receiving the signal (312) by launching one of the second processors (104);

wherein the one of the second processors (104) is configured to launch another one of the second processors (104) in response to being launched; and

wherein each of the second processors (104) is configured to continue execution from the second synchronization point (304) in response to being launched.

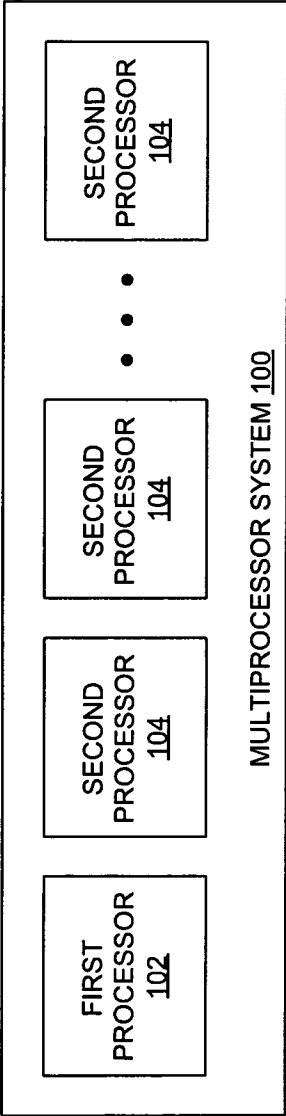


FIG. 1

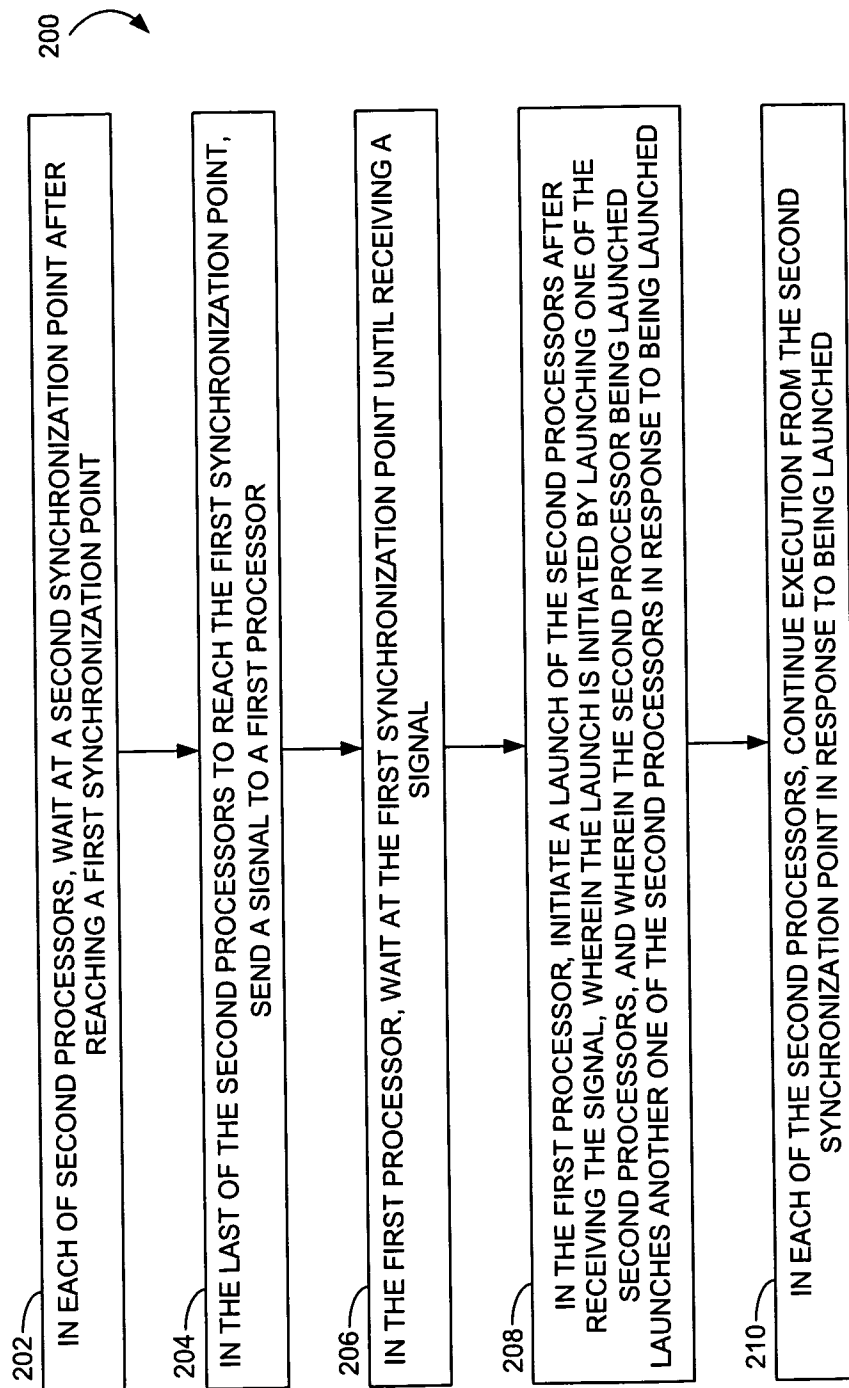


FIG. 2

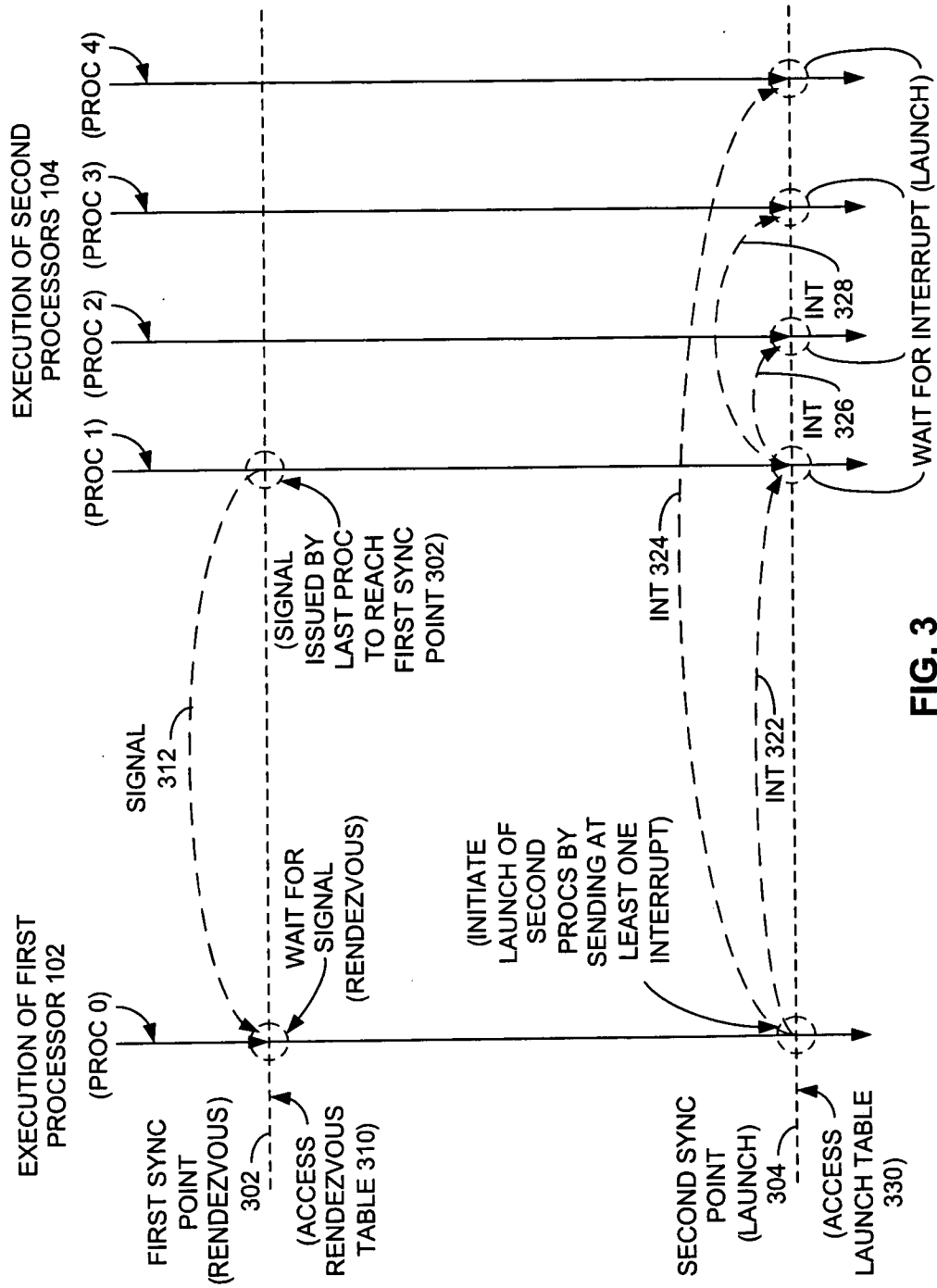


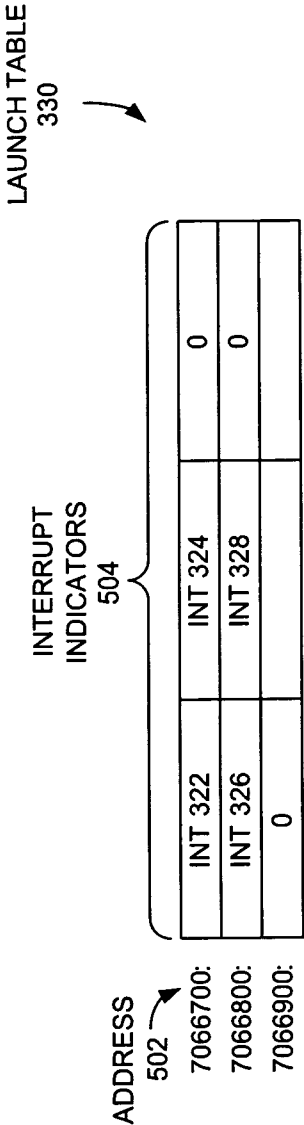
FIG. 3

RENDEZVOUS
TABLE
310

ADDRESS 402	PROCESSOR COUNT 404	PROCESSOR THRESHOLD 406	NEXT ADDRESS 408
7066000:	0	0	0000000
7066100:	0	1	7066000
7066200:	0	2	7066100

PROCESSOR	ASSIGNED RENDEZVOUS TABLE 310 ENTRY ADDRESS
0 (FIRST PROCESSOR 102)	7066000
1 (SECOND PROCESSOR 104)	7066200
2 (SECOND PROCESSOR 104)	7066200
3 (SECOND PROCESSOR 104)	7066200
4 (SECOND PROCESSOR 104)	7066100

FIG. 4



PROCESSOR	ASSIGNED LAUNCH TABLE 330 ENTRY ADDRESS	INTERRUPT RECEIVED
0 (FIRST PROCESSOR 102)	7066700	N/A
1 (SECOND PROCESSOR 104)	7066800	INT 322
2 (SECOND PROCESSOR 104)	7066900	INT 326
3 (SECOND PROCESSOR 104)	7066900	INT 328
4 (SECOND PROCESSOR 104)	7066900	INT 324

FIG. 5

A. CLASSIFICATION OF SUBJECT MATTER**G06F 9/46(2006.01)i**

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)

IPC : G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models since 1975

Japanese utility models 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)

eKIPASS(Kipo Internal), NDSL, Google

keywords: multi*, parallel, processor*, synchronization, point

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US2005/0193148 A1 (CHEN, C.J. et al.) 01 SEPTEMBER 2005 See figure 3; paragraphs [5][19][20][23].	1-10
A	US06032173 A (REINMUTH, A.) 29 FEBRUARY 2000 See Summary.	1-10
A	JP2708172 B2 (TOKYO SHIBAURA ELECTRIC CO.) 17 OCTOPBER 1997 See Claim 1.	1-10
A	JOHNSON, T.A. and HOARE, R.R. 'Cyclical Cascade Chains: A Dynamic Barrier Synchronization Mechanism for Multiprocessor Systems' In Parallel and Distributed Processing Symposium, 15th International Conference, APRIL 2001. See page 1, right colum 1st paragraph, page 5, left column 2nd paragraph.	1-10
PA	US2007/0240158 A1 (CHAUDHRY, S. et al.) 11 OCTOBER 2007 See figure 11 and its description.	1-10



Further documents are listed in the continuation of Box C.



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

"E" 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

"T" 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

"X" 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

22 JANUARY 2009 (22.01.2009)

Date of mailing of the international search report

22 JANUARY 2009 (22.01.2009)

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
Government Complex-Daejeon, 139 Seonsa-ro, Seo-
gu, Daejeon 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

YOON, Hye Sook

Telephone No. 82-42-481-8370



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2008/008625

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2005-0193148 A1	01.09.2005	NONE	
US 6032173 A	29.02.2000	AT 143154 T DE 4219005 A1 EP 0645034 A1 JP 7503565 T WO 93-25966 A1	15.10.1996 16.12.1993 29.03.1995 13.04.1995 23.12.1993
JP 2708172 B2	17.10.1997	NONE	
US 2007-0240158 A1	11.10.2007	NONE	